User's Manual

# R-Car V4M Series

## User's Manual: Hardware

arm

R-Car V4M Series (R8A779H2*) Products
R-Car V4M Series (R8A779H3*) Products
R-Car V4M Series (R8A779H4*) Products
R-Car V4M Series (R8A779H5*) Products

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.0.81 May 2025

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

© 2025 Renesas Electronics Corporation. All rights reserved.

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation.
Arm® and Cortex® are registered trademarks of Arm Limited.
The Arm logo is a registered trademark of Arm Limited.
For the “Cortex” notation, it is used as follows;
- Arm® Cortex®-A76
- Arm® Cortex®-R52

Note that after this page, they may be noted as Cortex-A76 and Cortex-R52 respectively.

## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

© 2025 Renesas Electronics Corporation. All rights reserved.

# General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

**List of Abbreviations**

| Abbreviations | Description |
|---|---|
| ADG | Audio Clock Generator |
| AES-ACC | Advanced Encryption Standard Accelerator |
| AOPS | Always On Persistent State |
| APMU | Advanced Power Management Unit |
| APSREG | AP-System register |
| AP-System Core | AP-System Core |
| AVS | Adaptive Voltage Scaling |
| AXI-bus | AXI-bus |
| GSX domain AXI | Graphics domain AXI |
| HSC domain AXI | High speed Communication domain AXI |
| IMP domain AXI | IMP domain AXI |
| MM domain AXI | Main Memory domain AXI |
| Peripheral domain AXI | Peripheral domain AXI |
| RT domain AXI | Real Time domain AXI |
| Slave Access Bus | Slave Access Bus |
| VC domain AXI | Video Codec domain AXI |
| VIO domain AXI | Video IO domain AXI |
| VIP domain AXI | Vision IP Processor domain AXI |
| BKBUF | Back-up Buffer |
| BOOTROM | Boot ROM |
| CCI | Cache Coherent Interconnect |
| CKM | Clock Monitor in Slave Access Bus domain |
| CKM3DG | Clock Monitor in GSX (Graphics) domain |
| CKMCNR | Clock Monitor in IMP (CNN) domain |
| CKMCR52 | Clock Monitor in RT (CR52) domain |
| CKMDSP | Clock Monitor in IMP (DSP) domain |
| CKMHSC | Clock Monitor in HSC (High speed Communication) domain |
| CKMIR | Clock Monitor in IMP domain |
| CKMMM | Clock Monitor in Main Memory domain |
| CKMPE0 | Clock Monitor in Peripheral domain |
| CKMRT | Clock Monitor in RT (Real Time) domain |
| CKMVC | Clock Monitor in VC (Video Codec) domain |
| CKMVIO | Clock Monitor in VIO (Video IO) domain |
| CKMVIP | Clock Monitor in VIP (Vision Processor) domain |
| CMT0 | Compare Match Timer Type0 |
| CMT1 | Compare Match Timer Type1 |
| CMT2 | Compare Match Timer Type2 |
| CMT3 | Compare Match Timer Type3 |
| CPG | Clock Pulse Generator |

| | |
|---|---|
| CRC | Cyclic Redundancy Check |
| CSI2 | CSI2 |
| CVE | Computer Vision Engine |
| DBE | JTAG Port Selector |
| DBG | Debug & Trace |
| DBSC5 | External Bus Controller for SDRAM |
| DDRPHY | DDRPHY |
| DOC | Display Output Checker |
| DOF or UMF | Dense type of Optical Flow vectors generation |
| DPAD | Display Unit PAD |
| DSI/CSI-2-TX | DSI/CSI-2 TX |
| DTA | Debug Trace Agent |
| DU | Display Unit |
| EAVB | EthernetAVB |
| ECM | Error Control Module |
| E-FUSE | E-FUSE |
| ETSN | EthernetTSN |
| FBA | Field BIST Activator |
| FBABUS | FBA Bus Module |
| FBC | Field BIST controller |
| FBJ | JTAG Sequencer |
| FCPRC, FCPRS | Data Compression Processor |
| FCPVD, FCPVX | Frame Compression Processor for VSP |
| FLXA | FlexRay |
| Generic counter | Generic counter |
| GPTP | gPTP |
| HSCIF | High Speed Serial Communication Interface with FIFO |
| HSSTP | High Speed Serial Trace Port |
| I2C | I2C Bus Interface |
| ICUMX | Intelligent Cryptographic Unit/Master Extension |
| IMP CNN | Convolution Neural Network |
| IMP core | IMP core |
| IMP DMAC | IMP DMAC |
| IMP PSC | IMP PSC |
| IMPSLV | IMPSLV |
| IMR-LX6 | IMR-LX6 |
| INTC | Interrupt Controller |
| ISP Core | Image Signal Processor |

| | |
|---|---|
| iVCP1E | Video Encoding Processor for Inter-device Video Transfer |
| LIFEC | Life Cycle |
| MFIS | Multifunctional Interface |
| MMC | SD Card/MMC Interface |
| MSIOF | Clock-Synchronized Serial Interface with FIFO |
| MTI | Measurement Trace Interface |
| OCDIDCMP | On-Chip Debug ID Comparator |
| OTP | One time program Memory |
| PCIEC | PCIe Controller |
| PFC | Pin Function |
| PWM | Pulse Width Modulation |
| RFSO | Failure Self-Detection Output |
| RGX | 3D Graphics Processing Unit |
| RPC-IF | SPI Multi I/O Bus Controller |
| RS-CANFD | CANFD Interface |
| RSENS | Remote sense |
| RT-DMAC | Direct Memory Access Controller |
| RT-VRAM | RT-VRAM |
| RWDT | RCLK Watchdog Timer |
| SCEG | Secure Boot Engine |
| SCIF | Serial Communication Interface with FIFO |
| SCMT | System Timer |
| SLIM-IMP DMAC | Slim-IMP DMAC |
| SMD | Structure from motion engines |
| SSIU | Serial Sound Interface Unit |
| SUCMT | System Up-Time Clock |
| SWDT | System Watchdog Timer |
| SYSC | System Controller |
| SYS-DMAC | Direct Memory Access Controller |
| TMU | Timer Unit |
| TPU | 16-Bit Timer Pulse Unit |
| TSC | Thermal Sensor Controller |
| VDSP | Vision DSP |
| VIN | Video Input Module |
| VSPD, VSPX | Video Signal Processor |

# Table of Contents

1. Overview

2. Area Map

3. Pin Assignment

4. Pin Multiplex

5. AP-System Core

6. Realtime Core

7. Pin Function (PFC/GPIO)

8. Clock Pulse Generator (CPG)

9. Module Standby, Software Reset

10. Advanced Power Management Unit

11. System Controller (SYSC)

12. Adaptive Voltage Scaling (AVS) [for specific customers only]

13. Thermal Sensor/Chip Internal Voltage Monitor/Core Voltage Monitor

14. Reset (RST)

15. Interrupt Controller (INTC)

16. Multifunctional Interface (MFIS)

17. Error Control Module (ECM)

18. AXI-Bus

19. IPMMU

30. Direct Memory Access Controller

31. Boot

33. External Bus Controller for SDRAM (DBSC5)

34. System RAM

36. RT-VRAM

37. Back-up Buffer (BKBUF)

50. 3D Graphics Processsing Unit (GSX)

53. CSI2

54. Image Signal Processor (ISP)/Channel Selector (CS)

55. Video Input Module (VIN)

56. IMR-LX6

57. Video Encoding Processor for inter-device video transfer (iVCP1E)

58. Image Processing Unit (IMP-X7)

60. Video Signal Processor (VSP2)

62. Frame Compression Processor (FCP)

63. Data Compression Processor (FCPR)

64. Display Unit (DU)

65. Display Output Checker (DOC)

67. DSI/CSI-2 TX

81. Serial Sound Interface Unit (SSIU)

82. Serial Sound Interface (SSI)

83. Audio Clock Generator (ADG)

100. Ethernet (EtherAVB-IF, gPTP, RGMII)

102. CAN-FD

103. FlexRay

104. PCIe Controller

105. Serial Communication Interface with FIFO(SCIF)

106. High Speed Serial Communication Interface with FIFO (HSCIF)

107. I2C Bus Interface (I2C)

109. Clock-Synchronized Serial Interface with FIFO (MSIOF)

110. PWM

112. SPI Multi I/O Bus Controller (RPC-IF)

132. AES Accelerator (AES-ACC)

133. Life Cycle

134. ICUMX

135. CRC

136. Failure Self-Detection Output(RFSO)

150. SD Card/MMC Interface

154. Watchdog Timer (RCLKWDT, WindowWDT, SystemWDT, ...)

155. Timer Unit (TPU, CMT0, CMT1, TMU, SCMT, SUCMT, ...)

168. Boundary Scan

169. HSSTP

170. Debug and Trace

171. Power On Self Test (POST) and Runtime Test

200. Electrical Characteristics

U10.1. E-FUSE

U10.2. OTP_MEM

U11. ROM

U20. Measurement Trace Interface (MTI)

U25. Vision-IP

U30. IMP DMAC

U31. IMP PSC

U32. IMP-X7 CVengine

U33. Convolutional Neural Network (IMP CNN)

U34. IMP-X7_system

U36. Debug Trace Agent (IMP DTA)

U39. Vision DSP (VDSP)

U61. DDR PHY

# 1. Overview

## 1.1 Introduction

The R-Car Series, 4th Generation are SoCs that feature the basic functions for next-generation ADAS(Advanced Driver Assistance System) products.

- Quad 1.0 GHz Arm® Cortex®-A76 cores, 4 cores x 1 cluster
- 3 x 1.4 GHz Arm® Cortex®-R52 core,
  Hardware Lock step is supported.
- Low Power Mode support
- Memory controller for LPDDR4x-4266/LPDDR5-6400
- Image recognition engine (IMP-X7),
- Image signal processor (ISP) x 1 channel,
- Video Output (4 lanes x 1 channel MIPI-DSI), Switchable to CSI2 interface 2.5 Gbps/lane
- Video Input (4 lanes × 2 channels MIPI-CSI2) 2.5 Gbps(D-PHY)/lane
- 1 channel FlexRay (A, B 2 channels) interface
- eMMC
- PCI Express Gen4.0 interface, (Dual lane)
- ICUMX
- AES Accelerator (x 8 channels)
- Video processing units,
- CAN FD interface x 4 channels
- Gbit-EtherAVB x 3 channels.
- Vision-IP
- PAP System

- Use TSMC 7nm process technology
- AVS (Adaptive Voltage Scaling) option supports optimization of power consumption for each SoC
- ISO26262 standard compliant for functional safety
- 19x19 mm2 FCBGA package with 729 pins
- Maximum operating temperature is Tj=125 degreeC (Ta=105 degreeC for AEC-Q100 Grade 2)

Remarks: For items noted as "option", please contact a Renesas Electronics sales representative.

### 1.1.1 R-Car V4M Series Lineup

V4M variants specification difference

| Specification | | R-Car V4M-7 | R-Car V4M-5 | R-Car V4M-3 | R-Car V4M-2 |
|---|---|---|---|---|---|
| Parts Number | | R8A779H2 | R8A779H3 | R8A779H4 | R8A779H5 |
| Number of Cortex-A76<br>Max Operating frequency (PLL2/Z0φ) | | 4 x<br>1.0 GHz | 4 x<br>0.8GHz | 3 x *1<br>0.933 GHz | 3 x *1<br>0.933GHz |
| Number of Cortex-R52<br>Max Operating frequency<br>(PLL6/ZRxφ(x=0-2)) | | 3 x<br>1.4 GHz | 3 x<br>1.1 GHz | 3 x<br>0.9 GHz | 2 x *2<br>0.9GHz |
| Number of VDSP core | | 2 | 2 | 1*3 | 1*3 |
| GPU<br>Max Operating frequency (PLL4/ZGφ) | | 1.0 GHz | 0.75 GHz | 0.5 GHz | Not Support |
| DRAM Max Operating frequency (PLL3/ZB3φ) | LP5 | 800 MHz | 625 MHz | 600 MHz | 600 MHz |
| | LP4 | 1066.66 MHz | 833.33 MHz | 800 MHz | 800 MHz |
| Identification Register<br>Address: OTP_MEM_1 + H'144<br>OTPMONITOR17.ID_Reg[7:0] | | 0x00 | 0x01 | 0x02 | 0x04 |

Notes.

Access to the modules or channels that have been removed from the R-Car V4M-7 product specifications is prohibited. If the user accesses these modules or channels, the correct operation of the LSI is not guaranteed.

*1: V4M-3 and V4M-2 do not support Cortex-A76 CPU core3, and operation is not guaranteed if accessed.

*2: V4M-2 does not support Cortex-R52 core2, and operation is not guaranteed if accessed.

*3: V4M-3 and V4M-2 do not support VDSP1, and operation is not guaranteed if accessed.

## 1.2 List of Specifications

### 1.2.1 Arm-based Application system Core

| Item | Description |
|---|---|
| System CPU | • Quad Core Arm Cortex-A76 (quad core per cluster) 1.0 GHz<br>L1 : I/ 64 KB per core [w/ Parity]<br>D/64 KB per core [w/ ECC]<br>L2 : 256 KB per core [w/ ECC]<br>L3 : 512 KB per cluster [w/ ECC] |

### 1.2.2 Arm-based Realtime Core

| Item | Description |
|---|---|
| Realtime Core | • Arm Cortex-R52 1.4 GHz<br>• L1 I/D cache 32K/32K bytes (ECC)<br>• ATCM/BTCM/CTCM (16K/16K/16K bytes) (ECC)<br>• Three CPU with Dual Lock-Step supported<br>• Armv8-R architecture |

### 1.2.3 Debug and Trace

| Item | Description |
|---|---|
| Debug and Trace | • JTAG interface<br>Supports dedicated 5-pin JTAG, SWD, and LPD*1<br>Additional pin-multiplexed 1-JTAG port.<br>• The cross trigger function between following IPs<br>Cortex-A76 CPU core<br>Cortex-R52 CPU core<br>Vision DSP (CEVA-SP500™)<br>IMP<br>CoreSight<br>ICUMX<br>PAP System<br>• Tracing function<br>10 Gbps or 5 Gbps HSSTP output bandwidth through PCIe*2<br>32 Gbps or 16 Gbps ETR output bandwidth through PCIe*4<br>32 Kbyte embedded trace FIFO for Cortex-A76<br>32 Kbyte embedded trace FIFO for Cortex-R52<br>32 Kbyte embedded trace FIFO for CoreSight<br>16 Kbyte embedded trace FIFO for Vision DSP (CEVA-SP500™)<br>System Trace Macrocell (ISP/MTI, AXI Masters)<br>• Secure debug<br><br>Debugging is disabled if ICUMX and Life Cycle are used in Secure LCS*6 and it can be enabled with secure debugging authentication. |

*1: LPD stands for Low Pin Debug, which is used for ICUMX
*2: Available bandwidth selectable from either 5Gbps per lane or 2.5Gbps per lane
*4: Available bandwidth selectable from either 16Gbps per lane or 8Gbps per lane
*6: ICUMX and Life Cycle define Life Cycle State (LCS). Secure (SE) LCS is one of Life Cycle states.

### 1.2.4 CPU Core Peripherals

| Item | Description |
|---|---|
| Clock Pulse Generator (CPG) | • Generates the clocks from external clock (EXTAL)<br>– Maximum Cortex-A76 clock: 1000 MHz<br>– Maximum Cortex-R52 clock: 1400 MHz<br>– Maximum AXI-bus clock: 400 MHz<br>– Maximum SDRAM bus clock: 800 MHz or 1066 MHz (LPDDR5-6400 Mbps, LPDDR4x-4266 Mbps)<br>– Maximum media clock: 533 MHz<br>– Maximum Vision IP clock: 640 MHz<br>• System-CPU shut down mode control supported<br>• Module-standby mode supported (to see Chapter 9. Module Standby, Software Reset)<br>• Includes module reset registers to control reset operation of individual on-chip peripheral modules |
| System Controller (SYSC) | • Shuts down and restores power to target modules<br>Target modules:<br>– Cortex-A76 (with independent shutting down of each CPUs and SCU+L3 cache) (*1)<br>– 3D Graphics Engine<br>– Vison DSP (CEVA-SP500™)<br>– IMP-X7-SS<br>– ISP<br>(*1): SCU and L3 cache are treated as one power-domain. When CPU is working, SCU+L3 cache can't be powered off.<br>• Low leakage standby mode supported |
| RESET | • Includes one reset-signal external output port for external modules<br>• Includes Boot Address Register etc. |
| Pin function controller (PFC) | • Setting multiplexed pin functions for LSI pins<br>Function of the LSI pin selectable by setting the registers in the PFC module<br>• Module selection<br>Enable and disable the functions of LSI pins to which pin functions from multiple pin groups are assigned by setting the registers in the PFC module.<br>• Pull-up/down control for each LSI pin<br>On/off and up/down of the pull register on each LSI pin can be controlled by setting the registers in the PFC module. |
| General-purpose I/O (GPIO) | • General-purpose I/O ports<br>• Supports GPIO interrupts |
| Thermal sensor / Core Voltage Monitor (THS/CVM) | • 2 channels of thermal sensor<br>• Programmable 3 temperature level for the sensor, to indicate the temperature level<br>• Interrupt to CPU/ECM when the temperature reaches programmed<br>• This module detects the lower or upper value of VDD’s voltage and notifies the state thereof<br>• Includes voltage monitoring module that measures the voltage supply inside the LSI |

### 1.2.5 External Bus Module

| Item | Description |
|---|---|
| External Flash Controller | • Supports Hyper Flash memory or QSPI flash memory<br>Maximum Frequency 160 MHz (320 MB/s) for Hyper Flash,<br>Maximum Frequency 133 MHz (133 MB/s) for QSPI (QSPI0)<br>• Dual QSPI operation for two 4-bit serial flash memories is also available; 80 MHz (160 MB/s) for Dual QSPI (QSPI0+QSPI1). |
| External Bus Controller for LPDDR5/4x (DBSC5) | • 2x 16-bit bus DRAM interfaces<br>• 2 x channels 16-bit bus per DRAM interface<br>• LPDDR4x-4266/LPDDR5-6400 can be connected directly<br>• Memory Size: Up to 8 GB(*)<br>• Auto Refresh/Self Refresh/Partial Array Self Refresh supported<br>• Auto Pre-charge Mode<br>• DDR Back Up supported<br>Cache memory for DDR-Memory access efficiency<br>(*) LPDDR4X-SDRAM compliant with JEDEC JESD209-4C and JEDEC JESD209-4-1A.<br>(One LPDDR4X-SDRAM can be connected to one channel. two 16bit channels within a pair must have same sizes. Support memory with sizes from 2 Gbits to 16 Gbits per 16 bits channel per rank.)<br>LPDDR5-SDRAM compliant with JEDEC JESD209-5B<br>(One LPDDR5-SDRAM can be connected to one channel. two 16bit channels within a pair must have same sizes. Support memory with sizes from 2 Gbits to 32 Gbits per 16 bit channel per rank.)<br>2 ranks/channel support. (Two ranks must have same sizes.) |

### 1.2.6 Internal Bus Module

| Item | Description |
|---|---|
| AXI-bus | • On-chip main bus<br>– Bus protocol: AXI3 with QoS control<br>– Frequency: 400 MHz<br>– Bus width: 512 bits/256 bits/128 bits |
| Direct Memory Access Controller for System (SYS-DMAC) | • 24 channels for PER0 domain (SYDM) with Dual core lock step<br>• Address space: 1 TBytes on architecture<br>• Data transfer length: Byte, word (2 Bytes), longword (4 Bytes), 8 Bytes, 16 Bytes, 32 Bytes and 64 Bytes<br>• Maximum number of transfer times: 16,777,215 times<br>• Transfer request:<br>Selectable from on-chip peripheral module request and auto request<br>• Bus mode:<br>Selectable from normal mode and slow mode<br>• Priority: Selectable from fixed channel priority mode and round-robin mode<br>• Interrupt request: Supports interrupt request to CPU at the end of data transfer<br>• Repeat function: Automatically resets the transfer source, destination, and count at the end of DMA transfer (by descriptor function)<br>• Descriptor function (each channel) supported<br>• MMU (each channel) supported<br>• Channel bandwidth arbiter (each channel) |
| Boot | • System startup with selectable boot mode at power-on reset<br>• In on-chip ROM boot, Hyper Flash or QSPI serial ROM boot is supported<br>• Program downloaded to internal memory<br>Autorun function for the downloaded program<br>• Secure boot supported. Integrity check of boot image is proceeded before executing<br>• On secure chip, boot operation is restricted to on-chip ROM boot mode<br>• About the detail information of BOOT, refer to Chapter 31. Boot |
| Realtime Direct Memory Access Controller (RT-DMAC) | • 32 channels for Realtime domain with Dual core lock step<br>• Address space: 1 TBytes on architecture<br>• Data transfer length: Byte, word (2 Bytes), longword (4 Bytes), 8 Bytes, 16 Bytes, 32 Bytes and 64 Bytes<br>• Maximum number of transfer times: 16,777,215 times<br>• Transfer request:<br>Selectable from on-chip peripheral module request and auto request<br>• Bus mode:<br>Selectable from normal mode and slow mode<br>• Priority: Selectable from fixed channel priority mode and round-robin mode<br>• Interrupt request: Supports interrupt request to CPU at the end of data transfer<br>• Repeat function: Automatically resets the transfer source, destination, and count at the end of DMA transfer (by descriptor function)<br>• Descriptor function (each channel) supported<br>• MMU (each channel) supported<br>• Channel bandwidth arbiter (each channel) |
| IPMMU | • An IPMMU is a memory management unit (MMU) which provides address translation and access protection functionalities to processing units and interconnect networks |

| Item | Description |
|---|---|
| Interrupt Controller (INTC) | • 6 interrupt pins which can detect external interrupts<br>• Fall/rise/high level/low level detection is selectable<br>• On-chip peripheral interrupts: Priority can be specified for each module<br>• 16 software interrupts that have been generated and 6 private peripheral interrupts supported<br>• 32-level priority selectable<br>• Trust Zone supported<br>• ITS (Interrupt Translation Service) support (for Message Signal Interrupt from PCI-Express) |
| Multifunctional Interface (MFIS) | • Interrupt generation between SYS domain and RT domain<br>• Fifteen external source bits for controlling 32 Ktypes of interrupts<br>• Lock function for exclusive access support |
| Error Control Module (ECM) | • Error checking function control and arbitrate error signal from other module |

### 1.2.7 Internal Memory

| Item | Description |
|---|---|
| RT-VRAM | 2 Mbytes, each 1 Mbytes in compatible mode and 28 Mbytes cache in extended mode |
| System RAM | 1 Mbytes |

### 1.2.8 Graphics Units

| Item | Description | |
|---|---|---|
| 3D Graphics Engine (GSX) | • Imagination Technologies PowerVR BXS-4-64 GPU<br>• 64 FLOPs/clk (FP32)<br>• System Level Cache provides cashing of all types of workload data. 64 Kbytes capacity<br>• OpenGL ES 3.2 supported | |
| Display Unit (DU) | MIPI-DSI interface | The following is Key features of the DSI/CSI-2-TX Transmitter module.<br>Supports 1 channel (4 lane x 1 channel) interface<br>• Supports MIPI DSI version 1.3.1<br>Supports MIPI D-PHY 2.5 Gbps with de-skew function (for 2.5 G)<br>• Supports MIPI CSI2 version 1<br>Supports MIPI D-PHY 2.5 Gbps with de-skew function |
| | DSI | • Supports D-PHY specification version 1.2<br>• Protocol Interface<br>Video mode and Packet Generation from Input Video signal<br>Supports RGB format (16 bpp, 18 bpp, 18 bpp Loosely and 24 bpp)<br>Supports YCbCr 4:2:2 format (16 bpp)<br>Supports Blanking Packet or LP-11 selection during each of HSA, HBP and HFP Blanking interval<br>Not supports Burst transfer mode<br>Not supports Interlaced Video<br>Not supports 3D control payload<br>Supports Command Packet Tx/Rx (maximum one packet per frame on blanking period)<br>• Supports Command mode which supports Command Packet Tx/Rx by Register<br>• Supports automated insertion of EoTp |
| | CSI2 | • Supports D-PHY specification version 1.2<br>• Protocol Interface<br>Video mode and Packet Generation from Input Video signal through DU (Display Unit ).<br>Supports RGB565/666/888<br>Supports YUV422 8 bit<br>Not support Interlaced Video<br>Not support Interleaving<br>Not support Line Start Packet and Line End Packet<br>Supports Tx Command Packet (maximum one packet per frame on working Video Interface)<br>• Supports Command Transfer mode which supports Tx/Rx Command Packet by Register |
| Display Output Checker (DOC)<br>(safety function option) | • The DOC allows to check whether the display content (warning, light, and so on) is correctly output by the Display Unit (DU) | |

| | |
|---|---|
| Video Input Module(VIN)<br>(MIPI-CSI2 interface) | • 2 channels (4 lane x 2 channels)<br>• 2.5 Gbps/Lane(D-PHY)<br>RAW data (ISP)<br>RAW8, RAW10*, RAW12*, RAW14* (HDR sensor Input), RAW16* (HDR sensor Input), RAW20* (HDR sensor Input), RAW24* (HDR sensor Input) are supported<br>*: RAW10, RAW12, RAW14, RAW16, and RAW20, RAW24 can be input only via ISP module.<br>YUV420 8/10bit, YUV422 8/10bit, RGB565, RGB888, Embedded 8bit, User Defined 8bit are supported |

### 1.2.9 Video Processing

| Item | Description |
|---|---|
| Video Signal Processor (VSPD) | VSPD has the following features. 1 set of VSPD is integrated.<br>(1) Supports various data formats and conversion<br>• Supports YCbCr444/422/420, RGB, αRGB, αplane<br>• Color space conversion and changes to the number of colors by dithering<br>• Color keying<br>• Supports combination between pixel alpha and global alpha<br>• Supports generating pre multiplied alpha<br>(2) Video processing<br>• Blending of 5 picture layers and raster operations (ROPs)<br>• Vertical flipping in case of output to memory<br>(3) Direct connection to display module<br>• Supports 6,144 pixels in horizontal direction<br>• Writing back image data which is transferred to Display Unit (DU) to memory<br>(4) Supports DISCOM function<br>• Support 2 sets of DISCOM per 1 VSPD<br>• Support same size with VSPD<br>• CRC calculation area is configurable<br>• Support detection of unmatching the expectation CRC value<br>• Support detection of matching image data over multiple frames for detecting image frozen<br>• Support both RGB and YCbCr color space |
| Video Encoding Processor for inter-device video transfer (iVCP1E) | • Support H.264/AVC with 8/10/12-bit depth<br>• Low-latency encoding with precise CBR (Constant Bit Rate) control<br>• Support 1,920 x 1,080 resolution<br>• Data handling on a picture-by-picture basis<br>Use the software from Renesas to handle iVCP1E functions. |
| IMR-LX6 | • 4 Image Renders for camera image distortion correction application<br>• Pixel Performance: 533 Mpixel/sec (peak@533MHz) per one IMR-LX6<br>• Rotator/Scaler function support (2 channels)<br>– YUV422 semi-planar input mode<br>– 3 x 3 Gaussian Filter on the Scaler function<br>Stride setting should be extended to support 4K width input |
| ISP<br>(Image Signal Processor) | • Defect Pixel Correction<br>• Noise Filter<br>• Local Tone Mapping<br>• Histogram Generation (4 colors x 256 bin)<br>• Demosaicing<br>• Color Correction<br>• Dual Video Output (Human Vision and Machine Vision) |

### 1.2.10 Sound Interface

| Item | Description |
|---|---|
| Serial Sound Interface (SSI) | • TDM format is supported.<br>• Monaural mode (8 bits or 16 bits) is supported<br>• Operating mode: non-compressed mode (Not support compressed mode)<br>• Supports versatile serial audio formats (I2S/left justified/right justified)<br>• Supports master/slave functions<br>• Programmable word clock, bit clock generation functions. |
| Audio Clock Generator (ADG) | • 1 channel<br>• Supplying Audio Clock to SSI (Serial Sound Interface) |

### 1.2.11 Storage

| Item | Description |
|---|---|
| Multi Media Card Interface (MMC) | • 1 channel<br>• eMMC 5.01 base, support HS400 class transfer rate |

### 1.2.12 Network

| Item | Description |
|---|---|
| CAN-FD | • 4 interfaces CANFD instance<br>• Data phase max 8 Mbps |
| PCIE Controller | • PCI Express Base Specification Revision 4.0<br>• PHY integrated<br>• Operates as a PCI Express Endpoint or a PCI Express Root Port<br>• 2 lanes x 1 channel |
| Ethernet AVB-IF | • Supports IEEE802.1BA, IEEE802.1AS, IEEE802.1Qav and IEEE1722 functions<br>• Supports transfer at 1000 Mbps and 100 Mbps<br>• Magic packet detection<br>• Supports Reception Filtering to separate streaming frames from different sources<br>• Supports interface conforming to IEEE802.3 PHYRGMII (Reduced Gigabit Media Independent Interface)<br>• RGMII v1.3<br>• 3 channels |
| FlexRay | • 2 channels (A, B interfaces)<br>• Comply with FlexRay protocol specification v2.1<br>• Data transfer rate: up to 10 Mbps on each channel |

### 1.2.13 Timer

| Item | Description |
|---|---|
| RCLK Watchdog Timer | • 1 channel<br>• Internal 16 bits watchdog timer operated by RCLK<br>• Programmable overflow time period: more than 1 hour count capable |
| 16-Bit Timer Pulse Unit (TPU) | • 4 channels 16 bits timers<br>• Each channel outputs PWM |
| System Watchdog Timer | • 1 channel<br>• Internal 16 bits watchdog timer<br>• Programmable overflow time period: more than 1 hour count capable<br>initial counter value 171[s] |
| Window WDT | • 7 channels (4 channels for Arm Cortex-A76 CPU cores, 3 channels for Arm Cortex-R52 cores)<br>• Window watchdog function; Window & 75% interrupt<br>• Generates Reset or NMI on error detection<br>• 3 start mode selectable (Automatic, Intelligent automatic, software trigger)<br>• Configuration can be done by configuration terminals (flash/mask option or clamp) information |
| Compare Match Timer Type0 (CMT0) | • 2 channels<br>• 32 bits timer (16 bits/32 bits can be selected)<br>• Source clock: RCLK clock<br>• Compare match function provided<br>• Interrupt requests |
| Compare Match Timer Type1/2/3 (CMT1/2/3) | • 8 channels<br>• 48 bits timer (16 bits/32 bits/48 bits can be selected)<br>• Source clock: RCLK/system clock<br>• Compare match function provided<br>• Interrupt requests |
| System Timer | • 32 bits timer, 1 channel (16 bits/32 bits is selectable)<br>• Compare match function provided<br>• Interrupt requests |
| System up-time clock | • 1 channel<br>• Internal 32 bits timer<br>• Programmable overflow time period: maximum 24 hours |
| Timer Unit (TMU) | • 15 channels<br>• 32 bits timer<br>• Auto-reload type 32 bits down counter<br>• Internal prescaler<br>• Interrupt request<br>• 4 channels for input capture |

### 1.2.14 Peripheral Module

| Item | Description |
|---|---|
| ICUMX | • The ICUMX integrates an Intelligent Cryptographic Unit Processor (ICUP) that controls the ICUMX sub-system and runs the user-defined security services<br>• The ICUMX includes an accelerator that supports a block cipher algorithm based on the AES (Advanced Encryption Standard, FIPS PUB 197) and supports DMA transfer for AES block data<br>• The ICUMX integrates several public key cryptographic algorithms like RSA, ECC<br>• The ICUMX integrates an accelerator that supports HASH function based on SHA-1, SHA-224, SHA-256 (Secure Hash Algorithm, FIPS-PUB 180) and SHA-3 (Secure Hash Algorithm, FIPS PUB 202)<br>• The ICUMX integrates a true random number generator<br>• The ICUMX can be used as a slave peripheral to run security services upon request. The ICUMX handles processing of other user-defined security services (for example, checking contents of memory for completeness and integrity) and operates as a master handling procedures to strengthen measures for security<br>• The ICUMX has exclusive read access and programming access to a specific area of memory which holds confidential data (keys and certificates) and highly confidential code for operating security services<br>• The ICUMX zeroizes key materials in FA mode |
| AES accelerator | Accelerating AES encryption/decryption operation for high speed communication<br>• Support 128 bits and 256 bits key length<br>• Support operation mode, ECB, CBC, CMAC, XTS, CTR and GCM<br>Key management function protects encryption keys from being exposed on memory or falsified by other functions |
| I2C Bus Interface (I2C) | • 4 channels for open drain buffer<br>• NXP I2C bus interface method supported<br>• Master/slave functions<br>• Multi-master functions<br>• Fast mode plus compatible<br>• Programmable clock generation from the system clock<br>• Master and Slave function DMA supported |

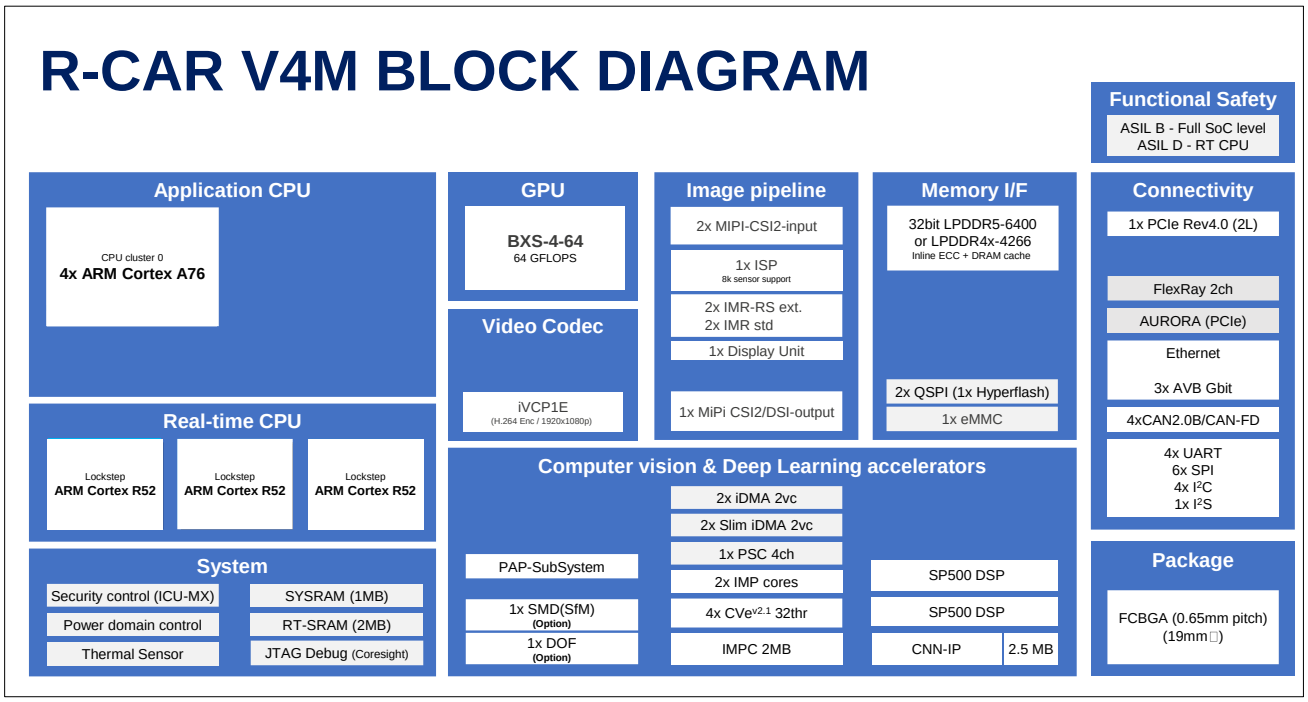
| Item | Description |
|---|---|
| Serial communication interface with FIFO (SCIF) | • 4 channels<br>• Asynchronous, clock-synchronized modes<br>• Asynchronous serial communication mode<br>The SCIF performs serial data communication based on a character-by-character asynchronous system. This feature enables serial data communication with standard asynchronous communication chips that support Universal Asynchronous Receiver/Transmitter (UART) or Asynchronous Communication Interface Adapter (ACIA). There is a choice of eight serial data transfer formats.<br>— Data length: 7 bits or 8 bits<br>— Stop bits: 1 bit or 2 bits<br>— Parity: Even/odd/none<br>— Receive error detection: Parity, framing, and overrun errors<br>— Break detection:<br>A break is detected when a framing error lasts for more than 1 frame length at Space 0 (low level).<br>When a framing error occurs, a break can also be detected by reading the RX pin level directly from the serial port register (SCSPTR).<br>• Clock synchronous serial communication mode<br>The SCIF performs serial data communication synchronized with a clock. This feature enables serial data communication with other LSIs that support synchronous communication. There is a single serial data communication format for clock synchronous serial communication.<br>— Data length: 8 bits<br>— Receive error detection: Overrun errors<br>• Full-duplex communication capability<br>The SCIF has an independent transmitter and receiver that enable simultaneous transmission and reception. The transmitter and receiver both have a 16-stage FIFO buffer structure, enabling continuous serial data transmission and reception.<br>• On-chip baud rate generator, enabling any bit rate to be selected<br>The SCIF enables choice of a clock source for transmission/reception: a clock from the on-chip baud rate generator based on the internal clock or an external clock.<br>• 8 interrupt sources<br>• The SCIF has eight types of interrupt sources. receive-data-ready, receive-FIFO-data-full, break, transmit-FIFO-data-empty, transmit-end, receive-error, overrun-error and time-out and enables any of them to be requested independently. |
| Serial Communication Interface with FIFO (SCIF) | • DMA data transfer<br>When the transmit FIFO register is empty or the receive FIFO register has received data, issuing a DMA transfer request activates the DMA controller (DMAC) to execute a data transfer.<br>• The amount of data in the transmit/receive FIFO registers and the number of receive errors in receive data in the receive FIFO register are available<br>• In asynchronous mode, a receive data ready (DR) or a timeout error (TO) can be detected during reception |

| Item | Description |
|---|---|
| Clock-Synchronized Serial Interface with FIFO (MSIOF) | • 6 channels<br>• Max. speed: 40 MHz for Master Reception or Master Transmission<br>• Max. speed: 60 MHz for Slave Reception<br>• Max. speed: 25 MHz for Slave Transmission<br>• Internal 32 bits x 256-stage transmit FIFOs/internal 32 bits x 256-stage receive FIFOs<br>• Supports master and slave modes<br>• Internal prescaler<br>• Supports serial formats: IIS, SPI (master and slave modes)<br>• Interrupt request, DMAC request |
| High Speed Serial Communication Interface with FIFO (HSCIF) | • 4 channels<br>• Asynchronous serial communication mode<br>• Capable of full-duplex communication<br>• On-chip baud rate generator, enabling any bit rate to be selected<br>• Eight interrupt sources<br>• DMA data transfer<br>• Modem control functions (HRTS# and HCTS#) are stored<br>• The amount of data in the transmit/receive FIFO registers and the number of receive errors in receive data in the receive FIFO register are available<br>• A receive data ready (DR) or a timeout error (TO) can be detected during reception |
| PWM | • 5 channels<br>• High-level width (10 bits) of PWM output can be set.<br>• Output cycle periods (10 bits) of PWM can be set.<br>• Periods in the range from two to $2^{24}$ x 1024 cycles of the Pϕ clock can be set.<br>• Continuous pulse or single pulse output selectable |

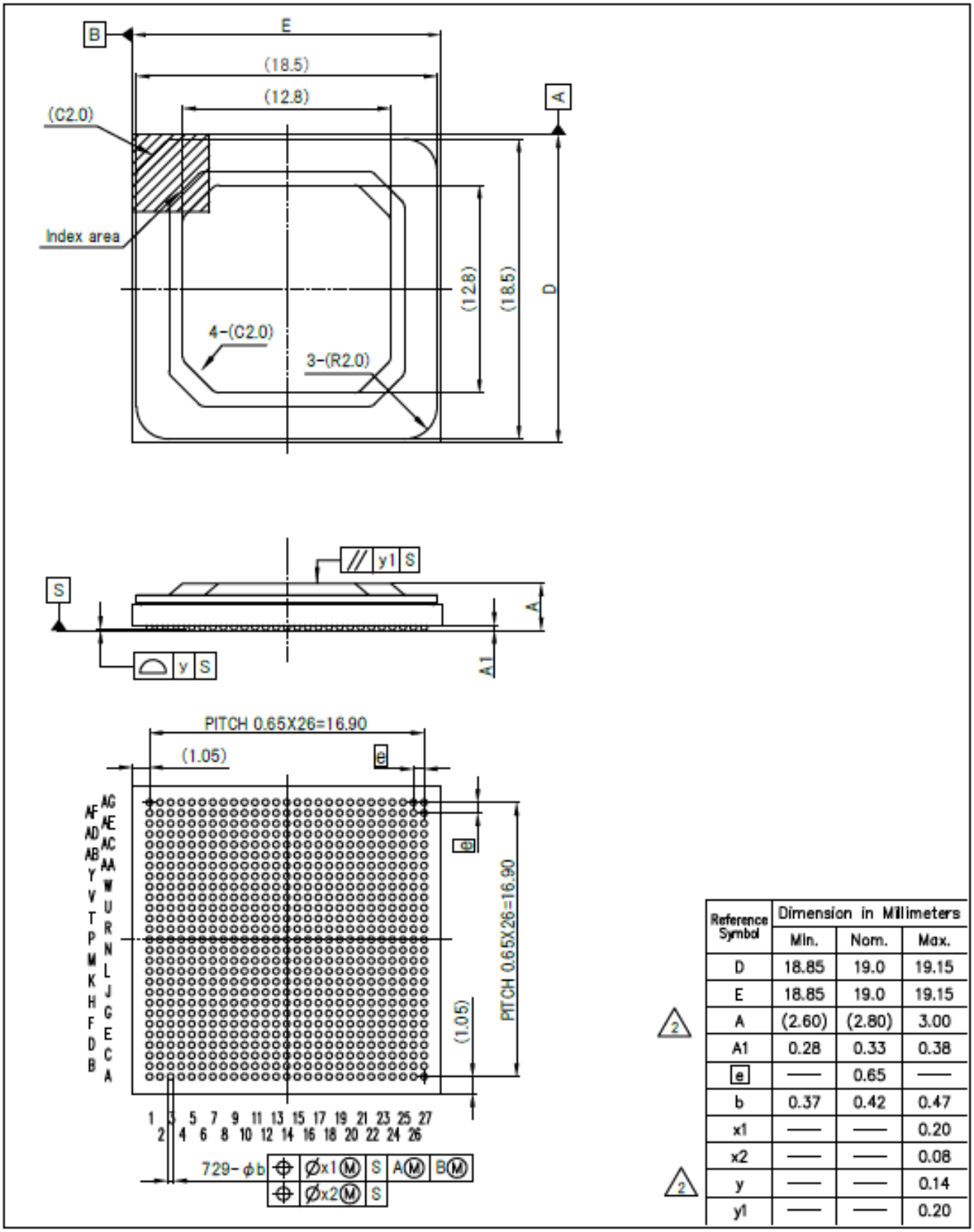
### 1.2.15 Special Edition Modules

| Item | Description |
|---|---|
| Image Processing Unit (IMP-X7) | • 2 Image Processor Cores<br>• 4 CVe Cores<br>• 2 Mbyte local memory<br>• 2 DMAC for transferring image data from DDR to local memory in IMP-X7<br>• 1 IMP PSC for image scaling engine<br>• 1 CNN engine (included 2 DSP(CEVA-SP500™). There are (2MB + 0.5MB= 2.5MB) memory per a CNN.) |
| Vision IP | • Structure from Motion engine<br>• Dense Optical Flow engine |
| PAPSystem | • Custom logic for Vision computing<br>• PAP Sub-System consist of PAP, Host-REG, PAP-DMAC and PAP-BUSs |
| CRC | The cyclic redundancy check (CRC) block is a function block to generate and check CRC codes of input data via the APB bus. The 32 bits Ethernet polynomial (CRC-32-IEEE 802.3) is used as the generator polynomial. The CRC codes based on this generator polynomial are stored in the register. |
| Failure Self-Detection Output (RFSO) | This module distinguishes between permanent failure and temporary failure detected by CPU or other important circuit module; moreover, detects time-out error in the process of detecting failures. The failure of CPU or other important circuit will be detected by other circuit (Runtime Test). Also, this module will be used for managing FTTI of this LSI. |

## 1.3 Block Diagram



## 1.4 Package outline




| Reference Symbol | Dimension in Millimeters | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| D | 18.85 | 19.0 | 19.15 |
| E | 18.85 | 19.0 | 19.15 |
| A | (2.60) | (2.80) | 3.00 |
| A1 | 0.28 | 0.33 | 0.38 |
| e | — | 0.65 | — |
| b | 0.37 | 0.42 | 0.47 |
| x1 | — | — | 0.20 |
| x2 | — | — | 0.08 |
| y | — | — | 0.14 |
| y1 | — | — | 0.20 |

## 1.5 Power Control

### 1.5.1 Power domain structure

R-Car V4M SoC consists from three kinds of power domains, which are Always ON, C4 and PSO (Power Shut OFF).
Table 11.1 (Power Domain Structure) shows the detail definition of each power domain.
PSO corresponds with A3*, A2*. A1* as shown in Table 11.1.
Table 11.1.2.1 (Function located on Always ON power domain), Table 11.1.2.2 (Function located on C4 power domain) and Table 11.1.2.3 (Function located on PSO) show the relationship between functions and its power domain.

**Always ON**
- Always Power ON.

**C4**
- Power ON when the chip operation mode is Full Run Mode.
- Power ON or OFF when the chip operation mode is Low Power Mode.

**PSO (Power Shut OFF)**
- Power ON or OFF when the chip operation mode is Full Run Mode.
- Power OFF when the chip operation mode is Low Power Mode.

### 1.5.2 Chip mode transition

Figure 1.5.2.1 and Figure 1.5.2.2 show the chip mode transition specification on selecting ICUMX or Cortex-R52 as a master boot processor, respectively.

As shown in Figure 1.5.2.1 and Figure 1.5.2.2. "C5" means "Always ON".

**Low Power Mode:**
All modules in PSO domain are power-off condition by register settings. Each module in C4 domain can be power-off condition by register settings.
In Low Power Mode, ICUMX and Cortex-R52 core0 can control only modules in C5 domain.
Chapter 11. System Controller (SYSC) shows how to turn on C4, which leads chip mode transition from Low Power Mode to Full Run Mode.

**Full Run Mode:**
All modules in not only C5 but also C4 domain are power-on condition by register settings.
Setting registers described in Chapter 11. System Controller (SYSC) turns on corresponding power domains, which can activate all CPUs and modules in PSO domain.

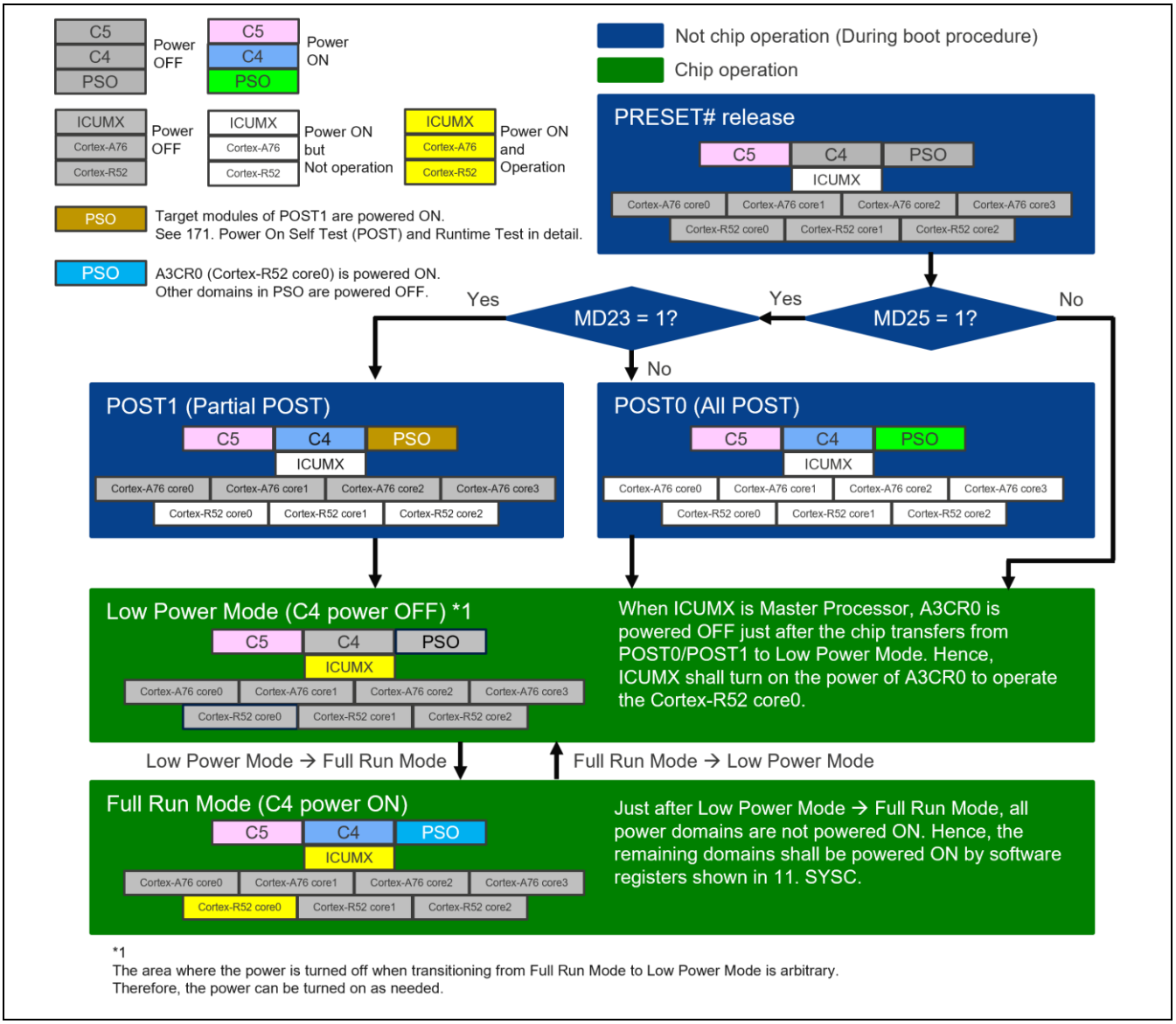



**Figure 1.5.2.1 R-CarV4M chip mode transition (Master Processor = ICUMX)**

**Notes:**
"C5" is equal to "Always ON".

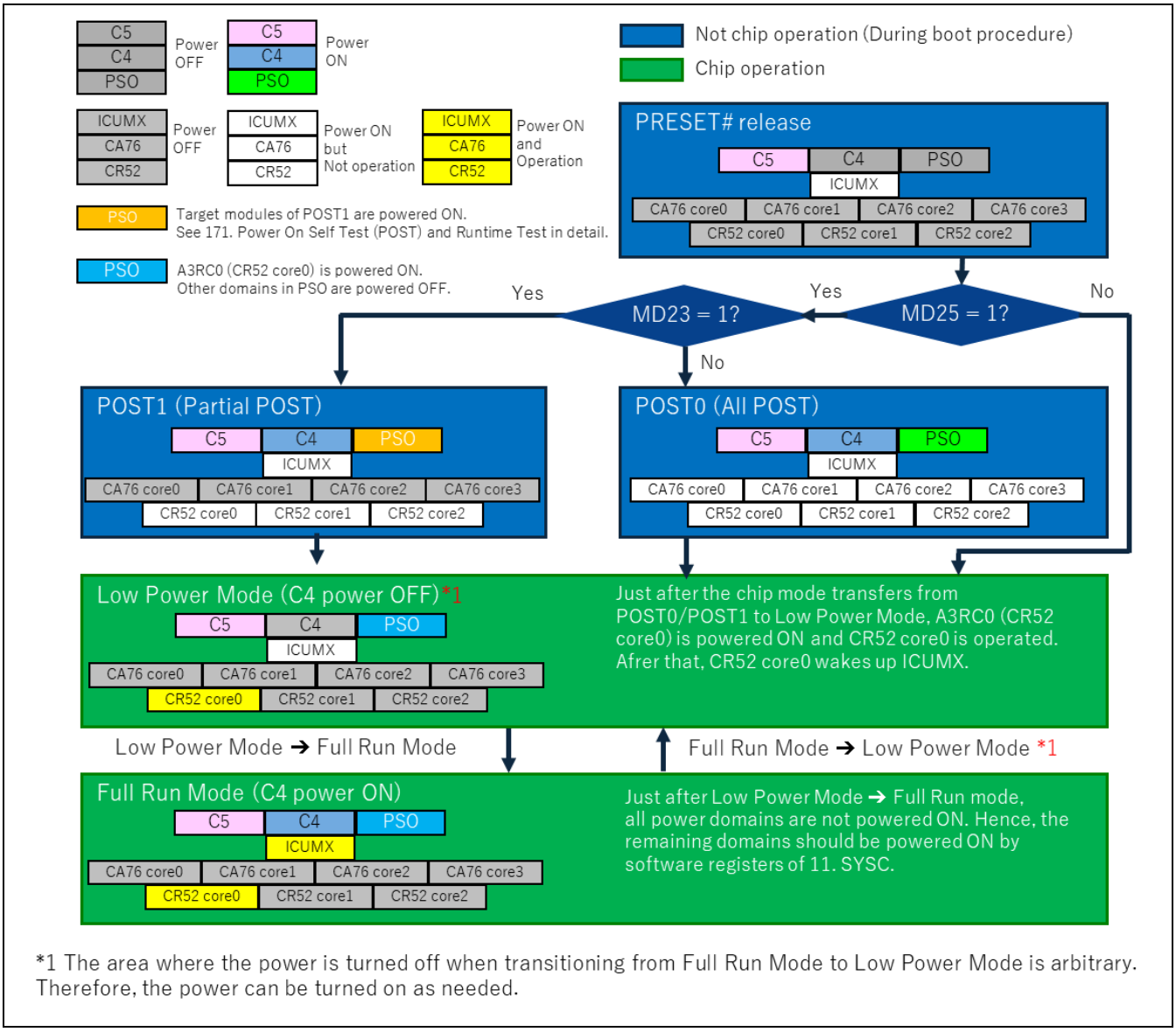



**Figure 1.5.2.2 R-CarV4M chip mode transition (Master Processor = Cortex-R52 core 0)**

**Notes:**
"C5" is equal to "Always ON".

**POST0:**
When the chip boot procedure is started from Cold startup, all modules shall be tested by POST (Power on Self Test) according to Functional Safety instruction. See Chapter 171. Power On Self Test (POST) and Runtime Test about the detail description related to POST0.

**POST1:**
When the chip boot procedure is re-executed by POR (Power On Reset shown in Table 14.1.1), it is not mandatory to execute the POST0 for all modules. The target of POST1 is modules which are categorized to ASIL-D. When the POST0 is re-executed for all modules again, there are high possibilities that the chip power consumption is increased. The POST1 is effective for solving it. See Chapter 171. Power On Self Test (POST) and Runtime Test about the detail description related to POST1.

**Full Run Mode → Low Power Mode**

- When the chip power consumption increases so largely in Full Run Mode, the power consumption can be reduced by the mode transition from Full Run Mode to Low Power Mode. The mode transition can be executed by turning off C4 and PSO domain. See Chapter 11. System Controller (SYSC) about a usage of turning off their domains. In addition, see Section 1.5.4 before executing the mode transition.

- Just after mode transition from Full Run Mode to Low Power Mode, any clocks which are provided for all modules in C4 and PSO domain are still provided. The clocks can be stopped for the further power consumption reduction (See Section 1.5.3.).

- The area where the power is turned off when transitioning from Full Run Mode to Low Power Mode is arbitrary. Therefore, the power can be turned on as needed.
  As an example of use, when controlling the power supply with Cortex-R52, it is necessary to turn on "1 Cortex-R52 core" at least.
-

**Low Power Mode → Full Run Mode**

- When it is judged the chip power consumption is reduced as expected, the chip mode can transition from Low Power Mode to Full Run Mode by turning on C4 domain. See Chapter 11. System Controller (SYSC) about a usage of turning on C4 domain. In addition, see Section 1.5.4. before executing the mode transition.

- If any clocks which are provided for all modules in C4 and PSO domain are stopped, the clocks shall be supplied with the corresponding modules by software registers in Chapter 8. Clock Pulse Generator (CPG). See Section 1.5.3. in detail.

### 1.5.3 Stopping unused clocks in Low Power Mode

Following clocks are allowed to be stopped in Low Power Mode, because any modules which are summarized in Table 11.1.2.2 (Function located on C4 power domain) and Table 11.1.2.3 (Function located on PSO) are powered OFF.

**Table 1.5.3.1 Clocks which are allowed to be stopped in Low Power Mode**

| Clock | Clock divider shown in Figure 8.1.1 | Clock control register shown in 8.2 (*1) |
|---|---|---|
| ZGφ | 3DGE divider | FRQCRB |
| ZDφ | DSU divider | FRQCRB |
| ZR1φ | RT-CPU divider1 | FRQCRC0 |
| ZC0φ | SYS-CPU divider0 | FRQCRC0 |
| ZC1φ | SYS-CPU divider1 | FRQCRC0 |
| ZR2φ | RT-CPU divider2 | FRQCRC1 |
| ZC2φ | SYS-CPU divider2 | FRQCRC1 |
| ZC3φ | SYS-CPU divider3 | FRQCRC1 |
| ZXφ | AXCC divider | CPGMA0CKSTPCR |
| S0_VIOφ | Common divider | CPGMA0CKSTPCR |
| S0_VCφ | Common divider | CPGMA0CKSTPCR |
| SV_VIPφ | Common divider | CPGMA0CKSTPCR |
| SV_IRφ | Common divider | CPGMA0CKSTPCR |
| IMPASRCφ | IMPASRC divider | CPGMA0CKSTPCR |
| IMPBSRCφ | IMPBSRC divider | CPGMA0CKSTPCR |
| VIOSRCφ | VIOSRC divider | CPGMA0CKSTPCR |
| VCSRCφ | VCSRC divider | CPGMA0CKSTPCR |
| MSOφ | MSIOF divider | MSOCKCR |
| CSIφ | CSI divider | CSICKCR |
| DSIEXTφ | DSIEXT divider | DSIEXTCKCR |
| ADGHφ | ADGH divider | ADGHCKCR |

### 1.5.4 Usage Notes

Before the mode transition b/w Full Run Mode → Low Power Mode, all functions operation which are located on C4 and PSO power domain shall be stopped to avoid the system deadlock. See each chapter about the usage of stopping the functions.

## 1.6 Sequence of Activation for the R-Car V4M

The following describes the initialization sequence of R-Car V4M.

**(1) Selection of Master Boot Processor and External Boot Device**

Select the master boot processor by placing the required levels on the MD6 and MD7 pins and external boot device by placing the required levels on the MD1, MD2, MD3 and MD4 pins before booting up the LSI. For details, see Chapter 31. Boot.

Note: The following steps from (2) to (6) are not included in the sequence described in Chapter 31. Boot because these steps are not relevant to booting. Note, however, those steps (2) to (6) shall be executed before restoring the power supply to modules, including CPUs other than the master boot processor, to which the power has been cut off.

**(2) Stopping the RWDT and System WDT**

To prevent the watchdog timer issuing an unexpected reset or interrupt while the activation sequence is in progress, be sure to disable the RCLK watchdog timer (RDWT) and System watchdog timer (System WDT). Enable their operation at an appropriate time after the activation sequence is completed. For details, see Chapter 14. Reset (RST), Chapter 15. Interrupt Controller (INTC), Section 154.1. RCLK Watchdog Timer, and Section 154.3. SystemWatchDog Timer.

Notes: Use the RWTCSRA register to disable the RWDT. Since the detailed specifications of the System WDT are not disclosed in this manual, contact us to inquire about the details or use the sample driver included in the board support package (BSP).

Currently, the reset and interrupt signals caused by the watchdog timer overflowing are initially masked in R-Car V4M. Accordingly, this step of disabling the watchdog timers within the activation sequence is not mandatory.

**(3) Selecting the Functions of Multiplexed Pins**

When a pin function other than that selected by the initial setting is to be used, set the relevant registers in the pin function controller (PFC). Note that, before setting these registers, the LSI multiplexed pin setting register (PMMR) shall be appropriately set to prevent incorrect writing to registers. For details, see Chapter 7. Pin Function Controller (PFC).

Note: When a GPIO function which is different from that selected by the initial setting is to be used, it is necessary to set the GPIO attributes before setting the pin function controller.

**(4) Initial Settings for Operating Frequency Required by the Main Processor, Cache, MMU, and Interrupt Controller**

Make initial settings for operating frequency required by the master processor, cache, MMU, and interrupt control. The operating frequencies are set by using the registers FRQCRC0, FRQCRC1, PLL2CR0, PLL2CR1, PLL6CR0, and PLL6CR1 (for Cortex-A76 and Cortex-R52). The details of the functional specifications for the CPU cores, cache, MMU, and interrupt control are not included in this manual. Please refer to the following references.

References: ARM® Architecture Reference Manual ARMv8, for ARMv8-A architecture profile

Arm® Cortex-R52 Processor Technical Reference Manual Revision: r1p4

Arm® Cortex®-A76 Core Technical Reference Manual, Revision: r4p1

Arm® DynamIQ™ Shared Unit Technical Reference Manual, Revision: r4p1

GICv3 and GICv4 Software Overview

**(5) Initial Settings for Memory Controllers**

When a program is to be stored in and runs from SDRAM, make initial settings for the required registers in the DBSC. For activation sequence, see Chapter 33. External Bus Controller for SDRAM (DBSC5).

Note: Each product has different initial setting. Renesas is preparing the document of each product.

**(6) QoS Setting**

**(7) Settings for Restoring Power Supply to Functional Modules to Be Used Immediately After Activation**

In the initial state, power may not be supplied to some modules to be used. Accordingly, for modules to be used immediately after activation, set the control registers in the system controller (SYSC) as required and read the status registers to check that the power supply has been restored. For details on the sequence for restoring the power supply, see Chapter 11. System Controller (SYSC).

**(8) Settings for Release from the Module Reset State and the Supply of Clock Signals to Functional Modules to Be Used Immediately After Activation (Other Than CPUs)**

Depends on module, in the initial state, clock signals may not be being supplied to all modules to be used. Accordingly, for modules to be used immediately after activation, set the Clock Control registers (Section 8.2. Register Description in Chapter 8. Clock Pulse Generator (CPG)) for the respective modules in the clock pulse generator (CPG) or set the MSTPCRn registers (n = 0 to 29) in the module standby and software reset control block of the CPG. For details, see Chapter 8. Clock Pulse Generator (CPG) and Chapter 9. Module Standby and Software Reset.

All registers above can be protected from unexpected write operation. See Section 8.4.1. and Section 8.4.2. for the registers in Chapter 8. Clock Pulse Generator (CPG). In addition, see Section 9.4.1. and Section 9.4.2. for the registers in Chapter 9. Module Standby, Software Reset.

**(9) Setting of Boot Address of the CPU (as Required)**

To switch the boot address to an area in the DDR memory to which high-speed access is possible, start by downloading the required program to the specified area and set the starting address for execution in the boot address register (CR52BAR, CR52BARP, RVBARLCz, RVBARHCz, RVBARPLCz or RVBARPHCz) for the relevant CPU. This setting is not required if the boot address is not to be changed. For details, refer to Chapter 10. Advanced Power Management Unit.

**(10) Restoring the Power Supply to A CPU and Releasing CPUs from the Reset State**

Before using Cortex-A76 CPU cores, start by checking the PRR register described in Chapter 16. Multifunctional Interface (MFIS) to confirm which CPU is available for use. If activation of a CPU disabled by the PRR register is attempted, the clock pulse generator may not operate properly.*[1]

To use a CPU (Cortex-A76), set 1 to WUP_REQ bit of PWRCTRLCz in the APMU module. When the power status needs to be confirmed, the status can be confirmed by WUP_REQ bit of PWRCTRLCz. For details, refer to Chapter 10. Advanced Power Management Unit.

Notes: 1. This step can be skipped if you already know which CPU is available for use.

**(11) Adaptive voltage scaling (AVS) support [for specific customer only]**

Adaptive voltage scaling (AVS) mitigates power variation in SoC level via adjusting VDD voltage. Chapter 12. Adaptive Voltage Scaling (AVS) shows the details of AVS feature [for specific customers only]

**(12) Cortex-A76 frequency settings**

Cortex-A76 operates at 1.0 GHz.

# 2. Area Map

Regarding section 2, refer to EXCEL file “002_R-CarV4M_Address_Map_r0p81.xlsx”.

To access to 40bits address area from masters who can use only 32bits address, please use IPMMU (section 19).

# 3. Pin Assignment

## 3.1 Pin Assignment

For details, refer to 003_R-CarV4M_pin_arrangement_r0p40.xlsx.

## 3.2 Mode Pin Settings

**Table 3.1 TEST3, TEST2, TEST1 Settings**

| TEST3 | TEST2 | TEST1 | Test Mode Select |
|---|---|---|---|
| 0 | 0 | 0 | User Mode* |

*Other settings are prohibited.

**Table 3.2 BSMODE Settings**

| BSMODE | JTAG Pin Operating Mode Switching |
|---|---|
| 0 | Normal operation |
| 1 | Operates in boundary scan mode. When PRESET# is at low level, pin IO control is disabled. |

**Table 3.3 MD0 Settings**

| MD0 | Free Run Mode / Step Up mode Selection |
|---|---|
| 0 | Free Run mode |
| 1 | Step up mode |

**Table 3.4 MD4, MD3, MD2 and MD1 Settings**

| MD4 | MD3 | MD2 | MD1 | Boot Device Selection |
|---|---|---|---|---|
| — | — | — | — | See section 31. Booting "Table 31.1 Mode Pin Configuration (1)" |

**Table 3.5 MD5 Settings**

| MD5 | Reserved |
|---|---|
| 1 | Other setting is prohibited |

**Table 3.6 MD7 and MD6 Settings**

| MD7 | MD6 | Master Boot Processor Selection |
|---|---|---|
| — | — | See section 31. Booting "Table 31.1 Mode Pin Configuration (2)" |

**Table 3.7 MD8 Settings**

| MD8 | Reserved |
|---|---|
| 1 | Other setting is prohibited |

**Table 3.8 MD9 Settings**

| MD9 | EXTAL/XTAL Pin Setting (See the Clock Pulse Generator section.) |
|---|---|
| 0 | Inputs an external clock to the EXTAL pin. |
| 1 | Connects a crystal resonator to the EXTAL/XTAL pin |

**Table 3.9 MD21, MD20, MD11, MD10 and MDT0 Settings**

| MDT0 | MD21 | MD20 | MD11 | MD10 | JTAG Selection |
|---|---|---|---|---|---|
| — | — | — | — | — | See Table 170.7 and Table 170.8 in section 170 “Debug and Trace”. |

**Table 3.10 MD14 and MD13 Settings**

| MD14 | MD13 | Clock Frequency Selection |
|---|---|---|
| — | — | See section 8. Clock Pulse Generator (CPG) "8.1.5 Clock Operating Mode" |

**Table 3.11 MD16 Settings**

| MD16 | Reserved |
|---|---|
| 1 | Other setting is prohibited |

**Table 3.12 MD17 and MD19 Settings**

| MD19 | MD17 | V4M-7 | V4M-5 | V4M-3 | V4M-2 |
|---|---|---|---|---|---|
| 0 | 0 | LPDDR4x-4266 or LPDDR5-6400 | LPDDR4x-3333 or LPDDR5-5000 | LPDDR4x-3200 or LPDDR5-4800 | LPDDR4x-3200 or LPDDR5-4800 |
| 0 | 1 | LPDDR4x-3733 or LPDDR5-6000 | LPDDR4x-3333 or LPDDR5-5000 | LPDDR4x-3200 or LPDDR5-4800 | LPDDR4x-3200 or LPDDR5-4800 |
| 1 | 0 | LPDDR4x-3200 or LPDDR5-5500 | LPDDR4x-3200 or LPDDR5-5000 | LPDDR4x-3200 or LPDDR5-4800 | LPDDR4x-3200 or LPDDR5-4800 |
| 1 | 1 | LPDDR4x-2133 or LPDDR5-4800 | LPDDR4x-2133 or LPDDR5-4800 | LPDDR4x-2133 or LPDDR5-4800 | LPDDR4x-2133 or LPDDR5-4800 |

**Table 3.13 MD22 and MD27 Settings**

| MD27 | MD22 | DDR Mode Selection |
|---|---|---|
| 0 | 0 | LPDDR5 |
| 0 | 1 | LPDDR4x |
| 1 | 0 | Reserved |
| 1 | 1 | Reserved |

**Table 3.14 MD23 Settings**

| MD25 | MD23 | POST target select |
|---|---|---|
| 0 | 0 | Field BIST is not activated. |
| 0 | 1 | Field BIST is not activated. |
| 1 | 0 | POST is executed on all hierarchies when MD25 = 1 |
| 1 | 1 | POST is executed on partial hierarchies when MD25 = 1. |

See Table 171.2, in which the detail specification is described.

**Table 3.15 MD24 Settings**

| MD24 | Reserved |
|---|---|
| 0 | Other setting is prohibited |

**Table 3.16 MD25 Settings**

| MD25 | Field BIST Control |
|---|---|
| 0 | Field BIST is not activated |
| 1 | Field BIST is activated |

**Table 3.17 MD29 Settings**

| MD29 | RCLK reference clock select |
|---|---|
| 0 | EXTALR select |
| 1 | On-Chip-Oscillator (OCO) select |

**Table 3.18 MD32 and MD31 Settings**

| MD32 | MD31 | SCIF Download mode speed select |
|---|---|---|
| — | — | See section 31. Booting "Table 31.2 Mode Pin Configuration (2)" |

**Table 3.19 MD33 Settings**

| MD33 | Reserved |
|---|---|
| 0 | Other setting is prohibited |

**Table 3.20 MD35 Settings**

| MD35 | Reserved |
|---|---|
| 0 | Other setting is prohibited |

**Table 3.21 MD18, MD36, and MD37 Settings**

| MD18 | MD37 | MD36 | PLL1 Operation Mode Select |
|---|---|---|---|
| 0 | 0 | 0 | Integer mode (SSCG OFF) |
| 0 | 0 | 1 | Fractional mode (SSCG OFF) |
| 0 | 1 | 0 | Fractional mode, SSCG (Down spread -1.0%) |
| 0 | 1 | 1 | Fractional mode, SSCG (Down spread -1.0%) |
| 1 | 0 | 0 | Fractional mode, SSCG (Center spread ±0.5%) |
| 1 | 0 | 1 | Fractional mode, SSCG (Center spread ±1.0%) |
| 1 | 1 | 0 | Fractional mode, SSCG (Center spread ±0.5%) |
| 1 | 1 | 1 | Fractional mode, SSCG (Center spread ±1.0%) |

See 8.1.5.1 in 008 Clock Pulse Generator (CPG), in which the detail specification for each mode is described.

# 4. Pin Multiplex

## 4.1 Pin Multiplex

Refer to 004_R-CarV4M_Pin_Multiplex_r0p81.xlsx

# 5. AP-System Core

## 5.1 Overview

The AP-System core is a core block equipped with Arm® Cortex®-A76. For details on the functions of Cortex-A76, see the relevant Technical Reference Manual.

### 5.1.1 Features

#### 5.1.1.1 Cortex-A76 CPU Cluster

Table 5.1 shows the features of Cortex-A76 CPU cluster in the AP-System Core.

Table 5.1 **Features of the AP-System Core Block (Cortex-A76)**

| | **R-Car V4M** |
|---|---|
| Version | r4p1 (Cortex-A76 CPU core)<br>r4p1 (DynamIQ Shared Unit) |
| # of Cores | R-CarV4M-7: 4 x Cortex-A76 CPU core<br>R-CarV4M-5: 4 x Cortex-A76 CPU core<br>R-CarV4M-3: 3 x Cortex-A76 CPU core (Not support Core 3)<br>R-CarV4M-2: 3 x Cortex-A76 CPU core (Not support Core 3) |
| Cluster Structure | R-CarV4M-7,V4M-5: 1 x (4-core/cluster)<br>R-CarV4M-3,V4M-2: 1 x (3-core/cluster) |
| L1 Cache | I=64KB [w/ Parity] + D=64KB [w/ ECC] |
| L2 Cache | 256 KB [w/ ECC] |
| L3 Cache | 512 KB [w/ ECC] |
| ECC | Supported |
| MMU | Supported |
| NEON/VFP | NEON supported (Advanced SIMD)<br>VFP supported |
| Cryptography Extension (Optional) | Supported |
| Max. Frequency | R-CarV4M-7: 1000 MHz<br>R-CarV4M-5: 800 MHz<br>R-CarV4M-3: 933 MHz<br>R-CarV4M-2: 933 MHz |
| Operating Clock | CPU core: ZC0ϕ, ZC1ϕ, ZC2ϕ, and ZC3ϕ<br>CPU Cluster: ZDϕ |
| | L2-Tag RAMs: ZCϕ, ZC1ϕ, ZC2ϕ, and ZC3ϕ<br>L2-Data RAMs: 2-divided clock of ZCϕ, ZC1ϕ, ZC2ϕ, and ZC3ϕ |
| | L3-Tag RAMs: ZDϕ<br>L3-Data RAMs: 2-divided clock of ZDϕ |
| | ACE bus: ZXϕ |
| Low Power Mode | Sleep mode (each CPU)<br>Core standby mode (each CPU)<br>Cortex-A76 CPU cluster power down mode (CPU cores in cluster and L3 cache shutdown: A2 = OFF) |
| Private Peripherals | Arm-based Generic Timer supported |

#### 5.1.1.2 AP Interconnect (AXCC)

AP Interconnect is an interconnect for AP System Core. It connects one ACE master and two AXI slaves.

Table 5.2 shows the module connection of each port.

The bus interconnection is connected to the CPU Clusters, Main Memory Domain AXI and Peripheral Domain AXI.

Table 5.2 Module Connection of Each Port

| | **R-Car V4M** |
|---|---|
| Slave port 0 | Cortex-A76 CPU Cluster 0 ACE |
| Master port 0 | Peripheral Domain AXI |
| Master port 1 | Main Memory Domain AXI |

#### 5.1.1.3 Generic Counter

For the general explanation about the Generic Counter, and the details of the relation between the Generic Counter and the Generic Timer, refer to the ARM manual[1]. This chapter explains the frequency for count-up of the Generic Counter. The Generic Counter serves as the CoreSight Timestamp function as well as the Generic Timer.

### 5.1.2 Block Diagram

Figure 5.1 shows the block diagram of the AP-System Core.

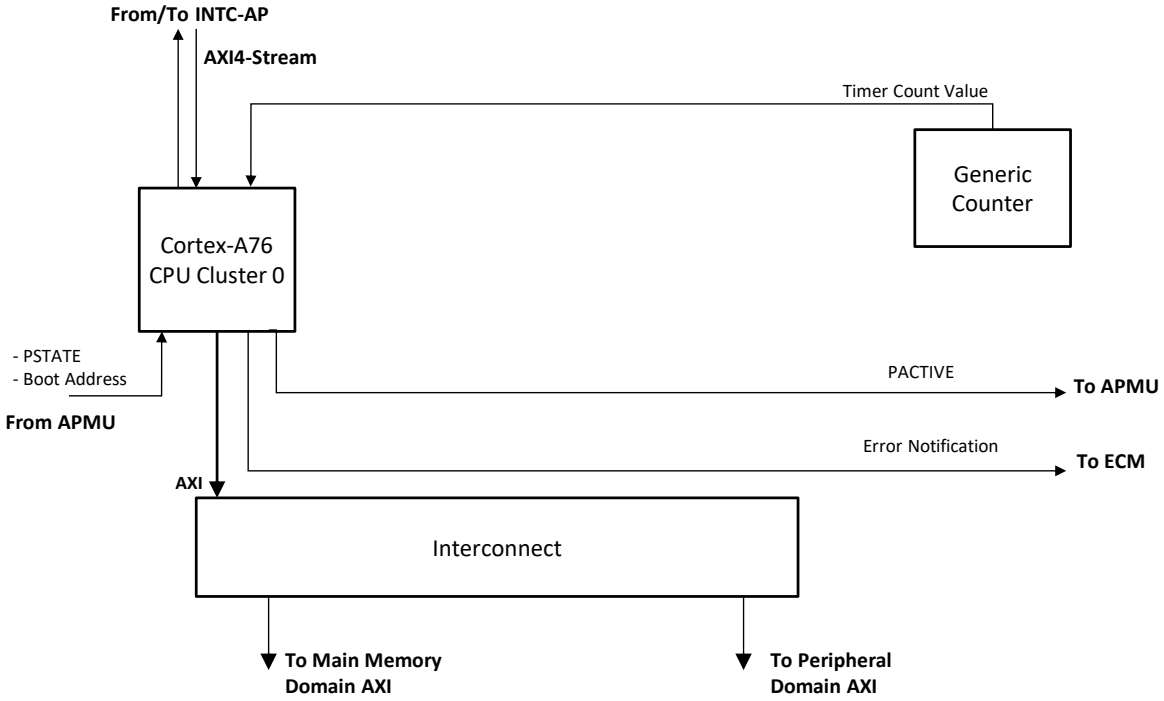



Figure 5.1 Ap-System Core Block Diagram

### 5.1.3 External Pins

In the AP-System Core, there are no external pins.

### 5.1.4 Register Configuration

**Table 5.3 Base Address**

Please refer to sheet “Base Address” of the file “R-CarV4M_UM_005_AP_System_Core_r0p81.xlsx”.

**Table 5.4 Register Configuration**

Please refer to sheet “Register Configuration” of the file “R-CarV4M_UM_005_AP_System_Core_r0p81.xlsx”.

**Table 5.6 Register Configuration (Generic Counter – Control Base: Arm-based registers)**

| Register Name | Abbreviation | R/W | Offset | Initial value | Size | R-Car V4M |
|---|---|---|---|---|---|---|
| Counter Control Register | CNTCR | R/W | H’0000 | H’0000 0000 | 32 | √ |
| Counter Status Register | CNTSR | R | H’0004 | H’0000 0000 | 32 | √ |
| Current Count Value Lower register | CNTCVL | R/W | H’0008 | H’0000 0000 | 32 | √ |
| Current Count Value Upper register | CNTCVU | R/W | H’000C | H’0000 0000 | 32 | √ |
| Base Frequency ID register | CNTFID0 | R/W | H’0020 | H’00C6 5D40 | 32 | √ |
| Counter ID Register 0 | CounterID0 | R | H’0FD0 | H’0000 0004 | 32 | √ |
| Counter ID Register 1 | CounterID1 | R | H’0FD4 | H’0000 0000 | 32 | √ |
| Counter ID Register 2 | CounterID2 | R | H’0FD8 | H’0000 0000 | 32 | √ |
| Counter ID Register 3 | CounterID3 | R | H’0FDC | H’0000 0000 | 32 | √ |
| Counter ID Register 4 | CounterID4 | R | H’0FE0 | H’0000 0001 | 32 | √ |
| Counter ID Register 5 | CounterID5 | R | H’0FE4 | H’0000 00B1 | 32 | √ |
| Counter ID Register 6 | CounterID6 | R | H’0FE8 | H’0000 001B | 32 | √ |
| Counter ID Register 7 | CounterID7 | R | H’0FEC | H’0000 0000 | 32 | √ |
| Counter ID Register 8 | CounterID8 | R | H’0FF0 | H’0000 000D | 32 | √ |
| Counter ID Register 9 | CounterID9 | R | H’0FF4 | H’0000 00F0 | 32 | √ |
| Counter ID Register 10 | CounterID10 | R | H’0FF8 | H’0000 0005 | 32 | √ |
| Counter ID Register 11 | CounterID11 | R | H’0FFC | H’0000 00B1 | 32 | √ |

Notes: 1. For details, refer to Arm architecture manual (DDI0487B_b_armv8_arm.pdf), Appendix I1 System Level Implementation of the Generic Timer.
2. Registers in Control Base region can be accessed only from secure world.

**Table 5.7 Register Configuration (Generic Counter – Read Base: Arm-based registers)**

| Register Name | Abbreviation | R/W | Offset | Initial value | Size | R-Car V4M |
|---|---|---|---|---|---|---|
| Current Count Value Lower register | CNTCVL | R | H'1000 | H'0000 0000 | 32 | √ |
| Current Count Value Upper register | CNTCVU | R | H'1004 | H'0000 0000 | 32 | √ |
| Counter ID Register 0 | CounterID0 | R | H'1FD0 | H'0000 0004 | 32 | √ |
| Counter ID Register 1 | CounterID1 | R | H'1FD4 | H'0000 0000 | 32 | √ |
| Counter ID Register 2 | CounterID2 | R | H'1FD8 | H'0000 0000 | 32 | √ |
| Counter ID Register 3 | CounterID3 | R | H'1FDC | H'0000 0000 | 32 | √ |
| Counter ID Register 4 | CounterID4 | R | H'1FE0 | H'0000 0001 | 32 | √ |
| Counter ID Register 5 | CounterID5 | R | H'1FE4 | H'0000 00B1 | 32 | √ |
| Counter ID Register 6 | CounterID6 | R | H'1FE8 | H'0000 001B | 32 | √ |
| Counter ID Register 7 | CounterID7 | R | H'1FEC | H'0000 0000 | 32 | √ |
| Counter ID Register 8 | CounterID8 | R | H'1FF0 | H'0000 000D | 32 | √ |
| Counter ID Register 9 | CounterID9 | R | H'1FF4 | H'0000 00F0 | 32 | √ |
| Counter ID Register 10 | CounterID10 | R | H'1FF8 | H'0000 0005 | 32 | √ |
| Counter ID Register 11 | CounterID11 | R | H'1FFC | H'0000 00B1 | 32 | √ |

Note: For details, refer to Arm architecture reference manual (DDI0487B_b_armv8_arm.pdf), Appendix I1 System Level Implementation of the Generic Timer.

### 5.1.5 Connected Module

Table 5.8 shows the connected modules to the AP-System Core. As for the features of the connected modules, refer to the chapter for each module.

Table 5.8 Connected Modules

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| AP-System Core | APMU | CPU power on/off sequence<br>Boot address setting<br>CPU reset control |
| | CPG | Clock supply |
| | SYSC | Power switch control |
| | RST | CPU reset control |
| | ECM | Error signal control |
| | INTC-AP | Interrupt handling |
| | AXI-bus | On-chip Interconnection bus |
| | DBSC4 | External bus controller |
| | Debug and Trace | Debug function |

### 5.1.6 Reference

[1] ARM® Architecture Reference Manual ARMv8, for ARMv8-A architecture profile, Issue B.b
[2] Arm® DynamIQ™ Shared Unit Technical Reference Manual, Revision: r4p1
[3] Arm® Cortex®-A76 Core Technical Reference Manual, Revision: r4p1

## 5.2 Register Description

As for register description of AP-System Core, refer to the following documents.

- ARM® Architecture Reference Manual ARMv8, for ARMv8-A architecture profile, Issue B.b
- Arm® DynamIQ™ Shared Unit Technical Reference Manual, Revision: r4p1
- Arm® Cortex®-A76 Core Technical Reference Manual, Revision: r4p1

Register descriptions of other registers are shown in the following subclauses.

Please refer to sheet “Register Description” of the file “R-CarV4M_UM_005_AP_System_Core_r0p81.xlsx”.

## 5.3 Operation

### 5.3.1 Boot Address Setting

Cortex-A76 supports only AArch64 mode, the boot address is controlled by RVBARL/HCx registers in APMU. For the initial value and details of these registers, refer to Section 10, Advanced Power Management Unit.

### 5.3.2 Power-Up/Down Mechanism

Each core has the following power modes.

- ON mode
- OFF mode
- OFF_EMU mode
- Pseudo OFF mode

And, each cluster has the following power modes.

- ON mode
- OFF mode
- FUNC_RET mode
- MEM_RET mode
- L3 cache partial power down mode
- Pseudo OFF mode

The detail of the above power modes is described in Section 10.3.1. And, the power mode transition sequence is described in Section 10.3.2.

### 5.3.3 Switching Between AArch32 and AArch64

There is no operation to switch between AArch32 and AArch64 with warm reset because Cortex-A76 does not support AArch32 in EL1, EL2, and EL3.

### 5.3.4 System Address Map

#### 5.3.4.1 Overview

The AXCC has the system address map. The address spaces defined in this map are split into 7 regions. The connection to each master port is defined by the value of ADDRMAP[39:0] in advance.

Table 5.11 is System Address Map of AXCC

**Table 5.11 System Address Map for AXCC**

| Address Region | Address Range | Destination |
|---|---|---|
| Region 0 | H'00 0000 0000 to H'00 3FFF FFFF | Master port 0 (Peripheral) |
| Region 1 | H'00 4000 0000 to H'00 BFFF FFFF | Master port 1 (Main Memory) |
| Region 2 | H'00 C000 0000 to H'00 DFFF FFFF | Master port 0 (Peripheral) |
| Region 3 | H'00 E000 0000 to H'00 E62F FFFF | Master port 0 (Peripheral) |
| Region 4 | H'00 E630 0000 to H'00 E63F FFFF | Master port 1 (System RAM) |
| Region 5 | H'00 E640 0000 to H'00 FFFF FFFF | Master port 0 (Peripheral) |
| Region 6 | H'01 0000 0000 to H'0F FFFF FFFF | Master port 1 (Main Memory) |

### 5.3.5 Frequency for Counting Up of Generic Counter

The generic timer counts up by one every clock edge of SASYNCRTφ clock continuously.

### 5.3.6 How to Start a Generic Counter

Set bit 0 in CNTCR to 1 to start the Generic Counter when the CPU is placed in the secure mode.

## 5.4 Usage Notes

### 5.4.1 Reserved

### 5.4.2 Reset Restriction for AXCC

The all of the Cortex-A76 CPU cores and cluster shall be reset when the AXCC is reset. The reasons are the following:

- There are safety mechanisms in ACE Bus on CPU side and AXCC side. They count the number of transactions. In master side the number is add to USER signal. In slave side, the USER signal is compared with its number. The numbers in both sides shall be reset simultaneously to prevent a false safety error.

When you reset a Cortex-A76 CPU core, caches inside the core need to be invalidated in advance. However, if the caches cannot be invalidated for some reasons (e.g. CPU hang-up), not only the corresponding core but also all other Cortex-A76 CPU cores and cluster, and AXCC need to be reset.

### 5.4.3 Register Initialization Before C4 power on

The registers in the Table 5.13 shall be initialized by Cortex-R52 before C4 power on.

**Table 5.13 The registers that need to be initialized before C4 power on**

| Register name | Register address | Setting value |
|---|---|---|
| APSREG_AP_CLUSTER*n*_AUX0<br>(*n* is cluster number [0]) | H'E628_*n*010<br>(*n* is cluster number [0]) | H'0000_0003 |
| APSREG_AXCC_AUX | H'E628_9010 | H'0000_0001 |

### 5.4.4 Virtual Memory Extensions for ARMv8.1

The Distributed Virtual Memory extensions for ARMv8.1 is not supported.

### 5.4.5 Prevention of unexpected phenomenon due to CPU speculative access

For Cortex-A76, set Device Memory and Execute Never, or do not map the speculative access prohibited regions[*1] in MMU page translation table.

Note *1: Other than RAM/ROM regions that are permitted to be accessed from the CPU and can access randomly.

## 5.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 5.5.1 Cortex-A76 CPU Cluster

#### 5.5.1.1 Overview

Cortex-A76 CPU Cluster has safety mechanisms such as parity, error correction code and hardware redundancy.

Table 5.14 shows ECC and parity safety mechanism list for memories of Cortexa-A76.

**Table 5.14 ECC/Parity list for memories (Cortex-A76)**

| RAM | Protection Type | Correction Behaviour*1 | Uncorrectable Error Behaviour |
|---|---|---|---|
| L1 instruction cache tag | Parity | The line that contains the error is invalidated from the L1 instruction cache and fetched again from the subsequent memory system. | - |
| L1 instruction cache data | SED | The line that contains the error is invalidated from the L1 instruction cache and fetched again from the subsequent memory system. | - |
| MMU translation cache | Parity | Entry invalidated, new pagewalk started to refetch the entry. | - |
| L1 data cache tag | SECDED | The cache line that contains the error gets evicted, corrected inline, and refilled to the core. | Fatal RAM error is flagged to ECM. |
| L1 data cache data | SECDED | The cache line that contains the error gets evicted, corrected inline, and refilled to the core. | Synchronous Data Abort, Prefetch Abort SystemError Interrupt. Fatal RAM error is flagged to ECM. |
| L2 cache tag | SECDED | Tag is corrected inline. | Fatal RAM error is flagged to ECM. |
| L2 cache data | SECDED | Data is corrected inline. | Synchronous Data Abort, Prefetch Abort or System Error Interrupt. Fatal RAM error is flagged to ECM. |
| L2 TQ data | SECDED | Data is corrected inline. | Fatal RAM error is flagged to ECM. |
| L3 cache data<br><br>Long-term data buffer | SECDED | Data is corrected inline. | Data is marked as poisoned, abort is generated when poisoned data is consumed. Fatal RAM error is flagged to ECM. |

| | | | |
|---|---|---|---|
| L3 cache tag | SECDED | The value is corrected and written back to the RAM. The lookup is replayed. | The line is invalidated. Fatal RAM error is flagged to ECM. |
| Snoop filter | SECDED | The value is corrected and written back to the RAM. The lookup is replayed. | Fatal RAM error is flagged to ECM. |

*1: RAM error is flagged to ECM if ERR0CTLR.CFI bit is set

#### 5.5.1.2 Parity

**(1) Overview**

Parity check is applied to the following internal RAMs in the AP-System Core.

— L1 instruction cache tag RAM, L1 instruction cache data RAM and MMU translation cache RAM in Cortex-A76.

**(2) Hardware Description**

As for register description of the AP-System Core, refer to the following documents.

Reference [3].

A parity error is treated as a correctable error. A correctable error is reported separately from uncorrectable error, where uncorrectable error is detected by ECC protection whenthe register in Cortex-A76 CPU core is set as follows:
— ERR0CTLR = H'0000 0105
By this setting, "RAM Error" in Figure 5.2 represents only the correctable error.

For more details of the error type definition and the error reporting, refer to chapter A8 "Reliability, Availability, and Serviceability (RAS)" in Reference [3].

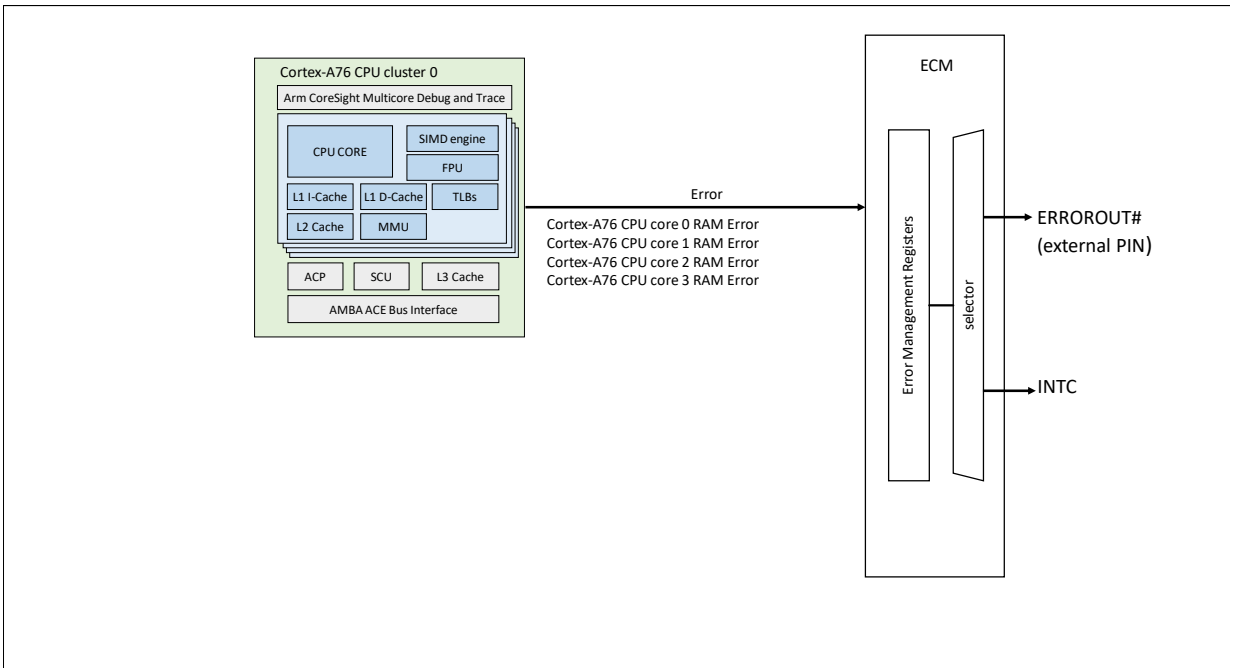



Figure 5.2 Parity check for AP-System Core

The following internal RAMs in Cortex-A76 CPU core (Table 5.15) are protected by parity checking.

Table 5.15 Parity protection (Cortex-A76)

| RAM | Protection method | Error signals |
|---|---|---|
| L1 instruction cache tag RAM | 1 parity bit | Cortex-A76 CPU core *n* RAM Error (correctable error) |
| L1 instruction cache data RAM | SED (parity) | Cortex-A76 CPU core *n* RAM Error (correctable error) |
| MMU translation cache RAM | 2 interleaved parity bits | Cortex-A76 CPU core *n* RAM Error (correctable error) |

Note: *n* is core number [0-3].
Note: For details of parity protection, refer to chapter A8 “Reliability, Availability, and Serviceability (RAS)” in Reference [3].

### 5.5.1.3 Error Correction Code (ECC)

#### (1) Overview

ECC protection is applied to the following RAMs.
— L1 data cache tag RAM and L1 data cache data RAM in Cortex-A76.
— L2 cache tag RAM, L2 cache data RAM, and L2 TQ RAM in Cortex-A76.
— L3 tag RAM, L3 data RAM, SCU snoop filter RAM, and SCU Long-Term Data Buffer RAM in Cortex-A76.

#### (2) Hardware Description

As for register description of the AP-System Core, refer to Reference [2] and Reference [3].

Cortex-A76 CPU cluster supports three types of errors: correctable error (single-bit ECC error), uncorrectable error (double-bit ECC error on tag RAM or snoop filter RAM) and deferred error (double-bit ECC error on data RAM or SCU Long-Term Data Buffer RAM, where the chunk of data with the error is marked as poisoned).

A correctable error and an uncorrectable error are reported separately when the registers in Cortex-A76 CPU cluster are set as follows:
— ERR0CTLR = H’0000 0105
— ERR1CTLR = H’0000 0105
By this setting, “RAM Error” in Figure 5.3 represents only the correctable error, “RAM Fatal Error” represents only the uncorrectable error. Deferred error is not reported to ECM and “RAM Fatal Error” will be reported when poisoned data are consumed by CPU.

For more details of the error type definition and the error reporting, refer to section A5.5 “L3 cache ECC and parity” in Reference [2] and chapter A8 “Reliability, Availability, and Serviceability (RAS)” in Reference [3].

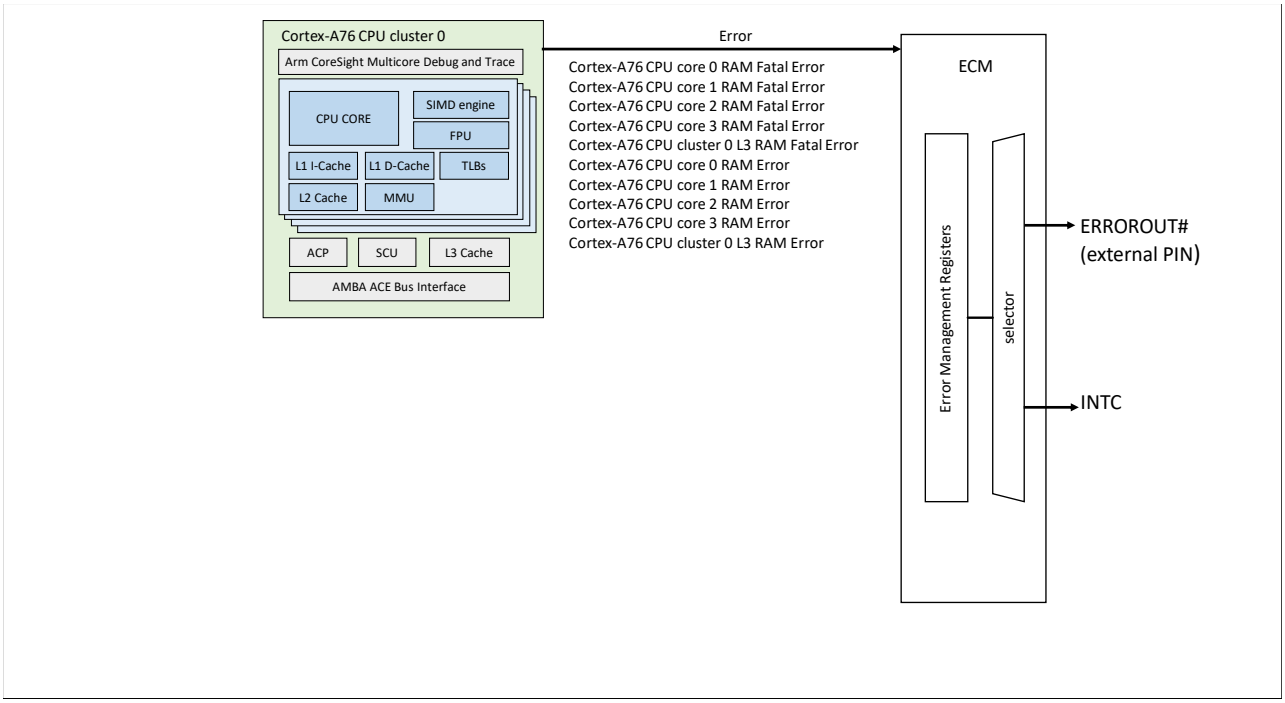



Figure 5.3 ECC Protection for AP-System Core

The following internal RAMs in Cortex-A76 CPU cluster (Table 5.16) are protected by ECC.

**Table 5.16 ECC protection (Cortex-A76 CPU cluster)**

| RAM | Protection method | Error signals |
|---|---|---|
| L1 data cache tag RAM | SECDED (ECC) | Cortex-A76 CPU core *n* RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU core *n* RAM Error (correctable error) |
| L1 data cache data RAM | SECDED (ECC) | Cortex-A76 CPU core *n* RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU core *n* RAM Error (correctable error) |
| L2 cache tag RAM | SECDED (ECC) | Cortex-A76 CPU core *n* RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU core *n* RAM Error (correctable error) |
| L2 cache data RAM | SECDED (ECC) | Cortex-A76 CPU core *n* RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU core *n* RAM Error (correctable error) |
| L2 TQ data RAM | SECDED (ECC) | Cortex-A76 CPU core *n* RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU core *n* RAM Error (correctable error) |
| L3 tag RAM | SECDED (ECC) | Cortex-A76 CPU cluster *m* L3 RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU cluster *m* L3 RAM Error (correctable error) |
| L3 data RAM | SECDED (ECC) | Cortex-A76 CPU cluster *m* L3 RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU cluster *m* L3 RAM Error (correctable error) |
| SCU snoop filter RAM | SECDED (ECC) | Cortex-A76 CPU cluster *m* L3 RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU cluster *m* L3 RAM Error (correctable error) |
| SCU Long-Term Data Buffer RAM | SECDED (ECC) | Cortex-A76 CPU cluster *m* L3 RAM Fatal Error (uncorrectable error) / Cortex-A76 CPU cluster *m* L3 RAM Error (correctable error) |

Note: *m* is cluster number [0] and *n* is core number [0-3].
Note: For details of ECC protection, refer to section A5.5 “L3 cache ECC and parity” in Reference [2] and chapter A8 “Reliability, Availability, and Serviceability (RAS)” in Reference [3].

#### 5.5.1.4 ACE Bus Protection

Please refer to 5.5.2.3(1).

#### 5.5.1.5 AXI4-Stream Bus Protection

**(1) Overview**

AXI4-Stream Bus are protected by EDC and ORDER gen/check.

**(2) Hardware Description**

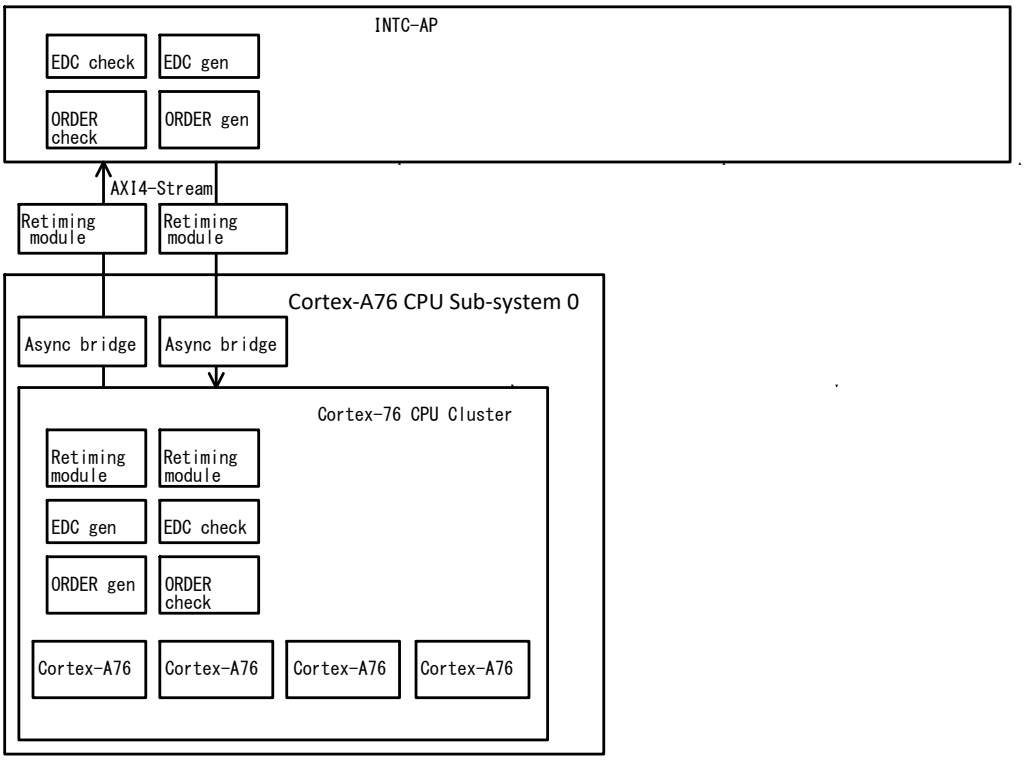



Figure 5.7 Block diagram of AXI4-Stream with safety mechanisms

**[EDC protection between AP-System Core and INTC-AP]**

EDC protection are implemented to protect each AXI4-Stream channel between AP-System Core and INTC-AP.

**[ORDER gen/check]**

The purpose of ORDER check is to detect the fault of AXI4-Stream Asynchronous bridges.

*ORDERgen* generates an incremental counter value for each transaction. Then, the counter value is attached to AXI4-Stream USER signal as a part of AXI4-Stream transaction. *ORDERchk* also generates an incremental counter value for each received transaction and compare this counter value with the value attached in USER signal. If 2 counter values are different, an error is informed to ECM module.

ORDER generator and ORDER checker are implemented as a pair for each AXI4-Stream channel.

#### 5.5.1.6 P-channel Interface Protection

P-channel interface is used to communicate between APMU and Cortex-A76. And Safety Mechanism for this interface is implemented in APMU and Cortex-A76. The block diagram is shown in Figure 5.9.

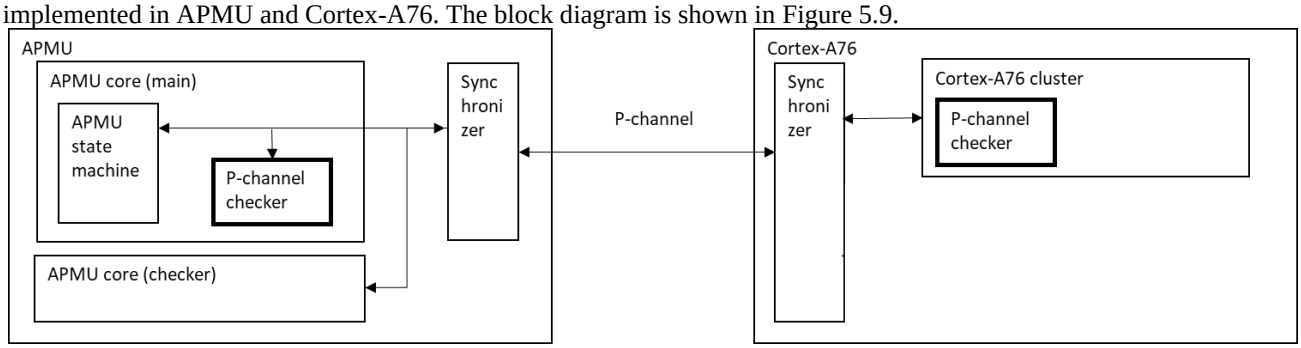



Figure 5.9 Safety Mechanism for P-channel between APMU and Cortex-A76

**(1) Safety Mechanism in Cortex-A76 side**

This Safety Mechanism can find two types of errors. The first is parity error and the second is P-channel signal protocol error. If either one error is detected, Cortex-A76 notifies error detection to ECM.

1. Parity error for P-channel signal

   A parity bit is added to P-channel signal from APMU to Cortex-A76. The parity checking logic is implemented in Cortex-A76. It detects the parity error.

2. P-channel protocol error

   Toggle ordering of P-channel signals is defined by Arm. About the detail, refer to AMBA Low Power Interface Specification. This mechanism detects failing of this ordering. For example, when PREQ signal from APMU is asserted, it has to be kept asserted until PACCEPT or PDENY signal from Cortex-A76 core is asserted. If PREQ signal are negated before PACCEPT or PDENY signal are asserted, this mechanism judges it as P-channel protocol error.

**(2) Safety Mechanism in APMU side**

This Safety Mechanism can find three types of errors.

1. P-channel protocol error
2. P-channel corruption error
3. FSM deadlock error

The error detection flag is stored to FSM Lock Register for each FSM. This register is in APMU. When error detection flag is set, 1 is set to each bit of ERRSTSR, and APMU notifies error detecting to ECM.

For more details, please refer to Section 10.5.3.

#### 5.5.1.7 Generic Counter Interface Protection

**(1) Overview**

Hardware Redundancy is applied to detect the fault of Generic Counter Interface between Generic Counter and AP-System Cores or Realtime Core.

**(2) Hardware Description**

Generic Counter Interface, Async. Absorber, and Gray code to Binary code converter are duplicated. If the difference between two values of output from two converters is more than 1, an error is notified to ECM. The block diagram is shown in Figure 5.11.

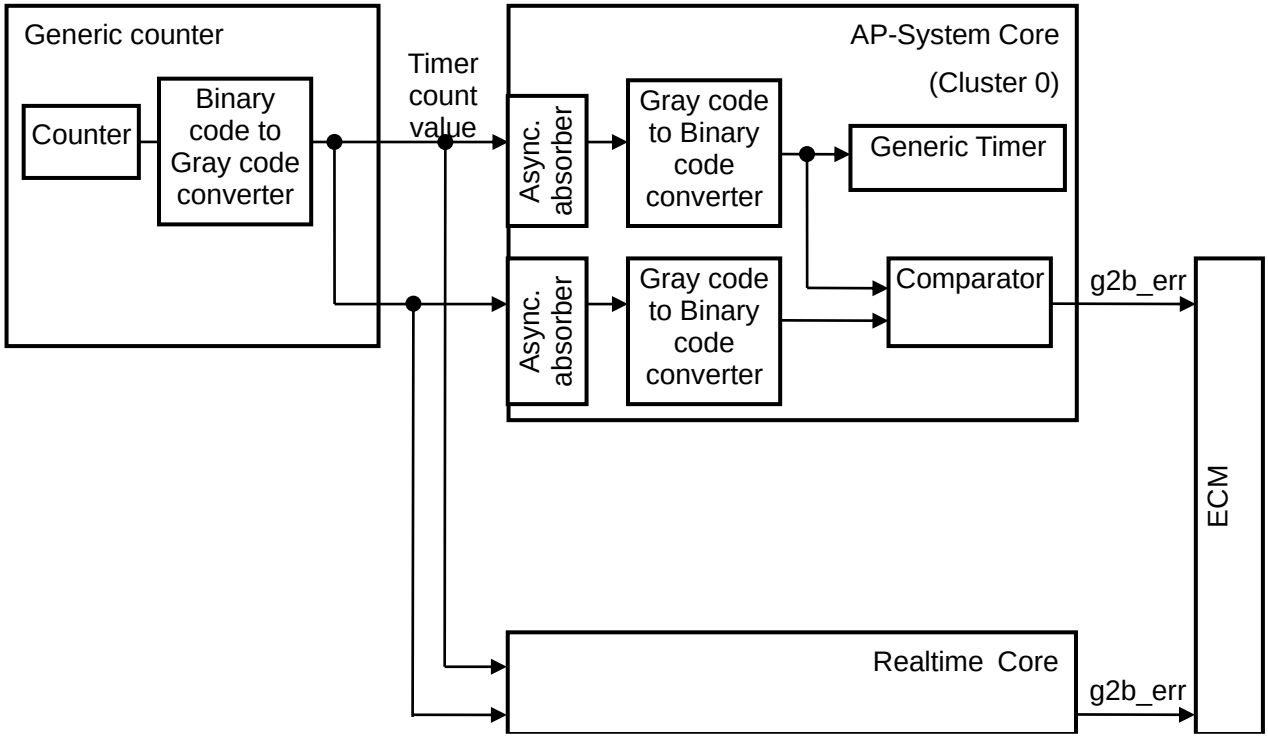



**Figure 5.11 Generic Counter Interface Protection**

#### 5.5.1.8 Debug Authentication Status Register Check

**(1) Overview**

This safety mechanism is a periodical check executed by software on the SoC.
This safety mechanism detects faults of debug authentication signals between the Debug module and the AP-System Core by checking whether the debug authentication status register value and the expected one match. These faults could cause unintended activation of debug mode of the AP-System Core.

**(2) Hardware Description**

None.

### 5.5.2 AP Interconnect (AXCC)

#### 5.5.2.1 Overview

Figure 5.13 shows the block diagram of AXCC module with Safety Mechanisms implemented.

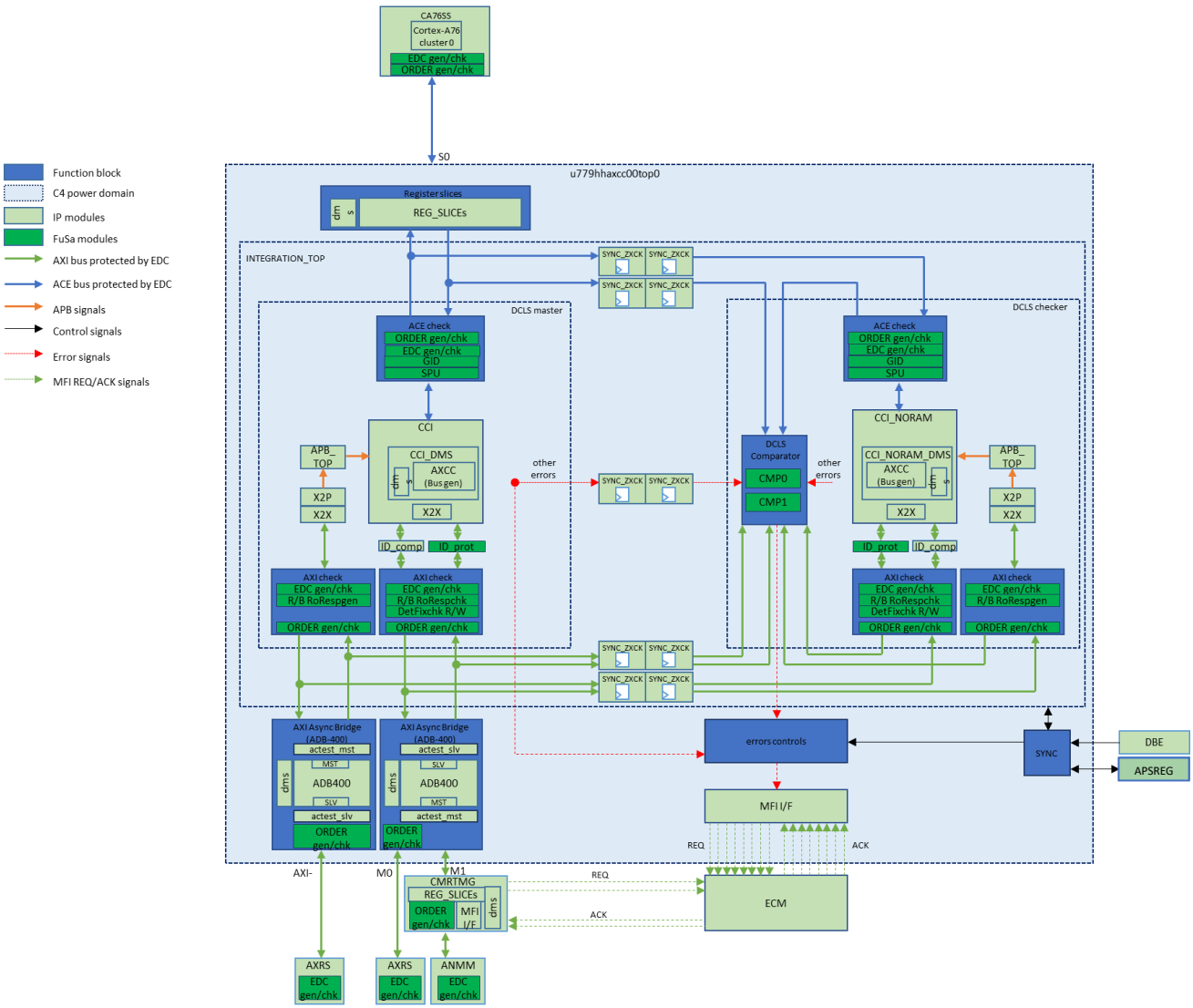



**Figure 5.13 Block diagram of AXCC module with safety mechanisms**

#### 5.5.2.2 Reserved section

#### 5.5.2.3 Bus Protection

**(1) Bus protection for ACE protocol**

**(a) Overview**

EDC protection is implemented to protect ACE Slave Interface between Cortex-A76 CPU cluster and AXCC

W/RACK check is implemented in ACE Slave Interface to detect error when no outstanding but having response.

MPU Wrapper: Region ID protection and ORDER gen/check function, as below:

- Region ID protection is implemented to protect memory access from Cortex-A76 CPU cores to memory.
- ORDER gen/check is implemented to detect faults of FIFO read/write pointers in ACE Asynchronous bridges.

**(b) Hardware Description**

**[EDC protection between Cortex-A76 CPU cluster and AXCC]**

EDC protection are implemented to protect AXI channel (AR, AW, R, W, B) between Cortex-A76 CPU cluster and AXCC.

**[W/RACK check]**

Figure 5.15 shows the AXCC WACK/RACK check diagram. WACK/RACK safety mechanism at ACE Slave Interfaces is used to detect error when no outstanding transactions (counter value of DMS function) but having response (WACK/RACK assert).

When an error detected, AXCC sub-system informs to ECM module.

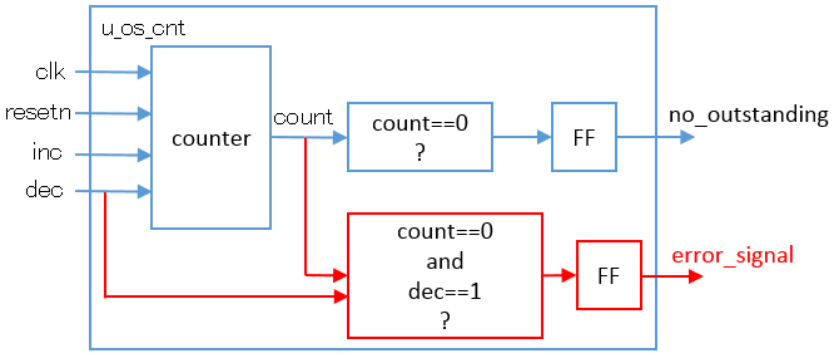



**Figure 5.15 AXCC W/RACK check diagram**

**[ORDER gen/check]**

Figure 5.16 shows the block diagram of ORDER check.

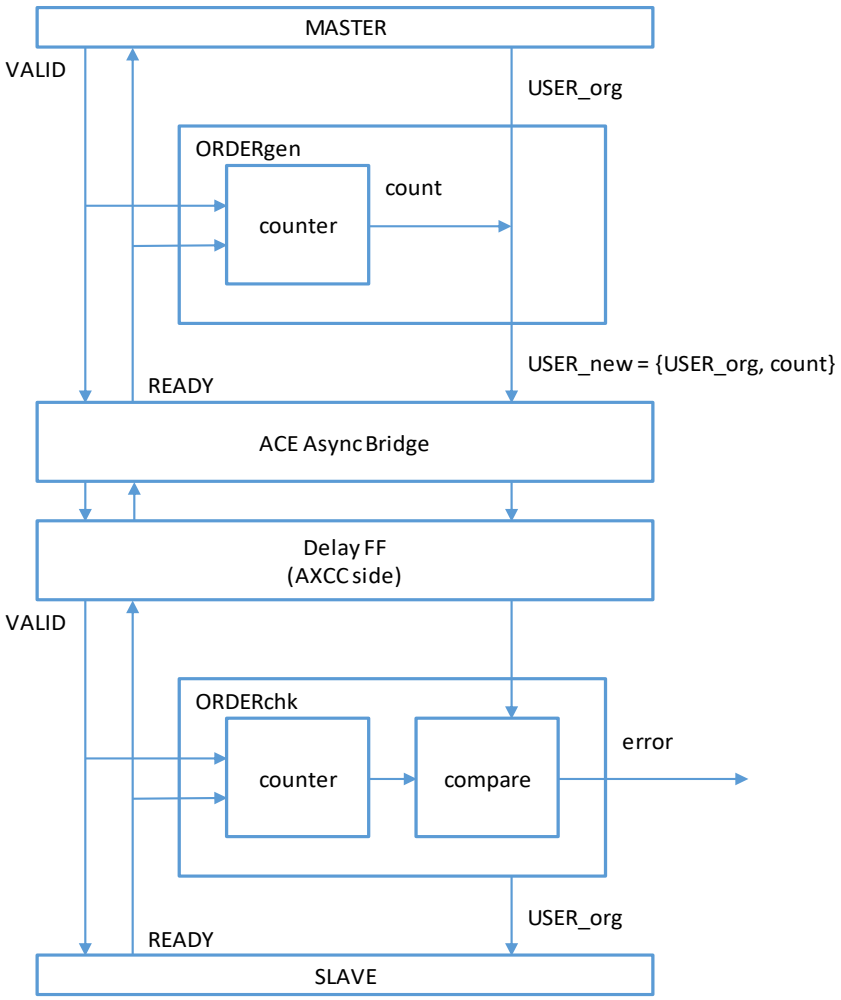



**Figure 5.16 AXCC ORDER gen/check diagram**

The purpose of ORDER check is to detect the wrong order of transaction of ACE Asynchronous bridges.

*ORDERgen* generates an incremental counter value for each transaction. Then, the counter value is attached to ACE USER signal as a part of ACE transaction. *ORDERchk* also generates an incremental counter value for each received transaction and compare this counter value with the value attached in USER signal. If 2 counter values are different, an error is informed to ECM module.

ORDER generator and ORDER checker are implemented as a pair for each ACE channel (AR, AW, R, W, B).

**[Region ID protection]**

Figure 5.18 and Figure 5.19 show the AXCC Region ID diagram. To achieve freedom from interference, the Region ID (RGID) protection concept is implemented. Region ID protection described in “18.5.9 Region ID protection”. For AP System Core, we applied it to AXCC.

Cortex-A76 CPU Cluster attaches Region ID in PA (physical address) bit [39:36]. MPU wrapper (included Region ID checker) compares Region ID of CPU’s transaction with authorized Region ID and only Authorized Region

ID can pass MPU. If the Region ID of CPU's transaction is not same Authorized Region ID, the transactions are protected and the CPU Cluster cannot issue transaction.

AXCC MPU GID Register 0 controls this protection feature. If bit 0 of this register is 1, any access with Region ID=0 is allowed. If bit 1 is 1, Region ID=1 is allowed, and same rule can be applied to bit 2-15 as well. If the access is not allowed, Region ID will be replaced with bit 16-19 of AXCC MPU GID Register.

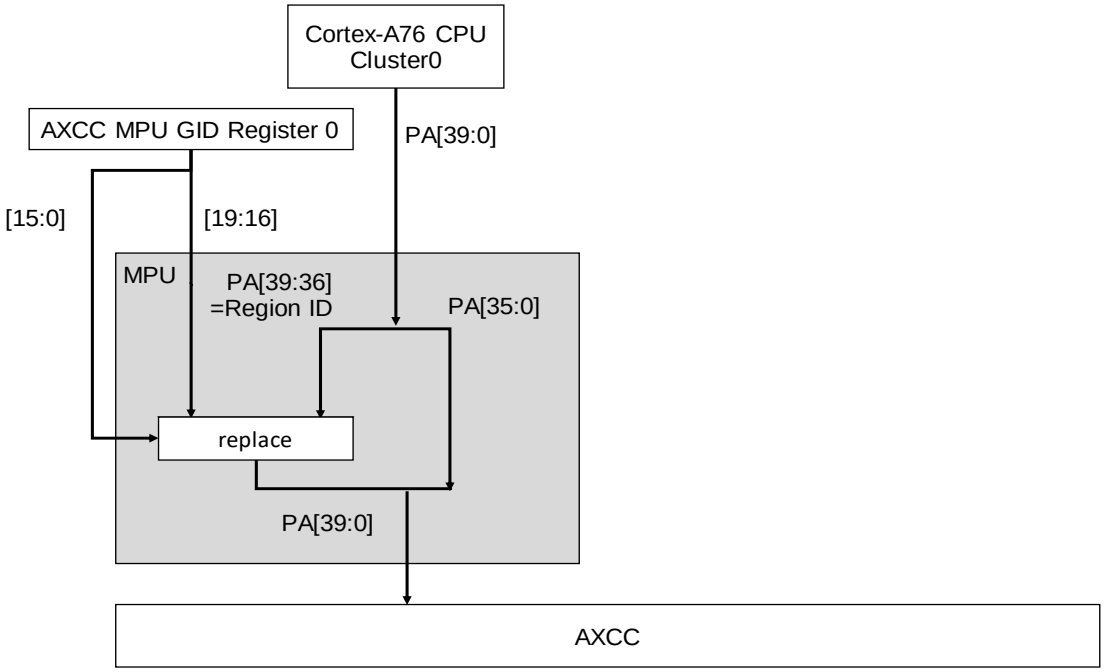



Figure 5.18 AXCC Region ID check & replacement

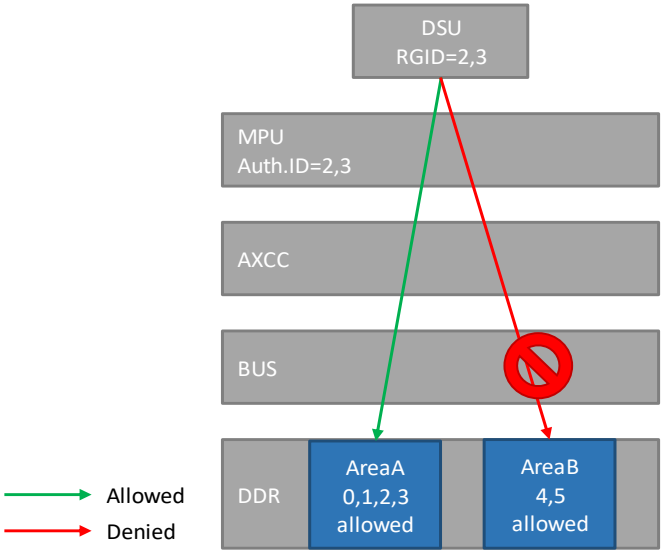



Figure 5.19 AXCC Region ID diagram

Note:

(1) Cortex-A76 CPU Cluster can have multiple Region ID.

(2) MPU has Authorized Region ID register (The register setting shall be set by Cortex-R52 before start Cortex-A76 CPU Cluster).

(3) The cache contents (L1/L2/L3) shall be flushed before changing RGID setting.

(4) When an access is protected by MPU, (i.e., when the replaced Region ID is output from MPU), the CPU cluster which issued that illegal access needs to be reset immediately. To do this, the replaced Region ID must be denied by all slave I/F.

### (2) Bus protection for AXI protocol

#### (a) Overview

AXCC has two AXI master interfaces ( M0 and M1 ) and one AXI slave interface (APB). The AXI interfaces are protected by following safety mechanism:

- EDC protection
- Routing Response check
- ORDER check
- Bus timeout detection
- Region ID
- Source ID
- AXI ID Duplication

#### (b) Hardware Description

Figure 5.21 shows Bus protection implemented in AXCC.

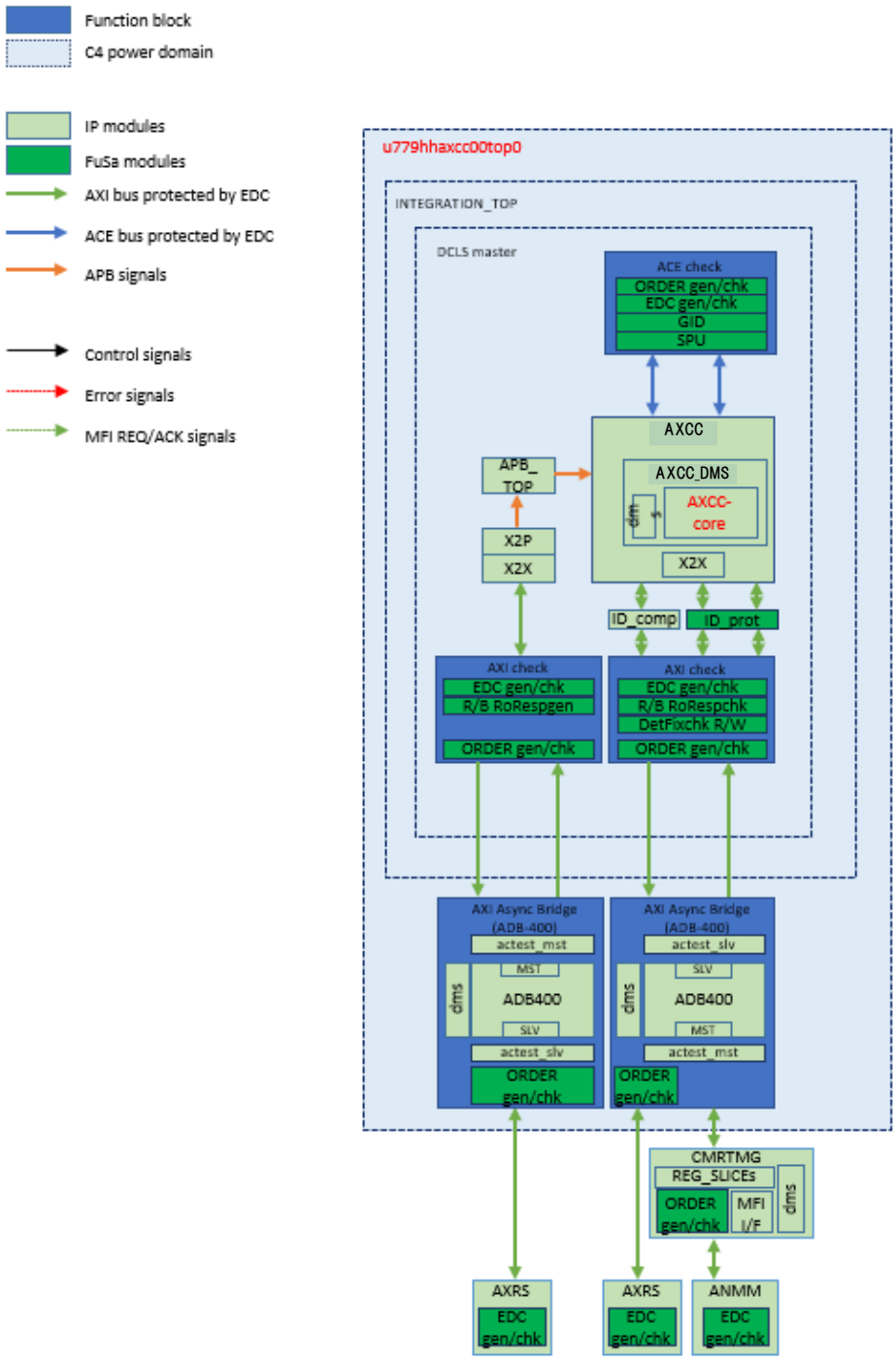



**Figure 5.21 AP Interconnect Bus protection diagram**

**[Bus timeout detection]**

Bus timeout detection (DetFIX) function which provides request timeout, illegal response. For detail, see “Section 15.4.5.5, DetFIX (inc. timeout counter)”.

**[ORDER check]**

Figure 5.22 shows the block diagram of ORDER check:

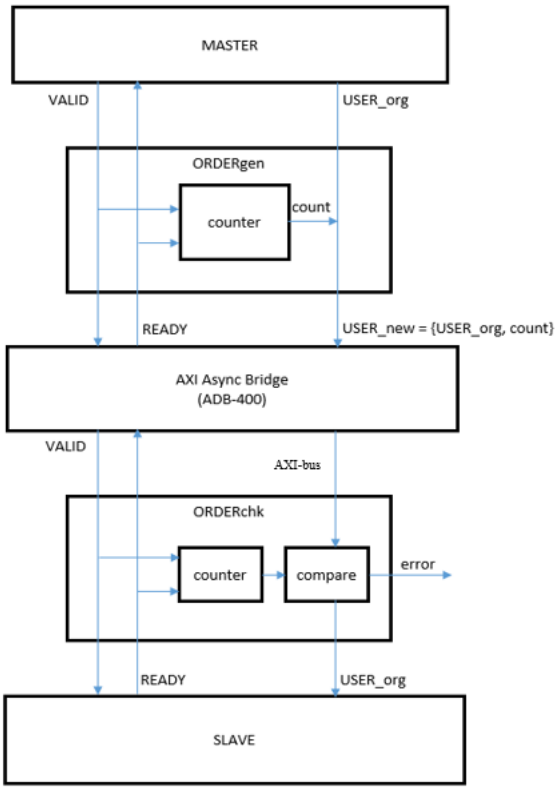



**Figure 5.22 AP Interconnect ORDER check for AXI diagram**

The purpose of ORDER check is to detect the fault of FIFO read/write pointer in AXI Asynchronous bridges.

ORDERgen generates an incremental counter value for each transaction. Then, the counter value is attached into AXI USER signal as a part of AXI transaction. ORDERchk also generates an incremental counter value for each received transaction and compare this counter value with the value attached in USER signal. If two counter values are different, an error is informed to ECM module.

For M0 and APB interfaces, ORDER generator and ORDER checker are implemented inside AXCC as a pair for each AXI channel (AR/AW/W/R/B).

For M1 interfaces, ORDER generator and ORDER checker are implemented inside AXCC and re-timing module for master port of AXCC as a pair for each AXI channel (AR/AW/W/R/B)

**[Region ID]**

This part is same as “For ACE protocol”. Please refer to “Region ID” section of “For ACE protocol”.

**[Source ID]**

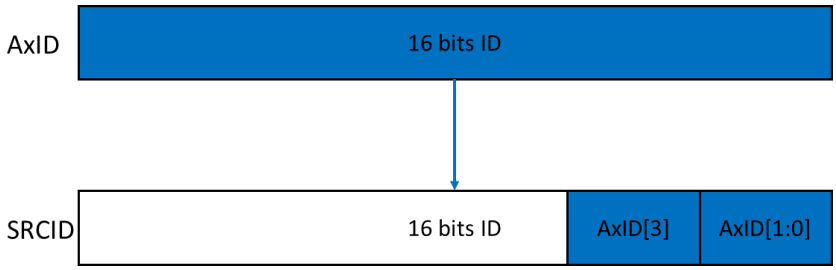



**Figure 5.23 Configuration for Source ID**

Figure 5.23 shows the Configuration for Source ID. The purpose of Source ID is to identify which Cortex-A76 CPU Cluster is accessing to bus system. It is used for BUS safety mechanism and debug purpose. For more detail, please refer to 18 AXI-bus.

Table 5.20 shows the Source ID for each Cortex-A76 CPU Cluster (AR/AW channel). It is just for the device (AXI protocol) access. For normal access, bit 0 is don't care. It means, cluster number can be recognized, but core number cannot be recognized.

**Table 5.20. Source ID of each Cortex-A76 CPU core**

| Cortex-A76 CPU core | Source ID (binary value) |
|---|---|
| Core 0 | B'00001000 |
| Core 1 | B'00001001 |
| Core 2 | B'00001010 |
| Core 3 | B'00001011 |

AXI Master interface uses AXI4 protocol, this protocol does not support ID for W channel. In this case, the ID's value is fixed by B'00001000.

For M0 interface, there is ID compress function to compress ID bit width from 12 to 8 bit. These bits are protected by xID protection function from bus system. But protected ID width is only 8 bit. Therefore, ID compress function is required.

**[EDC protection between AXCC and AXI (Slave Access Bus/CPU domain AXI)]**

EDC protection are implemented to protect all AXI channels (AR/AW/W/R/B) between AXCC AXI Master/Slave Interface and Slave Access Bus/CPU domain AXI.

**[Routing Response Gen and Check (RoRespgen and RoRespchk)]**

The purpose of Routing Response gen and check are to protect AXI channels (R/B) between AXCC and AXI-BUS.

Routing Response gen records the Source ID from ARUSER/AWUSER and attaches the Source ID on RUSER/BUSER when R/B arrives at the Routing Response gen. Routing Response check compares Source ID and the ID which are attached on RUSER/BUSER. If 2 values are different, an error is informed to ECM module.

For AXCC AXI Master/Slave interface, Routing Response gen and check are implemented. For more details, please refer to “18.5 Safety Mechanism for AXI-bus”.

**[ID Protection]**

Figure 5.25 shows the ID protection for AXI Master Interface. For AXI Master Interface 1 and AXI Master Interface 2, AXI_xID[7:0] are protected by EDC and AXI_xID[11:8] are protected by signal duplication.

The error signal is connected to other EDC errors.

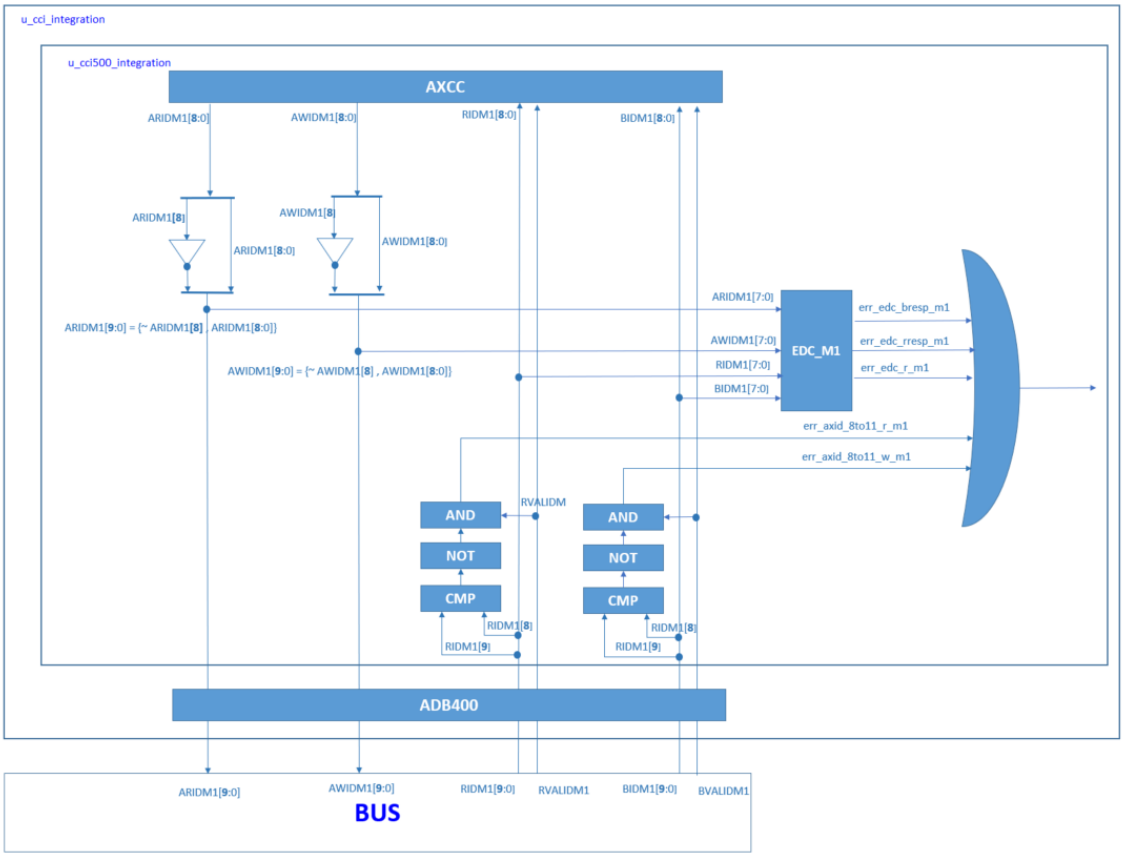



**Figure 5.25 ID Protection for AXI Master Interface**

### 5.5.2.4 Hardware Redundancy (DCLS)

**(1) Overview**

Figure 5.27 shows the dual core lockstep (DCLS) architecture is applied to the AP Interconnect (AXCC).

**(2) Hardware Description**

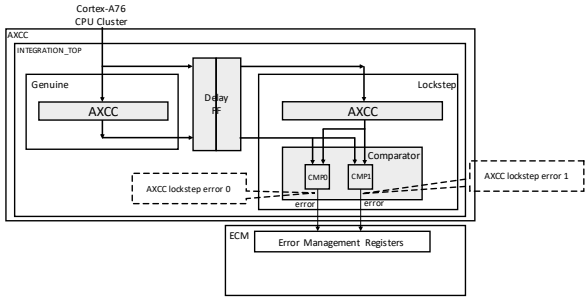



**Figure 5.27 Block diagram of AXCC with DCLS mechanism**

The dual core lockstep (DCLS) exists in the AP interconnect (AXCC) in order to check the malfunction of this module. When an error related to this is detected, the error is reported to the ECM module.

DCLS are implemented for AXCC:

- EDC generators/checkers, ORDER generators/checkers, GID and DetFix checkers are included in the target of DCLS.
    - Once the output values from the checker and master do not match, the comparator outputs an error signal. The comparator itself (called comparator 0) is also the target of duplication (called comparator 1).

Figure 5.29 shows DCLS structure implemented in AXCC in order to mitigate Common Cause Failure (CCF) by following key points:

- Two comparators are implemented. Comparator 0 and Comparator 1 are located in Checker. Errors from 2 comparators are informed to ECM independently.
- DCLS Checker, Comparator 0 and Comparator 1 are separated from DCLS Master by spatial separation.
- Two clock cycles delay for inputs of DCLS Cheker. Two clock cycles delay for output of DCLS Master.
- Clock tree is separated for DCLS Master and DCLS checker.

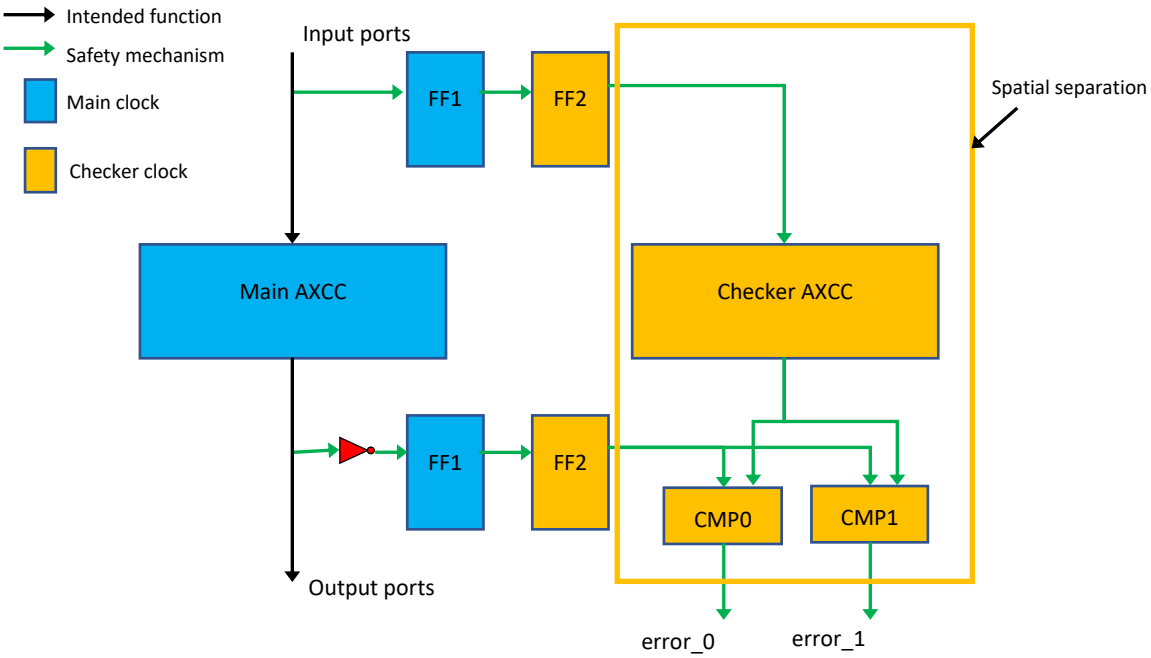


**Figure 5.29 DCLS structure of AXCC**

#### 5.5.2.5 Other Error notification

RAM is not implemented for AXCC.

### 5.5.3 Memory protection system

#### 5.5.3.1 Overview

AP System Core has the following safety mechanisms. Figure 5.31 shows block diagram of AP system Core with the safety mechanisms. The safety mechanisms are as follows:

- Runtime Test
- AXCC DCLS
- ACE/AXI4Stream/AXI EDC
- ACE/AXI4Stream/AXI ORDER check for Retiming and Async Register slice
- RGID
- Common AXI bus protection

"RGID" is for freedom from interference (FFI). The others are for fault detection. For "Common AXI bus protection", please refer to Section 18 AXI Bus.

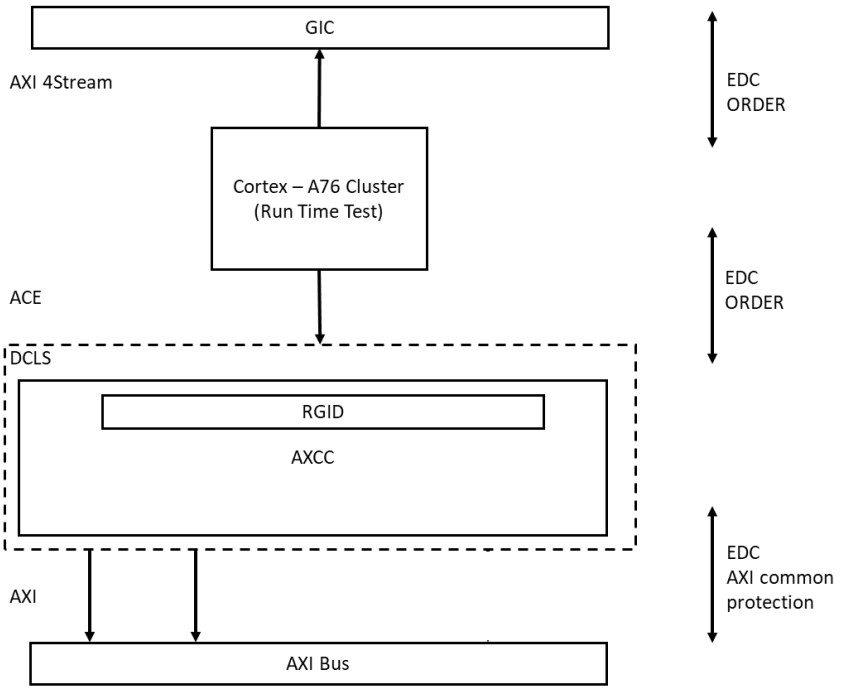



**Figure 5.31 Block diagram of AP system core safety mechanisms**

#### 5.5.3.2 Hardware Description

Table 5.22 shows summary of Safety Mechanisms.

Table 5.22 Safety mechanism list

| Safety Mechanism | Target module |
|---|---|
| Runtime Test | Cortex-A76 Cluster |
| DCLS | AXCC |
| BUS EDC | ACE/AXI4Stream/AXI |
| ORDER check | ACE/AXI4Stream/AXI |
| RGID | Bus Region ID protection |

Fault point and detection functions are following:

- Failures Runtime Test area of the cluster are detected by Runtime Test.
- Failure within the DCLS area of the AXCC is detected by DCLS.
- Failures inside ACE/AXI4Stream/AXI are detected by EDC.
- Infringement between different ASILs is protected by RGID.

#### 5.5.3.3 Usage Notes

When changing the RGID setting, all of the CPU cache shall be flushed.

### 5.5.4 Generic Counter

#### 5.5.4.1 Overview

Figure 5.33 shows safety mechanism in Generic Counter. Hatching modules in the figure are safety mechanism.

The list of safety mechanisms in the Generic Counter is as follows.

- Hardware Redundancy for Generic Counter
- Hardware Redundancy for Generic Counter Interface
- EDC for Bus protection

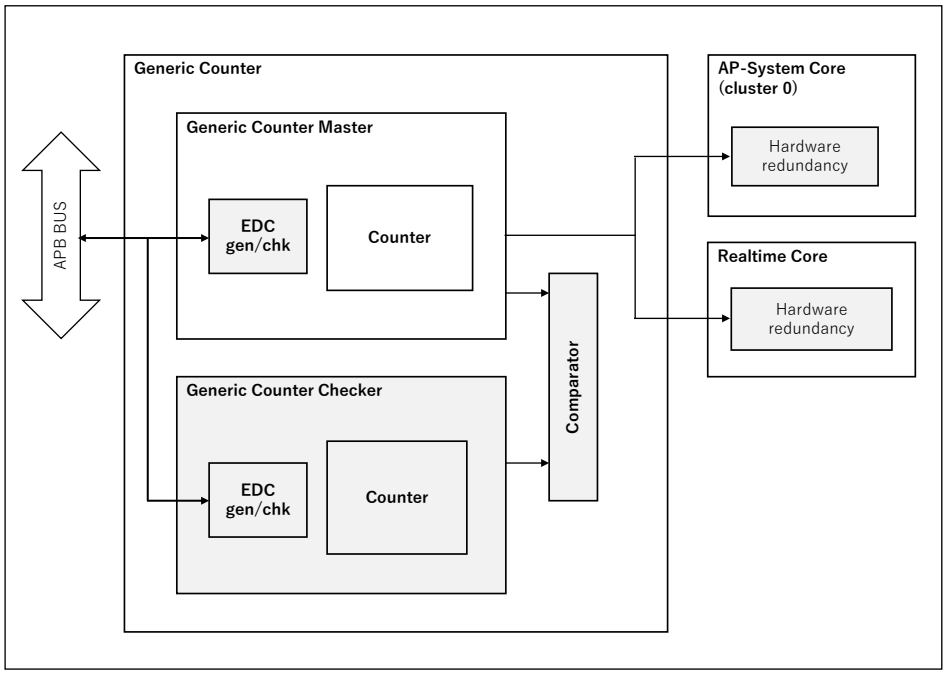



Figure 5.33 Generic Counter block diagram for safety mechanism

### 5.5.4.2 Hardware Redundancy for generic counter

#### (1) Hardware Description

Redundant logic exists in the generic counter in order to check the malfunction of this module. Figure 5.34 shows hardware redundancy for Generic Counter. When an error related to this is detected, the error is reported to the ECM modules.

Even if an error is detected in the counter logic circuit, the counter continues to count up.

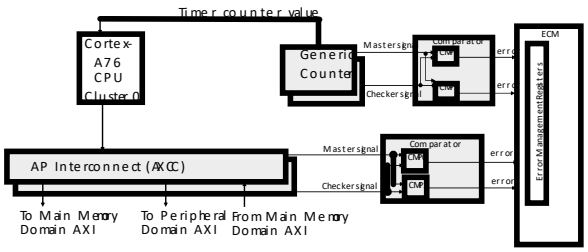



Figure 5.34 DCLS for AP-Syste

### 5.5.4.3 Hardware Redundancy for Generic Counter Interface.

Refer to section 5.5.1.8 for detail of hardware redundancy for generic counter interface.

### 5.5.4.4 Bus Protection

#### (1) Hardware Description

Figure 5.35 shows a block diagram of bus protection for Generic Counter. Address and data signals are protected by EDC protection for DED scheme. EDC checker (EDCchk) and EDC generator (EDCgen) are implemented in both Generic Counter Master and Generic Counter Checker, and the error in EDCgen or EDCchk can be detected as hardware redundancy error for Generic Counter.

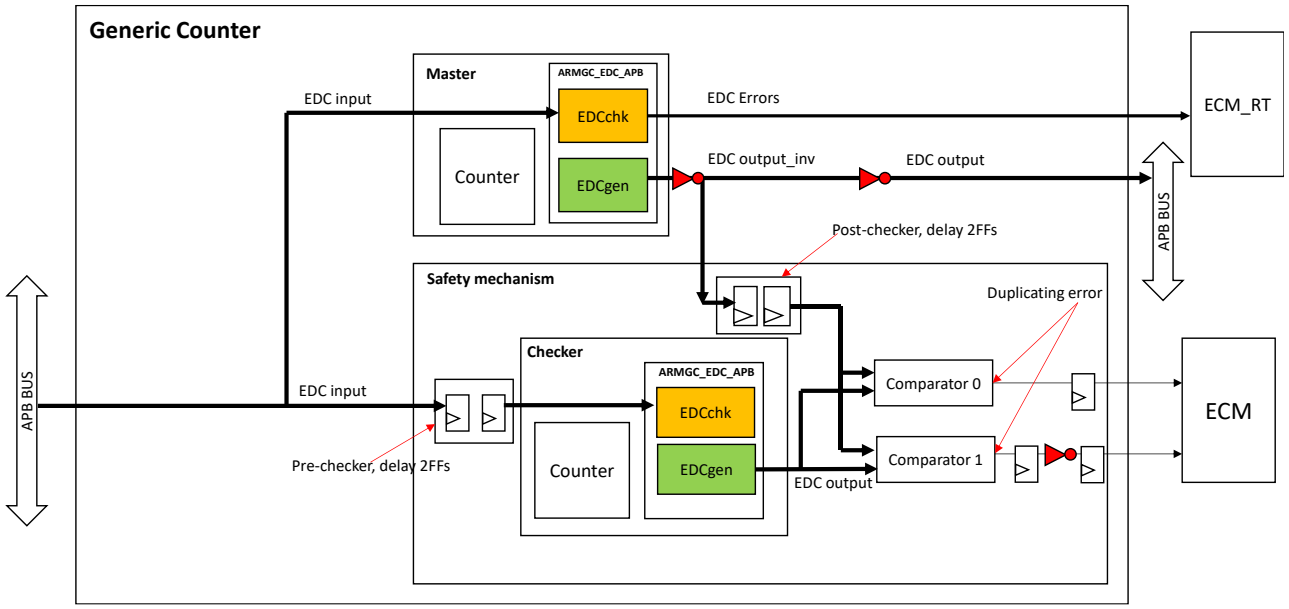



**Figure 5.35 Bus protection for Generic counter**

# 6. Realtime Core

## 6.1 Overview

The Realtime core is a core block equipped with Arm® Cortex®-R52. For details on the functions of Cortex-R52, see the relevant Technical Reference Manual.

### 6.1.1 Features

Table 6.1 shows the feature of the Realtime Core.

Table 6.1 Feature of the Realtime Core Block (Cortex-R52)

| | R-Car V4M |
|---|---|
| **Version** | r1p4 |
| **Cluster ID** | V4M-7,5,3: Cluster 0, Cluster 1, Cluster 2<br>V4M-2: Cluster 0, Cluster 1 |
| **Number of CPUs** | V4M-7,5,3: 3 x CPU with dual core lock step<br>V4M-2: 2 x CPU with dual core lock step |
| **Number of Clusters** | V4M-7,5,3: 3 x Cluster<br>V4M-2: 2 x Cluster |
| **L1 cache**<br>(per each Cortex-R52 CPU cluster) | I$ = 32KB (8 KB × 4 way, 64-byte lines) [w/ECC]<br>D$ = 32KB (8 KB × 4 way, 64-byte lines) [w/ECC] |
| **L2 cache** | None |
| **TCM support** | ATCM 16KB [w/ECC]<br>BTCM 16KB [w/ECC]<br>CTCM 16KB [w/ECC] |
| **ECC** | Supported |
| **MPU** | 24 regions |
| **NEON™/VFP** | NEON supported (Advanced SIMD)<br>VFP supported |
| **Operating clock** | CPU core:<br><br>V4M-7: 1.4 GHz<br>V4M-5: 1.1 GHz<br>V4M-3: 0.9GHz<br>V4M-2: 0.9GHz<br>(V4M-7,5,3: ZR0ϕ for Cluster 0, ZR1ϕ for Cluster 1, ZR2ϕ for Cluster 2<br>V4M-2: ZR0ϕ for Cluster 0, ZR1ϕ for Cluster 1) |
| | AXI bus:<br>400 MHz (S0D2ϕ) |
| **Low power mode** | Sleep mode (Clock stop by WFI)<br>Core standby mode (Before boot)<br>Core powerdown mode (A3RC* power domain is OFF) |
| **Private peripherals** | General Interrupt Controller, Generic Timer, Write buffer |

Note: For details, refer to Reference [1].

### 6.1.2 Block Diagram

Figure 6.1 shows the block diagram of Realtime Core.

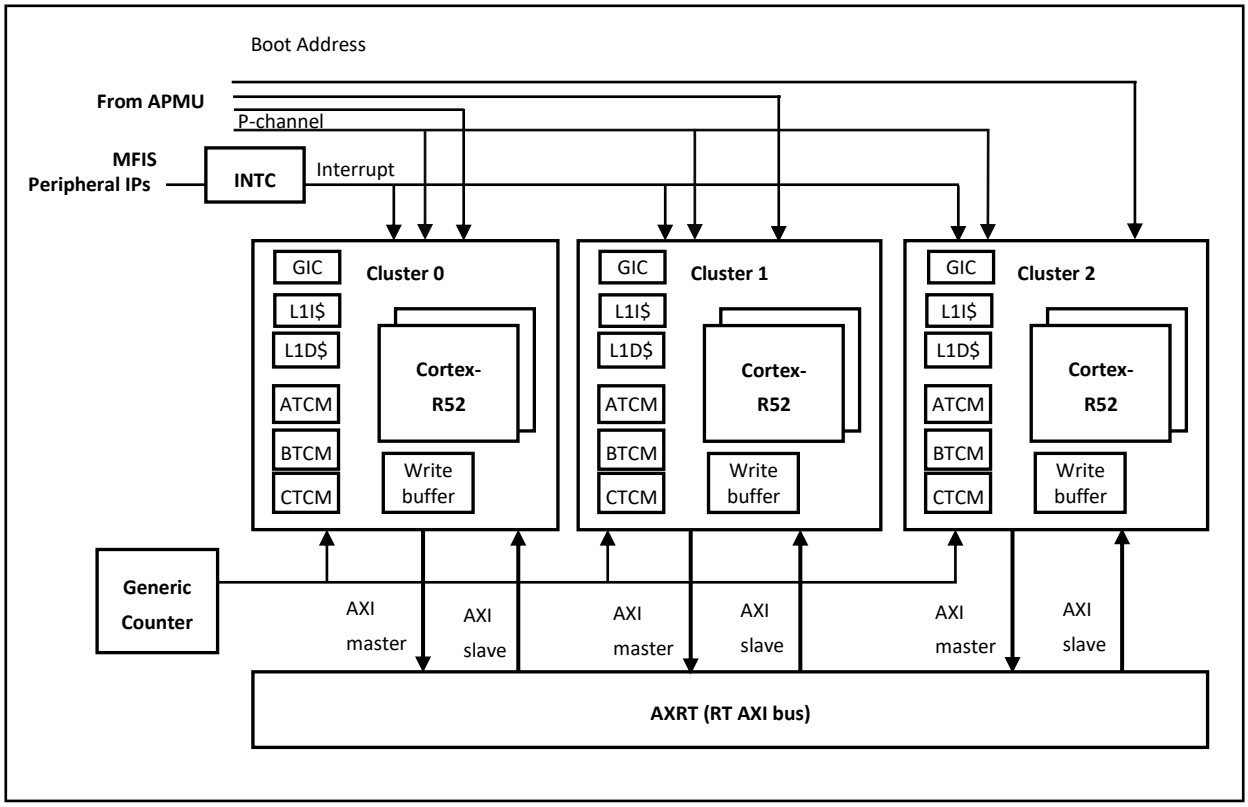



Figure 6.1 Realtime Core block diagram

### 6.1.3 External Pins

There are no external pins in this module.

### 6.1.4 Register Configuration

Table 6.2 shows the register address map of the Realtime Core. Only the access to the address defined below is allowed. Those areas are accessible only by the own Realtime Core. For example, the registers of the Cortex-R52 CPU core 1 can be accessed by the Cortex-R52 CPU core 1, only.

Table 6.2 Address Map of Realtime Core

| Address Range | Target module | Note |
|---|---|---|
| 0xF000_0000 – 0xF01F_FFFF | Generic Interrupt Controller (GIC) | For details, refer to section 9 in Reference [1] |
| 0xF020_0000 – 0xF020_0FFF | Write Buffer | - |
| 0xE628_0000 – 0xE62B_FFFF | APSREG | - |

As for the Register configuration of Write Buffer registers, refer to R-CarV4M_UM_006_Realtime_Core_r0p41.xlsx.

### 6.1.5 Connected Module

Table 6.3 shows the connected modules to the Realtime Core. As for the features of the connected modules, refer to the chapter for each module.

Table 6.3 Connected Modules

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| Realtime Core | CPG | Clock control, power management |
| | SYSC | Power management |
| | RST | Reset |
| | ECM | Error Control Management |
| | APMU | Advanced Power Management Unit |
| | AXRT | RT(Real Time) domain AXI |
| | Generic Counter (AP-System Core) | Generic Counter |
| | Debug and Trace | Debug |
| | MFIS | Multifunctional interface |

### 6.1.6 Reference document

[1] Arm Cortex-R52 Processor Technical Reference Manual Revision r1p4

[2] ARM® Architecture Reference Manual Supplement ARMv8, for the ARMv8-R AArch32 architecture profile, A.b

[3] AMBA AXI and ACE Protocol Specification.

[4] AMBA Low Power Interface Specification

## 6.2 Register Description

Refer to R-CarV4M_UM_006_Realtime_Core_r0p41.xlsx

## 6.3 Operation

### 6.3.1 Boot Address Setting

The boot address of Realtime Core can be controlled by Boot Address Register (BAR) in Advanced Power Management Unit (APMU) module. About the initial value of these registers, refer to Section 10, Advanced Power Management Unit (APMU).

Notice: APMU supports Reset Vector Base Address Register (RVBAR) that can also change the boot address of Realtime Core. However, in R-Car product, it must not be used to avoid confliction with BAR.

By default (MPU OFF), the area from H'8000_0000 to H'FFFF_FFFF is Execute-never. Therefore, we cannot use RVBAR for Secure boot whose boot address is H'EB10_0000. If boot address is from H'0000_0000 to H'7FFF_FFFF, software can use RVBAR. But in that case, please do not use BAR.

### 6.3.2 MPU (Memory Protection Unit) setting

Table 6.4, Table 6.5, Table 6.6 and Table 6.7 show the default memory map of Realtime Core. In the case of boot from Cortex-R52, MPU is set by the code in mask ROM as shown in Table 6.8.

**Table 6.4 Default Memory Map in Hypervisor mode (EL2) of instruction access (MPU OFF)**

| Address Range | Instruction caching (HSCTLR.I) | | Execute-never (XN) |
|---|---|---|---|
| | Enabled | Disabled | |
| H'0000_0000 to H'7FFF_FFFF | Normal memory, Non-sharable, Write-Through Cacheable | Normal memory, Non-cacheable | Instruction execution permitted |
| H'8000_0000 to H'FFFF_FFFF | - | - | Execute-never |

Note: For details, refer to Reference [1].

**Table 6.5 Default Memory Map in Hypervisor mode (EL2) of data access (MPU OFF)**

| Address Range | Data caching (HSCTLR.C) | |
|---|---|---|
| | Enabled | Disabled |
| H'0000_0000 to H'3FFF_FFFF | Normal memory, Non-sharable, Write-Back, Write-Allocate Cacheable | Normal memory, Shareable, Non-cacheable |
| H'4000_0000 to H'5FFF_FFFF | Normal memory, Non-sharable, Write-Through Cacheable | Normal memory, Shareable, Non-cacheable |
| H'6000_0000 to H'7FFF_FFFF | Normal memory, Sharable, Non-cacheable | Normal memory, Shareable, Non-cacheable |
| H'8000_0000 to H'BFFF_FFFF | Device-nGnRE | Device-nGnRE |
| H'C000_0000 to H'FFFF_FFFF | Device-nGnRnE | Device-nGnRnE |

Note: For details, refer to Reference [1].

**Table 6.6 Default Memory Map in EL1 of instruction access (MPU OFF)**

| Address Range | Instruction caching (SCTLR.I) | | Execute-never (XN) |
|---|---|---|---|
| | Enabled | Disabled | |
| H'0000_0000 to H'7FFF_FFFF | Normal memory, Non-sharable, Write-Through Cacheable | Normal memory, Non-cacheable | Instruction execution permitted |
| H'8000_0000 to H'FFFF_FFFF | - | - | Execute-never |

Note: For details, refer to Reference [1].

**Table 6.7 Default Memory Map in EL1 of data access (MPU OFF)**

| Address Range | Data caching (SCTLR.C) | |
|---|---|---|
| | Enabled | Disabled |
| H'0000_0000 to H'3FFF_FFFF | Normal memory, Non-sharable, Write-Back, Write-Allocate Cacheable | Normal memory, Shareable, Non-cacheable |
| H'4000_0000 to H'5FFF_FFFF | Normal memory, Non-sharable, Write-Through Cacheable | Normal memory, Shareable, Non-cacheable |
| H'6000_0000 to H'7FFF_FFFF | Normal memory, Sharable, Non-cacheable | Normal memory, Shareable, Non-cacheable |
| H'8000_0000 to H'BFFF_FFFF | Device-nGnRE | Device-nGnRE |
| H'C000_0000 to H'FFFF_FFFF | Device-nGnRnE | Device-nGnRnE |

Note: For details, refer to Reference [1].

Table 6.8 Memory Map (after ROM boot)

| Address Range | Memory Attribute | Instruction Execution |
|---|---|---|
| H'0000 0000 to H'3FFF FFFF | Normal memory, Non-sharable, Write-Back, Read/Write allocate | Instruction execution permitted |
| H'4000 0000 to H'BFFF FFFF | Normal memory, Non-sharable, Write-Back, Read/Write allocate | Instruction execution permitted |
| H'C000 0000 to H'DFFF FFFF | Device-nGnRnE, Non-sharable | Execute Never |
| H'E000 0000 to H'E3FF FFFF | Normal memory, Non-sharable, Write-Back, Read/Write allocate | Instruction execution permitted |
| H'E400 0000 to H'E4FF FFFF | Normal memory, Non-sharable, Write-Back, Read/Write allocate | Instruction execution permitted |
| H'E500 0000 to H'E62F FFFF | Device-nGnRnE, Non-sharable | Execute Never |
| H'E630 0000 to H'E63F FFFF | Normal memory, Non-sharable, Write-Back, Read/Write allocate | Instruction execution permitted |
| H'E640 0000 to H'EB0F FFFF | Device-nGnRnE, Non-sharable | Execute Never |
| H'EB10 0000 to H'EB12 7FFF | Normal memory, Non-sharable, Write-Back, Read/Write allocate | Instruction execution permitted |
| H'EB12 8000 to H'EB1F FFFF | Device-nGnRnE, Non-sharable | Execute Never |
| H'EB20 0000 to H'EB2F FFFF | Normal memory, Non-sharable, Write-Back, Read/Write allocate | Instruction execution permitted |
| H'EB30 0000 to H'FFFF FFFF | Device-nGnRnE, Non-sharable | Execute Never |

### 6.3.3 Power-Up/Down Mechanism

The Realtime Core supports the following five power modes.

- Core Standby mode
- Sleep mode (clock stop)
- Normal operation mode
- Core powerdown mode (A3CR* domain power is off)
- Debug over Core powerdown mode

Core Standby mode is a power mode before Cortex-R52 CPU core is booted. Cortex-R52 CPU core 0 is in Core Standby mode only in ICUMX boot. Other Cortex-R52 CPU cores are in Core Standby mode before they are woken up. Each Cortex-R52 CPU core is under reset and there is no operation in Core Standby mode. The Cortex-R52 CPU core leaves Core Standby mode when Cortex-R52 CPU core is woken up by ICUMX, AP-System Core or other Cortex-R52 CPU cores. After leaving Core Standby mode, it is impossible to go back to this mode.

Cortex-R52 CPU cores enter Sleep mode when CPU executing WFI instruction. Clock to CPU is stopped to save the power consumption until CPU receives a wake-up request.

In Core powerdown mode, CPU core power domain (A3RC* power domain) is powered-off. Core powerdown mode has less power than Sleep mode. Debug over Core powerdown mode is for debug Core powerdown mode.

Refer to section 10.3.2.5 for power control operation.

### 6.3.4 Write Buffer

The Write Buffer is a buffer equipped between Cortex-R52 core and AXI master port. Under the following [Condition], the write requests which the target address are successive can be merged by using the write buffer. Write buffer has three merge slots. Each merge slot can store up to 128-byte data. Figure 6.2 shows the block diagram of the Write Buffer.

**[Condition]**

A write request which fulfills both conditions:

- Access to Normal memory (Not to Device memory) (AWCACHE[1] = B'1)
- Normal access (Not Exclusive access) (AWLOCK = B'0)

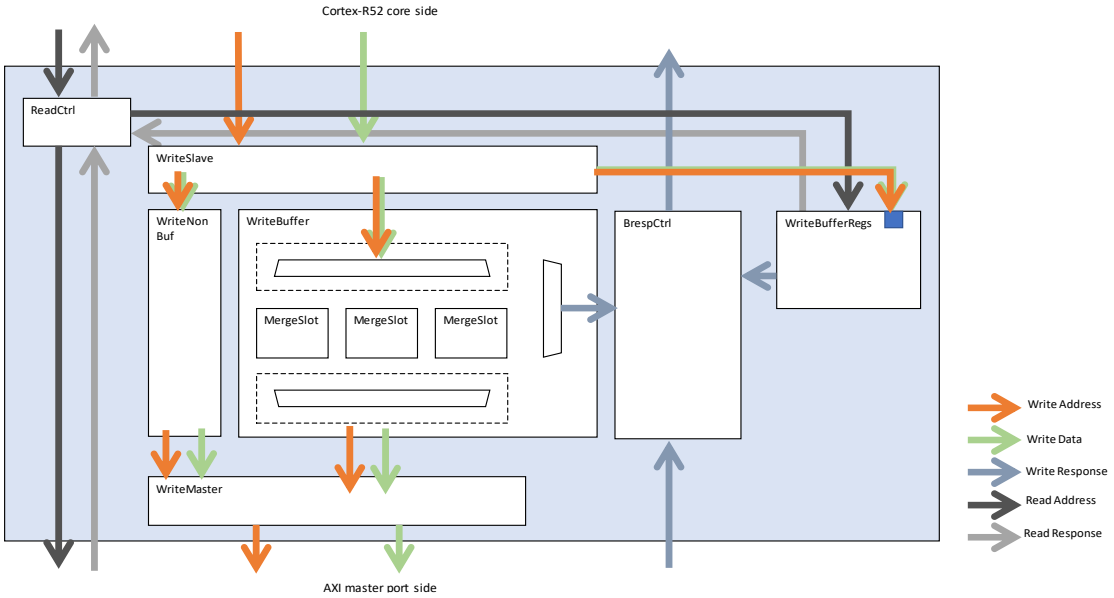



Figure 6.2 Block Diagram of Write Buffer

Once the write data is stored to the write buffer, the write response is returned to the bus master interface of Realtime Core before receiving the response from the target modules. When the corresponding response from the target modules are returned with the bus error attribute (DECERR/SLVERR), the interrupt related to this error can be asserted and notified to ECM.

To enable the write buffer, Realtime Core sets 1'b1 to WBCTLR register. For data eviction operation, there are two cases. In the first case, automatic eviction occurs for each slot individually when the slot becomes full or timeout occurs. Where, timeout period is 256 cycles of CPU clock (ZRϕ) after the last data is stored into the slot. In the second case, Realtime Core sets 1'b1 to WBSYNCR register to execute memory barrier operation accompanied with the data eviction from all slots in the write buffer. In order to complete the write transactions when the write buffer is used, data eviction operation from the write buffer is needed.

For details of the registers related to the write buffer, refer to Section 6.2.

Note: The data cache in Cortex-R52 always has Write-Though behavior. Enabling the write buffer might help AXI master write performance improvement.

### 6.3.5 Private Peripheral Modules

The private peripheral modules in Realtime Core are operated at ZR0ϕ, ZR1ϕ and ZR2ϕ.

#### 6.3.5.1 Base Address

Base address of private peripheral modules is defined by hardware. In R-Car V4H products, the base address is “H’F000 0000”.

### 6.3.6 Configuration signals setting

Table 6.9 shows the configuration signals setting of Cortex-R52. For details of these signals, refer to Section A.4 (Configuration inputs) in Reference [1].

Table 6.9 Configuration signals setting (Cortex-R52)

| Configuration Signal | Value | Description |
|---|---|---|
| CFGENDIANESS0 | B'0 | Little endian boot. |
| CFGAXISTCMBASEADDR[31:24] | H'E4 | Base address of the TCMs on system address map (access via AXIS IF). |
| CFGCLUSTERUTID[1:0] | B'00<br>B'01<br>B'02 | Cluster unique transaction identifier for the purpose of interconnect protection (Not used).<br>Cluster 0: B'00<br>Cluster 1: B'01<br>Cluster 2: B'02 |
| CFGDBGROMADDRV | B'1 | Top-level debug ROM table address enable. |
| CFGDBGROMADDR[31:12] | H'F8000 | Top-level debug ROM table base address on system address map. |
| CFGFLASHBASEADDR[31:27] | B'00000 | Base address of the Flash interface (No Flash memory). |
| CFGFLASHEN0 | B'0 | Flash interface is disabled (No Flash memory). |
| CFGFLASHPROTEN | B'0 | Flash memory protection is disabled (No Flash memory). |
| CFGFLASHPROTIMP | B'0 | Flash memory protection is not support (No Flash memory). |
| CFGFLASHIMP | B'0 | Flash region is not implemented in memory map (No Flash memory). |
| CFGLLPPIMP | B'0 | LLPP region is not implemented in memory map. |
| CFGLLPPBASEADDR[31:12] | H'00000 | Base address of the LLPP (Not used). |
| CFGLLPPSIZE[3:0] | H'0000 | Region size of the LLPP (Not used). |
| CFGMRPEN | B'0 | Memory reconstruction port is disabled. |
| CFGMPIDRAFF1[7:0] | H'00<br>H'01<br>H'02 | Cluster ID at affinity level 1.<br>Cluster 0: H'00<br>Cluster 1: H'01<br>Cluster 2: H'02 |
| CFGMPIDRAFF2[7:0] | H'00 | Cluster ID at affinity level 2. |
| CFGPERIPHBASE[31:21] | H'780 | Base address of the GIC Distributor Unit (GDU) registers is 0xF000_0000. |
| CFGRAMPROTEN | 0/1 | RAM (caches and TCMs) memory protection enable out of reset. This port can be controlled by APMU's CR52CR register. Default is 1'b1. |
| CFGTCMBOOT0 | 0/1 | ATCM enabled and at address 0x0 out of reset. This port can be controlled by APMU's CR52CR register. Default is 1'b0. |
| CFGTHUMBEXCEPTIONS0 | 0 | Instruction set state is A32 at power on reset. |
| CFGVECTABLE0[31:5] | Any | Vector table base address out of reset. This port can be controlled by APMU's RVBAR register. Default is 0x0. |

Note: For details of APMU registers, refer to Section 10, Advanced Power Management Unit (APMU).

## 6.4 Usage Notes

### 6.4.1 Interrupt handling for Realtime Core

Cortex-R52 does not support NMI (Non-Maskable Interrupt). EXTPPI[0] is used to connect NMI pin from INTC-EX. Table 6.10 shows ID number.

Table 6.10 EXTPPI (not allocated PPI) assignment

| EXTPPI | ID number | Description |
|---|---|---|
| EXTPPI[0] | ID16 | NMI interrupt from INTC-EX |
| EXTPPI[5:1] | ID17 – ID21 | Not used |
| EXTPPI[7:6] | ID28 – ID29 | Not used |
| EXTPPI[8] | ID31 | Not used |

For SPI (Shared Peripheral Interrupts), SGI (Software Generated Interrupts), and allocated PPI (Private Peripheral Interrupts), refer to Reference [1] – Section 9. About SPI number of each module, refer to Section 15.4.2.

### 6.4.2 Accessing to TCM regions

Notice: During Cortex-R52's reset is asserted, accessing TCM is not allowed. Before Cortex-R52's reset is asserted, accessing TCM has to be stopped. Before accessing TCM is started, checking Cortex-R52's reset negating is necessary.

#### 6.4.2.1 Accessing to TCM regions by CPU

Realtime Core can access to its TCMs directly. Software has to set the base address, size of each TCM and enable it by writing to IMP_ATCMREGIONR, IMP_BTCMREGIONR, IMP_CTCMREGIONR. Base address must be size-aligned. Separate enable bits are provided to enable the TCMs in EL2 and/or EL1/EL0. For details, refer to Section 7.2 (TCM memory) in Reference [1].

#### 6.4.2.2 Accessing to TCM regions via AXIS interface

The TCM address range (16MB region) defined in system address map is from H'E400 0000 to H'E4FF FFFF. Table 6.11 describes the valid TCM regions for each Cortex-R52 (Core 0, Core 1, Core 2). Those regions are accessed via AXIS port. Accessing to reserved areas will cause DECERR.

Table 6.11 TCM region address

| TCM region | Address range | Size | Target |
|---|---|---|---|
| ATCM | H'E400 0000 to H'E400 3FFF | 16KB | Cortex-R52 CPU core 0 |
| Reserved | H'E400 4000 to H'E40F FFFF | - | |
| BTCM | H'E410 0000 to H'E410 3FFF | 16KB | Cortex-R52 CPU core 0 |
| Reserved | H'E410 4000 to H'E41F FFFF | - | |
| CTCM | H'E420 0000 to H'E420 3FFF | 16KB | Cortex-R52 CPU core 0 |
| Reserved | H'E420 4000 to H'E42F FFFF | - | |
| Reserved | H'E430 0000 to H'E43F FFFF | - | |
| ATCM | H'E440 0000 to H'E440 3FFF | 16KB | Cortex-R52 CPU core 1 |
| Reserved | H'E440 4000 to H'E44F FFFF | - | |
| BTCM | H'E450 0000 to H'E450 3FFF | 16KB | Cortex-R52 CPU core 1 |
| Reserved | H'E450 4000 to H'E45F FFFF | - | |
| CTCM | H'E460 0000 to H'E460 3FFF | 16KB | Cortex-R52 CPU core 1 |
| Reserved | H'E460 4000 to H'E46F FFFF | - | |
| Reserved | H'E470 0000 to H'E47F FFFF | - | |
| ATCM | H'E480 0000 to H'E480 3FFF | 16KB | Cortex-R52 CPU core 2 |
| Reserved | H'E480 4000 to H'E48F FFFF | - | |
| BTCM | H'E490 0000 to H'E490 3FFF | 16KB | Cortex-R52 CPU core 2 |
| Reserved | H'E490 4000 to H'E49F FFFF | - | |
| CTCM | H'E4A0 0000 to H'E4A0 3FFF | 16KB | Cortex-R52 CPU core 2 |
| Reserved | H'E4A0 4000 to H'E4AF FFFF | - | |
| Reserved | H'E4B0 0000 to H'E4BF FFFF | - | |
| Reserved | H'E4C0 0000 to H'E4FF FFFF | - | |

Access control checks can be enabled for AXIS transactions by programming IMP_SLAVEPCTLR.TCMACCLVL. Access control allows either all transactions or only privileged transactions to access the TCM. If a transaction is not permitted, the AXIS generates a SLVERR response.
Privileged of transactions that access TCM can be overwritten by Cortex-R52 Secure Auxiliary Configuration Register (H'E62A 0010) bit 31 (for read transactions) and bit 30 (for write transactions). When those 2 bits are set to 1, accesses (read/write) to TCM always have privileged permission (ARPROT[0]/AWPROT[0] = 1). This register can be accessed only in secure mode.

When an external agent or core attempts to write to the TCM through the AXIS interface and there is a pending write to the same TCM line from the core locally, the AXIS interface returns a SLVERR. This is to indicate a software error because more than one agent are writing to the same TCM line (even if the bytes do not overlap) because there is no guarantee in which order the memory is updated.

**[Restriction]**
The AXIS implements a subset of the AXI transactions, the Accelerator Coherency Port Interface Restrictions, that are described in *AMBA® AXI™ and ACE™ Protocol Specification AXI3™, AXI4™, and AXI4-Lite™ ACE and ACE-Lite™.* For details of restrictions, refer to Appendix B (*Accelerator Coherency Port Interface Restrictions*) in Reference [3].

The AXIS interface supports two below burst lengths only:
- 16-byte transaction: AxLEN[7:0]=0x00, AxADDR[3:0] = 4'b0000
- 64-byte transaction: AxLEN[7:0]=0x03, AxADDR[5:0] = 6'b000000

For all other combinations of AxLEN and AxADDR, AXIS generates an error SLVERR.

Accessing TCMs with transactions whose AxSIZE[2:0] != 3'b100 and/or AxBURST[1:0] != 2'b01 is also not supported. An error will be notified to ECM when AXIS interface receives such illegal transactions. For details, refer to Section 6.5.3 in this document.

### 6.4.3 Watchdog Timer

Cortex-R52 does not support watchdog timer. Consider choosing watchdog timers outside of Cortex-R52.

### 6.4.4 Exclusive operation usage restriction

Exclusive operation using LDREX and STREX instruction pair is supported only in RT-VRAM area, SDRAM area (H'4000 0000 to H'BFFF FFFF) and System RAM area. Exclusive operation to other areas is not supported.

NOTE: Because Cortex-A76+AXCC changes ARch transaction ID and AWch transaction ID, so please don't use Store-Exclusive and Load-Exclusive instruction to access RT-VRAM from Cortex-A76. But Cortex-R52 has no this kind of issue. So Cortex-R52 can use Store-Exclusive and Load-Exclusive instruction to access RT-VRAM

### 6.4.5 AxPROT[1] overwrite function on AXI Master interface

Cortex-R52 always issues secure transactions (AxPROT[1] = 0) to AXI Master interface. Realtime Core has feature to change from secure transactions to Non-secure transactions (AxPROT[1] = 1) by writing 0 to register (address 0xFFC43018). This register can be changed from 1 (default value) to 0 only. Before changing this register from 1 to 0, it's mandatory to flush all data in Cortex-R52 Store Buffer and in Write Buffer to external memory. Software can use "DSB" instruction to flush all data in Cortex-R52 Store Buffer, then use Write Buffer Sync Operation Register to flush all data in Write Buffer to external memory. Refer to Section 6.2.8.

### 6.4.6 Prevention of unexpected phenomenon due to CPU speculative access

For Cortex-R52, configure MPU before enabling MPU and caches by HSCTLR or SCTLR register. For MPU setting, set Device Memory and Execute Never to the speculative access prohibited regions*1. Until finish of these MPU setting and enabling, set 1 (Stops supply of the clock signal to the corresponding module.) to PCIE related bits in Module Stop Control Register (MSTPCR). Until finish of these MPU setting and enabling, PCIe related bits in AXI-Bus ICISTPINTENHSC0 Register may be 1 (Error flag set by access to the corresponding module when clock is stopped). Then clear these bits to 0 after finish of these MPU setting and enabling.

Note *1: Other than RAM/ROM regions that are permitted to be accessed from the CPU and can access randomly.

## 6.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 6.5.1 Overview

Realtime Core system is equipped with various Safety Mechanism. Figure 6.3 shows Safety Mechanisms implemented in Realtime Core.

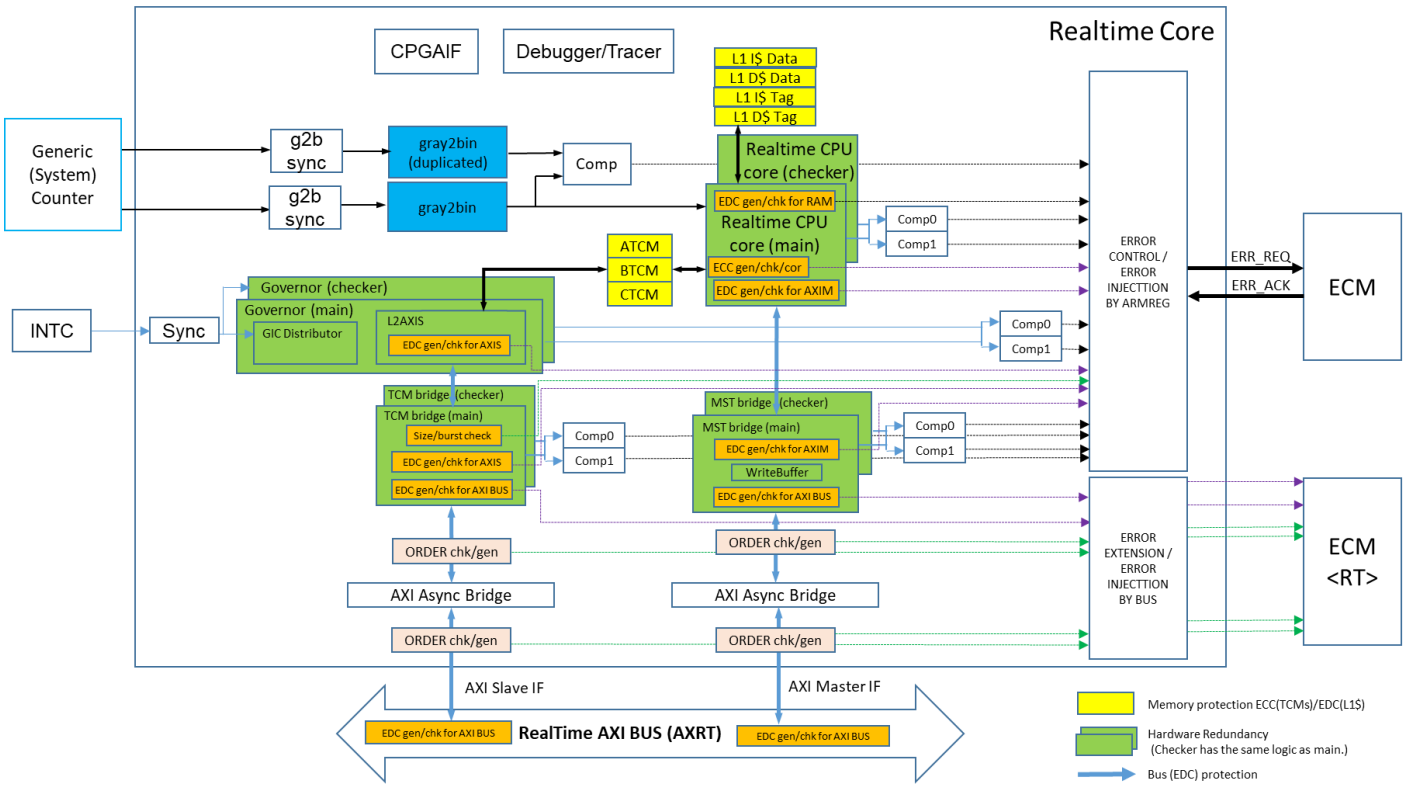



Figure 6.3 Realtime Core block diagram with Safety Mechanisms

### 6.5.2 Error Correction Code (ECC)

#### 6.5.2.1 Overview

ECC protection is applied to the following RAMs.

- L1 D-cache Tag RAM
- L1 D-cache Data RAM
- L1 I-cache Tag RAM
- L1 I-cache Data RAM
- ATCM
- BTCM
- CTCM

#### 6.5.2.2 Related Registers

About the related registers, refer to Reference [1] and Reference [2].

#### 6.5.2.3 Hardware Description

Figure 6.4 shows the block diagram of the Realtime Core.
All L1 Cache and TCM memories in Realtime Core is implemented with ECC protection.

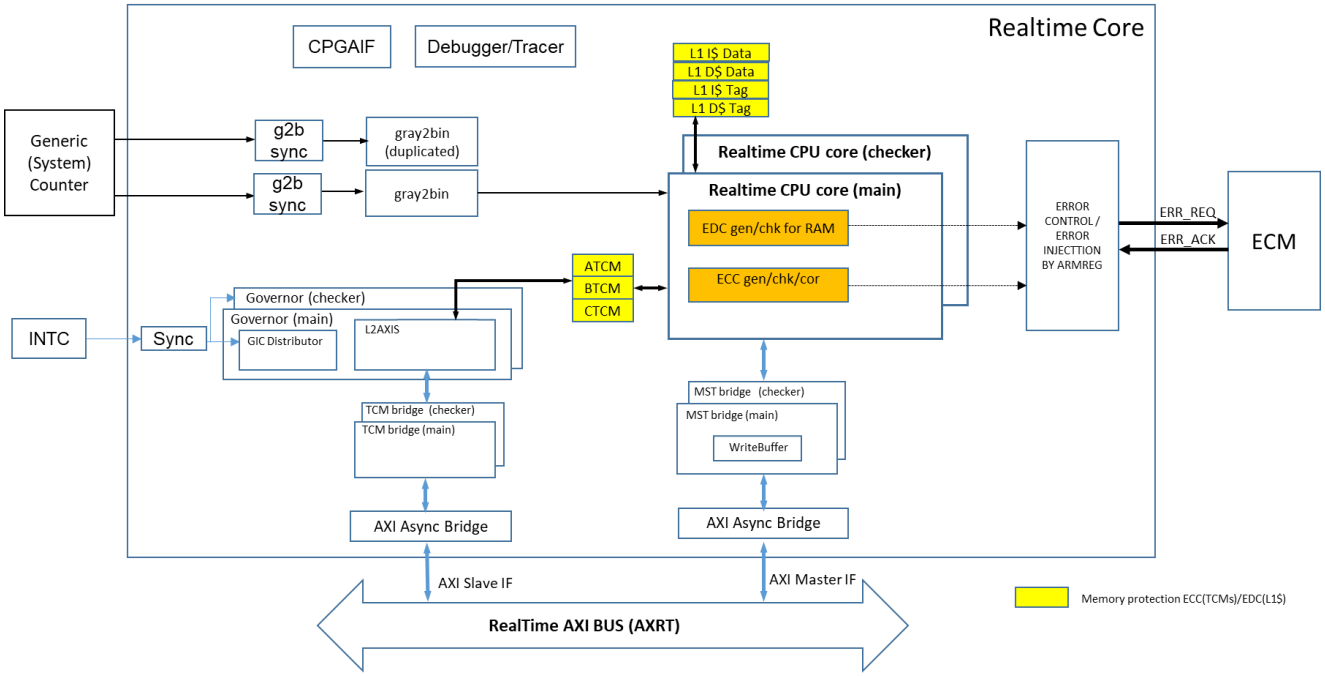



Figure 6.4 Realtime Core Memory Protection diagram

- TCMs memories are protected by SEC-DED error correction code (ECC) checking.
- L1 Cache memories are protected by DED ECC. Errors that are detected in the instruction and data cache are always correctable by invalidating the cache, flushing the core pipeline, re-fetching and executing the failed instruction. This is feasible because the caches implement Write-Through caching and can never hold

dirty data. This behavior allows the failing cache line to be read from main memory. Single-bit and double-bit hard errors are handled by cache logic to prevent repeatedly correcting the same error.

- Address decode and white noise protection are built into the ECC scheme. A fault in RAMs might also result in no lines being selected which might result in reading either all zeros or all ones. Such faults are referred to as white noise.
- TCM & L1 Cache hard error protection is implemented to avoid processor livelock.
- When an error in the RAM is detected, Realtime Core informs the ECM module of the error.

Table 6.12 ECC Protection of Realtime Core

| RAM | Protection method | Correction Behavior | Uncorrectable Error Behavior |
|---|---|---|---|
| L1 I-cache Tag RAM | DED ECC | Correction of both single and double-bit errors is achieved by invalidating the cacheline. A subsequent access fetches the data from external memory and no longer receives the error. | - |
| L1 I-cache Data RAM | DED ECC | Correction of both single and double-bit errors is achieved by invalidating the cacheline. A subsequent access fetches the data from external memory and no longer receives the error. | - |
| L1 D-cache Tag RAM | DED ECC | Correction of both single and double-bit errors is achieved by invalidating the cacheline. A subsequent access fetches the data from external memory and no longer receives the error. | - |
| L1 D-cache Data RAM | DED ECC | Correction of both single and double-bit errors is achieved by invalidating the cacheline. A subsequent access fetches the data from external memory and no longer receives the error. | - |
| ATCM | SECDED | Correct single-bit errors in each ECC chunk | Detect double bit errors in each ECC chunk and flag to ECM. |
| BTCM | SECDED | Correct single-bit errors in each ECC chunk | Detect double bit errors in each ECC chunk and flag to ECM. |
| CTCM | SECDED | Correct single-bit errors in each ECC chunk | Detect double bit errors in each ECC chunk and flag to ECM. |

### 6.5.3 Bus Protection

#### 6.5.3.1 Overview

- Signal Integrity Protection is implemented to protect AXI Interface between Realtime Core CPU & AXI bus bridges.
- Bus timeout detection is implemented in AXI-bus master Interface to detect transactions that fail to complete within a programmable time limit.
- EDC protection is implemented to protect AXI Interface between Realtime Core and RT (Realtime) domain AXI (AXRT).
- ORDER check is implemented to detect faults of FIFO read/write pointers in AXI Asynchronous bridges.
- SIZE-BUSRT check is implemented in AXI Slave interface (AXIS) to detect illegal AxSIZE/AxBURST transactions.

#### 6.5.3.2 Related Registers

About the related registers, refer to Reference [1] and Reference [2].

#### 6.5.3.3 Hardware Description

Figure 6.5 shows Bus protection implemented in Realtime Core.

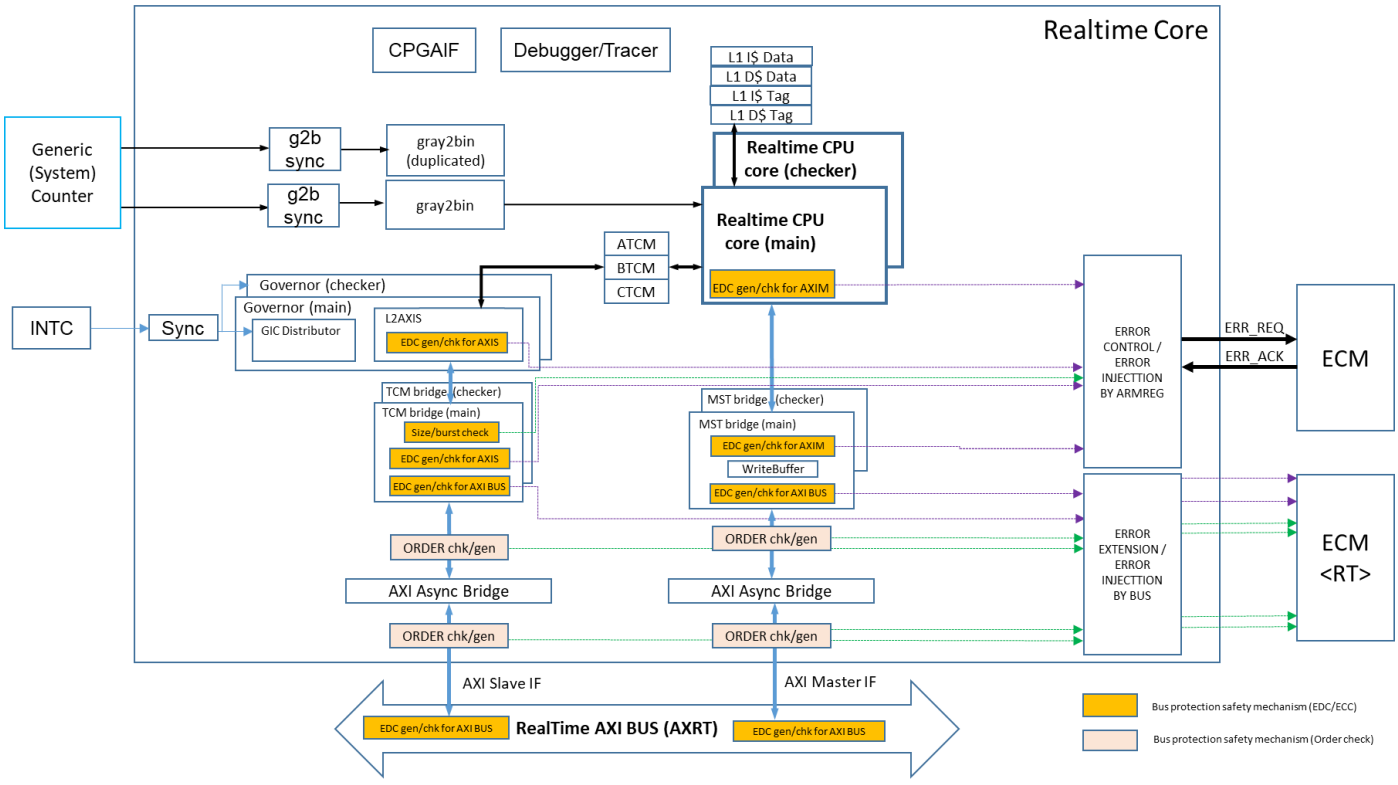



Figure 6.5 Realtime Core Bus protection diagram

**[Signal Integrity Protection]**

- Parity protection using an odd parity scheme on the handshake signals (xVALID/xREADY).
- ECC protection for DED scheme on payload signals.
- ECC protection for SECDED scheme on address and data signals.

For details, refer to Section 7.9.4 (Bus protection) in Reference [1].

When an Error is detected, Realtime Core inform ECM module of the error.

**[Bus timeout detection]**

When a transaction presented on a master port (AXIM) fails to complete, that is, a response is not received or an address or write data is not accepted resulting in a response not being received, within a programmable time limit, an error is signaled.

For details, refer to Section 7.11 (Bus timeouts) in Reference [1].

When an Error is detected, Realtime Core inform ECM module of the error.

**[ORDER check]**

Figure 6.6 shows the block diagram of ORDER check.

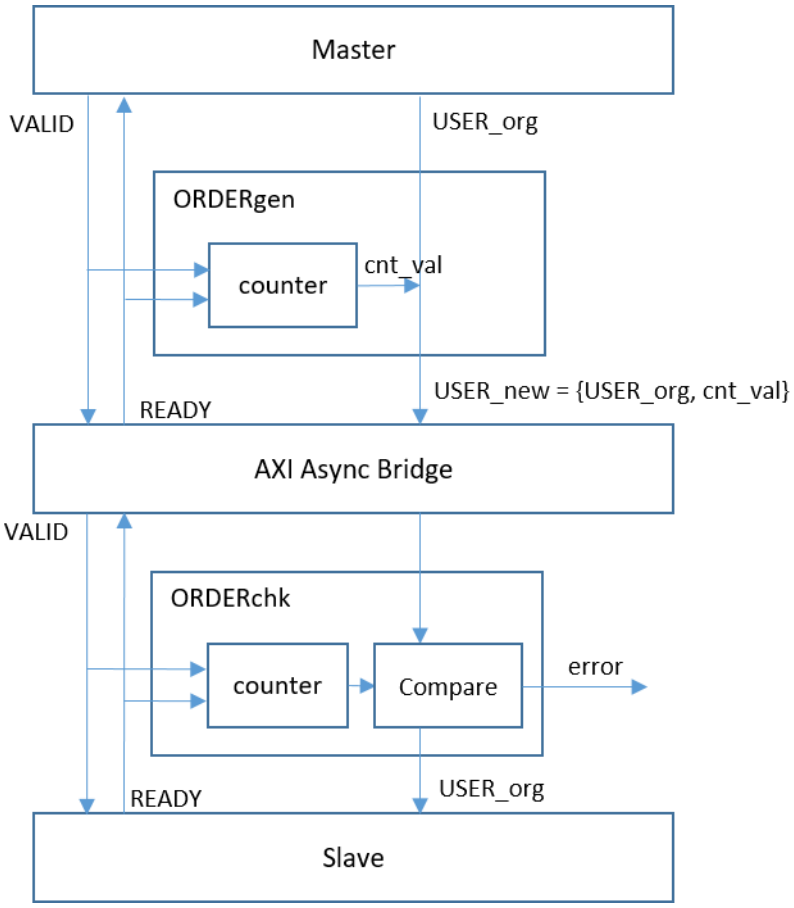



Figure 6.6 Realtime Core ORDER check diagram

- The purpose of ORDER check is to detect the fault of AXI Asynchronous bridges.
- *ORDERgen* generates an incremental counter value for each transaction. Then, the counter value is attached to AXI USER signal as a part of AXI transaction. *ORDERchk* also generates an incremental counter value for each received transaction and compare this counter value with the value attached in USER signal. If 2 counter values are different, an error is informed to AXI-bus ECM on RT domain.
- ORDER generator and ORDER checker are implemented as a pair for each AXI channel (AR/AW/W/R/B).

**[SIZE-BURST check]**

Realtime Core AXI Slave interface (AXIS) supports only a subset of AXI transactions, the Accelerator Coherency Port Interface Restrictions. Realtime Core implements a logic called SIZE-BURST check to detect the illegal transactions. When AXIS receives a transaction whose ARSIZE/AWSIZE is not 0x4 (16-byte) and/or ARBURST/AWBURST is not 0x1 (INCR), an error is informed to ECM module.

**[EDC protection between Realtime Core and RT (Realtime) domain AXI (AXRT)]**

- EDC protection is implemented to protect all AXI channels (AW/W/B/AR/R) between Realtime Core AXI Master/Slave Interfaces and AXRT.

### 6.5.4 Hardware Redundancy (DCLS)

#### 6.5.4.1 Overview

The dual core lock step (DCLS) architecture is implemented in Realtime Core.

#### 6.5.4.2 Related Registers

Table 6.13 Related Registers

| Register Name | Abbreviation |
|---|---|
| Cortex-R52 Comparator Enable Register | CR52CMPEN |

#### 6.5.4.3 Hardware Description

The function is realized by running duplicated CPU core, write buffer. The checker core has the same input of main core, and the checker write buffer has the same input of main write buffer.

When the output of main core and checker core, and/or the output of main write buffer and checker write buffer are different, an error is detected. The Realtime Core sends a notification of the error to the ECM module. To enable the DCLS operation, writing 1 to bit 0 of CR52CMPEN register is required.

Figure 6.7 shows block diagram of the Realtime Core (Cortex-R52).

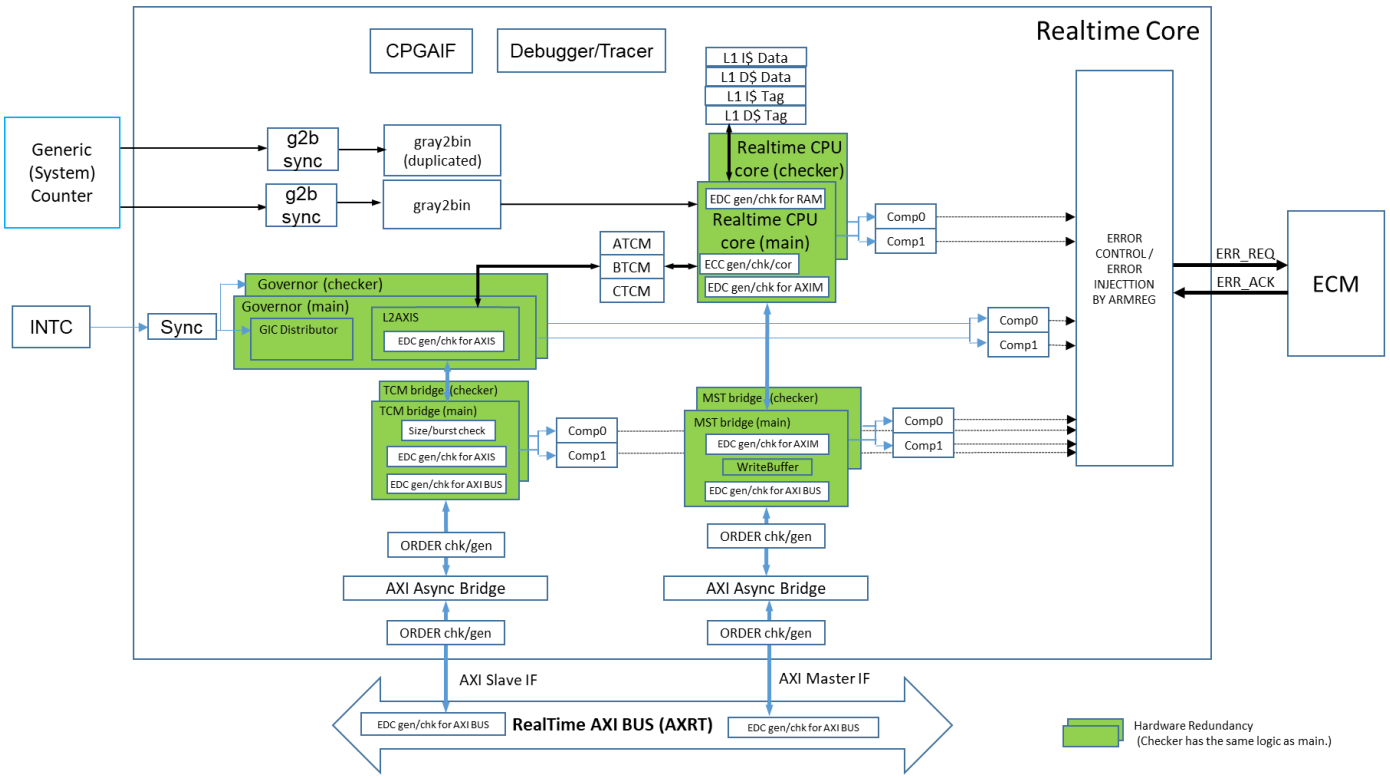



**Figure 6.7 Hardware Redundancy of Realtime Core (Cortex-R52)**

- DCLS is implemented for Realtime CPU and Write Buffer.
- EDC generators and checkers are included in the target of DCLS.
- Two comparators are implemented for each pair of Master core and Checker core. Each comparator notifys the error to ECM independently. Comparator 0 (Comp0) is put at Master while Comparator 1 (Comp1) is put at Checker core. Refer to Figure 6.8 in this document for more detail.

Table 6.14 shows register setting

**Table 6.14 Realtime Core Hardware Redundancy Register Configuration**

| Register Name | Abbreviation | R/W | Address | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|
| Cortex-R52 Comparator Enable Register | CR52CMPEN | R/W | H'E617 0328 | H'0000 0000 | 32 | √ |

### (1) Cortex-R52 Comparator Enable Register (CR52CMPEN)

This register is a 32-bit writable/readable register that controls the DCLS operation in Realtime Core. This register is accessed only in secure mode.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | NSRCMPEN | CMPEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | - | 0 | R | Reserved.<br>These bits are always read as 0.<br>The write value should always 0. |
| 1 | NSRCMPEN | 0 | R/W | Enable Non-safety related activity checking.<br>0: Disable Non-safety related activity checking.<br>1: Enable Non-safety related activity checking.<br>Notice: Please do not set this bit to 1 when debug is enabled. |
| 0 | CMPEN | 0 | R/W | Enable DCLS operation.<br>Setting this bit to enable or disable DCLS operation in Realtime Core.<br>0: Disable DCLS operation in Realtime Core.<br>1: Enable DCLS operation in Realtime Core. |

When NSRCMPEN is set to 1, Realtime Core detects the toggles of non-safety related operations and/or signals, for examples, Debug Trace data, Debug APB interface, etc. Such events are informed to ECM.
Notice:

1. Because of asynchronous reset assertion, it may cause different behavior for between Master core and Checker core when reset is asserted. Therefore, CMPEN & NSRCMPEN must be enabled after power reset (at initialization step) and must be disabled before issuing Software reset to Realtime Core.
2. NSRCMPEN shall not be set to 1 if debug is enabled. Where, debug is disabled only when mode pin setting is {MDT0, MD21, MD20, MD11, MD10} = 5'b00000. Othewise (at least one of these mode pins is set to 1), debug is enabled.

### 6.5.5 Generic Counter Interface Protection

#### 6.5.5.1 Overview

Hardware Redundancy is applied to detect if a fault of Generic Counter Interface between Generic Counter and Realtime Core.

#### 6.5.5.2 Related Registers

Table 6.15 Related Registers

| Register Name | Abbreviation |
|---|---|
| Generic Counter Error Enable Register | GCNTERREN |

### 6.5.5.3 Hardware Description

Generic Counter Interface, synchronizers & Gray to Binary converter are duplicated. The outputs of primary Gray to Binary converter and duplicated Gray to Binary converter are compared in each CPU clock cycle. If the values of two converters are different more than 1 or 3 (programmable), an error is notified to ECM.

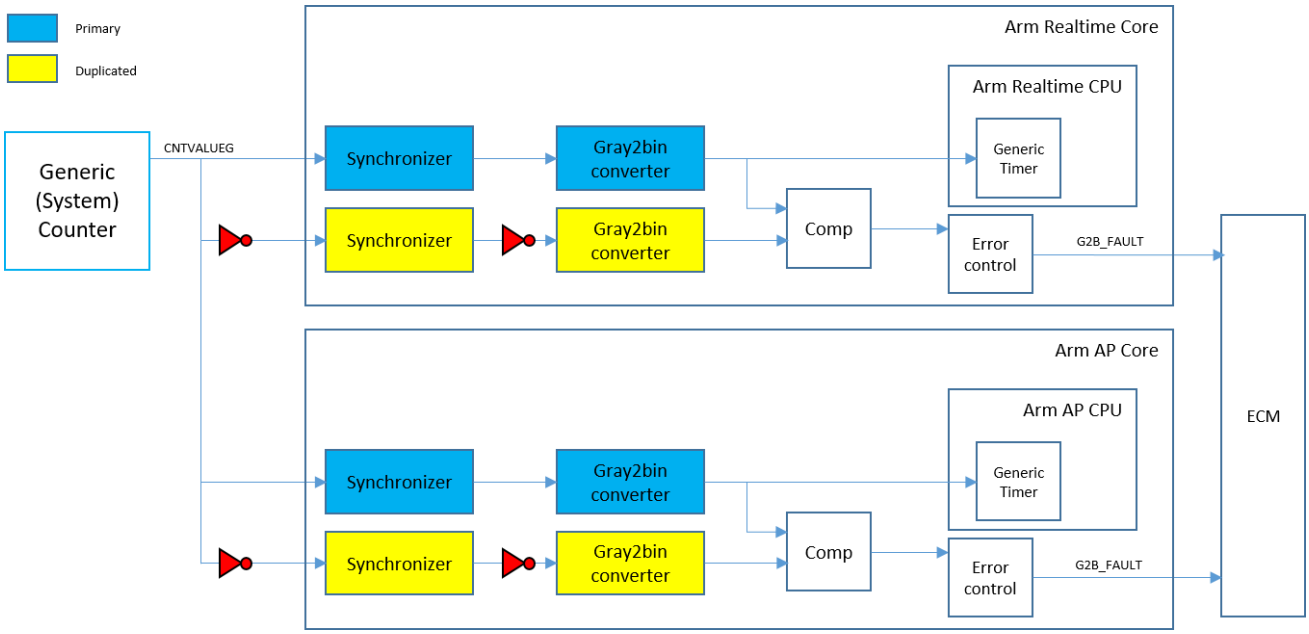



Figure 6.8 Realtime Core Generic Counter interface protection

Table 6.16 shows register setting.

Table 6.16 Realtime Core Generic Counter interface protection register configuration.

| Register Name | Abbreviation | R/W | Address | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|
| Generic Counter Error Enable Register | GCNTERREN | R/W | H'E617 032C | H'0000 0000 | 32 | √ |

#### (1) Generic Counter Error Enable Register (GCNTERREN)

This register is a 32-bit writable/readable register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | GCNTERREN |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | - | 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always 0. |
| 0 | GCNTERREN | 0 | R/W | Enable error notification of Generic Counter interface protection between Generic Counter and Realtime Core<br>0: Disable Generic Counter interface protection error<br>1: Enable Generic Counter interface protection error |

### 6.5.6 Address Feedback

#### 6.5.6.1 Overview

Cortex-R52 has method to detect the faults in the memory address decoder that are built into ECC scheme. The following is memories are applied address decode detection.

- L1 D-cache Tag RAM
- L1 D-cache Data RAM
- L1 I-cache Tag RAM
- L1 I-cache Data RAM
- ATCM
- BTCM
- CTCM

#### 6.5.6.2 Related Register

Please refer to 6.5.2.2 for more detail.

#### 6.5.6.3 Hardware Description

Figure 6.9 shows the block diagram of address decode detection in the Realtime Core.

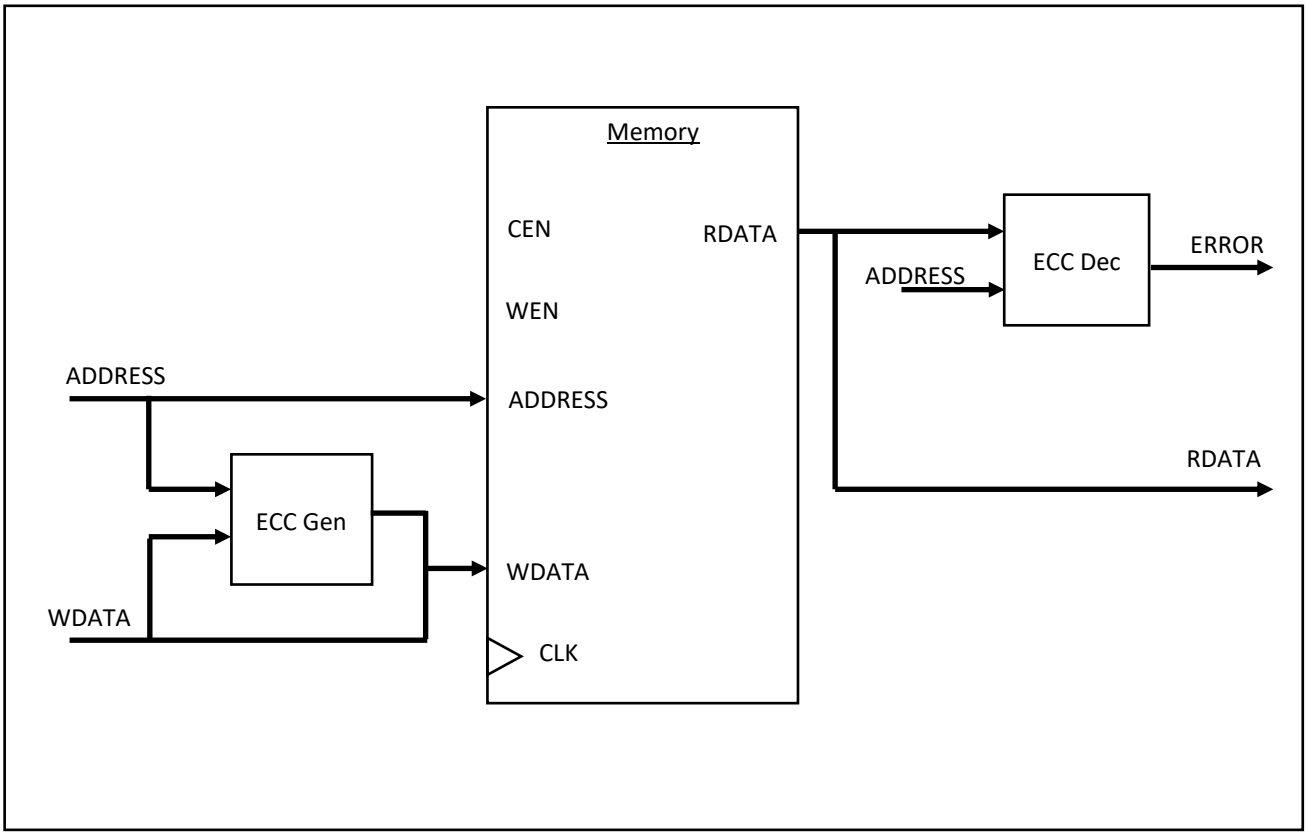



Figure 6.9 Realtime Core memory address decode detection

### 6.5.7 Other Error notification

#### 6.5.7.1 Overview

Realtime Core supports to notify several types of Errors/Events to ECM. Software can use these Errors/Events notification as needed.

- When Write Buffer is enabled, once the write data is stored to the Write Buffer, the write response is returned to the bus master interface of Realtime Core before receiving the response from the target modules. When the corresponding response from the target modules are returned with the bus error attribute (DECERR/SLVERR), the interrupt related to this error can be asserted and notified to ECM.

### 6.5.8 P-channel Interface Protection

#### 6.5.8.1 Overview

P-channel interface is used to communicate between APMU and Cortex-R52 which use to control power ON/OFF of Cortex-R52. And Safety Mechanism for this interface (P-channel checker) is implemented to APMU and Cortex-R52. Refer to Reference [4] for more detail. The block diagram is shown in Figure 6.10.

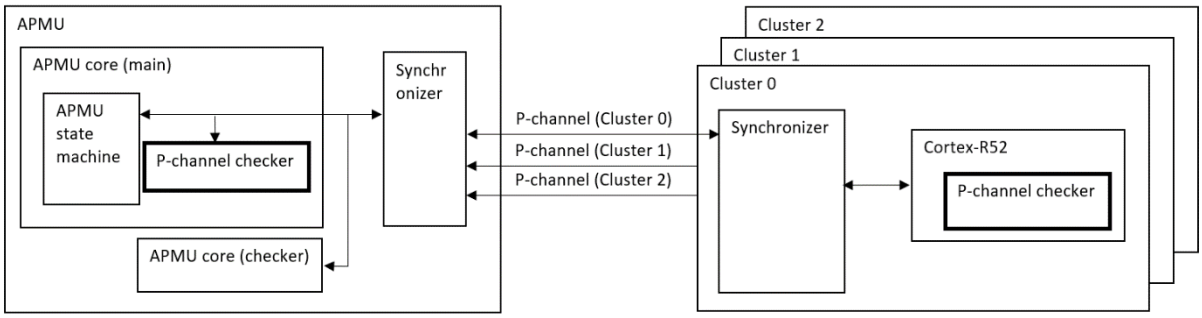



Figure 6.10 Safety Mechanism for P-channel between APMU and Cortex-R52

#### 6.5.8.2 Related Register

About the related registers, refer to Section 10 APMU.

#### 6.5.8.3 Hardware Description

This Safety Mechanism detects two types of errors. The first is parity error, the second is P-channel signal protocol error. If any of the errors are detected, Cortex-R52 notifies error detection to ECM.

1. Parity error for P-channel signal.

A parity bit is added to P-channel signal from APMU to Cortex-R52. The parity checking logic is implemented in Cortex-R52. It detects the parity error.

2. P-channel protocol error

Toggle ordering of P-channel signals is defined by Arm. Refer to Reference [4] for more detail. This mechanism detects failing of this ordering. For example, when PREQ signal from APMU is asserted, it must be kept asserted until PACCEPT or PDENY signal from Cortex-R52 core is asserted. If PREQ signal are negated before PACCEPT or PDENY signal are asserted, this mechanism judges it as P-channel protocol error.

# 7. Pin Function (PFC/GPIO)

## 7.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The pin function controller (PFC) is a module that consists of registers for selecting the function of the multiplexed pins and controlling the pull-up resistor on each LSI pin.

### 7.1.1 Features

- Setting multiplexed pin functions for LSI pins
  Function of the R-Car V4M pin selectable by setting the registers in the PFC module
- Bus Domain access: Please refer section 18 – AXI-bus
- DRV control for each LSI pin
  DRV control resistors can control the driving abilities of pins.
- TDSEL control for each LSI pin
  TDSEL control registers can control the driving abilities of pins in use for the SDHI.
  TDSEL control resistors on each LSI pin can be controlled by setting the registers in the PFC module.
  (Selection is handled by the return path for SDHI clock drive select register TDSEL0-1.).
- Pull-up/down control for each LSI pin.
  PUEN registers can on/off control of the pull resistors.
  On/off of the pull resistors on each LSI pin can be controlled by setting the registers in the PFC module.
  PUD registers can pull-up/pull-down control of the pull resistors.
  Pull-up/Pull-down control resistors on each LSI pin can be controlled by setting the registers in the PFC module.
- Module selection
  Module Select Register can select the group for multiple LSI pins with multiplexed pin functions.
  Enable and disable the functions of R-Car V4M LSI pins to which pin functions from multiple pin groups are assigned by setting the registers in the PFC module.
- Notes on configuring multiplexed pin functions
  The multiplexed LSI pins (MODSELn, GPSRn, IPiSRn i = 0 - 3) must be set in the initial sequence (*).
  Switching multiplexed LSI pins during operation is not guaranteed.

R-CarV4M product have 9 port groups GPIO group 0-7, System group 0.

### (1) Clock Supply

PFC clock supply is shown in the following table.

**Clock Supply**

| Unit Name | Unit Clock Name | Supply Clock Name |
|---|---|---|
| PFC | PFC clock sources | CPϕ |

### 7.1.2 Block Diagram

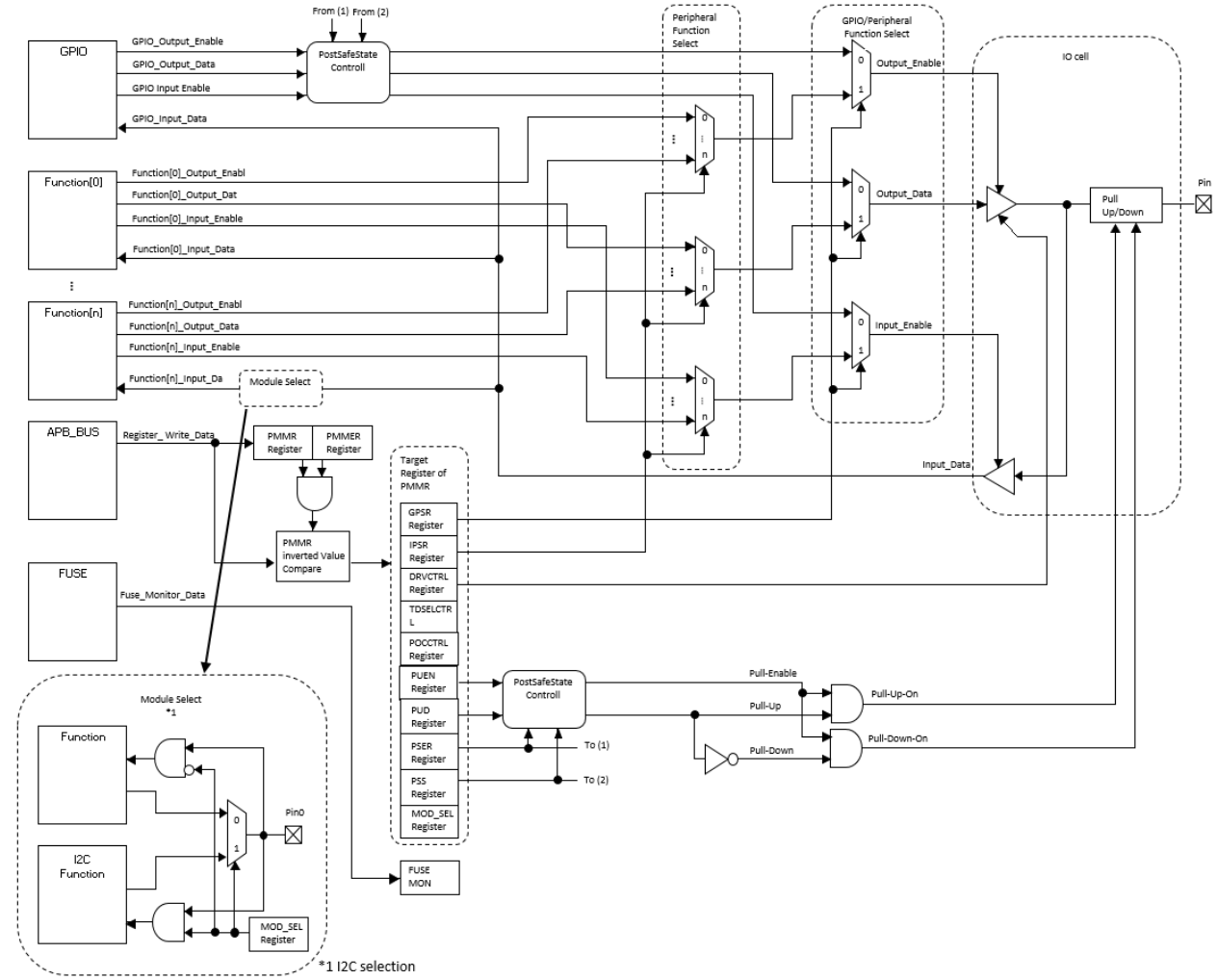



**Figure 7.1 PFC Block Diagram**

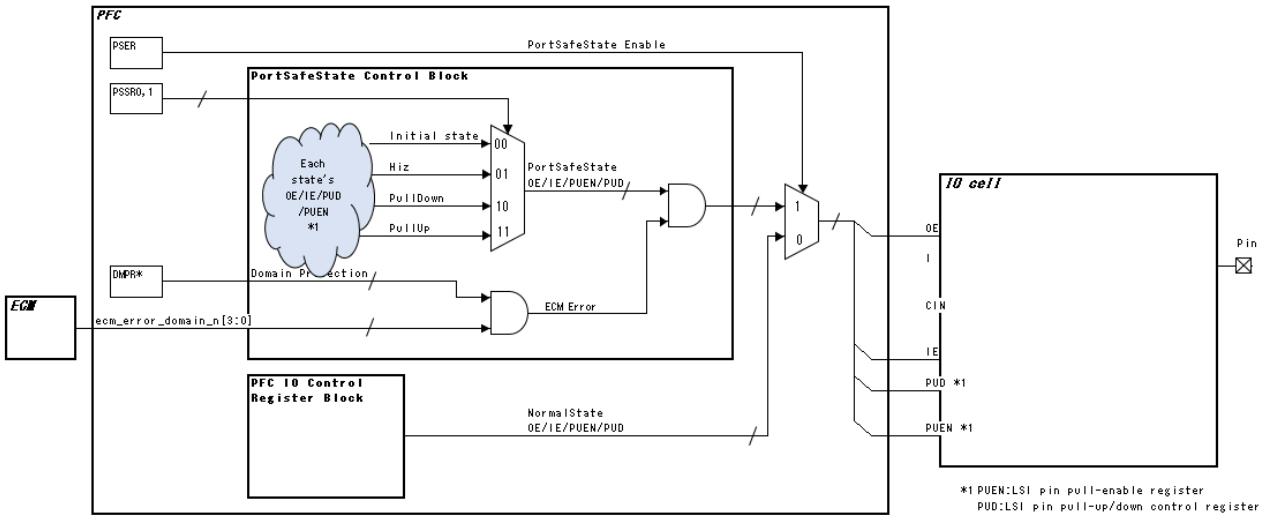



**Figure 7.2 Port Safe State Block**

### 7.1.3 External Pins

Table 7.1 shows the pin configuration of the GPIO.

**Table 7.1 Pin Configuration**

| | I/O | Function | Description |
|---|---|---|---|
| GP0_00 to GP7_20 | I/O | IO/interrupt ports | General input/output and interrupt input |

| Pin Name | GPIO | Pin Name | GPIO | Pin Name | GPIO |
|---|---|---|---|---|---|
| GP0_00 | GP0_00 | MSIOF1_SS2 | GP1_00 | FXR_TXDA | GP2_00 |
| GP0_01 | GP0_01 | MSIOF1_SS1 | GP1_01 | FXR_TXENA# | GP2_01 |
| GP0_02 | GP0_02 | MSIOF1_SYNC | GP1_02 | RXDA_EXTFXR | GP2_02 |
| IRQ3 | GP0_03 | MSIOF1_SCK | GP1_03 | CLK_EXTFXR | GP2_03 |
| IRQ2 | GP0_04 | MSIOF1_TXD | GP1_04 | RXDB_EXTFXR | GP2_04 |
| IRQ1 | GP0_05 | MSIOF1_RXD | GP1_05 | FXR_TXENB# | GP2_05 |
| IRQ0 | GP0_06 | MSIOF0_SS2 | GP1_06 | FXR_TXDB | GP2_06 |
| MSIOF5_SS2 | GP0_07 | MSIOF0_SS1 | GP1_07 | TPU0TO1 | GP2_07 |
| MSIOF5_SS1 | GP0_08 | MSIOF0_SYNC | GP1_08 | TPU0TO0 | GP2_08 |
| MSIOF5_SYNC | GP0_09 | MSIOF0_TXD | GP1_09 | CAN_CLK | GP2_09 |
| MSIOF5_TXD | GP0_10 | MSIOF0_SCK | GP1_10 | CANFD0_TX | GP2_10 |
| MSIOF5_SCK | GP0_11 | MSIOF0_RXD | GP1_11 | CANFD0_RX | GP2_11 |
| MSIOF5_RXD | GP0_12 | HTX0 | GP1_12 | CANFD2_TX | GP2_12 |
| MSIOF2_SS2 | GP0_13 | HCTS0# | GP1_13 | CANFD2_RX | GP2_13 |
| MSIOF2_SS1 | GP0_14 | HRTS0# | GP1_14 | CANFD3_TX | GP2_14 |
| MSIOF2_SYNC | GP0_15 | HSCK0 | GP1_15 | CANFD3_RX | GP2_15 |
| MSIOF2_TXD | GP0_16 | HRX0 | GP1_16 | CANFD1_TX | GP2_17 |
| MSIOF2_SCK | GP0_17 | SCIF_CLK | GP1_17 | CANFD1_RX | GP2_19 |
| MSIOF2_RXD | GP0_18 | SSI_SCK | GP1_18 | | |
| | | SSI_WS | GP1_19 | | |
| | | SSI_SD | GP1_20 | | |
| | | AUDIO_CLKOUT | GP1_21 | | |
| | | AUDIO_CLKIN | GP1_22 | | |
| | | GP1_23 | GP1_23 | | |
| | | HRX3 | GP1_24 | | |
| | | HSCK3 | GP1_25 | | |
| | | HRTS3# | GP1_26 | | |
| | | HCTS3# | GP1_27 | | |
| | | HTX3 | GP1_28 | | |
| | | ERROROUTC# | GP1_29 | | |

| Pin Name | GPIO | Pin Name | GPIO | Pin Name | GPIO |
|---|---|---|---|---|---|
| MMC_SD_D1 | GP3_00 | SCL0 | GP4_00 | AVB2_AVTP_PPS | GP5_00 |
| MMC_SD_D0 | GP3_01 | SDA0 | GP4_01 | AVB2_AVTP_CAPTURE | GP5_01 |
| MMC_SD_D2 | GP3_02 | SCL1 | GP4_02 | AVB2_AVTP_MATCH | GP5_02 |
| MMC_SD_CLK | GP3_03 | SDA1 | GP4_03 | AVB2_LINK | GP5_03 |
| MMC_DS | GP3_04 | SCL2 | GP4_04 | AVB2_PHY_INT | GP5_04 |
| MMC_SD_D3 | GP3_05 | SDA2 | GP4_05 | AVB2_MAGIC | GP5_05 |
| MMC_D5 | GP3_06 | SCL3 | GP4_06 | AVB2_MDC | GP5_06 |
| MMC_D4 | GP3_07 | SDA3 | GP4_07 | AVB2_TXCREFCLK | GP5_07 |
| MMC_D7 | GP3_08 | HRX2 | GP4_08 | AVB2_TD3 | GP5_08 |
| MMC_D6 | GP3_09 | HTX2 | GP4_09 | AVB2_RD3 | GP5_09 |
| MMC_SD_CMD | GP3_10 | HRTS2# | GP4_10 | AVB2_MDIO | GP5_10 |
| SD_CD | GP3_11 | SCIF_CLK2 | GP4_11 | AVB2_TD2 | GP5_11 |
| SD_WP | GP3_12 | HCTS2# | GP4_12 | AVB2_TD1 | GP5_12 |
| PWM1 | GP3_13 | HSCK2 | GP4_13 | AVB2_RD2 | GP5_13 |
| PWM2 | GP3_14 | PWM3 | GP4_14 | AVB2_RD1 | GP5_14 |
| QSPI0_SSL | GP3_15 | PWM4 | GP4_15 | AVB2_TD0 | GP5_15 |
| QSPI0_IO3 | GP3_16 | — | — | AVB2_TXC | GP5_16 |
| QSPI0_IO2 | GP3_17 | — | — | AVB2_RD0 | GP5_17 |
| QSPI0_MISO_IO1 | GP3_18 | — | — | AVB2_RXC | GP5_18 |
| QSPI0_MOSI_IO0 | GP3_19 | — | — | AVB2_TX_CTL | GP5_19 |
| QSPI0_SPCLK | GP3_20 | — | — | AVB2_RX_CTL | GP5_20 |
| QSPI1_MOSI_IO0 | GP3_21 | PCIE0_CLKREQ# | GP4_21 | | |
| QSPI1_SPCLK | GP3_22 | — | — | | |
| QSPI1_MISO_IO1 | GP3_23 | AVS0 | GP4_23 | | |
| QSPI1_IO2 | GP3_24 | AVS1 | GP4_24 | | |
| QSPI1_SSL | GP3_25 | | | | |
| QSPI1_IO3 | GP3_26 | | | | |
| RPC_RESET# | GP3_27 | | | | |
| RPC_WP# | GP3_28 | | | | |
| RPC_INT# | GP3_29 | | | | |
| TCLK3 | GP3_30 | | | | |
| TCLK4 | GP3_31 | | | | |

| Pin Name | GPIO | Pin Name | GPIO |
|---|---|---|---|
| AVB1_MDIO | GP6_00 | AVB0_AVTP_PPS | GP7_00 |
| AVB1_MAGIC | GP6_01 | AVB0_AVTP_CAPTURE | GP7_01 |
| AVB1_MDC | GP6_02 | AVB0_AVTP_MATCH | GP7_02 |
| AVB1_PHY_INT | GP6_03 | AVB0_TD3 | GP7_03 |
| AVB1_LINK | GP6_04 | AVB0_LINK | GP7_04 |
| AVB1_AVTP_MATCH | GP6_05 | AVB0_PHY_INT | GP7_05 |
| AVB1_TXC | GP6_06 | AVB0_TD2 | GP7_06 |
| AVB1_TX_CTL | GP6_07 | AVB0_TD1 | GP7_07 |
| AVB1_RXC | GP6_08 | AVB0_RD3 | GP7_08 |
| AVB1_RX_CTL | GP6_09 | AVB0_TXCREFCLK | GP7_09 |
| AVB1_AVTP_PPS | GP6_10 | AVB0_MAGIC | GP7_10 |
| AVB1_AVTP_CAPTURE | GP6_11 | AVB0_TD0 | GP7_11 |
| AVB1_TD1 | GP6_12 | AVB0_RD2 | GP7_12 |
| AVB1_TD0 | GP6_13 | AVB0_MDC | GP7_13 |
| AVB1_RD1 | GP6_14 | AVB0_MDIO | GP7_14 |
| AVB1_RD0 | GP6_15 | AVB0_TXC | GP7_15 |
| AVB1_TD2 | GP6_16 | AVB0_TX_CTL | GP7_16 |
| AVB1_RD2 | GP6_17 | AVB0_RD1 | GP7_17 |
| AVB1_TD3 | GP6_18 | AVB0_RD0 | GP7_18 |
| AVB1_RD3 | GP6_19 | AVB0_RXC | GP7_19 |
| AVB1_TXCREFCLK | GP6_20 | AVB0_RX_CTL | GP7_20 |

### 7.1.4 Register Configuration

All the registers in the PFC are mapped into the APB bus space. Table 7.2 shows the configuration of the registers provided in the PFC. Details on each register in the PFC are given in sections 7.2.

The register configurations of R/W registers area, SET registers area and CLR registers area in GPIO group 0 to 8 and System group are the same. Also, the register configurations of R/W registers area, SET registers area and CLR registers area in System group are the same.

**Table 7.2 Configuration of Registers in PFC**

| | Bus Domain | Group | Registers Area | Access Type | Address |
|---|---|---|---|---|---|
| PFC | Bus Domain 0 | GPIO Group 0 | R/W registers area | Read or Write | H'E605 0000 to H'E605 01FF |
| | | | SET register area | Set | H'E605 0200 to H'E605 03FF |
| | | | CLR register area | Clear | H'E605 0400 to H'E605 05FF |
| | | | Reserved | — | H'E605 0600 to H'E605 07FF |
| | | GPIO Group 1 | R/W registers area | Read or Write | H'E605 0800 to H'E605 09FF |
| | | | SET register area | Set | H'E605 0A00 to H'E605 0BFF |
| | | | CLR register area | Clear | H'E605 0C00 to H'E605 0DFF |
| | | | Reserved | — | H'E605 0E00 to H'E605 0FFF |
| | | — | Reserved | — | H'E605 1000 to H'E605 1FFF |
| | Bus Domain 1 | GPIO Group 0 | R/W registers area | Read or Write | H'E605 2000 to H'E605 21FF |
| | | | SET register area | Set | H'E605 2200 to H'E605 23FF |
| | | | CLR register area | Clear | H'E605 2400 to H'E605 25FF |
| | | | Reserved | — | H'E605 2600 to H'E605 27FF |
| | | GPIO Group 1 | R/W registers area | Read or Write | H'E605 2800 to H'E605 29FF |
| | | | SET register area | Set | H'E605 2A00 to H'E605 2BFF |
| | | | CLR register area | Clear | H'E605 2C00 to H'E605 2DFF |
| | | | Reserved | — | H'E605 2E00 to H'E605 2FFF |
| | | — | Reserved | — | H'E605 3000 to H'E605 3FFF |
| | Bus Domain 2 | GPIO Group 0 | R/W registers area | Read or Write | H'E605 4000 to H'E605 41FF |
| | | | SET register area | Set | H'E605 4200 to H'E605 43FF |
| | | | CLR register area | Clear | H'E605 4400 to H'E605 45FF |
| | | | Reserved | — | H'E605 4600 to H'E605 47FF |
| | | GPIO Group 1 | R/W registers area | Read or Write | H'E605 4800 to H'E605 49FF |
| | | | SET register area | Set | H'E605 4A00 to H'E605 4BFF |
| | | | CLR register area | Clear | H'E605 4C00 to H'E605 4DFF |
| | | | Reserved | — | H'E605 4E00 to H'E605 4FFF |
| | | — | Reserved | — | H'E605 5000 to H'E605 5FFF |
| | Bus Domain 3 | GPIO Group 0 | R/W registers area | Read or Write | H'E605 6000 to H'E605 61FF |
| | | | SET register area | Set | H'E605 6200 to H'E605 63FF |
| | | | CLR register area | Clear | H'E605 6400 to H'E605 65FF |
| | | | Reserved | — | H'E605 6600 to H'E605 67FF |
| | | GPIO Group 1 | R/W registers area | Read or Write | H'E605 6800 to H'E605 69FF |
| | | | SET register area | Set | H'E605 6A00 to H'E605 6BFF |
| | | | CLR register area | Clear | H'E650 6C00 to H'E605 6DFF |
| | | | Reserved | — | H'E605 6E00 to H'E605 6FFF |
| | | — | Reserved | — | H'E605 7000 to H'E605 7FFF |
| | Bus Domain 0 | GPIO Group 2 | R/W registers area | Read or Write | H'E605 8000 to H'E605 81FF |
| | | | SET register area | Set | H'E605 8200 to H'E605 83FF |
| | | | CLR register area | Clear | H'E605 8400 to H'E605 85FF |
| | | | Reserved | — | H'E605 8600 to H'E605 87FF |
| | | GPIO Group 3 | R/W registers area | Read or Write | H'E605 8800 to H'E605 89FF |
| | | | SET register area | Set | H'E605 8A00 to H'E605 8BFF |
| | | | CLR register area | Clear | H'E605 8C00 to H'E605 8DFF |
| | | | Reserved | — | H'E605 8E00 to H'E605 8FFF |
| | | — | Reserved | — | H'E650 9000 to H'E605 9FFF |
| | Bus Domain 1 | GPIO Group 2 | R/W registers area | Read or Write | H'E605 A000 to H'E605 A1FF |
| | | | SET register area | Set | H'E650 A200 to H'E605 A3FF |
| | | | CLR register area | Clear | H'E605 A400 to H'E605 A5FF |
| | | | Reserved | — | H'E605 A600 to H'E605 A7FF |
| | | GPIO Group 3 | R/W registers area | Read or Write | H'E605 A800 to H'E605 A9FF |
| | | | SET register area | Set | H'E605 AA00 to H'E605 ABFF |
| | | | CLR register area | Clear | H'E605 AC00 to H'E605 ADFF |
| | | | Reserved | — | H'E605 AE00 to H'E605 AFFF |
| | | — | Reserved | — | H'E605 B000 to H'E605 BFFF |
| | Bus Domain 2 | GPIO Group 2 | R/W registers area | Read or Write | H'E605 C000 to H'E605 C1FF |
| | | | SET register area | Set | H'E650 C200 to H'E605 C3FF |
| | | | CLR register area | Clear | H'E605 C400 to H'E605 C5FF |
| | | | Reserved | — | H'E605 C600 to H'E605 C7FF |

| | Bus Domain | Group | Registers Area | Access Type | Address |
|---|---|---|---|---|---|
| | | GPIO Group 3 | R/W registers area | Read or Write | H'E605 C800 to H'E605 C9FF |
| | | | SET register area | Set | H'E605 CA00 to H'E605 CBFF |
| | | | CLR register area | Clear | H'E605 CC00 to H'E605 CDFF |
| | | | Reserved | — | H'E605 CE00 to H'E605 CFFF |
| | | — | Reserved | — | H'E605 D000 to H'E605 DFFF |
| | Bus Domain 3 | GPIO Group 2 | R/W registers area | Read or Write | H'E605 E000 to H'E605 E1FF |
| | | | SET register area | Set | H'E650 E200 to H'E605 E3FF |
| | | | CLR register area | Clear | H'E605 E400 to H'E605 E5FF |
| | | | Reserved | — | H'E605 E600 to H'E605 E7FF |
| | | GPIO Group 3 | R/W registers area | Read or Write | H'E605 E800 to H'E605 E9FF |
| | | | SET register area | Set | H'E605 EA00 to H'E605 EBFF |
| | | | CLR register area | Clear | H'E605 EC00 to H'E605 EDFF |
| | | | Reserved | — | H'E605 EE00 to H'E605 EFFF |
| | | — | Reserved | — | H'E605 F000 to H'E605 FFFF |
| | Bus Domain 0 | GPIO Group 4 | R/W registers area | Read or Write | H'E606 0000 to H'E606 01FF |
| | | | SET register area | Set | H'E606 0200 to H'E606 03FF |
| | | | CLR register area | Clear | H'E606 0400 to H'E606 05FF |
| | | | Reserved | — | H'E606 0600 to H'E606 07FF |
| | | GPIO Group 5 | R/W registers area | Read or Write | H'E606 0800 to H'E606 09FF |
| | | | SET register area | Set | H'E606 0A00 to H'E606 0BFF |
| | | | CLR register area | Clear | H'E606 0C00 to H'E606 0DFF |
| | | | Reserved | — | H'E606 0E00 to H'E606 0FFF |
| | | GPIO Group 6 | R/W registers area | Read or Write | H'E606 1000 to H'E606 11FF |
| | | | SET register area | Set | H'E606 1200 to H'E606 13FF |
| | | | CLR register area | Clear | H'E606 1400 to H'E606 15FF |
| | | | Reserved | — | H'E606 1600 to H'E606 17FF |
| | | GPIO Group 7 | R/W registers area | Read or Write | H'E606 1800 to H'E606 19FF |
| | | | SET register area | Set | H'E606 1A00 to H'E606 1BFF |
| | | | CLR register area | Clear | H'E606 1C00 to H'E606 1DFF |
| | | | Reserved | — | H'E606 1E00 to H'E606 1FFF |
| | Bus Domain 1 | GPIO Group 4 | R/W registers area | Read or Write | H'E606 2000 to H'E606 21FF |
| | | | SET register area | Set | H'E606 2200 to H'E606 23FF |
| | | | CLR register area | Clear | H'E6062400 to H'E606 25FF |
| | | | Reserved | — | H'E606 2600 to H'E606 27FF |
| | | GPIO Group 5 | R/W registers area | Read or Write | H'E606 2800 to H'E606 29FF |
| | | | SET register area | Set | H'E606 2A00 to H'E606 2BFF |
| | | | CLR register area | Clear | H'E606 2C00 to H'E606 2DFF |
| | | | Reserved | — | H'E606 2E00 to H'E606 2FFF |
| | | GPIO Group 6 | R/W registers area | Read or Write | H'E606 3000 to H'E606 31FF |
| | | | SET register area | Set | H'E606 3200 to H'E606 33FF |
| | | | CLR register area | Clear | H'E606 3400 to H'E606 35FF |
| | | | Reserved | — | H'E606 3600 to H'E606 37FF |
| | | GPIO Group 7 | R/W registers area | Read or Write | H'E606 3800 to H'E606 39FF |
| | | | SET register area | Set | H'E606 3A00 to H'E606 3BFF |
| | | | CLR register area | Clear | H'E606 3C00 to H'E606 3DFF |
| | | | Reserved | — | H'E606 3E00 to H'E606 3FFF |
| | Bus Domain 2 | GPIO Group 4 | R/W registers area | Read or Write | H'E606 4000 to H'E606 41FF |
| | | | SET register area | Set | H'E606 4200 to H'E606 43FF |
| | | | CLR register area | Clear | H'E606 4400 to H'E606 45FF |
| | | | Reserved | — | H'E606 4600 to H'E606 47FF |
| | | GPIO Group 5 | R/W registers area | Read or Write | H'E606 4800 to H'E606 49FF |
| | | | SET register area | Set | H'E606 4A00 to H'E606 4BFF |
| | | | CLR register area | Clear | H'E606 4C00 to H'E606 4DFF |
| | | | Reserved | — | H'E606 4E00 to H'E606 4FFF |
| | | GPIO Group 6 | R/W registers area | Read or Write | H'E606 5000 to H'E606 51FF |
| | | | SET register area | Set | H'E606 5200 to H'E606 53FF |
| | | | CLR register area | Clear | H'E606 5400 to H'E606 55FF |
| | | | Reserved | — | H'E606 5600 to H'E606 57FF |
| | | GPIO Group 7 | R/W registers area | Read or Write | H'E606 5800 to H'E606 59FF |
| | | | SET register area | Set | H'E606 5A00 to H'E606 5BFF |
| | | | CLR register area | Clear | H'E606 5C00 to H'E606 5DFF |
| | | | Reserved | — | H'E606 5E00 to H'E606 5FFF |
| | Bus Domain 3 | GPIO Group 4 | R/W registers area | Read or Write | H'E606 6000 to H'E606 61FF |
| | | | SET register area | Set | H'E606 6200 to H'E606 63FF |
| | | | CLR register area | Clear | H'E606 6400 to H'E606 5FF |
| | | | Reserved | — | H'E606 6600 to H'E606 67FF |
| | | GPIO Group 5 | R/W registers area | Read or Write | H'E606 6800 to H'E606 69FF |
| | | | SET register area | Set | H'E606 6A00 to H'E606 6BFF |
| | | | CLR register area | Clear | H'E650 6C00 to H'E606 6DFF |
| | | | Reserved | — | H'E606 6E00 to H'E606 6FFF |
| | | GPIO Group 6 | R/W registers area | Read or Write | H'E606 7000 to H'E606 71FF |

| | Bus Domain | Group | Registers Area | Access Type | Address |
|---|---|---|---|---|---|
| | | | SET register area | Set | H'E606 7200 to H'E606 73FF |
| | | | CLR register area | Clear | H'E606 7400 to H'E606 75FF |
| | | | Reserved | — | H'E606 7600 to H'E606 77FF |
| | | GPIO Group 7 | R/W registers area | Read or Write | H'E606 7800 to H'E606 79FF |
| | | | SET register area | Set | H'E606 7A00 to H'E606 7BFF |
| | | | CLR register area | Clear | H'E606 7C00 to H'E606 7DFF |
| | | | Reserved | — | H'E606 7E00 to H'E606 7FFF |
| | Bus Domain 0 | System Group [for PRESETOUT#, DCURDY#/LPDCLKOUT, DCUTDO/LPDO, DCUTMS] | R/W registers area | Read or Write | H'E607 8000 to H'E607 81FF |
| | | | SET register area | Set | H'E607 8200 to H'E607 83FF |
| | | | CLR register area | Clear | H'E607 8400 to H'E607 85FF |
| | | | Reserved | — | H'E607 8600 to H'E607 87FF |
| | | — | Reserved | — | H'E607 8800 to H'E607 9FFF |
| | Bus Domain 1 | System Group [for PRESETOUT#, DCURDY#/LPDCLKOUT, DCUTDO/LPDO, DCUTMS] | R/W registers area | Read or Write | H'E607 A000 to H'E607 A1FF |
| | | | SET register area | Set | H'E607 A200 to H'E607 A3FF |
| | | | CLR register area | Clear | H'E607 A400 to H'E607 A5FF |
| | | | Reserved | — | H'E607 A600 to H'E607 A7FF |
| | | — | Reserved | — | H'E607 A800 to H'E607 BFFF |
| | Bus Domain 2 | System Group [for PRESETOUT#, DCURDY#/LPDCLKOUT, DCUTDO/LPDO, DCUTMS] | R/W registers area | Read or Write | H'E607 C000 to H'E607 C1FF |
| | | | SET register area | Set | H'E607 C200 to H'E607 C3FF |
| | | | CLR register area | Clear | H'E607 C400 to H'E607 C5FF |
| | | | Reserved | — | H'E607 C600 to H'E607 C7FF |
| | | — | Reserved | — | H'E607 C800 to H'E607 DFFF |
| | Bus Domain 3 | System Group [for PRESETOUT#, DCURDY#/LPDCLKOUT, DCUTDO/LPDO, DCUTMS] | R/W registers area | Read or Write | H'E607 E000 to H'E607 E1FF |
| | | | SET register area | Set | H'E607 E200 to H'E607 E3FF |
| | | | CLR register area | Clear | H'E607 E400 to H'E607 E5FF |
| | | | Reserved | — | H'E607 E600 to H'E607 E7FF |
| | | — | Reserved | — | H'E607 E800 to H'E607 FFFF |

**Table 7.3 Base address of Registers in PFC**

| Bus Domain | Group | Base address |
|---|---|---|
| Bus Domain 0 | GPIO Group 0 | H'E605 0000 |
| | GPIO Group 1 | H'E605 0800 |
| | GPIO Group 2 | H'E605 8000 |
| | GPIO Group 3 | H'E605 8800 |
| | GPIO Group 4 | H'E606 0000 |
| | GPIO Group 5 | H'E606 0800 |
| | GPIO Group 6 | H'E606 1000 |
| | GPIO Group 7 | H'E606 1800 |
| | — | — |
| | System Group | H'E607 8000 |

**Table 7.4 Offset table**

| | | R/W registers area | +H'200<br>SET registers area | +H'400<br>CLR registers area |
|---|---|---|---|---|
| | Bus Domain 0 | Substitute from Table 7.3 | +H'0200 | +H'0400 |
| +H'2000 | Bus Domain 1 | +H'2000 | +H'2200 | +H'2400 |
| +H'4000 | Bus Domain 2 | +H'4000 | +H'4200 | +H'4400 |
| +H'6000 | Bus Domain 3 | +H'6000 | +H'6200 | +H'6400 |

**Table 7.5 Configuration of R/W Registers in GPIO group n**

| Name | Abbr. | R/W | Initial Value | Offset Address [lower 9bits] | Access Size [bits] | Condition |
|---|---|---|---|---|---|---|
| LSI Multiplexed Pin Setting Mask Register | PMMRn (n = 0 to 7) | R/W | H'0000 0000 | H'000 | 32 | - |
| LSI Multiplexed Pin Setting Mask Enable Register | PMMERn | R/W | H'0000 0001 | H'004 | 32 | - |
| Domain protection register 0 | DM0PRn | R/W | H'FFFF FFFF | H'020 | 32 | be able to write only from Bus Domain 0 |
| Domain protection register 1 | DM1PRn | R/W | H'0000 0000 | H'024 | 32 | be able to write only from Bus Domain 0 |
| Domain protection register 2 | DM2PRn | R/W | H'0000 0000 | H'028 | 32 | be able to write only from Bus Domain 0 |
| Domain protection register 3 | DM3PRn | R/W | H'0000 0000 | H'02C | 32 | be able to write only from Bus Domain 0 |
| GPIO/Peripheral Function Select register | GPSRn | R/W | Port group 0: H'0007 FFFF<br>Port group 1: H'1FFF FFFF<br>Port group 2: H'0000 0007<br>Port group 3: H'3FFF 9FFF<br>Port group 4: H'0180 0000<br>Port group 5: H'001F FFFF<br>Port group 6: H'001F FFFF<br>Port group 7: H'001F FFFF | H'040 | 32 | - |
| Peripheral Function Select register 0 | IP0SRn | R/W | Port group 0: H'0000 0000<br>Port group 1: H'0000 0000<br>Port group 2: H'0000 0000<br>Port group 3: H'0000 0000<br>Port group 4: H'0000 0000<br>Port group 5: H'0000 0000<br>Port group 6: H'0000 0000<br>Port group 7: H'0000 0000 | H'060 | 32 | - |
| Peripheral Function Select register 1 | IP1SRn | R/W | Port group 0: H'0000 0000<br>Port group 1: H'0000 0000<br>Port group 2: H'0000 0000<br>Port group 3: H'0000 0000<br>Port group 4: H'0000 0000<br>Port group 5: H'0000 0000<br>Port group 6: H'0000 0000<br>Port group 7: H'0000 0000 | H'064 | 32 | - |

| Name | Abbr. | R/W | Initial Value | Offset Address [lower 9bits] | Access Size [bits] | Condition |
|---|---|---|---|---|---|---|
| Peripheral Function Select register 2 | IP2SRn | R/W | Port group 0: H'0000 0000<br>Port group 1: H'0000 0000<br>Port group 2: H'0000 0000<br>Port group 3: H'0000 0000<br>Port group 4: H'0000 0000<br>Port group 5: H'0000 0000<br>Port group 6: H'0000 0000<br>Port group 7: H'0000 0000 | H'068 | 32 | - |
| Peripheral Function Select register 3 | IP3SRn | R/W | Port group 1: H'0000 0000<br>Port group 3: H'0000 0000<br>Port group 4: H'0000 0000 | H'06C | 32 | - |
| DRV control register 0 | DRV0CTRLn | R/W | Port group 0: H'6666 6666<br>Port group 1: H'6666 6666<br>Port group 2: H'6666 6666<br>Port group 3: H'6666 6666<br>Port group 4: H'6666 6666<br>Port group 5: H'6666 6666<br>Port group 6: H'6666 6666<br>Port group 7: H'6666 6666 | H'080 | 32 | - |
| DRV control register 1 | DRV1CTRLn | R/W | Port group 0: H'6666 6666<br>Port group 1: H'6666 6666<br>Port group 2: H'6666 6666<br>Port group 3: H'2226 6666<br>Port group 4: H'6666 6666<br>Port group 5: H'6666 6666<br>Port group 6: H'6666 6666<br>Port group 7: H'6666 6666 | H'084 | 32 | - |
| DRV control register 2 | DRV2CTRLn | R/W | Port group 0: H'0000 0666<br>Port group 1: H'6666 6666<br>Port group 2: H'0000 6060<br>Port group 3: H'2222 2222<br>Port group 4: H'6060 0000<br>Port group 5: H'0006 6666<br>Port group 6: H'0006 6666<br>Port group 7: H'0006 6666 | H'088 | 32 | - |
| DRV control register 3 | DRV3CTRLn | R/W | Port group 1: H'0026 6666<br>Port group 3: H'2222 2222<br>Port group 4: H'0000 0006 | H'08C | 32 | - |
| POWER Condition control register | POCn | R/W | Port group 0: H'0007 FFFF<br>Port group 1: H'1FFF FFFF<br>Port group 3: H'0000 1FFF<br>Port group 4: H'0000 3FFF<br>Port group 5: H'0000 0001<br>Port group 6: H'0000 0001<br>Port group 7: H'0000 0001 | H'0A0 | 32 | - |
| LSI pin pull-enable register | PUENn | R/W | Port group 0: H'0000 1E01<br>Port group 1: H'3C7C 0100<br>Port group 2: H'000A D7FF<br>Port group 3: H'E000 6000<br>Port group 4: H'0000 C000<br>Port group 5: H'001F FF9C<br>Port group 6: H'001F FFFA<br>Port group 7: H'001F 9FFF | H'0C0 | 32 | - |
| LSI pin pull-up/down control register | PUDn | R/W | Port group 0: H'0000 1E01<br>Port group 1: H'1C7C 0100<br>Port group 2: H'000A D7FF<br>Port group 3: H'2000 0000<br>Port group 4: H'0000 C000<br>Port group 5: H'0000 0000<br>Port group 6: H'0000 0000<br>Port group 7: H'0000 0000 | H'0E0 | 32 | - |
| Module select register | MODSELn | R/W | H'0000 0000 | H'100 | 32 | - |

| Name | Abbr. | R/W | Initial Value | Offset Address [lower 9bits] | Access Size [bits] | Condition |
|---|---|---|---|---|---|---|
| TDSEL Control Register 0 | TD0SELn | R/W | H'0000 0000 | H'120 | 32 | - |
| TDSEL Control Register 1 | TD1SELn | R/W | H'0000 0000 | H'124 | 32 | - |
| Port Safe state Enable Register | PSERn | R/W | H'FFFF FFFF | H'160 | 32 | - |
| Port Safe state Select Register 0 | PS0SRn | R/W | H'0000 0000 | H'164 | 32 | - |
| Port Safe state Select Register 1 | PS1SRn | R/W | H'0000 0000 | H'168 | 32 | - |
| General IO/interrupt switching register | IOINTSELn | R/W | H'0000 0000 | H'180 | 32 | - |
| General input/output switching register | INOUTSELn | R/W | H'0000 0000 | H'184 | 32 | - |
| General output register | OUTDTn | R/W | H'0000 0000 | H'188 | 32 | - |
| General input register | INDTn | R | - | H'18C | 32 | - |
| Interrupt display register | INTDTn | R | H'0000 0000 | H'190 | 32 | - |
| Interrupt clear register | INTCLRn | R/W | H'0000 0000 | H'194 | 32 | - |
| Interrupt mask register | INTMSKn | R/W | H'0000 0000 | H'198 | 32 | - |
| Interrupt Mask Clear Register | MSKCLRn | R/W | H'0000 0000 | H'19C | 32 | - |
| Positive/negative logic select register | POSNEGn | R/W | H'0000 0000 | H'1A0 | 32 | - |
| Edge/level select register 0 | EDGLEVELn | R/W | H'0000 0000 | H'1A4 | 32 | - |
| Chattering prevention on/off register | FILONOFFn | R/W | H'0000 0000 | H'1A8 | 32 | - |
| Chattering prevention clock select register | FILCLKSELn | R/W | H'0000 411B | H'1AC | 32 | - |
| Output data select register | OUTDTSELn | R/W | H'0000 0000 | H'1C0 | 32 | - |
| Output data high register | OUTDTHn | R/W | H'0000 0000 | H'1C4 | 32 | - |
| Output data low register | OUTDTLn | R/W | H'0000 0000 | H'1C8 | 32 | - |
| One edge/both edge select register | BOTHEDGEn | R/W | H'0000 0000 | H'1CC | 32 | - |
| General input enable register | INENn | R/W | H'0000 0000 | H'1D0 | 32 | - |

**Table 7.6 Configuration of R/W Registers in System Group**

| Name | Abbr. | R/W | Initial Value | Offset Address [lower 9bits] | Access Size [bits] | Condition |
|---|---|---|---|---|---|---|
| LSI Multiplexed Pin Setting Mask Register | PMMRSYS | R/W | H'0000 0000 | H'000 | 32 | - |
| LSI Multiplexed Pin Setting Mask Enable Register | PMMERSYS | R/W | H'0000 0001 | H'004 | 32 | - |
| Domain protection register0 | DM0PRSYS | R/W | H'FFFF FFFF | H'020 | 32 | - |
| Domain protection register1 | DM1PRSYS | R/W | H'0000 0000 | H'024 | 32 | - |
| Domain protection register2 | DM2PRSYS | R/W | H'0000 0000 | H'028 | 32 | - |
| Domain protection register3 | DM3PRSYS | R/W | H'0000 0000 | H'02C | 32 | - |
| DRV control register 0 | DRV0CTRLSYS | R/W | H'0000 0006 | H'080 | 32 | - |
| DRV control register 1 | DRV1CTRLSYS | R/W | H'0000 3302 | H'084 | 32 | - |
| LSI pin pull-enable register | PUENSYS | R/W | H'0000 03C0 | H'0C0 | 32 | - |
| LSI pin pull-up/down control register | PUDSYS | R/W | H'0000 03C0 | H'0E0 | 32 | - |

## 7.2 Register Description

[Legend]

| | |
|---|---|
| Initial value: | Register value after a reset |
| - : | Undefined value |
| R/W: | Readable/writable. The written value can be read. |
| R: | Read-only. The write value should always be 0. |
| n: | Number of GPIO Group, it’s value is in the range 0 to 9. |

All the bits are active high unless otherwise specified, and deactivated on reset.

All access to registers is made in longword units.

The write value to a reserved bit shall always be 0.

[Register access type and access mode]

Register access type is decoded by address offset, and selected by address check.

- R/W : Read-Write enabled register.

  Offset: H’000
- SET : Controled per bit. Write case, applied bit is "1" then "1", or applied bit is "0" then keep value. Read case, return Reg value. (HWM described as WriteOnly)

  Offset: H’200
- CLR : Controled per bit. Write case, applied bit is "1" then "0", or applied bit is "0" then keep value. Read case, return Reg value. (HWM described WriteOnly)

  Offset: H’400
- Reserved:

  Offset: H’600

### 7.2.1 LSI Multiplexed Pin Setting Mask Register (PMMRn / PMMRSYS)

Function: PMMRn / PMMRSYS enables / disables writing to the multiplexed pin setting registers.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMRn[31:16] / PMMRSYS[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMMRn[15:0] / PMMRSYS[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PMMRn[31:0] | H'0000 0000 | R/W | Multiplexed Pin Setting Mask<br>Writing a value to any register from among the GPIO/peripheral function select register GPSR, peripheral function select registers, DRV control registers, TDSEL control register, POWER Condition control register, LSI pin pull-enable register PUEN, LSI pin pull-up/down control register PUD, and Module Select Register MODSEL Bus Domain Protection Register DMPR, Port Safe state Select Register PSSR and Port Safe state Enable Register PSER is enabled by writing the inverse of the value to this register. |

Note: This register must be set before setting each of the GPIO / peripheral function select register GPSR, peripheral function select registers, DRV control register, TDSEL control register, POWER Condition control register, LSI pin pull-enable register PUEN and LSI pin pull-up/down control register PUD, Module Select Register MODSEL, Bus Domain Protection Register DMPR, Port Safe state Select Register PSSR and Port Safe state Enable Register PSER.

### 7.2.2 LSI Multiplexed Pin Setting Mask Enable Register (PMMERn / PMMERSYS)

Function: PMMERn / PMMERSYS performs enables / disables control of the PMMR.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

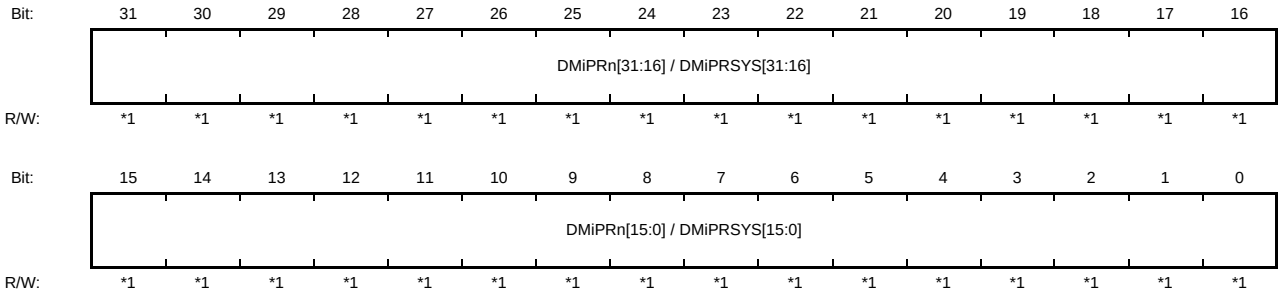
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PMMERn[0] / PMMERSYS[0] |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | — | R | — |
| 0 | PMMERn | 1 | R/W | Multiplexed Pin Setting Mask Enable<br>0: PMMR function is disabled.<br>1: PMMR function is enabled. |

### 7.2.3 Bus Domain Protection Register 0-3 (DM0PRn , DM1PRn , DM2PRn , DM3PRn / DM0PRSYS , DM1PRSYS , DM2PRSYS , DM3PRSYS)

Function: DMiPRn / DMiPRSYS enables / disables writing to the registers from bus domain.

Note: i = 0 to 3 (Bus Domain)



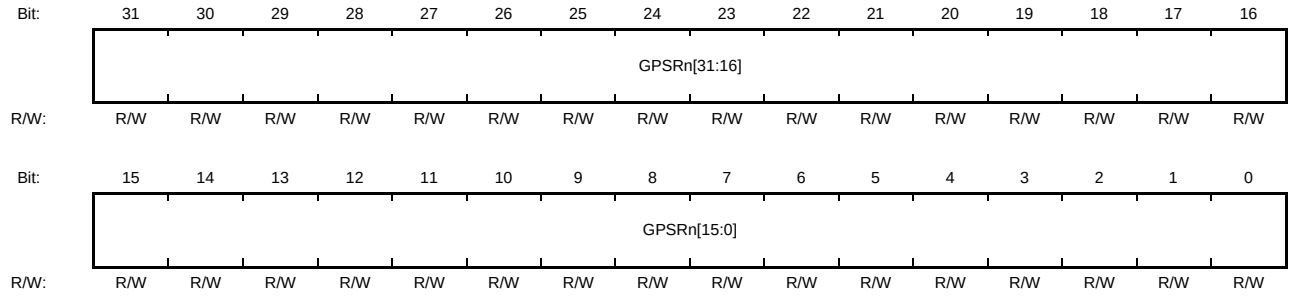

Note: *1 R/W. Be able to write only from Bus domain 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DM0PRn[31:0], DM0PRSYS[31:0] | H'FFFF FFFF | R/W | Bus Domain Protection<br>0: Disable write to register<br>1: Enable write to register<br><br>DMiPRn / DMiPRSYS and FILCLKSELn are not include in the potection target. |
| | DM1PRn[31:0], DM2PRn[31:0], DM3PRn[31:0], DM1PRSYS[31:0], DM2PRSYS[31:0], DM3PRSYS[31:0] | H'0000 0000 | R/W | |

Note: To enable this register to be set, appropriately set the multiplexed pin setting mask register (PMMR) immediately before setting this register.

### 7.2.4 GPIO / Peripheral Function Select Register (GPSRn)

Function: GPSRn selects the functions of the multiplexed LSI pins.




| Bit | Bit Name | R/W | Description |
|---|---|---|---|
| 31 to 0 | GPSRn[31:0] | R/W | 0: GPIO<br>1: Peripheral function<br>The functions of the LSI pins are selected according to the table below. |

Note: To enable this register to be set, appropriately set the multiplexed pin setting mask register (PMMR) immediately before setting this register.

**Table 7.7 Configuration of Registers in GPSR0-2**

| | GPSR0 | | GPSR1 | | GPSR2 | |
|---|---|---|---|---|---|---|
| | Pin name | Initial value (MD[4:1] ≠B'0000) | Pin name | Initial value (MD[4:1] ≠B'0000) | Pin name | Initial value (MD[4:1] ≠B'0000) |
| bit31 | — | 0 | — | 0 | — | 0 |
| bit30 | — | 0 | — | 0 | — | 0 |
| bit29 | — | 0 | ERROROUTC# | 0 | — | 0 |
| bit28 | — | 0 | HTX3 | 0 | — | 0 |
| bit27 | — | 0 | HCTS3# | 0 | — | 0 |
| bit26 | — | 0 | HRTS3# | 0 | — | 0 |
| bit25 | — | 0 | HSCK3 | 0 | — | 0 |
| bit24 | — | 0 | HRX3 | 0 | — | 0 |
| bit23 | — | 0 | GP1_23 | 0 | — | 0 |
| bit22 | — | 0 | AUDIO_CLKIN | 0 | — | 0 |
| bit21 | — | 0 | AUDIO_CLKOUT | 0 | — | 0 |
| bit20 | — | 0 | SSI_SD | 0 | — | 0 |
| bit19 | — | 0 | SSI_WS | 0 | CANFD1_RX | 0 |
| bit18 | MSIOF2_RXD | 0 | SSI_SCK | 0 | — | 0 |
| bit17 | MSIOF2_SCK | 0 | SCIF_CLK | 0 | CANFD1_TX | 0 |
| bit16 | MSIOF2_TXD | 0 | HRX0 | 0 | — | 0 |
| bit15 | MSIOF2_SYNC | 0 | HSCK0 | 0 | CANFD3_RX | 0 |
| bit14 | MSIOF2_SS1 | 0 | HRTS0# | 0 | CANFD3_TX | 0 |
| bit13 | MSIOF2_SS2 | 0 | HCTS0# | 0 | CANFD2_RX | 0 |
| bit12 | MSIOF5_RXD | 0 | HTX0 | 0 | CANFD2_TX | 0 |
| bit11 | MSIOF5_SCK | 0 | MSIOF0_RXD | 0 | CANFD0_RX | 0 |
| bit10 | MSIOF5_TXD | 0 | MSIOF0_SCK | 0 | CANFD0_TX | 0 |
| bit9 | MSIOF5_SYNC | 0 | MSIOF0_TXD | 0 | CAN_CLK | 0 |
| bit8 | MSIOF5_SS1 | 0 | MSIOF0_SYNC | 0 | TPU0TO0 | 0 |
| bit7 | MSIOF5_SS2 | 0 | MSIOF0_SS1 | 0 | TPU0TO1 | 0 |
| bit6 | IRQ0 | 0 | MSIOF0_SS2 | 0 | FXR_TXDB | 0 |
| bit5 | IRQ1 | 0 | MSIOF1_RXD | 0 | FXR_TXENB# | 0 |
| bit4 | IRQ2 | 0 | MSIOF1_TXD | 0 | RXDB_EXTFXR | 0 |
| bit3 | IRQ3 | 0 | MSIOF1_SCK | 0 | CLK_EXTFXR | 0 |
| bit2 | GP0_02 | 0 | MSIOF1_SYNC | 0 | RXDA_EXTFXR | 0 |
| bit1 | GP0_01 | 0 | MSIOF1_SS1 | 0 | FXR_TXENA# | 0 |
| bit0 | GP0_00 | 0 | MSIOF1_SS2 | 0 | FXR_TXDA | 0 |

**Table 7.8 Configuration of Registers in GPSR3-5**

| | GPSR3 | | GPSR4 | | GPSR5 | |
|---|---|---|---|---|---|---|
| | Pin name | Initial value (MD[4:1] ≠B'0000) | Pin name | Initial value (MD[4:1] ≠B'0000) | Pin name | Initial value (MD[4:1] ≠B'0000) |
| bit31 | TCLK4 | 0 | — | 0 | — | 0 |
| bit30 | TCLK3 | 0 | — | 0 | — | 0 |
| bit29 | RPC_INT# | 1 | — | 0 | — | 0 |
| bit28 | RPC_WP# | 1 | — | 0 | — | 0 |
| bit27 | RPC_RESET# | 1 | — | 0 | — | 0 |
| bit26 | QSPI1_IO3 | 1 | — | 0 | — | 0 |
| bit25 | QSPI1_SSL | 1 | — | 0 | — | 0 |
| bit24 | QSPI1_IO2 | 1 | AVS1 | 1 | — | 0 |
| bit23 | QSPI1_MISO_IO1 | 1 | AVS0 | 1 | — | 0 |
| bit22 | QSPI1_SPCLK | 1 | — | 0 | — | 0 |
| bit21 | QSPI1_MOSI_IO0 | 1 | PCIE0_CLKREQ# | 0 | — | 0 |
| bit20 | QSPI0_SPCLK | 1 | — | 0 | AVB2_RX_CTL | 1 |
| bit19 | QSPI0_MOSI_IO0 | 1 | — | 0 | AVB2_TX_CTL | 1 |
| bit18 | QSPI0_MISO_IO1 | 1 | — | 0 | AVB2_RXC | 1 |
| bit17 | QSPI0_IO2 | 1 | — | 0 | AVB2_RD0 | 1 |
| bit16 | QSPI0_IO3 | 1 | — | 0 | AVB2_TXC | 1 |
| bit15 | QSPI0_SSL | 1 | PWM4 | 0 | AVB2_TD0 | 1 |
| bit14 | PWM2 | 0 | PWM3 | 0 | AVB2_RD1 | 1 |
| bit13 | PWM1 | 0 | HSCK2 | 0 | AVB2_RD2 | 1 |
| bit12 | SD_WP | 1 | HCTS2# | 0 | AVB2_TD1 | 1 |
| bit11 | SD_CD | 1 | SCIF_CLK2 | 0 | AVB2_TD2 | 1 |
| bit10 | MMC_SD_CMD | 1 | HRTS2# | 0 | AVB2_MDIO | 1 |
| bit9 | MMC_D6 | 1 | HTX2 | 0 | AVB2_RD3 | 1 |
| bit8 | MMC_D7 | 1 | HRX2 | 0 | AVB2_TD3 | 1 |
| bit7 | MMC_D4 | 1 | SDA3 | 0 | AVB2_TXCREFCLK | 1 |
| bit6 | MMC_D5 | 1 | SCL3 | 0 | AVB2_MDC | 1 |
| bit5 | MMC_SD_D3 | 1 | SDA2 | 0 | AVB2_MAGIC | 1 |
| bit4 | MMC_DS | 1 | SCL2 | 0 | AVB2_PHY_INT | 1 |
| bit3 | MMC_SD_CLK | 1 | SDA1 | 0 | AVB2_LINK | 1 |
| bit2 | MMC_SD_D2 | 1 | SCL1 | 0 | AVB2_AVTP_MATCH | 1 |
| bit1 | MMC_SD_D0 | 1 | SDA0 | 0 | AVB2_AVTP_CAPTURE | 1 |
| bit0 | MMC_SD_D1 | 1 | SCL0 | 0 | AVB2_AVTP_PPS | 1 |

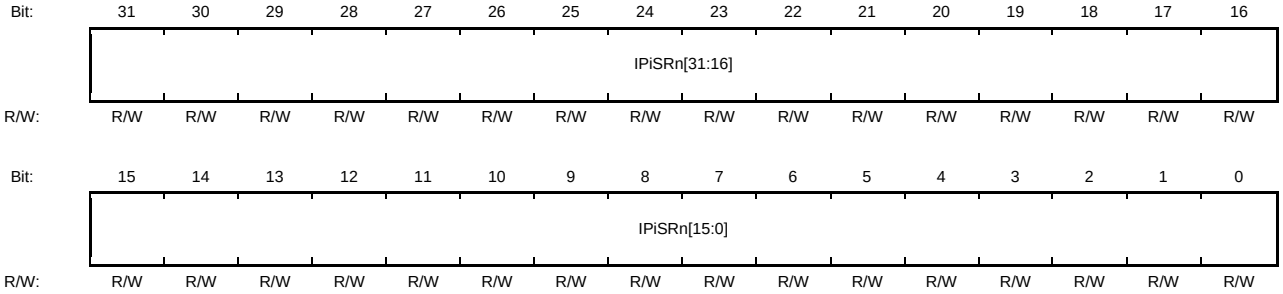
**Table 7.9 Configuration of Registers in GPSR6-7**

| | GPSR6 | | GPSR7 | |
|---|---|---|---|---|
| | Pin name | Initial value (MD[4:1] ≠ B'0000) | Pin name | Initial value (MD[4:1] ≠ B'0000) |
| bit31 | — | 0 | — | 0 |
| bit30 | — | 0 | — | 0 |
| bit29 | — | 0 | — | 0 |
| bit28 | — | 0 | — | 0 |
| bit27 | — | 0 | — | 0 |
| bit26 | — | 0 | — | 0 |
| bit25 | — | 0 | — | 0 |
| bit24 | — | 0 | — | 0 |
| bit23 | — | 0 | — | 0 |
| bit22 | — | 0 | — | 0 |
| bit21 | — | 0 | — | 0 |
| bit20 | AVB1_TXCREFCLK | 1 | AVB0_RX_CTL | 1 |
| bit19 | AVB1_RD3 | 1 | AVB0_RXC | 1 |
| bit18 | AVB1_TD3 | 1 | AVB0_RD0 | 1 |
| bit17 | AVB1_RD2 | 1 | AVB0_RD1 | 1 |
| bit16 | AVB1_TD2 | 1 | AVB0_TX_CTL | 1 |
| bit15 | AVB1_RD0 | 1 | AVB0_TXC | 1 |
| bit14 | AVB1_RD1 | 1 | AVB0_MDIO | 1 |
| bit13 | AVB1_TD0 | 1 | AVB0_MDC | 1 |
| bit12 | AVB1_TD1 | 1 | AVB0_RD2 | 1 |
| bit11 | AVB1_AVTP_CAPTURE | 1 | AVB0_TD0 | 1 |
| bit10 | AVB1_AVTP_PPS | 1 | AVB0_MAGIC | 1 |
| bit9 | AVB1_RX_CTL | 1 | AVB0_TXCREFCLK | 1 |
| bit8 | AVB1_RXC | 1 | AVB0_RD3 | 1 |
| bit7 | AVB1_TX_CTL | 1 | AVB0_TD1 | 1 |
| bit6 | AVB1_TXC | 1 | AVB0_TD2 | 1 |
| bit5 | AVB1_AVTP_MATCH | 1 | AVB0_PHY_INT | 1 |
| bit4 | AVB1_LINK | 1 | AVB0_LINK | 1 |
| bit3 | AVB1_PHY_INT | 1 | AVB0_TD3 | 1 |
| bit2 | AVB1_MDC | 1 | AVB0_AVTP_MATCH | 1 |
| bit1 | AVB1_MAGIC | 1 | AVB0_AVTP_CAPTURE | 1 |
| bit0 | AVB1_MDIO | 1 | AVB0_AVTP_PPS | 1 |

### 7.2.5 Peripheral Function Select Register 0-3 (IP0SRn , IP1SRn , IP2SRn , IP3SRn)

Function: IPiSRn selects the functions of the multiplexed LSI pins.

Note: i = 0 to 3 (Bus Domain)




Note: i = 0 to 3

| Bit | R/W | Description |
|---|---|---|
| 31 to 0 | R/W | The functions of the LSI pins are selected according to the table below. |

Note: To enable this register to be set, appropriately set the multiplexed pin setting mask register (PMMR) immediately before setting this register.

**Table 7.10 Configuration of Registers in IP0SR0 , IP1SR0 , IP2SR0**

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| GP0_00 | IP0SR0[3:0] | — | ERROROUTC# | TCLK2 | — |
| GP0_01 | IP0SR0[7:4] | — | MSIOF3_SS1 | — | — |
| GP0_02 | IP0SR0[11:8] | — | MSIOF3_SS2 | — | — |
| IRQ3 | IP0SR0[15:12] | IRQ3 | MSIOF3_SCK | — | — |
| IRQ2 | IP0SR0[19:16] | IRQ2 | MSIOF3_TXD | — | — |
| IRQ1 | IP0SR0[23:20] | IRQ1 | MSIOF3_RXD | — | — |
| IRQ0 | IP0SR0[27:24] | IRQ0 | MSIOF3_SYNC | — | — |
| MSIOF5_SS2 | IP0SR0[31:28] | MSIOF5_SS2 | — | — | — |
| MSIOF5_SS1 | IP1SR0[3:0] | MSIOF5_SS1 | — | — | — |
| MSIOF5_SYNC | IP1SR0[7:4] | MSIOF5_SYNC | — | — | — |
| MSIOF5_TXD | IP1SR0[11:8] | MSIOF5_TXD | — | — | — |
| MSIOF5_SCK | IP1SR0[15:12] | MSIOF5_SCK | — | — | — |
| MSIOF5_RXD | IP1SR0[19:16] | MSIOF5_RXD | — | — | — |
| MSIOF2_SS2 | IP1SR0[23:20] | MSIOF2_SS2 | TCLK1 | IRQ2 | — |
| MSIOF2_SS1 | IP1SR0[27:24] | MSIOF2_SS1 | HTX1 | TX1 | — |
| MSIOF2_SYNC | IP1SR0[31:28] | MSIOF2_SYNC | HRX1 | RX1 | — |
| MSIOF2_TXD | IP2SR0[3:0] | MSIOF2_TXD | HCTS1# | CTS1# | — |
| MSIOF2_SCK | IP2SR0[7:4] | MSIOF2_SCK | HRTS1# | RTS1# | — |
| MSIOF2_RXD | IP2SR0[11:8] | MSIOF2_RXD | HSCK1 | SCK1 | — |
| – | IP2SR0[15:12] | — | — | — | — |
| – | IP2SR0[19:16] | — | — | — | — |
| – | IP2SR0[23:20] | — | — | — | — |
| – | IP2SR0[27:24] | — | — | — | — |
| – | IP2SR0[31:28] | — | — | — | — |

**Table 7.11 Configuration of Registers in IP0SR1, IP1SR1, IP2SR1, IP3SR1**

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| MSIOF1_SS2 | IP0SR1[3:0] | MSIOF1_SS2 | HTX3 | TX3 | — |
| MSIOF1_SS1 | IP0SR1[7:4] | MSIOF1_SS1 | HCTS3# | RX3 | — |
| MSIOF1_SYNC | IP0SR1[11:8] | MSIOF1_SYNC | HRTS3# | RTS3# | — |
| MSIOF1_SCK | IP0SR1[15:12] | MSIOF1_SCK | HSCK3 | CTS3# | — |
| MSIOF1_TXD | IP0SR1[19:16] | MSIOF1_TXD | HRX3 | SCK3 | — |
| MSIOF1_RXD | IP0SR1[23:20] | MSIOF1_RXD | — | — | — |
| MSIOF0_SS2 | IP0SR1[27:24] | MSIOF0_SS2 | HTX1 | TX1 | — |
| MSIOF0_SS1 | IP0SR1[31:28] | MSIOF0_SS1 | HRX1 | RX1 | — |
| MSIOF0_SYNC | IP1SR1[3:0] | MSIOF0_SYNC | HCTS1# | CTS1# | — |

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| MSIOF0_TXD | IP1SR1[7:4] | MSIOF0_TXD | HRTS1# | RTS1# | — |
| MSIOF0_SCK | IP1SR1[11:8] | MSIOF0_SCK | HSCK1 | SCK1 | — |
| MSIOF0_RXD | IP1SR1[15:12] | MSIOF0_RXD | — | — | — |
| HTX0 | IP1SR1[19:16] | HTX0 | TX0 | — | — |
| HCTS0# | IP1SR1[23:20] | HCTS0# | CTS0# | — | — |
| HRTS0# | IP1SR1[27:24] | HRTS0# | RTS0# | PWM0 | — |
| HSCK0 | IP1SR1[31:28] | HSCK0 | SCK0 | PWM0 | — |
| HRX0 | IP2SR1[3:0] | HRX0 | RX0 | — | — |
| SCIF_CLK | IP2SR1[7:4] | SCIF_CLK | IRQ4 | — | — |
| SSI_SCK | IP2SR1[11:8] | SSI_SCK | TCLK3 | — | — |
| SSI_WS | IP2SR1[15:12] | SSI_WS | TCLK4 | — | — |
| SSI_SD | IP2SR1[19:16] | SSI_SD | IRQ0 | — | — |
| AUDIO_CLKOUT | IP2SR1[23:20] | AUDIO_CLKOUT | IRQ1 | — | — |
| AUDIO_CLKIN | IP2SR1[27:24] | AUDIO_CLKIN | PWM3 | — | — |
| GP1_23 | IP2SR1[31:28] | — | TCLK2 | MSIOF4_SS1 | IRQ3 |
| HRX3 | IP3SR1[3:0] | HRX3 | SCK3 | MSIOF4_SS2 | — |
| HSCK3 | IP3SR1[7:4] | HSCK3 | CTS3# | MSIOF4_SCK | TPU0TO0 |
| HRTS3# | IP3SR1[11:8] | HRTS3# | RTS3# | MSIOF4_TXD | TPU0TO1 |
| HCTS3# | IP3SR1[15:12] | HCTS3# | RX3 | MSIOF4_RXD | — |
| HTX3 | IP3SR1[19:16] | HTX3 | TX3 | MSIOF4_SYNC | — |
| ERROROUTC# | IP3SR1[23:20] | ERROROUTC# | — | — | — |
| — | IP3SR1[27:24] | — | — | — | — |
| — | IP3SR1[31:28] | — | — | — | — |

**Table 7.12 Configuration of Registers in IP0SR2, IP1SR2, IP2SR2**

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| FXR_TXDA | IP0SR2[3:0] | FXR_TXDA | — | TPU0TO2 | — |
| FXR_TXENA# | IP0SR2[7:4] | FXR_TXENA# | — | TPU0TO3 | — |
| RXDA_EXTFXR | IP0SR2[11:8] | RXDA_EXTFXR | — | IRQ5 | — |
| CLK_EXTFXR | IP0SR2[15:12] | CLK_EXTFXR | — | IRQ4 | — |
| RXDB_EXTFXR | IP0SR2[19:16] | RXDB_EXTFXR | — | — | — |
| FXR_TXENB# | IP0SR2[23:20] | FXR_TXENB# | — | — | — |
| FXR_TXDB | IP0SR2[27:24] | FXR_TXDB | — | — | — |
| TPU0TO1 | IP0SR2[31:28] | TPU0TO1 | — | — | TCLK2 |
| TPU0TO0 | IP1SR2[3:0] | TPU0TO0 | — | — | TCLK1 |
| CAN_CLK | IP1SR2[7:4] | CAN_CLK | FXR_TXENA# | — | — |
| CANFD0_TX | IP1SR2[11:8] | CANFD0_TX | FXR_TXENB# | — | — |
| CANFD0_RX | IP1SR2[15:12] | CANFD0_RX | — | — | — |
| CANFD2_TX | IP1SR2[19:16] | CANFD2_TX | TPU0TO2 | — | TCLK3 |
| CANFD2_RX | IP1SR2[23:20] | CANFD2_RX | TPU0TO3 | PWM1 | TCLK4 |
| CANFD3_TX | IP1SR2[27:24] | CANFD3_TX | — | PWM2 | — |
| CANFD3_RX | IP1SR2[31:28] | CANFD3_RX | — | PWM3 | — |
| — | IP2SR2[3:0] | — | — | — | — |
| CANFD1_TX | IP2SR2[7:4] | CANFD1_TX | — | PWM1 | — |
| — | IP2SR2[11:8] | — | — | — | — |
| CANFD1_RX | IP2SR2[15:12] | CANFD1_RX | — | PWM2 | — |
| — | IP2SR2[19:16] | — | — | — | — |
| — | IP2SR2[23:20] | — | — | — | — |
| — | IP2SR2[27:24] | — | — | — | — |
| — | IP2SR2[31:28] | — | — | — | — |

**Table 7.13 Configuration of Registers in IP0SR3, IP1SR3, IP2SR3, IP3SR3**

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| MMC_SD_D1 | IP0SR3[3:0] | MMC_SD_D1 | — | — | — |
| MMC_SD_D0 | IP0SR3[7:4] | MMC_SD_D0 | — | — | — |
| MMC_SD_D2 | IP0SR3[11:8] | MMC_SD_D2 | — | — | — |
| MMC_SD_CLK | IP0SR3[15:12] | MMC_SD_CLK | — | — | — |
| MMC_DS | IP0SR3[19:16] | MMC_DS | — | — | — |
| MMC_SD_D3 | IP0SR3[23:20] | MMC_SD_D3 | — | — | — |
| MMC_D5 | IP0SR3[27:24] | MMC_D5 | — | — | — |
| MMC_D4 | IP0SR3[31:28] | MMC_D4 | — | — | — |
| MMC_D7 | IP1SR3[3:0] | MMC_D7 | — | — | — |
| MMC_D6 | IP1SR3[7:4] | MMC_D6 | — | — | — |
| MMC_SD_CMD | IP1SR3[11:8] | MMC_SD_CMD | — | — | — |
| SD_CD | IP1SR3[15:12] | SD_CD | — | — | — |

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| SD_WP | IP1SR3[19:16] | SD_WP | — | — | — |
| PWM1 | IP1SR3[23:20] | PWM1 | — | — | — |
| PWM2 | IP1SR3[27:24] | PWM2 | — | — | — |
| QSPI0_SSL | IP1SR3[31:28] | QSPI0_SSL | — | — | — |
| QSPI0_IO3 | IP2SR3[3:0] | QSPI0_IO3 | — | — | — |
| QSPI0_IO2 | IP2SR3[7:4] | QSPI0_IO2 | — | — | — |
| QSPI0_MISO_IO1 | IP2SR3[11:8] | QSPI0_MISO_IO1 | — | — | — |
| QSPI0_MOSI_IO0 | IP2SR3[15:12] | QSPI0_MOSI_IO0 | — | — | — |
| QSPI0_SPCLK | IP2SR3[19:16] | QSPI0_SPCLK | — | — | — |
| QSPI1_MOSI_IO0 | IP2SR3[23:20] | QSPI1_MOSI_IO0 | — | — | — |
| QSPI1_SPCLK | IP2SR3[27:24] | QSPI1_SPCLK | — | — | — |
| QSPI1_MISO_IO1 | IP2SR3[31:28] | QSPI1_MISO_IO1 | — | — | — |
| QSPI1_IO2 | IP3SR3[3:0] | QSPI1_IO2 | — | — | — |
| QSPI1_SSL | IP3SR3[7:4] | QSPI1_SSL | — | — | — |
| QSPI1_IO3 | IP3SR3[11:8] | QSPI1_IO3 | — | — | — |
| RPC_RESET# | IP3SR3[15:12] | RPC_RESET# | — | — | — |
| RPC_WP# | IP3SR3[19:16] | RPC_WP# | — | — | — |
| RPC_INT# | IP3SR3[23:20] | RPC_INT# | — | — | — |
| TCLK3 | IP3SR3[27:24] | TCLK3 | — | — | — |
| TCLK4 | IP3SR3[31:28] | TCLK4 | — | — | — |

**Table 7.14 Configuration of Registers in IP0SR4, IP1SR4, IP2SR4, IP3SR4**

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| SCL0 | IP0SR4[3:0] | SCL0 | — | — | — |
| SDA0 | IP0SR4[7:4] | SDA0 | — | — | — |
| SCL1 | IP0SR4[11:8] | SCL1 | — | — | — |
| SDA1 | IP0SR4[15:12] | SDA1 | — | — | — |
| SCL2 | IP0SR4[19:16] | SCL2 | — | — | — |
| SDA2 | IP0SR4[23:20] | SDA2 | — | — | — |
| SCL3 | IP0SR4[27:24] | SCL3 | — | — | — |
| SDA3 | IP0SR4[31:28] | SDA3 | — | — | — |
| HRX2 | IP1SR4[3:0] | HRX2 | SCK4 | — | — |
| HTX2 | IP1SR4[7:4] | HTX2 | CTS4# | — | — |
| HRTS2# | IP1SR4[11:8] | HRTS2# | RTS4# | — | — |
| SCIF_CLK2 | IP1SR4[15:12] | SCIF_CLK2 | — | — | — |
| HCTS2# | IP1SR4[19:16] | HCTS2# | TX4 | — | — |
| HSCK2 | IP1SR4[23:20] | HSCK2 | RX4 | — | — |
| PWM3 | IP1SR4[27:24] | PWM3 | — | — | — |
| PWM4 | IP1SR4[31:28] | PWM4 | — | — | — |
| — | IP2SR4[3:0] | — | — | — | — |
| — | IP2SR4[7:4] | — | — | — | — |
| — | IP2SR4[11:8] | — | — | — | — |
| — | IP2SR4[15:12] | — | — | — | — |
| — | IP2SR4[19:16] | — | — | — | — |
| PCIE0_CLKREQ# | IP2SR4[23:20] | PCIE0_CLKREQ# | — | — | — |
| — | IP2SR4[27:24] | — | — | — | — |
| AVS0 | IP2SR4[31:28] | AVS0 | — | — | — |
| AVS1 | IP3SR4[3:0] | AVS1 | — | — | — |
| — | IP3SR4[7:4] | — | — | — | — |
| — | IP3SR4[11:8] | — | — | — | — |
| — | IP3SR4[15:12] | — | — | — | — |
| — | IP3SR4[19:16] | — | — | — | — |
| — | IP3SR4[23:20] | — | — | — | — |
| — | IP3SR4[27:24] | — | — | — | — |
| — | IP3SR4[31:28] | — | — | — | — |

**Table 7.15 Configuration of Registers in IP0SR5, IP1SR5, IP2SR5**

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| AVB2_AVTP_PPS | IP0SR5[3:0] | AVB2_AVTP_PPS | Ether_GPTP_PPS0 | — | — |
| AVB2_AVTP_CAPTURE | IP0SR5[7:4] | AVB2_AVTP_CAPTURE | Ether_GPTP_CAPTURE | — | — |
| AVB2_AVTP_MATCH | IP0SR5[11:8] | AVB2_AVTP_MATCH | Ether_GPTP_MATCH | — | — |
| AVB2_LINK | IP0SR5[15:12] | AVB2_LINK | — | — | — |
| AVB2_PHY_INT | IP0SR5[19:16] | AVB2_PHY_INT | — | — | — |
| AVB2_MAGIC | IP0SR5[23:20] | AVB2_MAGIC | Ether_GPTP_PPS1 | — | — |
| AVB2_MDC | IP0SR5[27:24] | AVB2_MDC | — | — | — |
| AVB2_TXCREFCLK | IP0SR5[31:28] | AVB2_TXCREFCLK | — | — | — |

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| AVB2_TD3 | IP1SR5[3:0] | AVB2_TD3 | — | — | — |
| AVB2_RD3 | IP1SR5[7:4] | AVB2_RD3 | — | — | — |
| AVB2_MDIO | IP1SR5[11:8] | AVB2_MDIO | — | — | — |
| AVB2_TD2 | IP1SR5[15:12] | AVB2_TD2 | — | — | — |
| AVB2_TD1 | IP1SR5[19:16] | AVB2_TD1 | — | — | — |
| AVB2_RD2 | IP1SR5[23:20] | AVB2_RD2 | — | — | — |
| AVB2_RD1 | IP1SR5[27:24] | AVB2_RD1 | — | — | — |
| AVB2_TD0 | IP1SR5[31:28] | AVB2_TD0 | — | — | — |
| AVB2_TXC | IP2SR5[3:0] | AVB2_TXC | — | — | — |
| AVB2_RD0 | IP2SR5[7:4] | AVB2_RD0 | — | — | — |
| AVB2_RXC | IP2SR5[11:8] | AVB2_RXC | — | — | — |
| AVB2_TX_CTL | IP2SR5[15:12] | AVB2_TX_CTL | — | — | — |
| AVB2_RX_CTL | IP2SR5[19:16] | AVB2_RX_CTL | — | — | — |
| — | IP2SR5[23:20] | — | — | — | — |
| — | IP2SR5[27:24] | — | — | — | — |
| — | IP2SR5[31:28] | — | — | — | — |

**Table 7.16 Configuration of Registers in IP0SR6 , IP1SR6 , IP2SR6**

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| AVB1_MDIO | IP0SR6[3:0] | AVB1_MDIO | — | — | — |
| AVB1_MAGIC | IP0SR6[7:4] | AVB1_MAGIC | — | — | — |
| AVB1_MDC | IP0SR6[11:8] | AVB1_MDC | — | — | — |
| AVB1_PHY_INT | IP0SR6[15:12] | AVB1_PHY_INT | — | — | — |
| AVB1_LINK | IP0SR6[19:16] | AVB1_LINK | AVB1_MII_TX_ER | — | — |
| AVB1_AVTP_MATCH | IP0SR6[23:20] | AVB1_AVTP_MATCH | AVB1_MII_RX_ER | — | — |
| AVB1_TXC | IP0SR6[27:24] | AVB1_TXC | AVB1_MII_TXC | — | — |
| AVB1_TX_CTL | IP0SR6[31:28] | AVB1_TX_CTL | AVB1_MII_TX_EN | — | — |
| AVB1_RXC | IP1SR6[3:0] | AVB1_RXC | AVB1_MII_RXC | — | — |
| AVB1_RX_CTL | IP1SR6[7:4] | AVB1_RX_CTL | AVB1_MII_RX_DV | — | — |
| AVB1_AVTP_PPS | IP1SR6[11:8] | AVB1_AVTP_PPS | AVB1_MII_COL | — | — |
| AVB1_AVTP_CAPTURE | IP1SR6[15:12] | AVB1_AVTP_CAPTURE | AVB1_MII_CRS | — | — |
| AVB1_TD1 | IP1SR6[19:16] | AVB1_TD1 | AVB1_MII_TD1 | — | — |
| AVB1_TD0 | IP1SR6[23:20] | AVB1_TD0 | AVB1_MII_TD0 | — | — |
| AVB1_RD1 | IP1SR6[27:24] | AVB1_RD1 | AVB1_MII_RD1 | — | — |
| AVB1_RD0 | IP1SR6[31:28] | AVB1_RD0 | AVB1_MII_RD0 | — | — |
| AVB1_TD2 | IP2SR6[3:0] | AVB1_TD2 | AVB1_MII_TD2 | — | — |
| AVB1_RD2 | IP2SR6[7:4] | AVB1_RD2 | AVB1_MII_RD2 | — | — |
| AVB1_TD3 | IP2SR6[11:8] | AVB1_TD3 | AVB1_MII_TD3 | — | — |
| AVB1_RD3 | IP2SR6[15:12] | AVB1_RD3 | AVB1_MII_RD3 | — | — |
| AVB1_TXCREFCLK | IP2SR6[19:16] | AVB1_TXCREFCLK | — | — | — |
| — | IP2SR6[23:20] | — | — | — | — |
| — | IP2SR6[27:24] | — | — | — | — |
| — | IP2SR6[31:28] | — | — | — | — |

**Table 7.17 Configuration of Registers in IP0SR7 , IP1SR7 , IP2SR7**

| | | Register Value [4 bits] | | | |
|---|---|---|---|---|---|
| Pin name | Register | H'0 | H'1 | H'2 | H'3 |
| AVB0_AVTP_PPS | IP0SR7[3:0] | AVB0_AVTP_PPS | AVB0_MII_COL | — | — |
| AVB0_AVTP_CAPTURE | IP0SR7[7:4] | AVB0_AVTP_CAPTURE | AVB0_MII_CRS | — | — |
| AVB0_AVTP_MATCH | IP0SR7[11:8] | AVB0_AVTP_MATCH | AVB0_MII_RX_ER | — | — |
| AVB0_TD3 | IP0SR7[15:12] | AVB0_TD3 | AVB0_MII_TD3 | — | — |
| AVB0_LINK | IP0SR7[19:16] | AVB0_LINK | AVB0_MII_TX_ER | — | — |
| AVB0_PHY_INT | IP0SR7[23:20] | AVB0_PHY_INT | — | — | — |
| AVB0_TD2 | IP0SR7[27:24] | AVB0_TD2 | AVB0_MII_TD2 | — | — |
| AVB0_TD1 | IP0SR7[31:28] | AVB0_TD1 | AVB0_MII_TD1 | — | — |
| AVB0_RD3 | IP1SR7[3:0] | AVB0_RD3 | AVB0_MII_RD3 | — | — |
| AVB0_TXCREFCLK | IP1SR7[7:4] | AVB0_TXCREFCLK | — | — | — |
| AVB0_MAGIC | IP1SR7[11:8] | AVB0_MAGIC | — | — | — |
| AVB0_TD0 | IP1SR7[15:12] | AVB0_TD0 | AVB0_MII_TD0 | — | — |
| AVB0_RD2 | IP1SR7[19:16] | AVB0_RD2 | AVB0_MII_RD2 | — | — |
| AVB0_MDC | IP1SR7[23:20] | AVB0_MDC | — | — | — |
| AVB0_MDIO | IP1SR7[27:24] | AVB0_MDIO | — | — | — |
| AVB0_TXC | IP1SR7[31:28] | AVB0_TXC | AVB0_MII_TXC | — | — |
| AVB0_TX_CTL | IP2SR7[3:0] | AVB0_TX_CTL | AVB0_MII_TX_EN | — | — |
| AVB0_RD1 | IP2SR7[7:4] | AVB0_RD1 | AVB0_MII_RD1 | — | — |
| AVB0_RD0 | IP2SR7[11:8] | AVB0_RD0 | AVB0_MII_RD0 | — | — |
| AVB0_RXC | IP2SR7[15:12] | AVB0_RXC | AVB0_MII_RXC | — | — |
| AVB0_RX_CTL | IP2SR7[19:16] | AVB0_RX_CTL | AVB0_MII_RX_DV | — | — |
| — | IP2SR7[23:20] | — | — | — | — |
| — | IP2SR7[27:24] | — | — | — | — |
| — | IP2SR7[31:28] | — | — | — | — |

### 7.2.6 DRV Control Register 0-3 (DRV0CTRLn , DRV1CTRLn , DRV2CTRLn , DRV3CTRLn / DRV0CTRLSYS , DRV1CTRLSYS)

Function: DRViCTRLn / DRVjCTRLSYS controls the driving abilities of pins. This setting is related to only the output.

Note: i = 0 to 3 (Bus Domain) , j= 0 to 1 (Bus Domain)

**Table 7.19 DRV truth table**

Output buffer

| DRV1 | DRV2 | DRV3 | Drive capability | | |
|---|---|---|---|---|---|
| | | | Voltage type | | |
| | | | for 1.8V/3.3V | for 1.8V/2.5V | for 3.3V |
| L | L | L | Invalid | Invalid | Invalid |
| H | L | L | 2/8 | 2/8 | 2/8 |
| L | H | L | 3/8 | 3/8 | 3/8 |
| H | H | L | 4/8 | 4/8 | 4/8 |
| L | L | H | 5/8 | 5/8 | 5/8 |
| H | L | H | 6/8 | 6/8 | 6/8 |
| L | H | H | 7/8 | 7/8 | 7/8 |
| H | H | H | Full | Full | Full |

Output buffer

| DRV1 | DRV2 | Drive capability |
|---|---|---|
| | | Voltage type |
| | | for 1.8V |
| L | L | 1/4 |
| H | L | 2/4 |
| L | H | 3/4 |
| H | H | Full |

The initial value of DRViCTRLn as following:

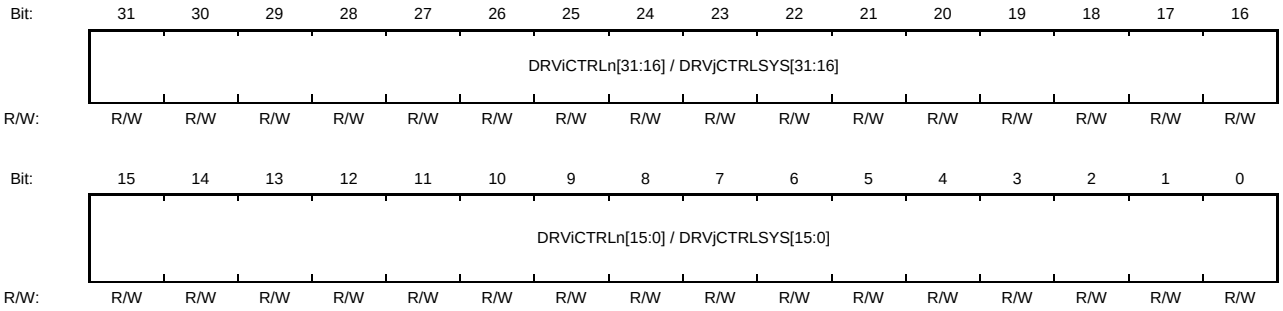
-IO 1.8V/3.3V : b’110

-IO 2.5V/3.3V : b’110

-IO 3.3V : b’110

-IO 1.8V (DCUTDO_LPDO, DCURDY#_LPDCLKOUT) : b’11

-IO 1.8V (Other than DCUTDO_LPDO, DCURDY#_LPDCLKOUT) : b’10




Note: i = 0 to 3 , j = 0 to 1

Note: To enable this register to be set, appropriately set the multiplexed pin setting mask register (PMMR) immediately before setting this register.

**Table 7.20 Configuration of Registers in DRV0CTRL0 , DRV1CTRL0 , DRV2CTRL0**

| | DRV0CTRL0 | | | DRV1CTRL0 | | | DRV2CTRL0 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | — | — | — | 0 |
| **bit30** | MSIOF5_SS2 | DRV3 | 1 | MSIOF2_SYNC | DRV3 | 1 | — | — | 0 |
| **bit29** | MSIOF5_SS2 | DRV2 | 1 | MSIOF2_SYNC | DRV2 | 1 | — | — | 0 |
| **bit28** | MSIOF5_SS2 | DRV1 | 0 | MSIOF2_SYNC | DRV1 | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit26** | IRQ0 | DRV3 | 1 | MSIOF2_SS1 | DRV3 | 1 | — | — | 0 |
| **bit25** | IRQ0 | DRV2 | 1 | MSIOF2_SS1 | DRV2 | 1 | — | — | 0 |
| **bit24** | IRQ0 | DRV1 | 0 | MSIOF2_SS1 | DRV1 | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit22** | IRQ1 | DRV3 | 1 | MSIOF2_SS2 | DRV3 | 1 | — | — | 0 |
| **bit21** | IRQ1 | DRV2 | 1 | MSIOF2_SS2 | DRV2 | 1 | — | — | 0 |
| **bit20** | IRQ1 | DRV1 | 0 | MSIOF2_SS2 | DRV1 | 0 | — | — | 0 |
| **bit19** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit18** | IRQ2 | DRV3 | 1 | MSIOF5_RXD | DRV3 | 1 | — | — | 0 |
| **bit17** | IRQ2 | DRV2 | 1 | MSIOF5_RXD | DRV2 | 1 | — | — | 0 |
| **bit16** | IRQ2 | DRV1 | 0 | MSIOF5_RXD | DRV1 | 0 | — | — | 0 |
| **bit15** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit14** | IRQ3 | DRV3 | 1 | MSIOF5_SCK | DRV3 | 1 | — | — | 0 |
| **bit13** | IRQ3 | DRV2 | 1 | MSIOF5_SCK | DRV2 | 1 | — | — | 0 |
| **bit12** | IRQ3 | DRV1 | 0 | MSIOF5_SCK | DRV1 | 0 | — | — | 0 |
| **bit11** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit10** | GP0_02 | DRV3 | 1 | MSIOF5_TXD | DRV3 | 1 | MSIOF2_RXD | DRV3 | 1 |
| **bit9** | GP0_02 | DRV2 | 1 | MSIOF5_TXD | DRV2 | 1 | MSIOF2_RXD | DRV2 | 1 |
| **bit8** | GP0_02 | DRV1 | 0 | MSIOF5_TXD | DRV1 | 0 | MSIOF2_RXD | DRV1 | 0 |
| **bit7** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit6** | GP0_01 | DRV3 | 1 | MSIOF5_SYNC | DRV3 | 1 | MSIOF2_SCK | DRV3 | 1 |
| **bit5** | GP0_01 | DRV2 | 1 | MSIOF5_SYNC | DRV2 | 1 | MSIOF2_SCK | DRV2 | 1 |
| **bit4** | GP0_01 | DRV1 | 0 | MSIOF5_SYNC | DRV1 | 0 | MSIOF2_SCK | DRV1 | 0 |
| **bit3** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit2** | GP0_00 | DRV3 | 1 | MSIOF5_SS1 | DRV3 | 1 | MSIOF2_TXD | DRV3 | 1 |
| **bit1** | GP0_00 | DRV2 | 1 | MSIOF5_SS1 | DRV2 | 1 | MSIOF2_TXD | DRV2 | 1 |
| **bit0** | GP0_00 | DRV1 | 0 | MSIOF5_SS1 | DRV1 | 0 | MSIOF2_TXD | DRV1 | 0 |

**Table 7.21 Configuration of Registers in DRV0CTRL1 , DRV1CTRL1 , DRV2CTRL1 , DRV3CTRL1**

| | DRV0CTRL1 | | | DRV1CTRL1 | | | DRV2CTRL1 | | | DRV3CTRL1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit30** | MSIOF0_SS1 | DRV3 | 1 | HSCK0 | DRV3 | 1 | GP1_23 | DRV3 | 1 | — | — | 0 |
| **bit29** | MSIOF0_SS1 | DRV2 | 1 | HSCK0 | DRV2 | 1 | GP1_23 | DRV2 | 1 | — | — | 0 |
| **bit28** | MSIOF0_SS1 | DRV1 | 0 | HSCK0 | DRV1 | 0 | GP1_23 | DRV1 | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit26** | MSIOF0_SS2 | DRV3 | 1 | HRTS0# | DRV3 | 1 | AUDIO_CLKIN | DRV3 | 1 | — | — | 0 |
| **bit25** | MSIOF0_SS2 | DRV2 | 1 | HRTS0# | DRV2 | 1 | AUDIO_CLKIN | DRV2 | 1 | — | — | 0 |
| **bit24** | MSIOF0_SS2 | DRV1 | 0 | HRTS0# | DRV1 | 0 | AUDIO_CLKIN | DRV1 | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit22** | MSIOF1_RXD | DRV3 | 1 | HCTS0# | DRV3 | 1 | AUDIO_CLKOUT | DRV3 | 1 | — | — | 0 |
| **bit21** | MSIOF1_RXD | DRV2 | 1 | HCTS0# | DRV2 | 1 | AUDIO_CLKOUT | DRV2 | 1 | ERROROUTC# | DRV2 | 1 |
| **bit20** | MSIOF1_RXD | DRV1 | 0 | HCTS0# | DRV1 | 0 | AUDIO_CLKOUT | DRV1 | 0 | ERROROUTC# | DRV1 | 0 |
| **bit19** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit18** | MSIOF1_TXD | DRV3 | 1 | HTX0 | DRV3 | 1 | SSI_SD | DRV3 | 1 | HTX3 | DRV3 | 1 |
| **bit17** | MSIOF1_TXD | DRV2 | 1 | HTX0 | DRV2 | 1 | SSI_SD | DRV2 | 1 | HTX3 | DRV2 | 1 |
| **bit16** | MSIOF1_TXD | DRV1 | 0 | HTX0 | DRV1 | 0 | SSI_SD | DRV1 | 0 | HTX3 | DRV1 | 0 |
| **bit15** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit14** | MSIOF1_SCK | DRV3 | 1 | MSIOF0_RXD | DRV3 | 1 | SSI_WS | DRV3 | 1 | HCTS3# | DRV3 | 1 |
| **bit13** | MSIOF1_SCK | DRV2 | 1 | MSIOF0_RXD | DRV2 | 1 | SSI_WS | DRV2 | 1 | HCTS3# | DRV2 | 1 |
| **bit12** | MSIOF1_SCK | DRV1 | 0 | MSIOF0_RXD | DRV1 | 0 | SSI_WS | DRV1 | 0 | HCTS3# | DRV1 | 0 |
| **bit11** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit10** | MSIOF1_SYNC | DRV3 | 1 | MSIOF0_SCK | DRV3 | 1 | SSI_SCK | DRV3 | 1 | HRTS3# | DRV3 | 1 |
| **bit9** | MSIOF1_SYNC | DRV2 | 1 | MSIOF0_SCK | DRV2 | 1 | SSI_SCK | DRV2 | 1 | HRTS3# | DRV2 | 1 |
| **bit8** | MSIOF1_SYNC | DRV1 | 0 | MSIOF0_SCK | DRV1 | 0 | SSI_SCK | DRV1 | 0 | HRTS3# | DRV1 | 0 |
| **bit7** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit6** | MSIOF1_SS1 | DRV3 | 1 | MSIOF0_TXD | DRV3 | 1 | SCIF_CLK | DRV3 | 1 | HSCK3 | DRV3 | 1 |
| **bit5** | MSIOF1_SS1 | DRV2 | 1 | MSIOF0_TXD | DRV2 | 1 | SCIF_CLK | DRV2 | 1 | HSCK3 | DRV2 | 1 |
| **bit4** | MSIOF1_SS1 | DRV1 | 0 | MSIOF0_TXD | DRV1 | 0 | SCIF_CLK | DRV1 | 0 | HSCK3 | DRV1 | 0 |
| **bit3** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit2** | MSIOF1_SS2 | DRV3 | 1 | MSIOF0_SYNC | DRV3 | 1 | HRX0 | DRV3 | 1 | HRX3 | DRV3 | 1 |
| **bit1** | MSIOF1_SS2 | DRV2 | 1 | MSIOF0_SYNC | DRV2 | 1 | HRX0 | DRV2 | 1 | HRX3 | DRV2 | 1 |
| **bit0** | MSIOF1_SS2 | DRV1 | 0 | MSIOF0_SYNC | DRV1 | 0 | HRX0 | DRV1 | 0 | HRX3 | DRV1 | 0 |

**Table 7.22 Configuration of Registers in DRV0CTRL2 , DRV1CTRL2 , DRV2CTRL2**

| | DRV0CTRL2 | | | DRV1CTRL2 | | | DRV2CTRL2 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit30** | TPU0TO1 | DRV3 | 1 | CANFD3_RX | DRV3 | 1 | — | — | 0 |
| **bit29** | TPU0TO1 | DRV2 | 1 | CANFD3_RX | DRV2 | 1 | — | — | 0 |
| **bit28** | TPU0TO1 | DRV1 | 0 | CANFD3_RX | DRV1 | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit26** | FXR_TXDB | DRV3 | 1 | CANFD3_TX | DRV3 | 1 | — | — | 0 |
| **bit25** | FXR_TXDB | DRV2 | 1 | CANFD3_TX | DRV2 | 1 | — | — | 0 |
| **bit24** | FXR_TXDB | DRV1 | 0 | CANFD3_TX | DRV1 | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit22** | FXR_TXENB# | DRV3 | 1 | CANFD2_RX | DRV3 | 1 | — | — | 0 |
| **bit21** | FXR_TXENB# | DRV2 | 1 | CANFD2_RX | DRV2 | 1 | — | — | 0 |
| **bit20** | FXR_TXENB# | DRV1 | 0 | CANFD2_RX | DRV1 | 0 | — | — | 0 |
| **bit19** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit18** | RXDB_EXTFXR | DRV3 | 1 | CANFD2_TX | DRV3 | 1 | — | — | 0 |
| **bit17** | RXDB_EXTFXR | DRV2 | 1 | CANFD2_TX | DRV2 | 1 | — | — | 0 |
| **bit16** | RXDB_EXTFXR | DRV1 | 0 | CANFD2_TX | DRV1 | 0 | — | — | 0 |
| **bit15** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit14** | CLK_EXTFXR | DRV3 | 1 | CANFD0_RX | DRV3 | 1 | CANFD1_RX | DRV3 | 1 |
| **bit13** | CLK_EXTFXR | DRV2 | 1 | CANFD0_RX | DRV2 | 1 | CANFD1_RX | DRV2 | 1 |
| **bit12** | CLK_EXTFXR | DRV1 | 0 | CANFD0_RX | DRV1 | 0 | CANFD1_RX | DRV1 | 0 |
| **bit11** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit10** | RXDA_EXTFXR | DRV3 | 1 | CANFD0_TX | DRV3 | 1 | — | — | 0 |
| **bit9** | RXDA_EXTFXR | DRV2 | 1 | CANFD0_TX | DRV2 | 1 | — | — | 0 |
| **bit8** | RXDA_EXTFXR | DRV1 | 0 | CANFD0_TX | DRV1 | 0 | — | — | 0 |
| **bit7** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit6** | FXR_TXENA# | DRV3 | 1 | CAN_CLK | DRV3 | 1 | CANFD1_TX | DRV3 | 1 |
| **bit5** | FXR_TXENA# | DRV2 | 1 | CAN_CLK | DRV2 | 1 | CANFD1_TX | DRV2 | 1 |
| **bit4** | FXR_TXENA# | DRV1 | 0 | CAN_CLK | DRV1 | 0 | CANFD1_TX | DRV1 | 0 |
| **bit3** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit2** | FXR_TXDA | DRV3 | 1 | TPU0TO0 | DRV3 | 1 | — | — | 0 |
| **bit1** | FXR_TXDA | DRV2 | 1 | TPU0TO0 | DRV2 | 1 | — | — | 0 |
| **bit0** | FXR_TXDA | DRV1 | 0 | TPU0TO0 | DRV1 | 0 | — | — | 0 |

**Table 7.23 Configuration of Registers in DRV0CTRL3 , DRV1CTRL3 , DRV2CTRL3 , DRV3CTRL3**

| | DRV0CTRL3 | | | DRV1CTRL3 | | | DRV2CTRL3 | | | DRV3CTRL3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit30** | MMC_D4 | DRV3 | 1 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit29** | MMC_D4 | DRV2 | 1 | QSPI0_SSL | DRV2 | 1 | QSPI1_MISO_IO1 | DRV2 | 1 | TCLK4 | DRV2 | 1 |
| **bit28** | MMC_D4 | DRV1 | 0 | QSPI0_SSL | DRV1 | 0 | QSPI1_MISO_IO1 | DRV1 | 0 | TCLK4 | DRV1 | 0 |
| **bit27** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit26** | MMC_D5 | DRV3 | 1 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit25** | MMC_D5 | DRV2 | 1 | PWM2 | DRV2 | 1 | QSPI1_SPCLK | DRV2 | 1 | TCLK3 | DRV2 | 1 |
| **bit24** | MMC_D5 | DRV1 | 0 | PWM2 | DRV1 | 0 | QSPI1_SPCLK | DRV1 | 0 | TCLK3 | DRV1 | 0 |
| **bit23** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit22** | MMC_SD_D3 | DRV3 | 1 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit21** | MMC_SD_D3 | DRV2 | 1 | PWM1 | DRV2 | 1 | QSPI1_MOSI_IO0 | DRV2 | 1 | RPC_INT# | DRV2 | 1 |
| **bit20** | MMC_SD_D3 | DRV1 | 0 | PWM1 | DRV1 | 0 | QSPI1_MOSI_IO0 | DRV1 | 0 | RPC_INT# | DRV1 | 0 |
| **bit19** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit18** | MMC_DS | DRV3 | 1 | SD_WP | DRV3 | 1 | — | — | 0 | — | — | 0 |
| **bit17** | MMC_DS | DRV2 | 1 | SD_WP | DRV2 | 1 | QSPI0_SPCLK | DRV2 | 1 | RPC_WP# | DRV2 | 1 |
| **bit16** | MMC_DS | DRV1 | 0 | SD_WP | DRV1 | 0 | QSPI0_SPCLK | DRV1 | 0 | RPC_WP# | DRV1 | 0 |
| **bit15** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit14** | MMC_SD_CLK | DRV3 | 1 | SD_CD | DRV3 | 1 | — | — | 0 | — | — | 0 |
| **bit13** | MMC_SD_CLK | DRV2 | 1 | SD_CD | DRV2 | 1 | QSPI0_MOSI_IO0 | DRV2 | 1 | RPC_RESET# | DRV2 | 1 |
| **bit12** | MMC_SD_CLK | DRV1 | 0 | SD_CD | DRV1 | 0 | QSPI0_MOSI_IO0 | DRV1 | 0 | RPC_RESET# | DRV1 | 0 |
| **bit11** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit10** | MMC_SD_D2 | DRV3 | 1 | MMC_SD_CMD | DRV3 | 1 | — | — | 0 | — | — | 0 |
| **bit9** | MMC_SD_D2 | DRV2 | 1 | MMC_SD_CMD | DRV2 | 1 | QSPI0_MISO_IO1 | DRV2 | 1 | QSPI1_IO3 | DRV2 | 1 |
| **bit8** | MMC_SD_D2 | DRV1 | 0 | MMC_SD_CMD | DRV1 | 0 | QSPI0_MISO_IO1 | DRV1 | 0 | QSPI1_IO3 | DRV1 | 0 |
| **bit7** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit6** | MMC_SD_D0 | DRV3 | 1 | MMC_D6 | DRV3 | 1 | — | — | 0 | — | — | 0 |
| **bit5** | MMC_SD_D0 | DRV2 | 1 | MMC_D6 | DRV2 | 1 | QSPI0_IO2 | DRV2 | 1 | QSPI1_SSL | DRV2 | 1 |
| **bit4** | MMC_SD_D0 | DRV1 | 0 | MMC_D6 | DRV1 | 0 | QSPI0_IO2 | DRV1 | 0 | QSPI1_SSL | DRV1 | 0 |
| **bit3** | — | — | 0 | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit2** | MMC_SD_D1 | DRV3 | 1 | MMC_D7 | DRV3 | 1 | — | — | 0 | — | — | 0 |
| **bit1** | MMC_SD_D1 | DRV2 | 1 | MMC_D7 | DRV2 | 1 | QSPI0_IO3 | DRV2 | 1 | QSPI1_IO2 | DRV2 | 1 |
| **bit0** | MMC_SD_D1 | DRV1 | 0 | MMC_D7 | DRV1 | 0 | QSPI0_IO3 | DRV1 | 0 | QSPI1_IO2 | DRV1 | 0 |

**Table 7.24 Configuration of Registers in DRV0CTRL4 , DRV1CTRL4 , DRV2CTRL4 , DRV3CTRL4**

| | DRV0CTRL4 | | | DRV1CTRL4 | | |
|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | 0 |
| **bit30** | SDA3 | DRV3 | 1 | PWM4 | DRV3 | 1 |
| **bit29** | SDA3 | DRV2 | 1 | PWM4 | DRV2 | 1 |
| **bit28** | SDA3 | DRV1 | 0 | PWM4 | DRV1 | 0 |
| **bit27** | — | — | 0 | — | — | 0 |
| **bit26** | SCL3 | DRV3 | 1 | PWM3 | DRV3 | 1 |
| **bit25** | SCL3 | DRV2 | 1 | PWM3 | DRV2 | 1 |
| **bit24** | SCL3 | DRV1 | 0 | PWM3 | DRV1 | 0 |
| **bit23** | — | — | 0 | — | — | 0 |
| **bit22** | SDA2 | DRV3 | 1 | HSCK2 | DRV3 | 1 |
| **bit21** | SDA2 | DRV2 | 1 | HSCK2 | DRV2 | 1 |
| **bit20** | SDA2 | DRV1 | 0 | HSCK2 | DRV1 | 0 |
| **bit19** | — | — | 0 | — | — | 0 |
| **bit18** | SCL2 | DRV3 | 1 | HCTS2# | DRV3 | 1 |
| **bit17** | SCL2 | DRV2 | 1 | HCTS2# | DRV2 | 1 |
| **bit16** | SCL2 | DRV1 | 0 | HCTS2# | DRV1 | 0 |
| **bit15** | — | — | 0 | — | — | 0 |
| **bit14** | SDA1 | DRV3 | 1 | SCIF_CLK2 | DRV3 | 1 |
| **bit13** | SDA1 | DRV2 | 1 | SCIF_CLK2 | DRV2 | 1 |
| **bit12** | SDA1 | DRV1 | 0 | SCIF_CLK2 | DRV1 | 0 |
| **bit11** | — | — | 0 | — | — | 0 |
| **bit10** | SCL1 | DRV3 | 1 | HRTS2# | DRV3 | 1 |
| **bit9** | SCL1 | DRV2 | 1 | HRTS2# | DRV2 | 1 |
| **bit8** | SCL1 | DRV1 | 0 | HRTS2# | DRV1 | 0 |
| **bit7** | — | — | 0 | — | — | 0 |
| **bit6** | SDA0 | DRV3 | 1 | HTX2 | DRV3 | 1 |
| **bit5** | SDA0 | DRV2 | 1 | HTX2 | DRV2 | 1 |
| **bit4** | SDA0 | DRV1 | 0 | HTX2 | DRV1 | 0 |
| **bit3** | — | — | 0 | — | — | 0 |
| **bit2** | SCL0 | DRV3 | 1 | HRX2 | DRV3 | 1 |
| **bit1** | SCL0 | DRV2 | 1 | HRX2 | DRV2 | 1 |
| **bit0** | SCL0 | DRV1 | 0 | HRX2 | DRV1 | 0 |

| | DRV2CTRL4 | | | DRV3CTRL4 | | |
|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | — |
| **bit30** | AVS0 | DRV3 | 1 | — | — | — |
| **bit29** | AVS0 | DRV2 | 1 | — | — | — |
| **bit28** | AVS0 | DRV1 | 0 | — | — | — |
| **bit27** | — | — | 0 | — | — | — |
| **bit26** | — | — | 0 | — | — | — |
| **bit25** | — | — | 0 | — | — | — |
| **bit24** | — | — | 0 | — | — | — |
| **bit23** | — | — | 0 | — | — | — |
| **bit22** | PCIE0_CLKREQ# | DRV3 | 1 | — | — | — |
| **bit21** | PCIE0_CLKREQ# | DRV2 | 1 | — | — | — |
| **bit20** | PCIE0_CLKREQ# | DRV1 | 0 | — | — | — |
| **bit19** | — | — | 0 | — | — | — |
| **bit18** | — | — | 0 | — | — | — |
| **bit17** | — | — | 0 | — | — | — |
| **bit16** | — | — | 0 | — | — | — |
| **bit15** | — | — | 0 | — | — | — |
| **bit14** | — | — | 0 | — | — | — |
| **bit13** | — | — | 0 | — | — | — |
| **bit12** | — | — | 0 | — | — | — |
| **bit11** | — | — | 0 | — | — | — |
| **bit10** | — | — | 0 | — | — | — |
| **bit9** | — | — | 0 | — | — | — |
| **bit8** | — | — | 0 | — | — | — |
| **bit7** | — | — | 0 | — | — | — |
| **bit6** | — | — | 0 | — | — | — |
| **bit5** | — | — | 0 | — | — | — |
| **bit4** | — | — | 0 | — | — | — |
| **bit3** | — | — | 0 | — | — | — |
| **bit2** | — | — | 0 | AVS1 | DRV3 | 1 |
| **bit1** | — | — | 0 | AVS1 | DRV2 | 1 |
| **bit0** | — | — | 0 | AVS1 | DRV1 | 0 |

**Table 7.25 Configuration of Registers in DRV0CTRL5 , DRV1CTRL5 , DRV2CTRL5**

| | DRV0CTRL5 | | | DRV1CTRL5 | | | DRV2CTRL5 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit30** | AVB2_TXCREFCLK | DRV3 | 1 | AVB2_TD0 | DRV3 | 1 | — | — | 0 |
| **bit29** | AVB2_TXCREFCLK | DRV2 | 1 | AVB2_TD0 | DRV2 | 1 | — | — | 0 |
| **bit28** | AVB2_TXCREFCLK | DRV1 | 0 | AVB2_TD0 | DRV1 | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit26** | AVB2_MDC | DRV3 | 1 | AVB2_RD1 | DRV3 | 1 | — | — | 0 |
| **bit25** | AVB2_MDC | DRV2 | 1 | AVB2_RD1 | DRV2 | 1 | — | — | 0 |
| **bit24** | AVB2_MDC | DRV1 | 0 | AVB2_RD1 | DRV1 | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit22** | AVB2_MAGIC | DRV3 | 1 | AVB2_RD2 | DRV3 | 1 | — | — | 0 |
| **bit21** | AVB2_MAGIC | DRV2 | 1 | AVB2_RD2 | DRV2 | 1 | — | — | 0 |
| **bit20** | AVB2_MAGIC | DRV1 | 0 | AVB2_RD2 | DRV1 | 0 | — | — | 0 |
| **bit19** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit18** | AVB2_PHY_INT | DRV3 | 1 | AVB2_TD1 | DRV3 | 1 | AVB2_RX_CTL | DRV3 | 1 |
| **bit17** | AVB2_PHY_INT | DRV2 | 1 | AVB2_TD1 | DRV2 | 1 | AVB2_RX_CTL | DRV2 | 1 |
| **bit16** | AVB2_PHY_INT | DRV1 | 0 | AVB2_TD1 | DRV1 | 0 | AVB2_RX_CTL | DRV1 | 0 |
| **bit15** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit14** | AVB2_LINK | DRV3 | 1 | AVB2_TD2 | DRV3 | 1 | AVB2_TX_CTL | DRV3 | 1 |
| **bit13** | AVB2_LINK | DRV2 | 1 | AVB2_TD2 | DRV2 | 1 | AVB2_TX_CTL | DRV2 | 1 |
| **bit12** | AVB2_LINK | DRV1 | 0 | AVB2_TD2 | DRV1 | 0 | AVB2_TX_CTL | DRV1 | 0 |
| **bit11** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit10** | AVB2_AVTP_MATCH | DRV3 | 1 | AVB2_MDIO | DRV3 | 1 | AVB2_RXC | DRV3 | 1 |
| **bit9** | AVB2_AVTP_MATCH | DRV2 | 1 | AVB2_MDIO | DRV2 | 1 | AVB2_RXC | DRV2 | 1 |
| **bit8** | AVB2_AVTP_MATCH | DRV1 | 0 | AVB2_MDIO | DRV1 | 0 | AVB2_RXC | DRV1 | 0 |
| **bit7** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit6** | AVB2_AVTP_CAPTURE | DRV3 | 1 | AVB2_RD3 | DRV3 | 1 | AVB2_RD0 | DRV3 | 1 |
| **bit5** | AVB2_AVTP_CAPTURE | DRV2 | 1 | AVB2_RD3 | DRV2 | 1 | AVB2_RD0 | DRV2 | 1 |
| **bit4** | AVB2_AVTP_CAPTURE | DRV1 | 0 | AVB2_RD3 | DRV1 | 0 | AVB2_RD0 | DRV1 | 0 |
| **bit3** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit2** | AVB2_AVTP_PPS | DRV3 | 1 | AVB2_TD3 | DRV3 | 1 | AVB2_TXC | DRV3 | 1 |
| **bit1** | AVB2_AVTP_PPS | DRV2 | 1 | AVB2_TD3 | DRV2 | 1 | AVB2_TXC | DRV2 | 1 |
| **bit0** | AVB2_AVTP_PPS | DRV1 | 0 | AVB2_TD3 | DRV1 | 0 | AVB2_TXC | DRV1 | 0 |

**Table 7.26 Configuration of Registers in DRV0CTRL6 , DRV1CTRL6 , DRV2CTRL6**

| | DRV0CTRL6 | | | DRV1CTRL6 | | | DRV2CTRL6 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit30** | AVB1_TX_CTL | DRV3 | 1 | AVB1_RD0 | DRV3 | 1 | — | — | 0 |
| **bit29** | AVB1_TX_CTL | DRV2 | 1 | AVB1_RD0 | DRV2 | 1 | — | — | 0 |
| **bit28** | AVB1_TX_CTL | DRV1 | 0 | AVB1_RD0 | DRV1 | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit26** | AVB1_TXC | DRV3 | 1 | AVB1_RD1 | DRV3 | 1 | — | — | 0 |
| **bit25** | AVB1_TXC | DRV2 | 1 | AVB1_RD1 | DRV2 | 1 | — | — | 0 |
| **bit24** | AVB1_TXC | DRV1 | 0 | AVB1_RD1 | DRV1 | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit22** | AVB1_AVTP_MATCH | DRV3 | 1 | AVB1_TD0 | DRV3 | 1 | — | — | 0 |
| **bit21** | AVB1_AVTP_MATCH | DRV2 | 1 | AVB1_TD0 | DRV2 | 1 | — | — | 0 |
| **bit20** | AVB1_AVTP_MATCH | DRV1 | 0 | AVB1_TD0 | DRV1 | 0 | — | — | 0 |
| **bit19** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit18** | AVB1_LINK | DRV3 | 1 | AVB1_TD1 | DRV3 | 1 | AVB1_TXCREFCLK | DRV3 | 1 |
| **bit17** | AVB1_LINK | DRV2 | 1 | AVB1_TD1 | DRV2 | 1 | AVB1_TXCREFCLK | DRV2 | 1 |
| **bit16** | AVB1_LINK | DRV1 | 0 | AVB1_TD1 | DRV1 | 0 | AVB1_TXCREFCLK | DRV1 | 0 |
| **bit15** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit14** | AVB1_PHY_INT | DRV3 | 1 | AVB1_AVTP_CAPTURE | DRV3 | 1 | AVB1_RD3 | DRV3 | 1 |
| **bit13** | AVB1_PHY_INT | DRV2 | 1 | AVB1_AVTP_CAPTURE | DRV2 | 1 | AVB1_RD3 | DRV2 | 1 |
| **bit12** | AVB1_PHY_INT | DRV1 | 0 | AVB1_AVTP_CAPTURE | DRV1 | 0 | AVB1_RD3 | DRV1 | 0 |
| **bit11** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit10** | AVB1_MDC | DRV3 | 1 | AVB1_AVTP_PPS | DRV3 | 1 | AVB1_TD3 | DRV3 | 1 |
| **bit9** | AVB1_MDC | DRV2 | 1 | AVB1_AVTP_PPS | DRV2 | 1 | AVB1_TD3 | DRV2 | 1 |
| **bit8** | AVB1_MDC | DRV1 | 0 | AVB1_AVTP_PPS | DRV1 | 0 | AVB1_TD3 | DRV1 | 0 |
| **bit7** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit6** | AVB1_MAGIC | DRV3 | 1 | AVB1_RX_CTL | DRV3 | 1 | AVB1_RD2 | DRV3 | 1 |
| **bit5** | AVB1_MAGIC | DRV2 | 1 | AVB1_RX_CTL | DRV2 | 1 | AVB1_RD2 | DRV2 | 1 |
| **bit4** | AVB1_MAGIC | DRV1 | 0 | AVB1_RX_CTL | DRV1 | 0 | AVB1_RD2 | DRV1 | 0 |
| **bit3** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit2** | AVB1_MDIO | DRV3 | 1 | AVB1_RXC | DRV3 | 1 | AVB1_TD2 | DRV3 | 1 |
| **bit1** | AVB1_MDIO | DRV2 | 1 | AVB1_RXC | DRV2 | 1 | AVB1_TD2 | DRV2 | 1 |
| **bit0** | AVB1_MDIO | DRV1 | 0 | AVB1_RXC | DRV1 | 0 | AVB1_TD2 | DRV1 | 0 |

**Table 7.27 Configuration of Registers in DRV0CTRL7 , DRV1CTRL7 , DRV2CTRL7**

| | DRV0CTRL7 | | | DRV1CTRL7 | | | DRV2CTRL7 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit30** | AVB0_TD1 | DRV3 | 1 | AVB0_TXC | DRV3 | 1 | — | — | 0 |
| **bit29** | AVB0_TD1 | DRV2 | 1 | AVB0_TXC | DRV2 | 1 | — | — | 0 |
| **bit28** | AVB0_TD1 | DRV1 | 0 | AVB0_TXC | DRV1 | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit26** | AVB0_TD2 | DRV3 | 1 | AVB0_MDIO | DRV3 | 1 | — | — | 0 |
| **bit25** | AVB0_TD2 | DRV2 | 1 | AVB0_MDIO | DRV2 | 1 | — | — | 0 |
| **bit24** | AVB0_TD2 | DRV1 | 0 | AVB0_MDIO | DRV1 | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit22** | AVB0_PHY_INT | DRV3 | 1 | AVB0_MDC | DRV3 | 1 | — | — | 0 |
| **bit21** | AVB0_PHY_INT | DRV2 | 1 | AVB0_MDC | DRV2 | 1 | — | — | 0 |
| **bit20** | AVB0_PHY_INT | DRV1 | 0 | AVB0_MDC | DRV1 | 0 | — | — | 0 |
| **bit19** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit18** | AVB0_LINK | DRV3 | 1 | AVB0_RD2 | DRV3 | 1 | AVB0_RX_CTL | DRV3 | 1 |
| **bit17** | AVB0_LINK | DRV2 | 1 | AVB0_RD2 | DRV2 | 1 | AVB0_RX_CTL | DRV2 | 1 |
| **bit16** | AVB0_LINK | DRV1 | 0 | AVB0_RD2 | DRV1 | 0 | AVB0_RX_CTL | DRV1 | 0 |
| **bit15** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit14** | AVB0_TD3 | DRV3 | 1 | AVB0_TD0 | DRV3 | 1 | AVB0_RXC | DRV3 | 1 |
| **bit13** | AVB0_TD3 | DRV2 | 1 | AVB0_TD0 | DRV2 | 1 | AVB0_RXC | DRV2 | 1 |
| **bit12** | AVB0_TD3 | DRV1 | 0 | AVB0_TD0 | DRV1 | 0 | AVB0_RXC | DRV1 | 0 |
| **bit11** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit10** | AVB0_AVTP_MATCH | DRV3 | 1 | AVB0_MAGIC | DRV3 | 1 | AVB0_RD0 | DRV3 | 1 |
| **bit9** | AVB0_AVTP_MATCH | DRV2 | 1 | AVB0_MAGIC | DRV2 | 1 | AVB0_RD0 | DRV2 | 1 |
| **bit8** | AVB0_AVTP_MATCH | DRV1 | 0 | AVB0_MAGIC | DRV1 | 0 | AVB0_RD0 | DRV1 | 0 |
| **bit7** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit6** | AVB0_AVTP_CAPTURE | DRV3 | 1 | AVB0_TXCREFCLK | DRV3 | 1 | AVB0_RD1 | DRV3 | 1 |
| **bit5** | AVB0_AVTP_CAPTURE | DRV2 | 1 | AVB0_TXCREFCLK | DRV2 | 1 | AVB0_RD1 | DRV2 | 1 |
| **bit4** | AVB0_AVTP_CAPTURE | DRV1 | 0 | AVB0_TXCREFCLK | DRV1 | 0 | AVB0_RD1 | DRV1 | 0 |
| **bit3** | — | — | 0 | — | — | 0 | — | — | 0 |
| **bit2** | AVB0_AVTP_PPS | DRV3 | 1 | AVB0_RD3 | DRV3 | 1 | AVB0_TX_CTL | DRV3 | 1 |
| **bit1** | AVB0_AVTP_PPS | DRV2 | 1 | AVB0_RD3 | DRV2 | 1 | AVB0_TX_CTL | DRV2 | 1 |
| **bit0** | AVB0_AVTP_PPS | DRV1 | 0 | AVB0_RD3 | DRV1 | 0 | AVB0_TX_CTL | DRV1 | 0 |

**Table 7.29 Configuration of Registers in DRV0CTRLSYS, DRV1CTRLSYS**

| | DRV0CTRLSYS | | | DRV1CTRLSYS | | |
|---|---|---|---|---|---|---|
| | Pin name | | Initial value | Pin name | | Initial value |
| **bit31** | — | — | 0 | — | — | 0 |
| **bit30** | — | — | 0 | — | — | 0 |
| **bit29** | — | — | 0 | — | — | 0 |
| **bit28** | — | — | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 |
| **bit26** | — | — | 0 | — | — | 0 |
| **bit25** | — | — | 0 | — | — | 0 |
| **bit24** | — | — | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 |
| **bit22** | — | — | 0 | — | — | 0 |
| **bit21** | — | — | 0 | — | — | 0 |
| **bit20** | — | — | 0 | — | — | 0 |
| **bit19** | — | — | 0 | — | — | 0 |
| **bit18** | — | — | 0 | — | — | 0 |
| **bit17** | — | — | 0 | — | — | 0 |
| **bit16** | — | — | 0 | — | — | 0 |
| **bit15** | — | — | 0 | — | — | 0 |
| **bit14** | — | — | 0 | — | — | 0 |
| **bit13** | — | — | 0 | DCURDY#_LPDCLKOUT | DRV2 | 1 |
| **bit12** | — | — | 0 | — | DRV1 | 1 |
| **bit11** | — | — | 0 | — | — | 0 |
| **bit10** | | | 1 | — | — | 0 |
| **bit9** | | | 0 | DCUTDO_LPDO | DRV2 | 1 |
| **bit8** | | | 0 | | DRV1 | 1 |
| **bit7** | — | — | 0 | — | — | 0 |
| **bit6** | — | — | 0 | — | — | 0 |
| **bit5** | — | — | 0 | — | — | 0 |
| **bit4** | — | — | 0 | — | — | 0 |
| **bit3** | — | — | 0 | — | — | 0 |
| **bit2** | PRESETOUT# | DRV3 | 1 | — | — | 0 |
| **bit1** | | DRV2 | 1 | DCUTMS | DRV2 | 1 |
| **bit0** | | DRV1 | 0 | | DRV1 | 0 |

### 7.2.7 POWER Condition Control Register (POCn)

Function: Each bit in POCn must be set according to IO voltage level that is supplied to the pin.
About the how to switch voltage value, refer at “7.3.10 Switching signal level voltage”.

0: 1.8V 1: 3.3V

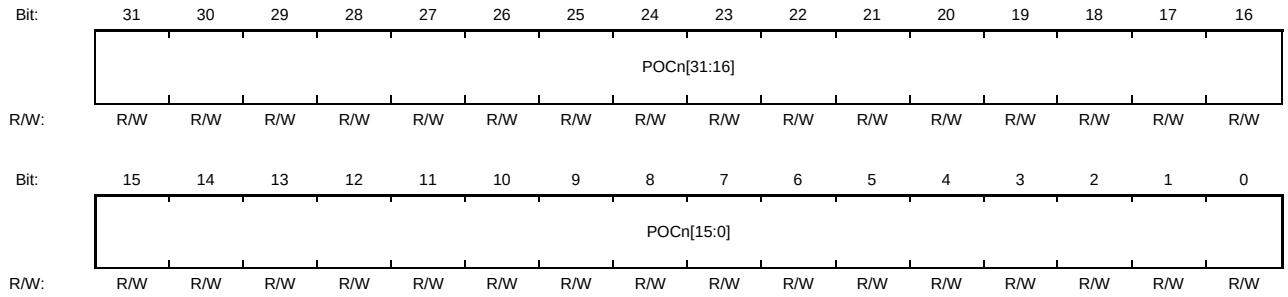
| Supply voltage | Setting register | Usage |
|---|---|---|
| 1.8V | 0 | Possible |
| 1.8V | 1 | Possible (limited*) |
| 3.3V | 0 | Do not set (broken) |
| 3.3V | 1 | Possible |

* : Only the operation to capture Mode Pin value is guaranteed. Other usage is not guaranteed.

0: 1.8V 1: 2.5V

| Supply voltage | Setting register | Usage |
|---|---|---|
| 1.8V | 0 | Possible |
| 1.8V | 1 | Possible (limited*) |
| 2.5V | 0 | Do not set (broken) |
| 2.5V | 1 | Possible |

* : Only the operation to capture Mode Pin value is guaranteed. Other usage is not guaranteed.




Note: To enable this register to be set, appropriately set the multiplexed pin setting mask register (PMMR) immediately before setting this register.

Note: The power domain is common for each function.

Note: When any GPIO are not used, they must be set POC register’s bits fitting with their Power Domain.

Note: No register setting is necessary in case of supplying 3.3V or 2.5V.

**Table 7.30 Configuration of Registers in POC0 , POC1**

| | POC0 | | | POC1 | | |
|---|---|---|---|---|---|---|
| | Pin name | Voltage | Initial value | Pin name | Voltage | Initial value |
| **bit31** | — | — | 0 | — | — | 0 |
| **bit30** | — | — | 0 | — | — | 0 |
| **bit29** | — | — | 0 | — | — | 0 |
| **bit28** | — | — | 0 | HTX3 | 1.8 / 3.3 | 1 |
| **bit27** | — | — | 0 | HCTS3# | 1.8 / 3.3 | 1 |
| **bit26** | — | — | 0 | HRTS3# | 1.8 / 3.3 | 1 |
| **bit25** | — | — | 0 | HSCK3 | 1.8 / 3.3 | 1 |
| **bit24** | — | — | 0 | HRX3 | 1.8 / 3.3 | 1 |
| **bit23** | — | — | 0 | GP1_23 | 1.8 / 3.3 | 1 |
| **bit22** | — | — | 0 | AUDIO_CLKIN | 1.8 / 3.3 | 1 |
| **bit21** | — | — | 0 | AUDIO_CLKOUT | 1.8 / 3.3 | 1 |
| **bit20** | — | — | 0 | SSI_SD | 1.8 / 3.3 | 1 |
| **bit19** | — | — | 0 | SSI_WS | 1.8 / 3.3 | 1 |
| **bit18** | MSIOF2_RXD | 1.8 / 3.3 | 1 | SSI_SCK | 1.8 / 3.3 | 1 |
| **bit17** | MSIOF2_SCK | 1.8 / 3.3 | 1 | SCIF_CLK | 1.8 / 3.3 | 1 |
| **bit16** | MSIOF2_TXD | 1.8 / 3.3 | 1 | HRX0 | 1.8 / 3.3 | 1 |
| **bit15** | MSIOF2_SYNC | 1.8 / 3.3 | 1 | HSCK0 | 1.8 / 3.3 | 1 |
| **bit14** | MSIOF2_SS1 | 1.8 / 3.3 | 1 | HRTS0# | 1.8 / 3.3 | 1 |
| **bit13** | MSIOF2_SS2 | 1.8 / 3.3 | 1 | HCTS0# | 1.8 / 3.3 | 1 |
| **bit12** | MSIOF5_RXD | 1.8 / 3.3 | 1 | HTX0 | 1.8 / 3.3 | 1 |
| **bit11** | MSIOF5_SCK | 1.8 / 3.3 | 1 | MSIOF0_RXD | 1.8 / 3.3 | 1 |
| **bit10** | MSIOF5_TXD | 1.8 / 3.3 | 1 | MSIOF0_SCK | 1.8 / 3.3 | 1 |
| **bit9** | MSIOF5_SYNC | 1.8 / 3.3 | 1 | MSIOF0_TXD | 1.8 / 3.3 | 1 |
| **bit8** | MSIOF5_SS1 | 1.8 / 3.3 | 1 | MSIOF0_SYNC | 1.8 / 3.3 | 1 |
| **bit7** | MSIOF5_SS2 | 1.8 / 3.3 | 1 | MSIOF0_SS1 | 1.8 / 3.3 | 1 |
| **bit6** | IRQ0 | 1.8 / 3.3 | 1 | MSIOF0_SS2 | 1.8 / 3.3 | 1 |
| **bit5** | IRQ1 | 1.8 / 3.3 | 1 | MSIOF1_RXD | 1.8 / 3.3 | 1 |
| **bit4** | IRQ2 | 1.8 / 3.3 | 1 | MSIOF1_TXD | 1.8 / 3.3 | 1 |
| **bit3** | IRQ3 | 1.8 / 3.3 | 1 | MSIOF1_SCK | 1.8 / 3.3 | 1 |
| **bit2** | GP0_02 | 1.8 / 3.3 | 1 | MSIOF1_SYNC | 1.8 / 3.3 | 1 |
| **bit1** | GP0_01 | 1.8 / 3.3 | 1 | MSIOF1_SS1 | 1.8 / 3.3 | 1 |
| **bit0** | GP0_00 | 1.8 / 3.3 | 1 | MSIOF1_SS2 | 1.8 / 3.3 | 1 |

**Table 7.31 Configuration of Registers in POC3**

| | POC3 | | |
|---|---|---|---|
| | Pin name | Voltage | Initial value |
| **bit31** | — | — | 0 |
| **bit30** | — | — | 0 |
| **bit29** | — | — | 0 |
| **bit28** | — | — | 0 |
| **bit27** | — | — | 0 |
| **bit26** | — | — | 0 |
| **bit25** | — | — | 0 |
| **bit24** | — | — | 0 |
| **bit23** | — | — | 0 |
| **bit22** | — | — | 0 |
| **bit21** | — | — | 0 |
| **bit20** | — | — | 0 |
| **bit19** | — | — | 0 |
| **bit18** | — | — | 0 |
| **bit17** | — | — | 0 |
| **bit16** | — | — | 0 |
| **bit15** | — | — | 0 |
| **bit14** | — | — | 0 |
| **bit13** | — | — | 0 |
| **bit12** | SD_WP | 1.8 / 3.3 | 1 |
| **bit11** | SD_CD | 1.8 / 3.3 | 1 |
| **bit10** | MMC_SD_CMD | 1.8 / 3.3 | 1 |
| **bit9** | MMC_D6 | 1.8 / 3.3 | 1 |
| **bit8** | MMC_D7 | 1.8 / 3.3 | 1 |
| **bit7** | MMC_D4 | 1.8 / 3.3 | 1 |
| **bit6** | MMC_D5 | 1.8 / 3.3 | 1 |
| **bit5** | MMC_SD_D3 | 1.8 / 3.3 | 1 |
| **bit4** | MMC_DS | 1.8 / 3.3 | 1 |
| **bit3** | MMC_SD_CLK | 1.8 / 3.3 | 1 |
| **bit2** | MMC_SD_D2 | 1.8 / 3.3 | 1 |
| **bit1** | MMC_SD_D0 | 1.8 / 3.3 | 1 |
| **bit0** | MMC_SD_D1 | 1.8 / 3.3 | 1 |

**Table 7.32 Configuration of Registers in POC4 , POC5**

| | POC4 | | | POC5 | | |
|---|---|---|---|---|---|---|
| | Pin name | Voltage | Initial value | Pin name | Voltage | Initial value |
| **bit31** | — | — | 0 | — | — | 0 |
| **bit30** | — | — | 0 | — | — | 0 |
| **bit29** | — | — | 0 | — | — | 0 |
| **bit28** | — | — | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 |
| **bit26** | — | — | 0 | — | — | 0 |
| **bit25** | — | — | 0 | — | — | 0 |
| **bit24** | — | — | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 |
| **bit22** | — | — | 0 | — | — | 0 |
| **bit21** | — | — | 0 | — | — | 0 |
| **bit20** | — | — | 0 | — | — | 0 |
| **bit19** | — | — | 0 | — | — | 0 |
| **bit18** | — | — | 0 | — | — | 0 |
| **bit17** | — | — | 0 | — | — | 0 |
| **bit16** | — | — | 0 | — | — | 0 |
| **bit15** | — | — | 0 | — | — | 0 |
| **bit14** | — | — | 0 | — | — | 0 |
| **bit13** | HSCK2 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit12** | HCTS2# | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit11** | SCIF_CLK2 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit10** | HRTS2# | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit9** | HTX2 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit8** | HRX2 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit7** | SDA3 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit6** | SCL3 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit5** | SDA2 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit4** | SCL2 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit3** | SDA1 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit2** | SCL1 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit1** | SDA0 | 1.8 / 3.3 | 1 | — | — | 0 |
| **bit0** | SCL0 | 1.8 / 3.3 | 1 | VDDQ18_25_AVB for AVB2 | 1.8 / 2.5 | 1 |

**Table 7.33 Configuration of Registers in POC6 , POC7**

| | POC6 | | | POC7 | | |
|---|---|---|---|---|---|---|
| | Pin name | Voltage | Initial value | Pin name | Voltage | Initial value |
| **bit31** | — | — | 0 | — | — | 0 |
| **bit30** | — | — | 0 | — | — | 0 |
| **bit29** | — | — | 0 | — | — | 0 |
| **bit28** | — | — | 0 | — | — | 0 |
| **bit27** | — | — | 0 | — | — | 0 |
| **bit26** | — | — | 0 | — | — | 0 |
| **bit25** | — | — | 0 | — | — | 0 |
| **bit24** | — | — | 0 | — | — | 0 |
| **bit23** | — | — | 0 | — | — | 0 |
| **bit22** | — | — | 0 | — | — | 0 |
| **bit21** | — | — | 0 | — | — | 0 |
| **bit20** | — | — | 0 | — | — | 0 |
| **bit19** | — | — | 0 | — | — | 0 |
| **bit18** | — | — | 0 | — | — | 0 |
| **bit17** | — | — | 0 | — | — | 0 |
| **bit16** | — | — | 0 | — | — | 0 |
| **bit15** | — | — | 0 | — | — | 0 |
| **bit14** | — | — | 0 | — | — | 0 |
| **bit13** | — | — | 0 | — | — | 0 |
| **bit12** | — | — | 0 | — | — | 0 |
| **bit11** | — | — | 0 | — | — | 0 |
| **bit10** | — | — | 0 | — | — | 0 |
| **bit9** | — | — | 0 | — | — | 0 |
| **bit8** | — | — | 0 | — | — | 0 |
| **bit7** | — | — | 0 | — | — | 0 |
| **bit6** | — | — | 0 | — | — | 0 |
| **bit5** | — | — | 0 | — | — | 0 |
| **bit4** | — | — | 0 | — | — | 0 |
| **bit3** | — | — | 0 | — | — | 0 |
| **bit2** | — | — | 0 | — | — | 0 |
| **bit1** | — | — | 0 | — | — | 0 |
| **bit0** | VDDQ18_25_AVB for AVB1 | 1.8 / 2.5 | 1 | VDDQ18_25_AVB for AVB0 | 1.8 / 2.5 | 1 |

### 7.2.8 LSI pin pull-enable register (PUENn / PUENSYS)

Function: PUENn / PUENSYS performs on / off control of the pull resistors.
0: Pull-up / down function is disabled.
1: Pull-up / down function is enabled.

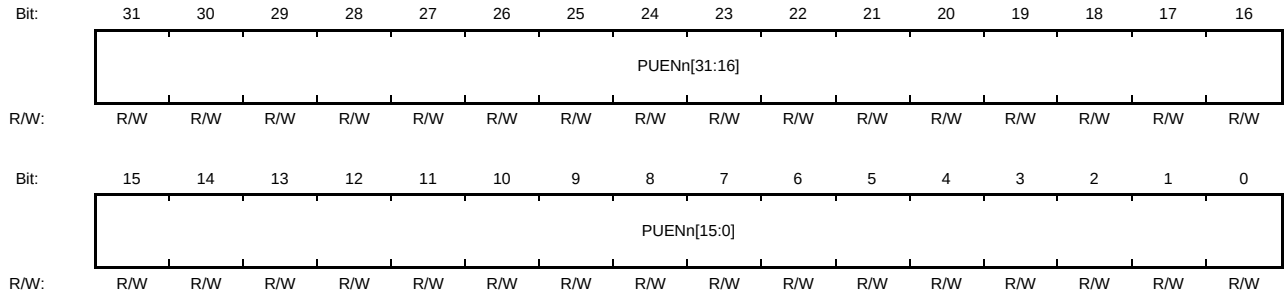



**Table 7.35 Configuration of Registers in PUEN0 , PUEN1**

| | PUEN0 | | PUEN1 | |
|---|---|---|---|---|
| | Pin name | Initial value | Pin name | Initial value |
| **bit31** | — | 0 | — | 0 |
| **bit30** | — | 0 | — | 0 |
| **bit29** | — | 0 | ERROROUTC# | 0 |
| **bit28** | — | 0 | HTX3 | 1 |
| **bit27** | — | 0 | HCTS3# | 1 |
| **bit26** | — | 0 | HRTS3# | 1 |
| **bit25** | — | 0 | HSCK3 | 1 |
| **bit24** | — | 0 | HRX3 | 0 |
| **bit23** | — | 0 | GP1_23 | 0 |
| **bit22** | — | 0 | AUDIO_CLKIN | 1 |
| **bit21** | — | 0 | AUDIO_CLKOUT | 1 |
| **bit20** | — | 0 | SSI_SD | 1 |
| **bit19** | — | 0 | SSI_WS | 1 |
| **bit18** | MSIOF2_RXD | 0 | SSI_SCK | 1 |
| **bit17** | MSIOF2_SCK | 0 | SCIF_CLK | 0 |
| **bit16** | MSIOF2_TXD | 0 | HRX0 | 0 |
| **bit15** | MSIOF2_SYNC | 0 | HSCK0 | 0 |
| **bit14** | MSIOF2_SS1 | 0 | HRTS0# | 0 |
| **bit13** | MSIOF2_SS2 | 0 | HCTS0# | 0 |
| **bit12** | MSIOF5_RXD | 1 | HTX0 | 0 |
| **bit11** | MSIOF5_SCK | 1 | MSIOF0_RXD | 0 |
| **bit10** | MSIOF5_TXD | 1 | MSIOF0_SCK | 0 |
| **bit9** | MSIOF5_SYNC | 0 | MSIOF0_TXD | 0 |
| **bit8** | MSIOF5_SS1 | 0 | MSIOF0_SYNC | 1 |
| **bit7** | MSIOF5_SS2 | 0 | MSIOF0_SS1 | 0 |
| **bit6** | IRQ0 | 0 | MSIOF0_SS2 | 0 |
| **bit5** | IRQ1 | 0 | MSIOF1_RXD | 0 |
| **bit4** | IRQ2 | 0 | MSIOF1_TXD | 0 |
| **bit3** | IRQ3 | 0 | MSIOF1_SCK | 0 |
| **bit2** | GP0_02 | 0 | MSIOF1_SYNC | 0 |
| **bit1** | GP0_01 | 0 | MSIOF1_SS1 | 0 |
| **bit0** | GP0_00 | 1 | MSIOF1_SS2 | 0 |

**Table 7.36 Configuration of Registers in PUEN2 , PUEN3**

| | PUEN2 | | PUEN3 | |
|---|---|---|---|---|
| | Pin name | Initial value | Pin name | Initial value |
| **bit31** | — | 0 | TCLK4 | 1 |
| **bit30** | — | 0 | TCLK3 | 1 |
| **bit29** | — | 0 | RPC_INT# | 1 |
| **bit28** | — | 0 | RPC_WP# | 0 |
| **bit27** | — | 0 | RPC_RESET# | 0 |
| **bit26** | — | 0 | QSPI1_IO3 | 0 |
| **bit25** | — | 0 | QSPI1_SSL | 0 |
| **bit24** | — | 0 | QSPI1_IO2 | 0 |
| **bit23** | — | 0 | QSPI1_MISO_IO1 | 0 |
| **bit22** | — | 0 | QSPI1_SPCLK | 0 |
| **bit21** | — | 0 | QSPI1_MOSI_IO0 | 0 |
| **bit20** | — | 0 | QSPI0_SPCLK | 0 |
| **bit19** | CANFD1_RX | 1 | QSPI0_MOSI_IO0 | 0 |
| **bit18** | — | 0 | QSPI0_MISO_IO1 | 0 |
| **bit17** | CANFD1_TX | 1 | QSPI0_IO2 | 0 |
| **bit16** | — | 0 | QSPI0_IO3 | 0 |
| **bit15** | CANFD3_RX | 1 | QSPI0_SSL | 0 |
| **bit14** | CANFD3_TX | 1 | PWM2 | 1 |
| **bit13** | CANFD2_RX | 0 | PWM1 | 1 |
| **bit12** | CANFD2_TX | 1 | SD_WP | 0 |
| **bit11** | CANFD0_RX | 0 | SD_CD | 0 |
| **bit10** | CANFD0_TX | 1 | MMC_SD_CMD | 0 |
| **bit9** | CAN_CLK | 1 | MMC_D6 | 0 |
| **bit8** | TPU0TO0 | 1 | MMC_D7 | 0 |
| **bit7** | TPU0TO1 | 1 | MMC_D4 | 0 |
| **bit6** | FXR_TXDB | 1 | MMC_D5 | 0 |
| **bit5** | FXR_TXENB# | 1 | MMC_SD_D3 | 0 |
| **bit4** | RXDB_EXTFXR | 1 | MMC_DS | 0 |
| **bit3** | CLK_EXTFXR | 1 | MMC_SD_CLK | 0 |
| **bit2** | RXDA_EXTFXR | 1 | MMC_SD_D2 | 0 |
| **bit1** | FXR_TXENA# | 1 | MMC_SD_D0 | 0 |
| **bit0** | FXR_TXDA | 1 | MMC_SD_D1 | 0 |

**Table 7.37 Configuration of Registers in PUEN4 , PUEN5**

| | PUEN4 | | PUEN5 | |
|---|---|---|---|---|
| | Pin name | Initial value | Pin name | Initial value |
| **bit31** | — | 0 | — | 0 |
| **bit30** | — | 0 | — | 0 |
| **bit29** | — | 0 | — | 0 |
| **bit28** | — | 0 | — | 0 |
| **bit27** | — | 0 | — | 0 |
| **bit26** | — | 0 | — | 0 |
| **bit25** | — | 0 | — | 0 |
| **bit24** | AVS1 | 0 | — | 0 |
| **bit23** | AVS0 | 0 | — | 0 |
| **bit22** | — | 0 | — | 0 |
| **bit21** | PCIE0_CLKREQ# | 0 | — | 0 |
| **bit20** | — | 0 | AVB2_RX_CTL | 1 |
| **bit19** | — | 0 | AVB2_TX_CTL | 1 |
| **bit18** | — | 0 | AVB2_RXC | 1 |
| **bit17** | — | 0 | AVB2_RD0 | 1 |
| **bit16** | — | 0 | AVB2_TXC | 1 |
| **bit15** | PWM4 | 1 | AVB2_TD0 | 1 |
| **bit14** | PWM3 | 1 | AVB2_RD1 | 1 |
| **bit13** | HSCK2 | 0 | AVB2_RD2 | 1 |
| **bit12** | HCTS2# | 0 | AVB2_TD1 | 1 |
| **bit11** | SCIF_CLK2 | 0 | AVB2_TD2 | 1 |
| **bit10** | HRTS2# | 0 | AVB2_MDIO | 1 |
| **bit9** | HTX2 | 0 | AVB2_RD3 | 1 |
| **bit8** | HRX2 | 0 | AVB2_TD3 | 1 |
| **bit7** | SDA3 | 0 | AVB2_TXCREFCLK | 1 |
| **bit6** | SCL3 | 0 | AVB2_MDC | 0 |
| **bit5** | SDA2 | 0 | AVB2_MAGIC | 0 |
| **bit4** | SCL2 | 0 | AVB2_PHY_INT | 1 |
| **bit3** | SDA1 | 0 | AVB2_LINK | 1 |
| **bit2** | SCL1 | 0 | AVB2_AVTP_MATCH | 1 |
| **bit1** | SDA0 | 0 | AVB2_AVTP_CAPTURE | 0 |
| **bit0** | SCL0 | 0 | AVB2_AVTP_PPS | 0 |

**Table 7.38 Configuration of Registers in PUEN6 , PUEN7**

| | PUEN6 | | PUEN7 | |
|---|---|---|---|---|
| | Pin name | Initial value | Pin name | Initial value |
| **bit31** | — | 0 | — | 0 |
| **bit30** | — | 0 | — | 0 |
| **bit29** | — | 0 | — | 0 |
| **bit28** | — | 0 | — | 0 |
| **bit27** | — | 0 | — | 0 |
| **bit26** | — | 0 | — | 0 |
| **bit25** | — | 0 | — | 0 |
| **bit24** | — | 0 | — | 0 |
| **bit23** | — | 0 | — | 0 |
| **bit22** | — | 0 | — | 0 |
| **bit21** | — | 0 | — | 0 |
| **bit20** | AVB1_TXCREFCLK | 1 | AVB0_RX_CTL | 1 |
| **bit19** | AVB1_RD3 | 1 | AVB0_RXC | 1 |
| **bit18** | AVB1_TD3 | 1 | AVB0_RD0 | 1 |
| **bit17** | AVB1_RD2 | 1 | AVB0_RD1 | 1 |
| **bit16** | AVB1_TD2 | 1 | AVB0_TX_CTL | 1 |
| **bit15** | AVB1_RD0 | 1 | AVB0_TXC | 1 |
| **bit14** | AVB1_RD1 | 1 | AVB0_MDIO | 0 |
| **bit13** | AVB1_TD0 | 1 | AVB0_MDC | 0 |
| **bit12** | AVB1_TD1 | 1 | AVB0_RD2 | 1 |
| **bit11** | AVB1_AVTP_CAPTURE | 1 | AVB0_TD0 | 1 |
| **bit10** | AVB1_AVTP_PPS | 1 | AVB0_MAGIC | 1 |
| **bit9** | AVB1_RX_CTL | 1 | AVB0_TXCREFCLK | 1 |
| **bit8** | AVB1_RXC | 1 | AVB0_RD3 | 1 |
| **bit7** | AVB1_TX_CTL | 1 | AVB0_TD1 | 1 |
| **bit6** | AVB1_TXC | 1 | AVB0_TD2 | 1 |
| **bit5** | AVB1_AVTP_MATCH | 1 | AVB0_PHY_INT | 1 |
| **bit4** | AVB1_LINK | 1 | AVB0_LINK | 1 |
| **bit3** | AVB1_PHY_INT | 1 | AVB0_TD3 | 1 |
| **bit2** | AVB1_MDC | 0 | AVB0_AVTP_MATCH | 1 |
| **bit1** | AVB1_MAGIC | 1 | AVB0_AVTP_CAPTURE | 1 |
| **bit0** | AVB1_MDIO | 0 | AVB0_AVTP_PPS | 1 |

**Table 7.39 Configuration of Registers in PUENSYS**

| | PUENSYS | |
|---|---|---|
| | Pin name | Initial value |
| **bit31** | — | 0 |
| **bit30** | — | 0 |
| **bit29** | — | 0 |
| **bit28** | — | 0 |
| **bit27** | — | 0 |
| **bit26** | — | 0 |
| **bit25** | — | 0 |
| **bit24** | — | 0 |
| **bit23** | — | 0 |
| **bit22** | — | 0 |
| **bit21** | — | 0 |
| **bit20** | — | 0 |
| **bit19** | — | 0 |
| **bit18** | — | 0 |
| **bit17** | — | 0 |
| **bit16** | — | 0 |
| **bit15** | — | 0 |
| **bit14** | — | 0 |
| **bit13** | — | 0 |
| **bit12** | — | 0 |
| **bit11** | — | 0 |
| **bit10** | — | 0 |
| **bit9** | DCUTDI_LPDI | 1 |
| **bit8** | DCUTMS | 1 |
| **bit7** | DCUTCK_LPDCLK | 1 |
| **bit6** | DCUTRST_N_LPDRST_N | 1 |
| **bit5** | — | 0 |
| **bit4** | EXTALR | 0 |
| **bit3** | — | 0 |
| **bit2** | — | 0 |
| **bit1** | — | 0 |
| **bit0** | PRESETOUT_N | 0 |

### 7.2.9 LSI pin pull-up / down control Register (PUDn / PUDSYS)

Function: PUDn / PUDSYS performs pull-up / pull-down control of the pull resistors.
0: Pull-down is enabled.
1: Pull-up is enabled.

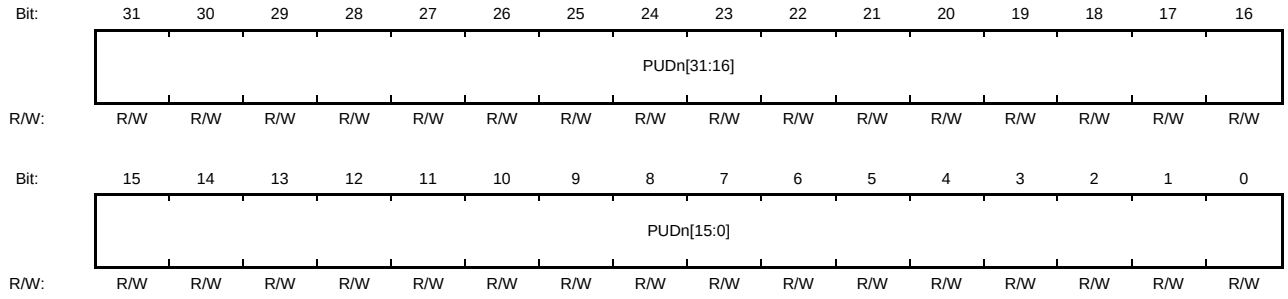



**Table 7.40 Configuration of Registers in PUD0 , PUD1**

| | PUD0 | | PUD1 | |
|---|---|---|---|---|
| | Pin name | Initial value | Pin name | Initial value |
| **bit31** | — | 0 | — | 0 |
| **bit30** | — | 0 | — | 0 |
| **bit29** | — | 0 | ERROROUTC# | 0 |
| **bit28** | — | 0 | HTX3 | 1 |
| **bit27** | — | 0 | HCTS3# | 1 |
| **bit26** | — | 0 | HRTS3# | 1 |
| **bit25** | — | 0 | HSCK3 | 1 |
| **bit24** | — | 0 | HRX3 | 0 |
| **bit23** | — | 0 | GP1_23 | 0 |
| **bit22** | — | 0 | AUDIO_CLKIN | 1 |
| **bit21** | — | 0 | AUDIO_CLKOUT | 1 |
| **bit20** | — | 0 | SSI_SD | 1 |
| **bit19** | — | 0 | SSI_WS | 1 |
| **bit18** | MSIOF2_RXD | 0 | SSI_SCK | 1 |
| **bit17** | MSIOF2_SCK | 0 | SCIF_CLK | 0 |
| **bit16** | MSIOF2_TXD | 0 | HRX0 | 0 |
| **bit15** | MSIOF2_SYNC | 0 | HSCK0 | 0 |
| **bit14** | MSIOF2_SS1 | 0 | HRTS0# | 0 |
| **bit13** | MSIOF2_SS2 | 0 | HCTS0# | 0 |
| **bit12** | MSIOF5_RXD | 1 | HTX0 | 0 |
| **bit11** | MSIOF5_SCK | 1 | MSIOF0_RXD | 0 |
| **bit10** | MSIOF5_TXD | 1 | MSIOF0_SCK | 0 |
| **bit9** | MSIOF5_SYNC | 0 | MSIOF0_TXD | 0 |
| **bit8** | MSIOF5_SS1 | 0 | MSIOF0_SYNC | 1 |
| **bit7** | MSIOF5_SS2 | 0 | MSIOF0_SS1 | 0 |
| **bit6** | IRQ0 | 0 | MSIOF0_SS2 | 0 |
| **bit5** | IRQ1 | 0 | MSIOF1_RXD | 0 |
| **bit4** | IRQ2 | 0 | MSIOF1_TXD | 0 |
| **bit3** | IRQ3 | 0 | MSIOF1_SCK | 0 |
| **bit2** | GP0_02 | 0 | MSIOF1_SYNC | 0 |
| **bit1** | GP0_01 | 0 | MSIOF1_SS1 | 0 |
| **bit0** | GP0_00 | 1 | MSIOF1_SS2 | 0 |

**Table 7.41 Configuration of Registers in PUD2 , PUD3**

| | PUD2 | | PUD3 | |
|---|---|---|---|---|
| | Pin name | Initial value | Pin name | Initial value |
| **bit31** | — | 0 | TCLK4 | 0 |
| **bit30** | — | 0 | TCLK3 | 0 |
| **bit29** | — | 0 | RPC_INT# | 1 |
| **bit28** | — | 0 | RPC_WP# | 0 |
| **bit27** | — | 0 | RPC_RESET# | 0 |
| **bit26** | — | 0 | QSPI1_IO3 | 0 |
| **bit25** | — | 0 | QSPI1_SSL | 0 |
| **bit24** | — | 0 | QSPI1_IO2 | 0 |
| **bit23** | — | 0 | QSPI1_MISO_IO1 | 0 |
| **bit22** | — | 0 | QSPI1_SPCLK | 0 |
| **bit21** | — | 0 | QSPI1_MOSI_IO0 | 0 |
| **bit20** | — | 0 | QSPI0_SPCLK | 0 |
| **bit19** | CANFD1_RX | 1 | QSPI0_MOSI_IO0 | 0 |
| **bit18** | — | 0 | QSPI0_MISO_IO1 | 0 |
| **bit17** | CANFD1_TX | 1 | QSPI0_IO2 | 0 |
| **bit16** | — | 0 | QSPI0_IO3 | 0 |
| **bit15** | CANFD3_RX | 1 | QSPI0_SSL | 0 |
| **bit14** | CANFD3_TX | 1 | PWM2 | 0 |
| **bit13** | CANFD2_RX | 0 | PWM1 | 0 |
| **bit12** | CANFD2_TX | 1 | SD_WP | 0 |
| **bit11** | CANFD0_RX | 0 | SD_CD | 0 |
| **bit10** | CANFD0_TX | 1 | MMC_SD_CMD | 0 |
| **bit9** | CAN_CLK | 1 | MMC_D6 | 0 |
| **bit8** | TPU0TO0 | 1 | MMC_D7 | 0 |
| **bit7** | TPU0TO1 | 1 | MMC_D4 | 0 |
| **bit6** | FXR_TXDB | 1 | MMC_D5 | 0 |
| **bit5** | FXR_TXENB# | 1 | MMC_SD_D3 | 0 |
| **bit4** | RXDB_EXTFXR | 1 | MMC_DS | 0 |
| **bit3** | CLK_EXTFXR | 1 | MMC_SD_CLK | 0 |
| **bit2** | RXDA_EXTFXR | 1 | MMC_SD_D2 | 0 |
| **bit1** | FXR_TXENA# | 1 | MMC_SD_D0 | 0 |
| **bit0** | FXR_TXDA | 1 | MMC_SD_D1 | 0 |

**Table 7.42 Configuration of Registers in PUD4, PUD5**

| | PUD4 | | PUD5 | |
|---|---|---|---|---|
| | Pin name | Initial value | Pin name | Initial value |
| **bit31** | — | — | — | — |
| **bit30** | — | — | — | — |
| **bit29** | — | — | — | — |
| **bit28** | — | — | — | — |
| **bit27** | — | — | — | — |
| **bit26** | — | — | — | — |
| **bit25** | — | — | — | — |
| **bit24** | AVS1 | 0 | — | — |
| **bit23** | AVS0 | 0 | — | — |
| **bit22** | — | 0 | — | — |
| **bit21** | PCIE0_CLKREQ# | 0 | — | — |
| **bit20** | — | 0 | AVB2_RX_CTL | 0 |
| **bit19** | — | 0 | AVB2_TX_CTL | 0 |
| **bit18** | — | 0 | AVB2_RXC | 0 |
| **bit17** | — | 0 | AVB2_RD0 | 0 |
| **bit16** | — | 0 | AVB2_TXC | 0 |
| **bit15** | PWM4 | 1 | AVB2_TD0 | 0 |
| **bit14** | PWM3 | 1 | AVB2_RD1 | 0 |
| **bit13** | HSCK2 | 0 | AVB2_RD2 | 0 |
| **bit12** | HCTS2# | 0 | AVB2_TD1 | 0 |
| **bit11** | SCIF_CLK2 | 0 | AVB2_TD2 | 0 |
| **bit10** | HRTS2# | 0 | AVB2_MDIO | 0 |
| **bit9** | HTX2 | 0 | AVB2_RD3 | 0 |
| **bit8** | HRX2 | 0 | AVB2_TD3 | 0 |
| **bit7** | SDA3 | 0 | AVB2_TXCREFCLK | 0 |
| **bit6** | SCL3 | 0 | AVB2_MDC | 0 |
| **bit5** | SDA2 | 0 | AVB2_MAGIC | 0 |
| **bit4** | SCL2 | 0 | AVB2_PHY_INT | 0 |
| **bit3** | SDA1 | 0 | AVB2_LINK | 0 |
| **bit2** | SCL1 | 0 | AVB2_AVTP_MATCH | 0 |
| **bit1** | SDA0 | 0 | AVB2_AVTP_CAPTURE | 0 |
| **bit0** | SCL0 | 0 | AVB2_AVTP_PPS | 0 |

**Table 7.43 Configuration of Registers in PUD6, PUD7**

| | PUD6 | | PUD7 | |
|---|---|---|---|---|
| | Pin name | Initial value | Pin name | Initial value |
| **bit31** | — | — | — | — |
| **bit30** | — | — | — | — |
| **bit29** | — | — | — | — |
| **bit28** | — | — | — | — |
| **bit27** | — | — | — | — |
| **bit26** | — | — | — | — |
| **bit25** | — | — | — | — |
| **bit24** | — | — | — | — |
| **bit23** | — | — | — | — |
| **bit22** | — | — | — | — |
| **bit21** | — | — | — | — |
| **bit20** | AVB1_TXCREFCLK | 0 | AVB0_RX_CTL | 0 |
| **bit19** | AVB1_RD3 | 0 | AVB0_RXC | 0 |
| **bit18** | AVB1_TD3 | 0 | AVB0_RD0 | 0 |
| **bit17** | AVB1_RD2 | 0 | AVB0_RD1 | 0 |
| **bit16** | AVB1_TD2 | 0 | AVB0_TX_CTL | 0 |
| **bit15** | AVB1_RD0 | 0 | AVB0_TXC | 0 |
| **bit14** | AVB1_RD1 | 0 | AVB0_MDIO | 0 |
| **bit13** | AVB1_TD0 | 0 | AVB0_MDC | 0 |
| **bit12** | AVB1_TD1 | 0 | AVB0_RD2 | 0 |
| **bit11** | AVB1_AVTP_CAPTURE | 0 | AVB0_TD0 | 0 |
| **bit10** | AVB1_AVTP_PPS | 0 | AVB0_MAGIC | 0 |
| **bit9** | AVB1_RX_CTL | 0 | AVB0_TXCREFCLK | 0 |
| **bit8** | AVB1_RXC | 0 | AVB0_RD3 | 0 |
| **bit7** | AVB1_TX_CTL | 0 | AVB0_TD1 | 0 |
| **bit6** | AVB1_TXC | 0 | AVB0_TD2 | 0 |
| **bit5** | AVB1_AVTP_MATCH | 0 | AVB0_PHY_INT | 0 |
| **bit4** | AVB1_LINK | 0 | AVB0_LINK | 0 |
| **bit3** | AVB1_PHY_INT | 0 | AVB0_TD3 | 0 |
| **bit2** | AVB1_MDC | 0 | AVB0_AVTP_MATCH | 0 |
| **bit1** | AVB1_MAGIC | 0 | AVB0_AVTP_CAPTURE | 0 |
| **bit0** | AVB1_MDIO | 0 | AVB0_AVTP_PPS | 0 |

**Table 7.44 Configuration of Registers in PUDSYS**

| | PUDSYS |
|---|---|
| | Pin name |
| **bit31** | — |
| **bit30** | — |
| **bit29** | — |
| **bit28** | — |
| **bit27** | — |
| **bit26** | — |
| **bit25** | — |
| **bit24** | — |
| **bit23** | — |
| **bit22** | — |
| **bit21** | — |
| **bit20** | — |
| **bit19** | — |
| **bit18** | — |
| **bit17** | — |
| **bit16** | — |
| **bit15** | — |
| **bit14** | — |
| **bit13** | — |
| **bit12** | — |
| **bit11** | — |
| **bit10** | — |
| **bit9** | DCUTDI_LPDI |
| **bit8** | DCUTMS |
| **bit7** | DCUTCK_LPDCLK |
| **bit6** | DCUTRST_N_LPDRST_N |
| **bit5** | — |
| **bit4** | EXTALR |
| **bit3** | — |
| **bit2** | — |
| **bit1** | — |
| **bit0** | PRESETOUT_N |

### 7.2.10 Module Select Register (MODSELn)

Function: MODSELn selects the group for multiple LSI pins with multiplexed pin functions.

The I2C cell uses two Pins of CLK and data, and STBN for making it unused is shared by two Pins of CLK and data, so the MODSELn register has 2 bits for one function.

In order to apply domain control, bit assignment is set to Port Group (GPIO).

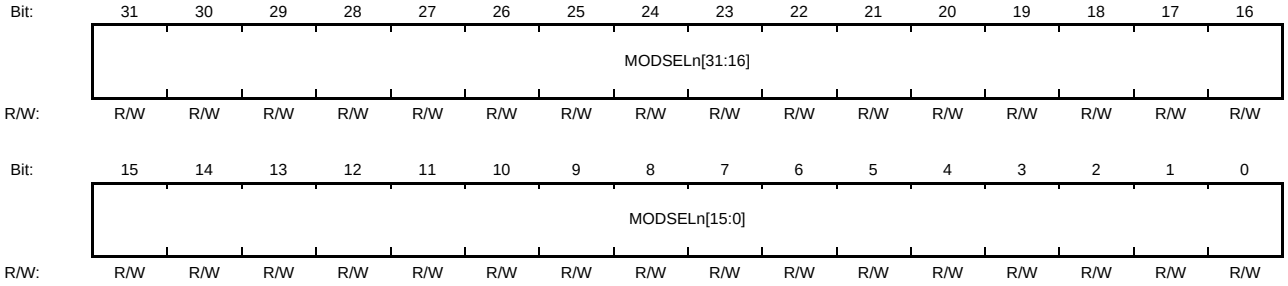



**Table 7.45 Configuration of Registers in MODSEL**

| | MODSEL4 | Initial value | Description |
|---|---|---|---|
| **bit31** | — | — | — |
| **bit30** | — | — | — |
| **bit29** | — | — | — |
| **bit28** | — | — | — |
| **bit27** | — | — | — |
| **bit26** | — | — | — |
| **bit25** | — | — | — |
| **bit24** | — | — | — |
| **bit23** | — | — | — |
| **bit22** | — | — | — |
| **bit21** | — | — | — |
| **bit20** | — | — | — |
| **bit19** | — | — | — |
| **bit18** | — | — | — |
| **bit17** | — | — | — |
| **bit16** | — | — | — |
| **bit15** | — | — | — |
| **bit14** | — | — | — |
| **bit13** | — | — | — |
| **bit12** | — | — | — |
| **bit11** | — | — | — |
| **bit10** | — | — | — |
| **bit9** | — | — | — |
| **bit8** | — | — | — |
| **bit7** | SDA3 | 0 | 0 : GPIO mode<br>1 : I2C mode |
| **bit6** | SCL3 | 0 | 0 : GPIO mode<br>1 : I2C mode |
| **bit5** | SDA2 | 0 | 0 : GPIO mode<br>1 : I2C mode |
| **bit4** | SCL2 | 0 | 0 : GPIO mode<br>1 : I2C mode |
| **bit3** | SDA1 | 0 | 0 : GPIO mode<br>1 : I2C mode |
| **bit2** | SCL1 | 0 | 0 : GPIO mode<br>1 : I2C mode |
| **bit1** | SDA0 | 0 | 0 : GPIO mode<br>1 : I2C mode |
| **bit0** | SCL0 | 0 | 0 : GPIO mode<br>1 : I2C mode |

Note: To enable this register to be set, appropriately set the multiplexed pin setting mask register (PMMR) immediately before setting this register.

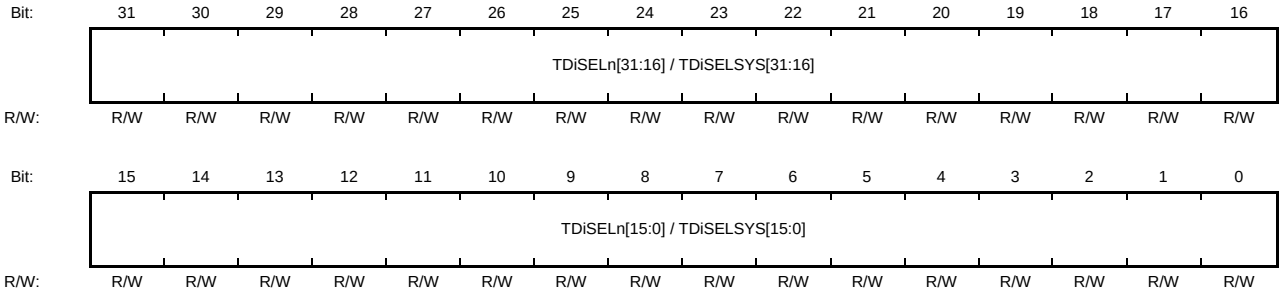
### 7.2.11 TDSEL Control Register 0-1 (TD0SELn , TD1SELn / TD0SELSYS , TD1SELSYS)

Function: TDiSELn / TDiSELSYS controls the driving abilities of pins in use for the SDHI.
This function is delay adjustment of the SDHI clock return path for the LSI inside.
Note: i = 0 to 1

| TDiSEL1 | TDiSEL0 | TDOUT<br>Target delay |
|---|---|---|
| L | L | 10pF (@1.8V operation) |
| H | L | 20pF (@1.8V operation) |
| L | H | 30pF (@1.8V operation) |
| H | H | 40pF (@1.8V operation) |

Note: TDiSEL1 SD0TDSEL1
TDiSEL0 SD0TDSEL0

The value of these bit must be 00.




Note: To enable this register to be set, appropriately set the multiplexed pin setting mask register (PMMR) immediately before setting this register.

**Table 7.46 Configuration of Registers in TD0SEL3 , TD1SEL3**

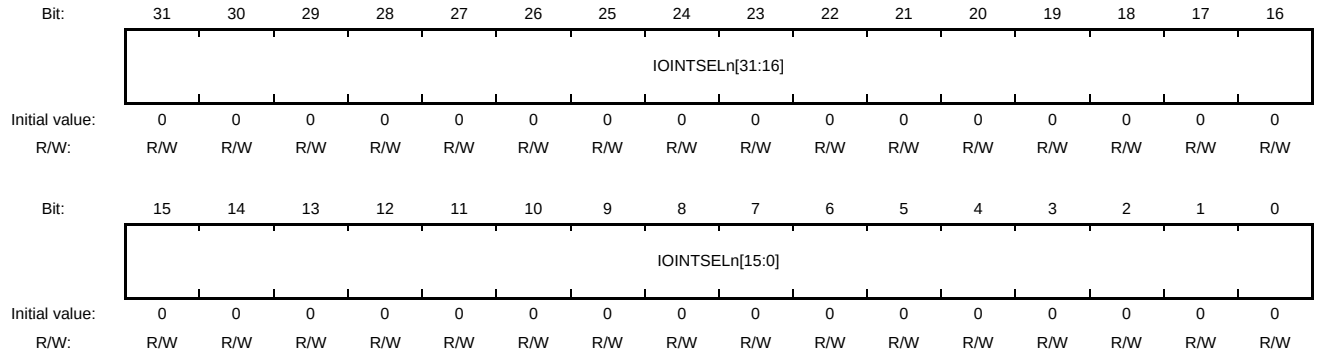
| | TD0SEL3 | | | | TD1SEL3 | | | |
|---|---|---|---|---|---|---|---|---|
| | Pin name | | [Renesas Internal] | Initial value | Pin name | | [Renesas Internal] | Initial value |
| **bit31** | — | — | — | — | — | — | — | — |
| **bit30** | — | — | — | — | — | — | — | — |
| **bit29** | — | — | — | — | — | — | — | — |
| **bit28** | — | — | — | — | — | — | — | — |
| **bit27** | — | — | — | — | — | — | — | — |
| **bit26** | — | — | — | — | — | — | — | — |
| **bit25** | — | — | — | — | — | — | — | — |
| **bit24** | — | — | — | — | — | — | — | — |
| **bit23** | — | — | — | — | — | — | — | — |
| **bit22** | — | — | — | — | — | — | — | — |
| **bit21** | — | — | — | — | — | — | — | — |
| **bit20** | — | — | — | — | — | — | — | — |
| **bit19** | — | — | — | — | — | — | — | — |
| **bit18** | — | — | — | — | — | — | — | — |
| **bit17** | — | — | — | — | — | — | — | — |
| **bit16** | — | — | — | — | — | — | — | — |
| **bit15** | — | — | — | — | — | — | — | — |
| **bit14** | — | — | — | — | — | — | — | — |
| **bit13** | — | — | — | — | — | — | — | — |
| **bit12** | — | — | — | — | — | — | — | — |
| **bit11** | — | — | — | — | — | — | — | — |
| **bit10** | — | — | — | — | — | — | — | — |
| **bit9** | — | — | — | — | — | — | — | — |
| **bit8** | — | — | — | — | — | — | — | — |
| **bit7** | MMC_SD_CLK | SD0TDSEL1 | — | 0 | — | — | — | — |
| **bit6** | | SD0TDSEL0 | — | 0 | — | — | — | — |

| | TD0SEL3 | | | | TD1SEL3 | | | |
|---|---|---|---|---|---|---|---|---|
| | Pin name | | [Renesas Internal] | Initial value | Pin name | | [Renesas Internal] | Initial value |
| **bit5** | — | — | — | — | — | — | — | — |
| **bit4** | — | — | — | — | — | — | — | — |
| **bit3** | — | — | — | — | — | — | — | — |
| **bit2** | — | — | — | — | — | — | — | — |
| **bit1** | — | — | — | — | — | — | — | — |
| **bit0** | — | — | — | — | — | — | — | — |

### 7.2.12 General IO / Interrupt Switching Register (IOINTSELn)

IOINTSELn selects either general input/output mode or interrupt input mode for each of the port pins 0 to 31 of the GPIO group. When general input / output mode is selected for a port, it is also necessary to select either input or output mode for the port using the corresponding bit in the general input/output switching register. When interrupt input mode is selected for a port, the setting of the general input / output switching register for the port is ignored.

[Hardware default value:H'0000 0000 = general input / output mode is selected for all the ports.]




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | IOINTSELn [31:0] | H'0000 0000 | R/W | Selects either general input/output mode or interrupt input mode for each port using the bits corresponding to the port numbers.<br>0: General input/output mode.<br>1: Interrupt input mode. |

Note: Unused bits should be set to the initial values.

| | IOINTSELn | | IOINTSELn |
|---|---|---|---|
| **Bit** | **GPIO** | **Bit** | **GPIO** |
| bit31 | GPn_31 | bit15 | GPn_15 |
| bit30 | GPn_30 | bit14 | GPn_14 |
| bit29 | GPn_29 | bit13 | GPn_13 |
| bit28 | GPn_28 | bit12 | GPn_12 |
| bit27 | GPn_27 | bit11 | GPn_11 |
| bit26 | GPn_26 | bit10 | GPn_10 |
| bit25 | GPn_25 | bit9 | GPn_9 |
| bit24 | GPn_24 | bit8 | GPn_8 |
| bit23 | GPn_23 | bit7 | GPn_7 |
| bit22 | GPn_22 | bit6 | GPn_6 |
| bit21 | GPn_21 | bit5 | GPn_5 |
| bit20 | GPn_20 | bit4 | GPn_4 |
| bit19 | GPn_19 | bit3 | GPn_3 |
| bit18 | GPn_18 | bit2 | GPn_2 |
| bit17 | GPn_17 | bit1 | GPn_1 |
| bit16 | GPn_16 | bit0 | GPn_0 |

### 7.2.13 General Input/Output Switching Register (INOUTSELn)

INOUTSELn is valid only for the ports for which general input/output mode is selected by the general IO/interrupt switching register. Specifically, INOUTSEL selects either general input or general output mode for a port using the bit corresponding to the port number. The INOUTSEL bits can be written to only when the corresponding bits in the general IO/interrupt switching register are 0. Note that after general input/output mode is changed to interrupt input mode, INOUTSEL retains the setting but is read as 0.

[Hardware default value: H'0000 0000 = general input mode is selected for all the ports.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INOUTSELn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INOUTSELn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | INOUTSELn [31:0] | H'0000 0000 | R/W | Selects either general input mode or general output mode for each port using the bits corresponding to the port numbers.<br>0: General input mode<br>1: General output mode |

Note: Unused bits should be set to the initial values.

### 7.2.14 General Output Register (OUTDTn)

OUTDTn is valid only for the ports for which general input/output mode is selected by the general IO/interrupt switching register and then general output mode is selected by the general input/output switching register. Specifically, the value of the bit in OUTDTn corresponding to the port number is inverted or not inverted depending on the setting of the positive/negative logic select register before being output from the corresponding port pin. Note that the polarity of the output signal should previously be set using the corresponding bit in the positive/negative logic select register. This register must be set after the output data select register is appropriately set to choose level of output data.

[Hardware default value: H'0000 0000 = 0 is output from all the ports.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTDTn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTDTn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | OUTDTn[31:0] | H'000 00000 | R/W | Allows the port to output the value set in the bit corresponding to the port number when the port is appropriately set by IOINTSEL, INOUTSEL and OUTDTSEL.<br>0: 0 is output.<br>1: 1 is output. |

Note: The values set in OUTDT are not directly output from the GPIO pins; the above set values are processed according to the settings of the positive / negative logic select register before being output.
Unused bits should be set to the initial values.

About write of output data: Output data can be selected exclusively of OUTDT or OUTDTH / OUTDTL, based on OUTDTSEL setting.

About read of output data: Either of OUTDT and OUTDTH / OUTDTL read as the actual output data.

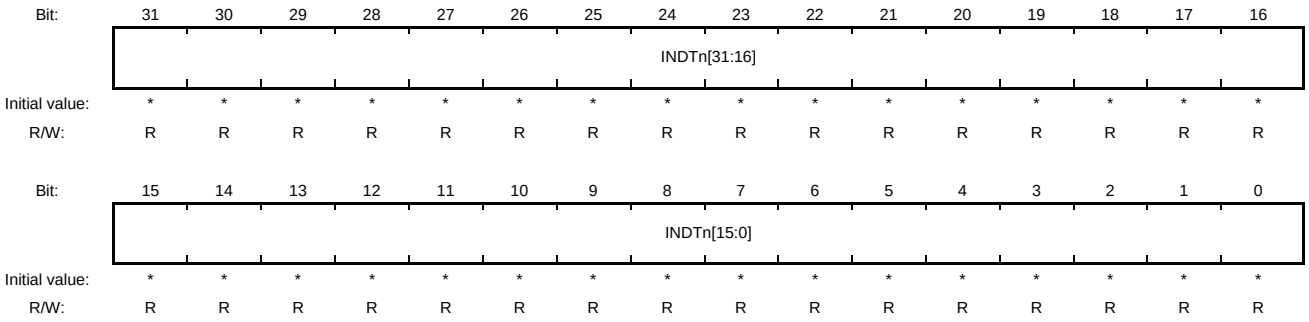
### 7.2.15 General Input Register (INDTn)

INDTn is a register that can read the status of General-Purpose Input / Output Ports.

Each bit reflects the value received through the corresponding port pin.

Note that when a bit in the positive / negative logic select register is 1, the corresponding bit in INDT indicates the inverted value of the input signal.

[Hardware default value: state of the signals input to the port pins.]




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | INDTn[31:0] | * | R | Each bit reflects the value received through the corresponding port pin.<br>0: Input is 0. (assuming positive logic)<br>1: Input is 1. (assuming positive logic) |

Note: Unused bits should be set to the initial values.
* State of the signals input to the port pins.

### 7.2.16 Interrupt Display Register (INTDTn)

INTDTn is valid only when interrupt input mode is selected by the general IO/interrupt switching register. Specifically, when an interrupt is input via a port pin when INTDTn is valid, the bit in INTDTn corresponding to the port indicates whether the port has received an interrupt input or not. In interrupt input mode, the polarity and detection conditions (one edge/both edge or level) of the external input signal can be set for each port pin. Before using a port pin for interrupt input, the corresponding bits in the positive/negative logic select register and edge/level select register (one edge/both edge register should be appropriately configured if edge detection mode is selected) should be set, respectively.

If a port is set for edge detection using the corresponding bit in the edge/level select register, even when an external pulse interrupt signal is input, the corresponding bit in INTDTn holds the input using the FF and allows the level interrupt signal to be output to the interrupt control block.

To stop all the interrupt signal outputs, all the bits in the interrupt clear register corresponding to the bits in INTDTn currently indicating the reception of the corresponding interrupt signals should be cleared to 0. Note that if a port is set for level detection using the corresponding bit in the edge/level select register and an external level interrupt signal is input, the corresponding bit in INTDTn does not use the FF to hold the input. Therefore, when an external input signal is stopped, the corresponding bit in INTDTn is cleared automatically. When all the bits in INTDTn are turned off (= 0), the GPIO stops outputting all the interrupt signals.

[Hardware default value: H'0000 0000 = no interrupt signals are input from ports.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTDTn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | INTDTn[31:0] | H'0000 0000 | R | Each bit indicates the input of an interrupt signal on the corresponding port pin.<br>0: No interrupt signal has been input.<br>1: Interrupt signal has been input. |

Conditions of Indicating Interrupt Input:

1. For level-sensitive interrupt input (EDGLEVEL = 0)
   - External input signals are constantly monitored and indicated. (When the negative logic is selected, the inverted value of the external input signal is indicated.)
2. For edge-sensitive interrupt input (EDGLEVEL = 1)
   - Clearing condition: When the interrupt clear register is cleared, indication is cleared regardless of the positive / negative logic select register.
   - Setting condition: With the positive logic (POSNEG = 0), when the rising edge of an external interrupt signal is detected, the interrupt input is indicated. With the negative logic (POSNEG = 1), when the falling edge is detected, the interrupt input is indicated. With both edge mode (BOTHEDGE = 1), when either the rising or falling edge, the interrupt input is indicated.

### 7.2.17 Interrupt Clear Register (INTCLRn)

When the interrupt display register is currently indicates the reception of the interrupt input on the port for which the edge detection is selected by the edge/level select register (with configuring for one edge/both edge select register) in interrupt input mode, INTCLRn clears the indication. Specifically, writing 1 to the bits in INTCLRn corresponding to port numbers can clear the corresponding bits in the interrupt display register. However, when the interrupt display register is currently indicates the reception of the interrupt input on the port for which the level detection is selected by the edge/level select register, writing 1 to the corresponding bits in INTCLRn cannot clear the corresponding bits in the interrupt display register. Only writing 1 to INTCLRn is effective; INTCLRn is always read as 0.

[Hardware default value: H'0000 0000 = interrupt indication is cleared for no ports.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLRn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCLRn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | INTCLRn [31:0] | H'0000 0000 | R/W | Writing 1 to bits corresponding to port numbers clears the corresponding bits in the interrupt display register.<br>0: No effect<br>1: Interrupt display register bit is cleared. |

Note: Unused bits should be set to the initial values.

### 7.2.18 Interrupt Mask Register (INTMSKn)

INTMSKn masks the interrupt requests indicated by the interrupt display register of GPIO.ch. Interrupts can be separately masked using the corresponding bits in INTMSKn. When all the bits currently indicating the reception of the interrupt signals are masked, no interrupt signals are output to the interrupt control block. Masks can be canceled by writing 1 to the corresponding bits in the interrupt mask clear register. Only writing 0 to this register is effective.

[Hardware default value: H'0000 0000 = all the ports are masked.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTMSKn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTMSKn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | INTMSKn [31:0] | H'0000 0000 | R/W | Setting a mask to the bit disables the corresponding interrupt signal to be output to the interrupt control block.<br>0: Interrupt is masked.<br>1: Interrupt is not masked. |

Note: Unused bits should be set to the initial values.

### 7.2.19 Interrupt Mask Clear Register (MSKCLRn)

MSKCLRn cancels masks that are set by the interrupt mask register of GPIO.ch*A. Each mask can be canceled (cleared) by writing 1 to the corresponding bit in MSKCLRn. Only writing 1 to MSKCLRn is effective; MSKCLRn is always read as 0.

[Hardware default value: H'0000 0000 = alternative interrupt masks are cleared for no ports.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSKCLRn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSKCLRn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

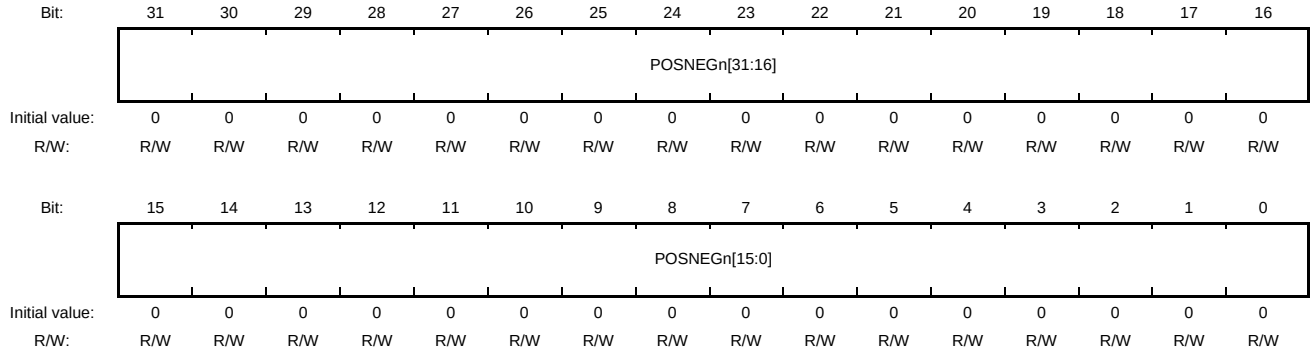
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | MSKCLRn [31:0] | H'0000 0000 | R/W | Setting a mask to the bit disables the corresponding alternative interrupt signal to be output to the interrupt control block.<br>0: No effect<br>1: Interrupt is not masked. |

Note: Unused bits should be set to the initial values. (When GPIO is not selected by the pin multiplex settings, do not cancel the alternative interrupt mask.)

### 7.2.20 Positive / Negative Logic Select Register (POSNEGn)

POSNEGn selects the polarity (positive or negative logic) of each port pin in general input mode, general output mode, or interrupt input mode. POSNEGn should be set before mode selection.

[Hardware default value: H'0000 0000 = positive logic is selected for all the ports.]




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | POSNEGn [31:0] | H'0000 0000 | R/W | Selects the polarity (positive or negative logic) of each port pin.<br>0: Positive logic<br>1: Negative logic |

Note: Unused bits should be set to the initial values.

### 7.2.21 Edge / level Select Register (EDGLEVELn)

EDGLEVELn is valid only for the ports for which interrupt input mode is selected by the general IO/interrupt switching register. Specifically, EDGLEVELn selects the detection conditions (edge or level) of the interrupt input signal on each port pin for which interrupt input mode is selected. EDGLEVELn should be set before selection of interrupt input mode.

[Hardware default value: H'0000 0000 = level detection is selected for all the ports.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EDGLEVELn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EDGLEVELn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

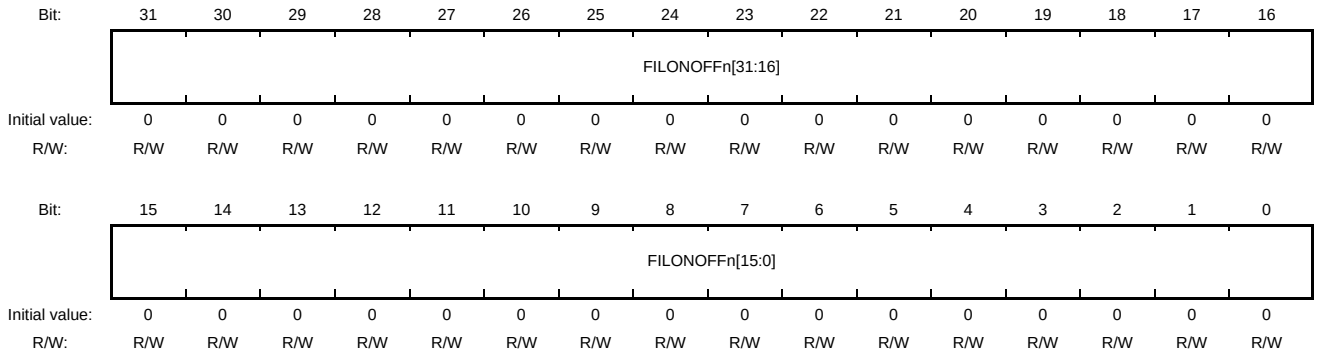
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | EDGLEVELn [31:0] | H'0000 0000 | R/W | Selects the level or edge as detection conditions of the interrupt input signal on each port pin for which interrupt input mode is selected.<br>0: Level<br>1: Edge |

Note: Unused bits should be set to the initial values.

### 7.2.22 Chattering Prevention On / Off Register (FILONOFFn)

FILONOFFn prevents chattering input to the port pins of each GPIO group. For details, refer to section 7.3.6, Handling of Input Signals on Port Pins.

[Hardware default value: H'0000 0000 = chattering prevention function is turned off for all the ports.]




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | FILONOFFn [31:0] | H'0000 0000 | R/W | Enables or disables the chattering prevention function.<br>0: Chattering prevention function is disabled.<br>1: Chattering prevention function is enabled |

### 7.2.23 Chattering Prevention Clock Select Register (FILCLKSELn)

FILCLKSELn controls the division ratio of clock CPϕ for prevent chattering input to the port pin GPIO group.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | | | | | | | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FILCLKSELn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | FILCLKSELn [15:0] | H'411B | R/W | Set the division ratio of filter CLOCK.<br>d: CPϕ / (d + 1)<br>d is setting value |

### 7.2.24 Output Data Select Register (OUTDTSELn)

OUTDTSELn is valid only for the ports for which general input/output mode is selected by the general IO/interrupt switching register and then general output mode is selected by the general input/output switching register. OUTDTSELn selects if OUTDTn or OUTDTHn / OUTDTLn will be the output data of GPIO. When choosing OUTDTn, configuration is performed as described in section 7.2.14, General Output Register (OUTDTn). When choosing OUTDTHn / OUTDTLn, output data will be output by writing the appropriate data to the corresponding bits in OUTDTHn or OUTDTLn. Note that the polarity of the output signal should be previously set using the corresponding bit in the positive/negative logic select register. Furthermore, this register should be set before writing data to OUTDTHn / OUTDTLn registers.

[Hardware default value: H'0000 0000 = Out data register is used to output data.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTDTSELn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTDTSELn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | OUTDTSELn [31:0] | H'0000 0000 | R/W | Choosing whether output data is output by general output register OUTDTn or output data high register OUTDTHn / output data low register OUTDTLn.<br>0: General output register is used to output the data.<br>1: Output data high register and output data low register is used to output the data. |

Note: Unused bits should be set to the initial values.

### 7.2.25 Output Data High Register (OUTDTHn)

OUTDTHn is valid only for the ports for which general input/output mode is selected by the general IO/interrupt switching register and then general output mode is selected by the general input/output switching register, and the output data select register OUTDTSELn is configured to choose OUTDTHn / OUTDTLn register to output the data of GPIO. Only writing 1 to OUTDTHn is effective. Otherwise, setting makes no changes. Reading OUTDTHn returns the values of the latest data set to OUTDTHn or OUTDTLn right before that. Note that the polarity of the output signal should be previously set using the corresponding bit in the positive/negative logic select register. Furthermore, this register should be written and read after output data select register OUTDTSELn is set. Reading OUTDTHn without appropriately configuring OUTDTSELn can return value of OUTDTn register.

[Hardware default value: H'0000 0000 = 0 is output from all the port swith setting OUTDTSELn.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTDTHn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTDTHn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | OUTDTHn [31:0] | H'0000 0000 | R/W | Outputting high value data.<br>0: Invalid data.<br>1: Valid data. |

Note: Unused bits should be set to the initial values.

### 7.2.26 Output Data Low Register (OUTDTLn)

OUTDTLn is valid only for the ports for which general input/output mode is selected by the general IO/interrupt switching register and then general output mode is selected by the general input/output switching register, and the output data select register OUTDTSELn is configured to choose OUTDTHn / OUTDTLn register to output the data of GPIO. Only writing 0 to OUTDTLn is effective. Otherwise, setting makes no changes. Reading OUTDTLn returns the values of the latest data set to OUTDTLn or OUTDTHn right before that. Note that the polarity of the output signal should be previously set using the corresponding bit in the positive/negative logic select register. Furthermore, this register should be written or read after output data select registers OUTDTSELn is set. Reading OUTDTHn without appropriately configuring OUTDTSELn can return value of OUTDTn register.

[Hardware default value: H'0000 0000 = 0 is output from all the ports with setting OUTDTSELn.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTDTLn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OUTDTLn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | OUTDTLn [31:0] | H'0000 0000 | R/W | Outputting low value data.<br>0: Valid data.<br>1: Invalid data. |

Note: Unused bits should be set to the initial values.

### 7.2.27 One Edge / Both Edge Select Register (BOTHEDGEn)

BOTHEDGEn is valid only when the edge detection mode is selected by the edge/level select registers. Specially, BOTHEDGEn selects the detection condition (one edge or both edges) of the interrupt input signal on each port pin for which interrupt input mode (selected by the general IO/interrupt switching registers) and edge detection mode are selected. BOTHEDGEn should be set before selection of interrupt input mode.

[Hardware default value: H'0000 0000 = both edge detection mode is disabled for all the ports.]

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BOTHEDGEn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BOTHEDGEn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BOTHEDGEn [31:0] | H'0000 0000 | R/W | Selecting one edge or both edge detection condition of the interrupt input signal on each port pin for which interrupt input mode and edge detection mode are selected.<br>0: One edge.<br>1: Both edges. |

Note: Unused bits should be set to the initial values.

### 7.2.28 General Input Enable register (INENn)

Create registers that can control IE in GPIO.
Assign 1 bit of register to 1 pin for IE of all pins to which GPIO is assigned. Also,as before, in order to maintain SW compatibility, set the initial value of IE to High (input ON) state.

The register specifications are shown below.
Create one register (32 bits) for each Port Group.
Setting value: 0 = Input Enable Off / 1 = Input Enable ON
Initial value: 0 = Input Enable OFF

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INENn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INENn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | INENn [31:0] | H'0000 0000 | R/W | 0: Input disable.<br>1: Input enable. |

### 7.2.29 Port Safe state Select Register0-1 (PS0SRn , PS1SRn)

Registers (PS*SRn) that select 4 types of Port Safe State (initial state or HiZ or Pull-Down or Pull-Up) with 2 bits.
Note: * 0 to 1

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PS*SRn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PS*SRn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note: It applies to Pin which multiplex with GPIO function
Note: About precautions of this register, refer to “7.4.2 Notes for PS0SRn/PS1SRn/PSERn”.

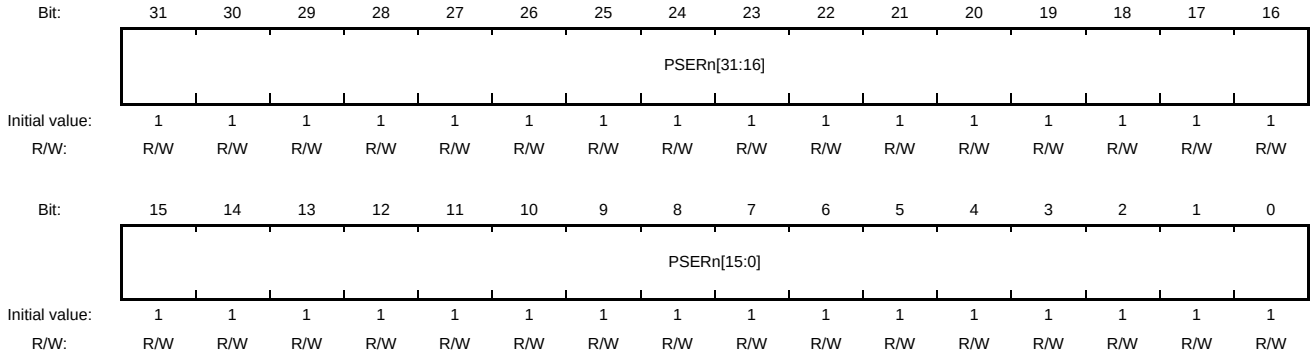
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PS*SRn[m×2+1:m×2]<br>m = 0 to 15 | b’00 | R/W | b'00=initial state<br>b'01=HiZ<br>b'10=Pull-Down<br>b'11=Pull-Up |

**Table 7.48 Configuration of Registers in PS0SRn , PS1SRn**

| | PS0SRn | | | PS1SRn | | |
|---|---|---|---|---|---|---|
| | **GPIO** | **Description** | **Initial value** | **GPIO** | **Description** | **Initial value** |
| **bit31**<br>**bit30** | GPn_15 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_31 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit29**<br>**bit28** | GPn_14 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_30 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit27**<br>**bit26** | GPn_13 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_29 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit25**<br>**bit24** | GPn_12 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_28 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit23**<br>**bit22** | GPn_11 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_27 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit21**<br>**bit20** | GPn_10 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_26 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit19**<br>**bit18** | GPn_9 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_25 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit17**<br>**bit16** | GPn_8 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_24 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit15**<br>**bit14** | GPn_7 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_23 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit13**<br>**bit12** | GPn_6 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_22 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit11**<br>**bit10** | GPn_5 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_21 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit9**<br>**bit8** | GPn_4 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_20 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit7**<br>**bit6** | GPn_3 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_19 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit5**<br>**bit4** | GPn_2 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_18 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit3**<br>**bit2** | GPn_1 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_17 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |
| **bit1**<br>**bit0** | GPn_0 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 | GPn_16 | b'00=initial state / b'01=HiZ<br>b'10=Pull-Down / b'11=Pull-Up | b'00 |

### 7.2.30 Port Safe state Enable Register (PSERn)

The register (PSERn) to enable / disable PSS Rregister.




Note: About precautions of this register, refer to “7.4.2 Notes for PS0SRn/PS1SRn/PSERn”.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PSERn [31:0] | H’FFFF FFFF | R/W | 0: PSSR disable.<br>1: PSSR enable. |

**Table 7.49 Configuration of Registers in PSER**

| | PSERn | | | PSERn | |
|---|---|---|---|---|---|
| **Bit** | **GPIO** | **Description** | **Bit** | **GPIO** | **Description** |
| **bit31** | GPn_31 | 0: PS1SRn[31:30] disable<br>1: PS1SRn[31:30] enable | **bit15** | GPn_15 | 0: PS0SRn[31:30] disable<br>1: PS0SRn[31:30] enable |
| **bit30** | GPn_30 | 0: PS1SRn[29:28] disable<br>1: PS1SRn[29:28] enable | **bit14** | GPn_14 | 0: PS0SRn[29:28] disable<br>1: PS0SRn[29:28] enable |
| **bit29** | GPn_29 | 0: PS1SRn[27:26] disable<br>1: PS1SRn[27:26] enable | **bit13** | GPn_13 | 0: PS0SRn[27:26] disable<br>1: PS0SRn[27:26] enable |
| **bit28** | GPn_28 | 0: PS1SRn[25:24] disable<br>1: PS1SRn[25:24] enable | **bit12** | GPn_12 | 0: PS0SRn[25:24] disable<br>1: PS0SRn[25:24] enable |
| **bit27** | GPn_27 | 0: PS1SRn[23:22] disable<br>1: PS1SRn[23:22] enable | **bit11** | GPn_11 | 0: PS0SRn[23:22] disable<br>1: PS0SRn[23:22] enable |
| **bit26** | GPn_26 | 0: PS1SRn[21:20] disable<br>1: PS1SRn[21:20] enable | **bit10** | GPn_10 | 0: PS0SRn[21:20] disable<br>1: PS0SRn[21:20] enable |
| **bit25** | GPn_25 | 0: PS1SRn[19:18] disable<br>1: PS1SRn[19:18] enable | **bit9** | GPn_9 | 0: PS0SRn[19:18] disable<br>1: PS0SRn[19:18] enable |
| **bit24** | GPn_24 | 0: PS1SRn[17:16] disable<br>1: PS1SRn[17:16] enable | **bit8** | GPn_8 | 0: PS0SRn[17:16] disable<br>1: PS0SRn[17:16] enable |
| **bit23** | GPn_23 | 0: PS1SRn[15:14] disable<br>1: PS1SRn[15:14] enable | **bit7** | GPn_7 | 0: PS0SRn[15:14] disable<br>1: PS0SRn[15:14] enable |
| **bit22** | GPn_22 | 0: PS1SRn[13:12] disable<br>1: PS1SRn[13:12] enable | **bit6** | GPn_6 | 0: PS0SRn[13:12] disable<br>1: PS0SRn[13:12] enable |
| **bit21** | GPn_21 | 0: PS1SRn[11:10] disable<br>1: PS1SRn[11:10] enable | **bit5** | GPn_5 | 0: PS0SRn[11:10] disable<br>1: PS0SRn[11:10] enable |
| **bit20** | GPn_20 | 0: PS1SRn[9:8] disable<br>1: PS1SRn[9:8] enable | **bit4** | GPn_4 | 0: PS0SRn[9:8] disable<br>1: PS0SRn[9:8] enable |
| **bit19** | GPn_19 | 0: PS1SRn[7:6] disable<br>1: PS1SRn[7:6] enable | **bit3** | GPn_3 | 0: PS0SRn[7:6] disable<br>1: PS0SRn[7:6] enable |
| **bit18** | GPn_18 | 0: PS1SRn[5:4] disable<br>1: PS1SRn[5:4] enable | **bit2** | GPn_2 | 0: PS0SRn[5:4] disable<br>1: PS0SRn[5:4] enable |
| **bit17** | GPn_17 | 0: PS1SRn[3:2] disable<br>1: PS1SRn[3:2] enable | **bit1** | GPn_1 | 0: PS0SRn[3:2] disable<br>1: PS0SRn[3:2] enable |
| **bit16** | GPn_16 | 0: PS1SRn[1:0] disable<br>1: PS1SRn[1:0] enable | **bit0** | GPn_0 | 0: PS0SRn[1:0] disable<br>1: PS0SRn[1:0] enable |

## 7.3 Operation

### 7.3.1 Overview

This is recomendation of Pin Function setting flow, “Figure7.3 Recomendation of Pin Function setting flow”.

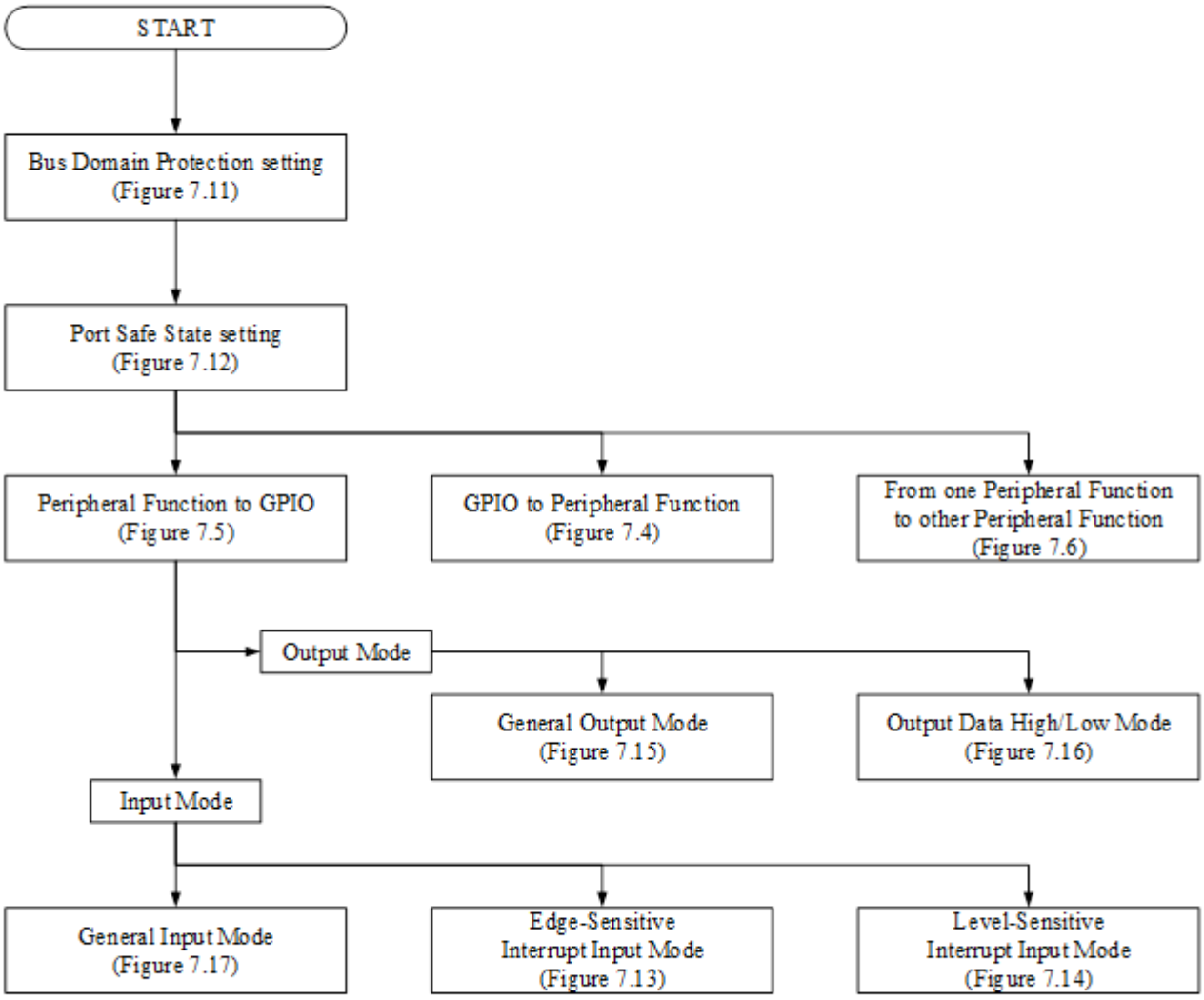



**Figure 7.3 Recomendation of Pin Function setting flow**

### 7.3.2 Function Setting for Multiplexed Pins

Setting the LSI multiplexed pin setting mask register (PMMRn) is necessary before setting each of the GPIO/peripheral function select register GPSRn, peripheral function select registers IP0SRn to IP3SRn, DRV control registers DRV0CTRLn to DRV3CTRLn, TDSEL control registers TD0SELn and TD1SELn, POWER Condition control registers POCn and Module select register MODSELn. Specifically, the inverse of the value to be set in the select register must be written to the LSI multiplexed pin setting mask register. Otherwise, the GPIO/peripheral function select register (GPSRn) and peripheral function select registers 0 to 3 (IP0SRn to IP3SRn) cannot be set.

Caution…

Some input functions is assigned to more than one pin. Only activate one single pin to one given peripheral input function. Do not activate a input function on multiple pins at the same time. Do not set peripheral selection to a place where the peripheral function is not assigned.

Ex) HSCIF3_A and HSCIF3_B

HSCIF3 is assigned to the following pins on this device. However, HSCIF3 function should not be activated on more than one pin. After activating the function on one pin, do not activate it on another.

#### (1) Procedure for changing pin function from GPIO to peripheral function

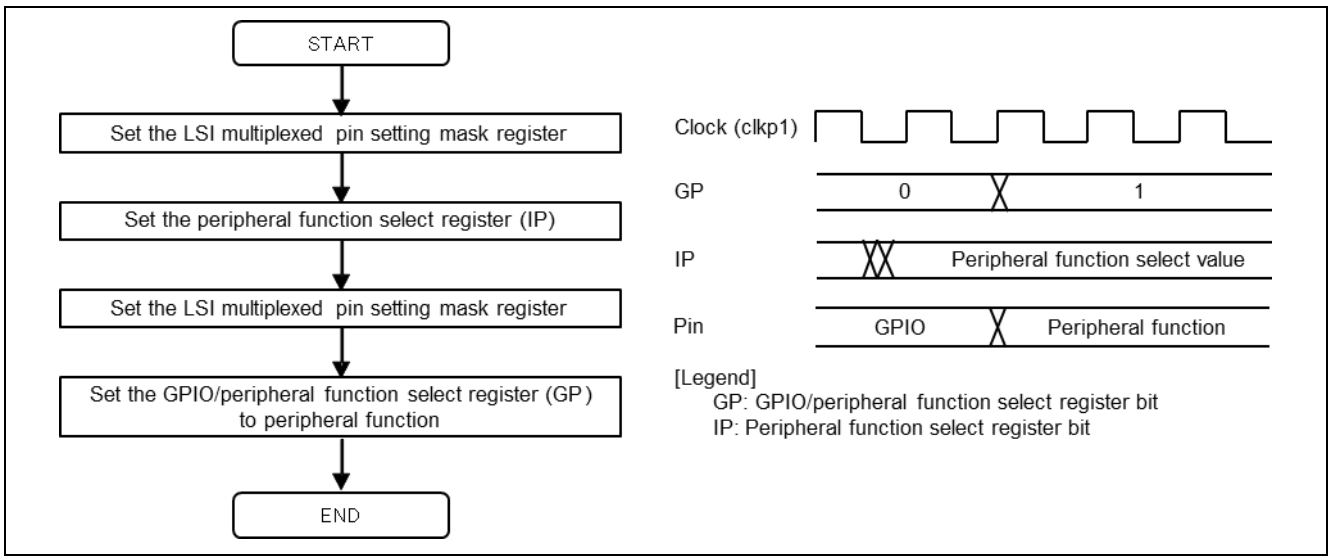



**Figure 7.4 Procedure for Changing Pin Function from GPIO to Peripheral Function**

#### (2) Procedure for changing pin function from peripheral function to GPIO

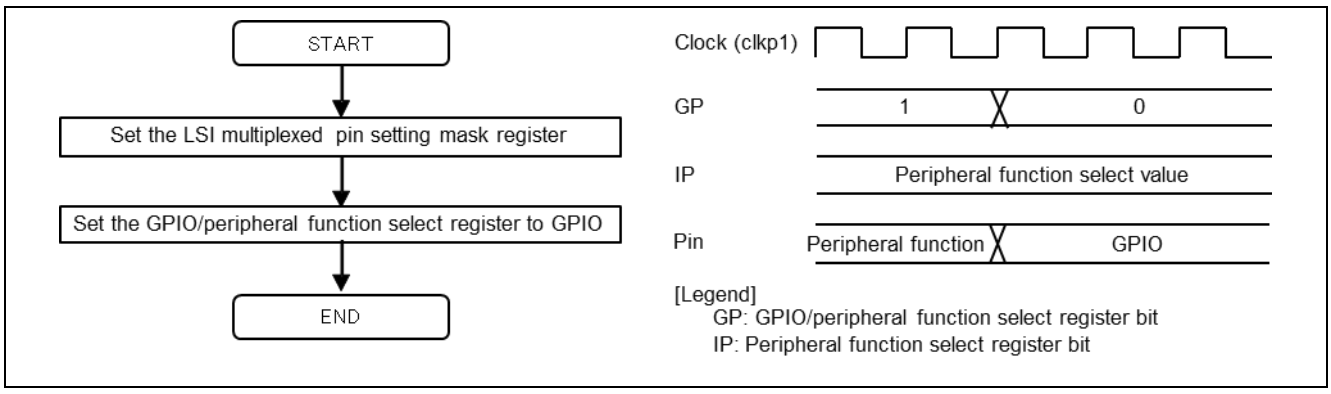



**Figure 7.5 Procedure for Changing Pin Function from Peripheral function GPIO**

**(3) Procedure 1 for changing pin function from one peripheral function to another peripheral function**

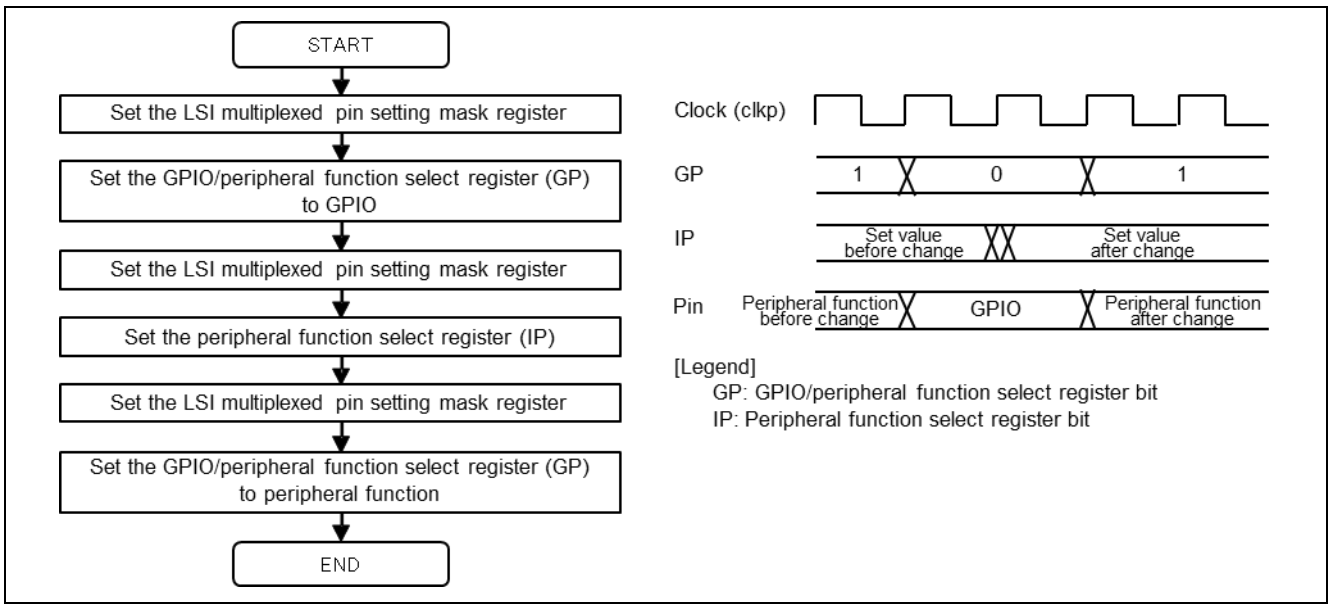



**Figure 7.6 Procedure for Changing Pin Function from One Peripheral Function to Another Peripheral Function (with GPIO Setting)**

### 7.3.3 Setting Pull-Up / Down Resistors

The LSI pin pull-on / off control register (PUENn) and pull-up / down control register (PUDn) areused.

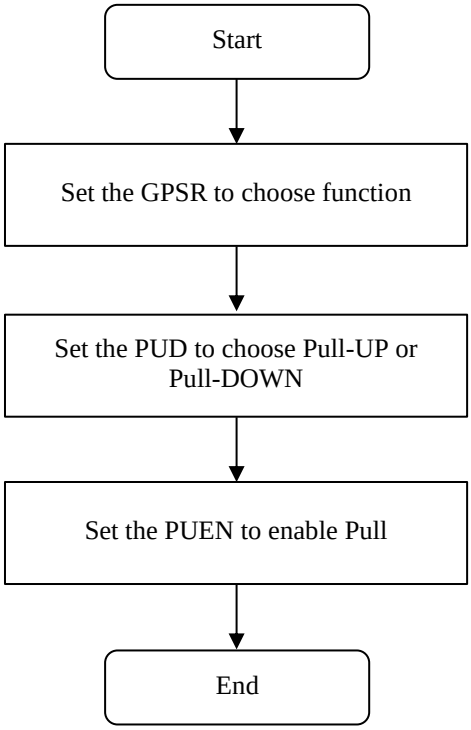



**Figure 7.7 Procedure for setting Pull-UP/DOWN**

### 7.3.4 Port Pin Specifications

Each GPIO group is provided with up to 32 port pins for general input / output and external interrupt input ports. Table 7.48 specifies these pins.

- Either general input / output mode or interrupt input mode can be set for each port.
- In general input mode, the polarity of input signals can be set for each port.
- In general output mode, the polarity of output signals can be set for each port.
- In interrupt input mode, the polarity of interrupt signal can be set for each port.
  In interrupt input mode, the detection conditions of interrupt signals can be set for each port.

**Table 7.50 Port Pin Specifications**

| Applicable Registers | Port Group | External Pin Name | Abbreviation | Name |
|---|---|---|---|---|
| IOINTSELn<br>INOUTSELn<br>OUTDTn<br>INDTn<br>INTDTn<br>INTCLRn<br>INTMSKn<br>MSKCLRn<br>POSNEGn<br>EDGLEVELn<br>FILONOFFn<br>OUTDTSELn<br>OUTDTHn<br>OUTDTLn<br>BOTHEDGEn<br>INENn<br>( n = 0 to 8) | GPIO Group 0 | GP0_00 | GP0_00 | IO/interrupt input port GP0_00 |
| | | GP0_01 | GP0_01 | IO/interrupt input port GP0_01 |
| | | GP0_02 | GP0_02 | IO/interrupt input port GP0_02 |
| | | IRQ3 | GP0_03 | IO/interrupt input port GP0_03 |
| | | IRQ2 | GP0_04 | IO/interrupt input port GP0_04 |
| | | IRQ1 | GP0_05 | IO/interrupt input port GP0_05 |
| | | IRQ0 | GP0_06 | IO/interrupt input port GP0_06 |
| | | MSIOF5_SS2 | GP0_07 | IO/interrupt input port GP0_07 |
| | | MSIOF5_SS1 | GP0_08 | IO/interrupt input port GP0_08 |
| | | MSIOF5_SYNC | GP0_09 | IO/interrupt input port GP0_09 |
| | | MSIOF5_TXD | GP0_10 | IO/interrupt input port GP0_10 |
| | | MSIOF5_SCK | GP0_11 | IO/interrupt input port GP0_11 |
| | | MSIOF5_RXD | GP0_12 | IO/interrupt input port GP0_12 |
| | | MSIOF2_SS2 | GP0_13 | IO/interrupt input port GP0_13 |
| | | MSIOF2_SS1 | GP0_14 | IO/interrupt input port GP0_14 |
| | | MSIOF2_SYNC | GP0_15 | IO/interrupt input port GP0_15 |
| | | MSIOF2_TXD | GP0_16 | IO/interrupt input port GP0_16 |
| | | MSIOF2_SCK | GP0_17 | IO/interrupt input port GP0_17 |
| | | MSIOF2_RXD | GP0_18 | IO/interrupt input port GP0_18 |
| | GPIO Group 1 | MSIOF1_SS2 | GP1_00 | IO/interrupt input port GP1_00 |
| | | MSIOF1_SS1 | GP1_01 | IO/interrupt input port GP1_01 |
| | | MSIOF1_SYNC | GP1_02 | IO/interrupt input port GP1_02 |
| | | MSIOF1_SCK | GP1_03 | IO/interrupt input port GP1_03 |
| | | MSIOF1_TXD | GP1_04 | IO/interrupt input port GP1_04 |
| | | MSIOF1_RXD | GP1_05 | IO/interrupt input port GP1_05 |
| | | MSIOF0_SS2 | GP1_06 | IO/interrupt input port GP1_06 |
| | | MSIOF0_SS1 | GP1_07 | IO/interrupt input port GP1_07 |
| | | MSIOF0_SYNC | GP1_08 | IO/interrupt input port GP1_08 |
| | | MSIOF0_TXD | GP1_09 | IO/interrupt input port GP1_09 |
| | | MSIOF0_SCK | GP1_10 | IO/interrupt input port GP1_10 |
| | | MSIOF0_RXD | GP1_11 | IO/interrupt input port GP1_11 |
| | | HTX0 | GP1_12 | IO/interrupt input port GP1_12 |
| | | HCTS0# | GP1_13 | IO/interrupt input port GP1_13 |
| | | HRTS0# | GP1_14 | IO/interrupt input port GP1_14 |
| | | HSCK0 | GP1_15 | IO/interrupt input port GP1_15 |
| | | HRX0 | GP1_16 | IO/interrupt input port GP1_16 |
| | | SCIF_CLK | GP1_17 | IO/interrupt input port GP1_17 |
| | | SSI_SCK | GP1_18 | IO/interrupt input port GP1_18 |
| | | SSI_WS | GP1_19 | IO/interrupt input port GP1_19 |
| | | SSI_SD | GP1_20 | IO/interrupt input port GP1_20 |
| | | AUDIO_CLKOUT | GP1_21 | IO/interrupt input port GP1_21 |
| | | AUDIO_CLKIN | GP1_22 | IO/interrupt input port GP1_22 |
| | | GP1_23 | GP1_23 | IO/interrupt input port GP1_23 |
| | | HRX3 | GP1_24 | IO/interrupt input port GP1_24 |
| | | HSCK3 | GP1_25 | IO/interrupt input port GP1_25 |

| Applicable Registers | Port Group | External Pin Name | Abbreviation | Name |
|---|---|---|---|---|
| | | HRTS3# | GP1_26 | IO/interrupt input port GP1_26 |
| | | HCTS3# | GP1_27 | IO/interrupt input port GP1_27 |
| | | HTX3 | GP1_28 | IO/interrupt input port GP1_28 |
| | | ERROROUTC# | GP1_29 | IO/interrupt input port GP1_29 |
| | GPIO Group 2 | FXR_TXDA | GP2_00 | IO/interrupt input port GP2_00 |
| | | FXR_TXENA# | GP2_01 | IO/interrupt input port GP2_01 |
| | | RXDA_EXTFXR | GP2_02 | IO/interrupt input port GP2_02 |
| | | CLK_EXTFXR | GP2_03 | IO/interrupt input port GP2_03 |
| | | RXDB_EXTFXR | GP2_04 | IO/interrupt input port GP2_04 |
| | | FXR_TXENB# | GP2_05 | IO/interrupt input port GP2_05 |
| | | FXR_TXDB | GP2_06 | IO/interrupt input port GP2_06 |
| | | TPU0TO1 | GP2_07 | IO/interrupt input port GP2_07 |
| | | TPU0TO0 | GP2_08 | IO/interrupt input port GP2_08 |
| | | CAN_CLK | GP2_09 | IO/interrupt input port GP2_09 |
| | | CANFD0_TX | GP2_10 | IO/interrupt input port GP2_10 |
| | | CANFD0_RX | GP2_11 | IO/interrupt input port GP2_11 |
| | | CANFD2_TX | GP2_12 | IO/interrupt input port GP2_12 |
| | | CANFD2_RX | GP2_13 | IO/interrupt input port GP2_13 |
| | | CANFD3_TX | GP2_14 | IO/interrupt input port GP2_14 |
| | | CANFD3_RX | GP2_15 | IO/interrupt input port GP2_15 |
| | | CANFD1_TX | GP2_17 | IO/interrupt input port GP2_17 |
| | | CANFD1_RX | GP2_19 | IO/interrupt input port GP2_19 |
| | GPIO Group 3 | MMC_SD_D1 | GP3_00 | IO/interrupt input port GP3_00 |
| | | MMC_SD_D0 | GP3_01 | IO/interrupt input port GP3_01 |
| | | MMC_SD_D2 | GP3_02 | IO/interrupt input port GP3_02 |
| | | MMC_SD_CLK | GP3_03 | IO/interrupt input port GP3_03 |
| | | MMC_DS | GP3_04 | IO/interrupt input port GP3_04 |
| | | MMC_SD_D3 | GP3_05 | IO/interrupt input port GP3_05 |
| | | MMC_D5 | GP3_06 | IO/interrupt input port GP3_06 |
| | | MMC_D4 | GP3_07 | IO/interrupt input port GP3_07 |
| | | MMC_D7 | GP3_08 | IO/interrupt input port GP3_08 |
| | | MMC_D6 | GP3_09 | IO/interrupt input port GP3_09 |
| | | MMC_SD_CMD | GP3_10 | IO/interrupt input port GP3_10 |
| | | SD_CD | GP3_11 | IO/interrupt input port GP3_11 |
| | | SD_WP | GP3_12 | IO/interrupt input port GP3_12 |
| | | PWM1 | GP3_13 | IO/interrupt input port GP3_13 |
| | | PWM2 | GP3_14 | IO/interrupt input port GP3_14 |
| | | QSPI0_SSL | GP3_15 | IO/interrupt input port GP3_15 |
| | | QSPI0_IO3 | GP3_16 | IO/interrupt input port GP3_16 |
| | | QSPI0_IO2 | GP3_17 | IO/interrupt input port GP3_17 |
| | | QSPI0_MISO_IO1 | GP3_18 | IO/interrupt input port GP3_18 |
| | | QSPI0_MOSI_IO0 | GP3_19 | IO/interrupt input port GP3_19 |
| | | QSPI0_SPCLK | GP3_20 | IO/interrupt input port GP3_20 |
| | | QSPI1_MOSI_IO0 | GP3_21 | IO/interrupt input port GP3_21 |
| | | QSPI1_SPCLK | GP3_22 | IO/interrupt input port GP3_22 |
| | | QSPI1_MISO_IO1 | GP3_23 | IO/interrupt input port GP3_23 |
| | | QSPI1_IO2 | GP3_24 | IO/interrupt input port GP3_24 |
| | | QSPI1_SSL | GP3_25 | IO/interrupt input port GP3_25 |
| | | QSPI1_IO3 | GP3_26 | IO/interrupt input port GP3_26 |
| | | RPC_RESET# | GP3_27 | IO/interrupt input port GP3_27 |
| | | RPC_WP# | GP3_28 | IO/interrupt input port GP3_28 |
| | | RPC_INT# | GP3_29 | IO/interrupt input port GP3_29 |
| | GPIO Group 4 | SCL0 | GP4_00 | IO/interrupt input port GP4_00 |
| | | SDA0 | GP4_01 | IO/interrupt input port GP4_01 |
| | | SCL1 | GP4_02 | IO/interrupt input port GP4_02 |
| | | SDA1 | GP4_03 | IO/interrupt input port GP4_03 |
| | | SCL2 | GP4_04 | IO/interrupt input port GP4_04 |
| | | SDA2 | GP4_05 | IO/interrupt input port GP4_05 |

| Applicable Registers | Port Group | External Pin Name | Abbreviation | Name |
|---|---|---|---|---|
| | | SCL3 | GP4_06 | IO/interrupt input port GP4_06 |
| | | SDA3 | GP4_07 | IO/interrupt input port GP4_07 |
| | | HRX2 | GP4_08 | IO/interrupt input port GP4_08 |
| | | HTX2 | GP4_09 | IO/interrupt input port GP4_09 |
| | | HRTS2# | GP4_10 | IO/interrupt input port GP4_10 |
| | | SCIF_CLK2 | GP4_11 | IO/interrupt input port GP4_11 |
| | | HCTS2# | GP4_12 | IO/interrupt input port GP4_12 |
| | | HSCK2 | GP4_13 | IO/interrupt input port GP4_13 |
| | | PWM3 | GP4_14 | IO/interrupt input port GP4_14 |
| | | PWM4 | GP4_15 | IO/interrupt input port GP4_15 |
| | | PCIE0_CLKREQ# | GP4_16 | IO/interrupt input port GP4_21 |
| | | AVS0 | GP4_17 | IO/interrupt input port GP4_23 |
| | | AVS1 | GP4_18 | IO/interrupt input port GP4_24 |
| | GPIO Group 5 | AVB2_AVTP_PPS | GP5_00 | IO/interrupt input port GP5_00 |
| | | AVB2_AVTP_CAPTURE | GP5_01 | IO/interrupt input port GP5_01 |
| | | AVB2_AVTP_MATCH | GP5_02 | IO/interrupt input port GP5_02 |
| | | AVB2_LINK | GP5_03 | IO/interrupt input port GP5_03 |
| | | AVB2_PHY_INT | GP5_04 | IO/interrupt input port GP5_04 |
| | | AVB2_MAGIC | GP5_05 | IO/interrupt input port GP5_05 |
| | | AVB2_MDC | GP5_06 | IO/interrupt input port GP5_06 |
| | | AVB2_TXCREFCLK | GP5_07 | IO/interrupt input port GP5_07 |
| | | AVB2_TD3 | GP5_08 | IO/interrupt input port GP5_08 |
| | | AVB2_RD3 | GP5_09 | IO/interrupt input port GP5_09 |
| | | AVB2_MDIO | GP5_10 | IO/interrupt input port GP5_10 |
| | | AVB2_TD2 | GP5_11 | IO/interrupt input port GP5_11 |
| | | AVB2_TD1 | GP5_12 | IO/interrupt input port GP5_12 |
| | | AVB2_RD2 | GP5_13 | IO/interrupt input port GP5_13 |
| | | AVB2_RD1 | GP5_14 | IO/interrupt input port GP5_14 |
| | | AVB2_TD0 | GP5_15 | IO/interrupt input port GP5_15 |
| | | AVB2_TXC | GP5_16 | IO/interrupt input port GP5_16 |
| | | AVB2_RD0 | GP5_17 | IO/interrupt input port GP5_17 |
| | | AVB2_RXC | GP5_18 | IO/interrupt input port GP5_18 |
| | | AVB2_TX_CTL | GP5_19 | IO/interrupt input port GP5_19 |
| | | AVB2_RX_CTL | GP5_20 | IO/interrupt input port GP5_20 |
| | GPIO Group 6 | AVB1_MDIO | GP6_00 | IO/interrupt input port GP6_00 |
| | | AVB1_MAGIC | GP6_01 | IO/interrupt input port GP6_01 |
| | | AVB1_MDC | GP6_02 | IO/interrupt input port GP6_02 |
| | | AVB1_PHY_INT | GP6_03 | IO/interrupt input port GP6_03 |
| | | AVB1_LINK | GP6_04 | IO/interrupt input port GP6_04 |
| | | AVB1_AVTP_MATCH | GP6_05 | IO/interrupt input port GP6_05 |
| | | AVB1_TXC | GP6_06 | IO/interrupt input port GP6_06 |
| | | AVB1_TX_CTL | GP6_07 | IO/interrupt input port GP6_07 |
| | | AVB1_RXC | GP6_08 | IO/interrupt input port GP6_08 |
| | | AVB1_RX_CTL | GP6_09 | IO/interrupt input port GP6_09 |
| | | AVB1_AVTP_PPS | GP6_10 | IO/interrupt input port GP6_10 |
| | | AVB1_AVTP_CAPTURE | GP6_11 | IO/interrupt input port GP6_11 |
| | | AVB1_TD1 | GP6_12 | IO/interrupt input port GP6_12 |
| | | AVB1_TD0 | GP6_13 | IO/interrupt input port GP6_13 |
| | | AVB1_RD1 | GP6_14 | IO/interrupt input port GP6_14 |
| | | AVB1_RD0 | GP6_15 | IO/interrupt input port GP6_15 |
| | | AVB1_TD2 | GP6_16 | IO/interrupt input port GP6_16 |
| | | AVB1_RD2 | GP6_17 | IO/interrupt input port GP6_17 |
| | | AVB1_TD3 | GP6_18 | IO/interrupt input port GP6_18 |
| | | AVB1_RD3 | GP6_19 | IO/interrupt input port GP6_19 |
| | | AVB1_TXCREFCLK | GP6_20 | IO/interrupt input port GP6_20 |
| | GPIO Group 7 | AVB0_AVTP_PPS | GP7_00 | IO/interrupt input port GP7_00 |
| | | AVB0_AVTP_CAPTURE | GP7_01 | IO/interrupt input port GP7_01 |
| | | AVB0_AVTP_MATCH | GP7_02 | IO/interrupt input port GP7_02 |

| Applicable Registers | Port Group | External Pin Name | Abbreviation | Name |
|---|---|---|---|---|
| | | AVB0_TD3 | GP7_03 | IO/interrupt input port GP7_03 |
| | | AVB0_LINK | GP7_04 | IO/interrupt input port GP7_04 |
| | | AVB0_PHY_INT | GP7_05 | IO/interrupt input port GP7_05 |
| | | AVB0_TD2 | GP7_06 | IO/interrupt input port GP7_06 |
| | | AVB0_TD1 | GP7_07 | IO/interrupt input port GP7_07 |
| | | AVB0_RD3 | GP7_08 | IO/interrupt input port GP7_08 |
| | | AVB0_TXCREFCLK | GP7_09 | IO/interrupt input port GP7_09 |
| | | AVB0_MAGIC | GP7_10 | IO/interrupt input port GP7_10 |
| | | AVB0_TD0 | GP7_11 | IO/interrupt input port GP7_11 |
| | | AVB0_RD2 | GP7_12 | IO/interrupt input port GP7_12 |
| | | AVB0_MDC | GP7_13 | IO/interrupt input port GP7_13 |
| | | AVB0_MDIO | GP7_14 | IO/interrupt input port GP7_14 |
| | | AVB0_TXC | GP7_15 | IO/interrupt input port GP7_15 |
| | | AVB0_TX_CTL | GP7_16 | IO/interrupt input port GP7_16 |
| | | AVB0_RD1 | GP7_17 | IO/interrupt input port GP7_17 |
| | | AVB0_RD0 | GP7_18 | IO/interrupt input port GP7_18 |
| | | AVB0_RXC | GP7_19 | IO/interrupt input port GP7_19 |
| | | AVB0_RX_CTL | GP7_20 | IO/interrupt input port GP7_20 |

### 7.3.5 Operations in Each Mode

#### (1) Mode Switching

Two registers are used to switch modes of the general IO/interrupt input pins of the GPIO groups. Each register is provided with up to 32 bits each controlling one of the GPIOn* port pins. The general IO/interrupt switching register is first used to select either general input/output mode or interrupt input mode for each port pin. When general input/output mode is selected, the setting of the relevant bit in the second register, i.e., the general input/output switching register, is used. Specifically, when a bit in the general input/output switching register is set for general output mode, the corresponding port pin is turned to the output direction and the route is formed so that the set value in the corresponding bit in the general output register should be output via the pin. Likewise, when set for the general input mode, the corresponding port pin is turned to the input direction and the route is formed so that the value received via the pin should be indicated by the corresponding bit in the general input register. When interrupt input mode is selected, the corresponding port pin is turned to the input direction and the route is formed so that the reception of the signal input via the pin should be indicated by the interrupt display register. Here, the setting of the second register, i.e., the general input/output switching register, is invalid.

Note: * n = 0 to 7 (The groups GP4 to GP7 have no interrupt input mode.)

#### (2) General Input / Output Mode

When setting a port for general input, set high value to INEN to enable general input. When a port is set for general input/output mode using the corresponding bit in the general IO/interrupt switching register, the corresponding port serves as a general input/output pin. In general input/output mode, either mode can be selected using the corresponding bit in the general input/output switching register. When a port is set for general output mode, the port outputs the value set in the corresponding bit in the general output register or output data high/output data low register with appropriate configuring in output data select register. Here, the polarity of the actual output signal is determined by the setting of the corresponding bit in the positive/negative logic select register. When a port is set for general input mode, the polarity of the input signal is also determined by the setting of the corresponding bit in the positive/negative logic select register. The general input register indicates the value accordingly. Note that the general input register does not hold the input signal using the FF.

#### (3) Interrupt Input Mode

When a port is set for interrupt input mode using the corresponding bit in the general IO/interrupt switching register, the corresponding port serves as an interrupt input pin. In interrupt input mode, when the port receives an external interrupt, the corresponding bit in the interrupt display register indicates the input of an interrupt signal on the corresponding port pin, and an interrupt signal is output to the interrupt control block. In this mode, the polarity and detection conditions (edge or level) of the external input signal can be set for each port. The corresponding bits in the positive/negative logic select register and edge/level select register, one edge/both edge select register should be used to set the polarity and detection conditions, respectively.

If a port is set for edge detection using the corresponding bit in the edge/level select register, even when an external pulse interrupt signal is input, the corresponding bit in the interrupt display register holds the input using the FF and allows the level interrupt signal to be output to the interrupt control block. To stop all the interrupt signal outputs, all the bits in the interrupt clear register corresponding to the bits in the interrupt display register currently indicating the reception of the corresponding interrupt signals should be cleared to 0. Note that if a port is set for level detection using the corresponding bit in the edge/level select register and an external level interrupt signal is input, the corresponding bit in the interrupt display register does not use the FF to hold the input.

Interrupts indicated by the interrupt display register can be separately masked using the corresponding bits in the interrupt mask register. When all the bits currently indicating the reception of the interrupt signals are masked, no interrupt signals are output to the interrupt control block. Masks can be canceled by writing 1 to the corresponding bits in the interrupt mask clear register depending on the interrupt mask register is used.

### 7.3.6 Handling of Input Signals on Port Pins

#### (1) Chattering

In general input mode and interrupt input modes, a filtering function can be used for the port pins 0 to 7 of each GPIO group to prevent external chattering input. Specifically, when a bit in the chattering prevention on / off register is set to use the function, the external input to the corresponding port pin is sampled four consecutive times based on the filter clock signal, which is internally generated by the GPIO. The external input is canceled except when the active input is detected four consecutive times. Therefore, when a filtering function is used, input to the port pins 0 to 7 of each GPIO group need to be at least four sampling clock cycles long (The sampling clock is generated from peripheral clock/k, where k is determined by FILONOFFn. CLKSEL.).

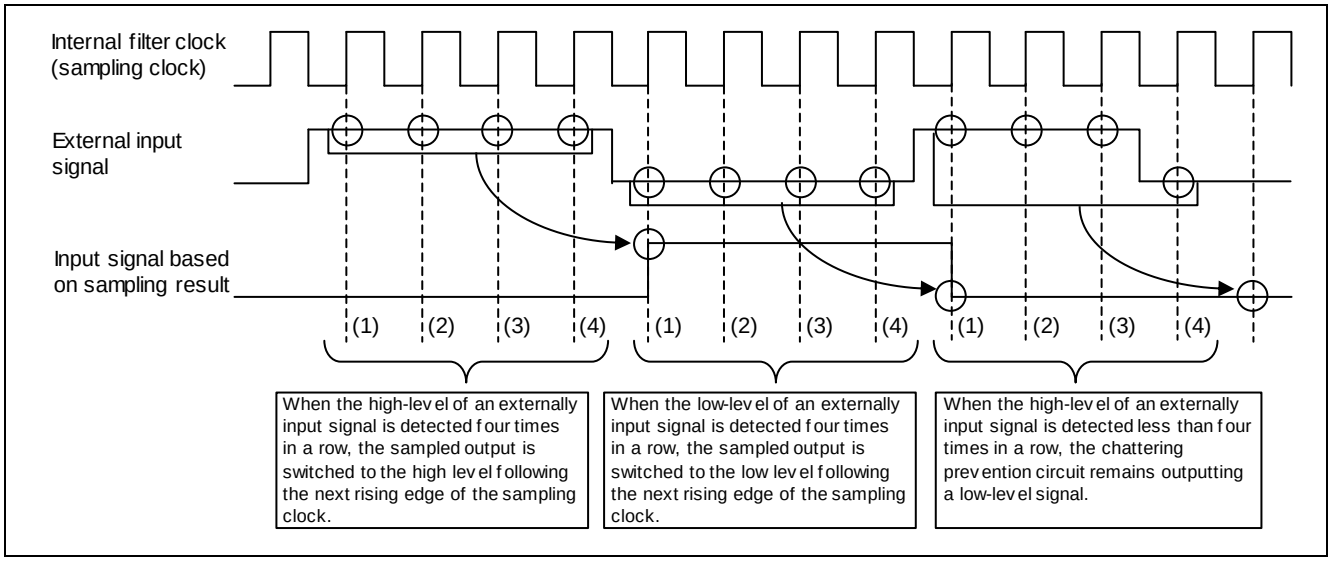



**Figure 7.8 Sampling Timing Chart**

#### (2) Input Signal Synchronization

Ingeneral input mode and interrupt input mode, external input signals on all port pins are synchronized with the GPIO clock (CPϕ).

### 7.3.7 Interrupt Display Timing Charts

Figure 7.9 shows the interrupt display timing and Figure 7.10 shows the note on the timing. In both figures, the positive logic and edge-sensitive input are assumed.

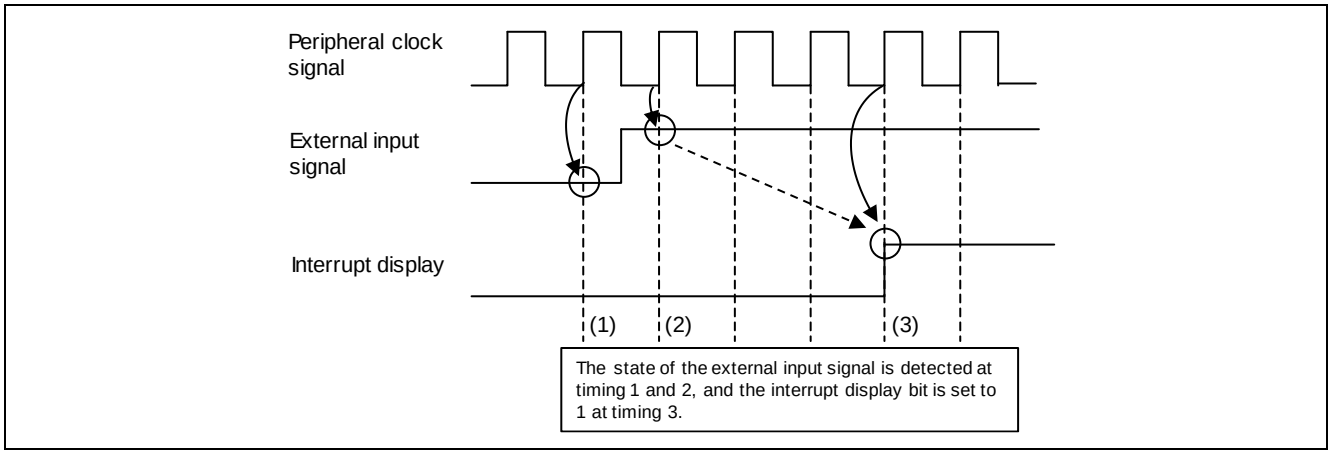



**Figure 7.9 Interrupt Display Timing**

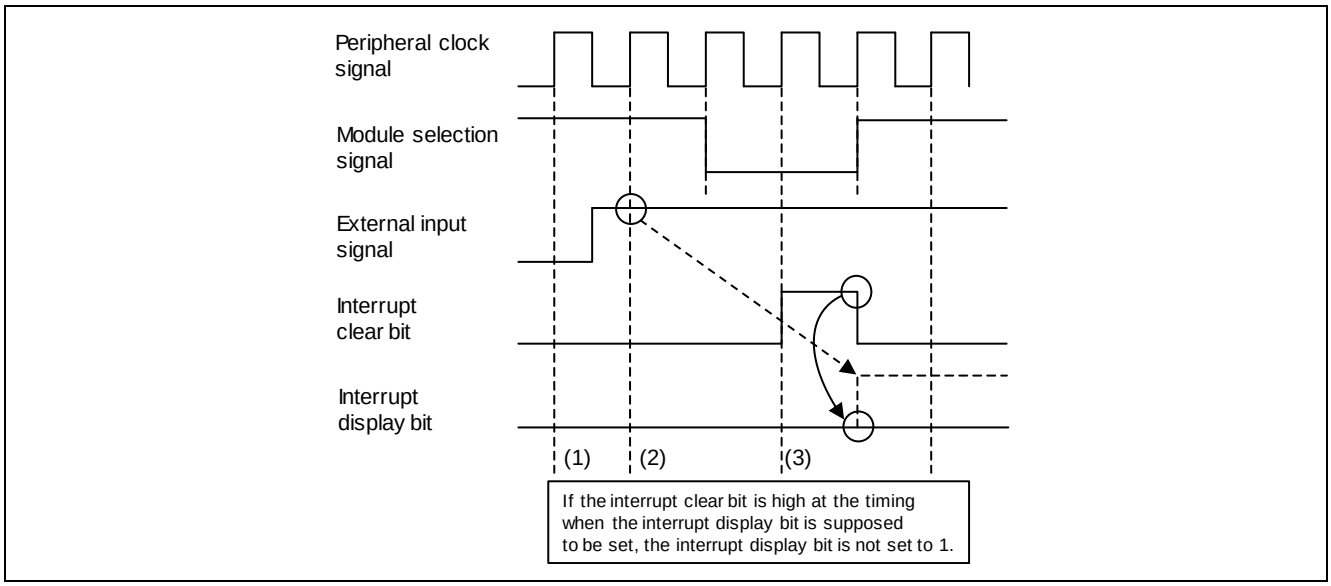



**Figure 7.10 Note on Interrupt Display Timing**

### 7.3.8 Using GPIO

The following sections describe how to use the GPIO. If the GPIO is not used according to the procedures shown here, operations are not guaranteed.

#### (1) Setting Bus Domain Protection

For setting Bus Domain Protection, refer to the procedure shown in Figure 7.11.

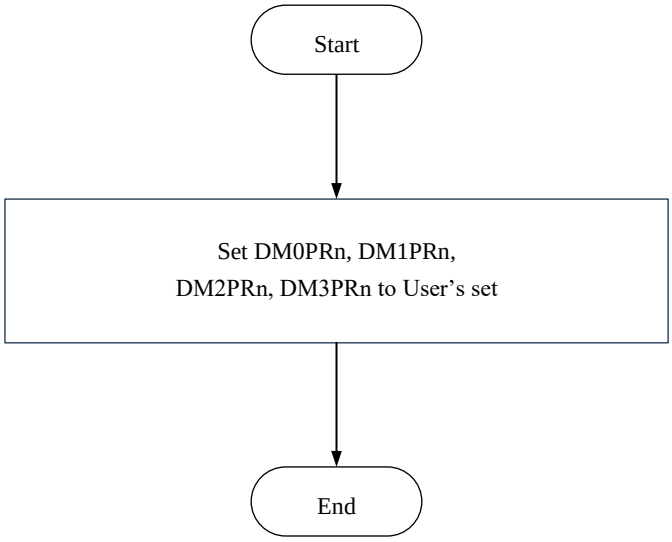



**Figure 7.11 Setting flow of Bus Domain Protection**

### (2) Setting Port Safe State

For setting Port Safe State, refer to the procedure shown in Figure 7.12.

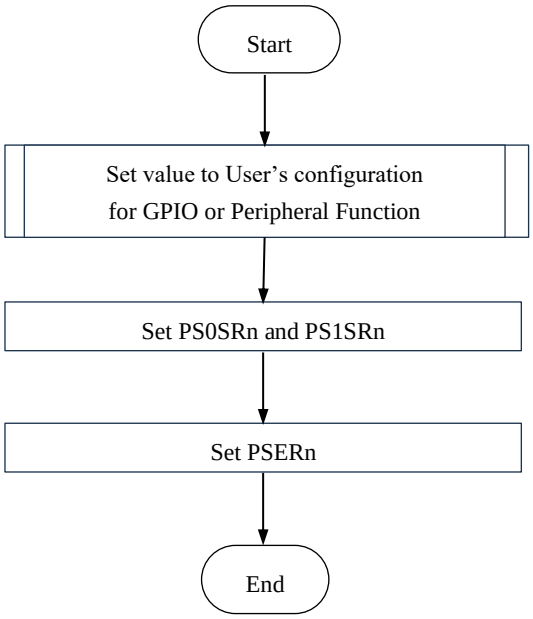



**Figure 7.12 Setting flow of Port Safe State**

### (3) Setting Edge-Sensitive Interrupt Input Mode

For setting edge-sensitive interrupt input mode, refer to the procedure shown in Figure 7.13.

Note that an unexpected interrupt might be generated in the module if setting (1), (2), (3) or (4) in the flowchart is changed. When changing the setting, (5) and (6) should be done.

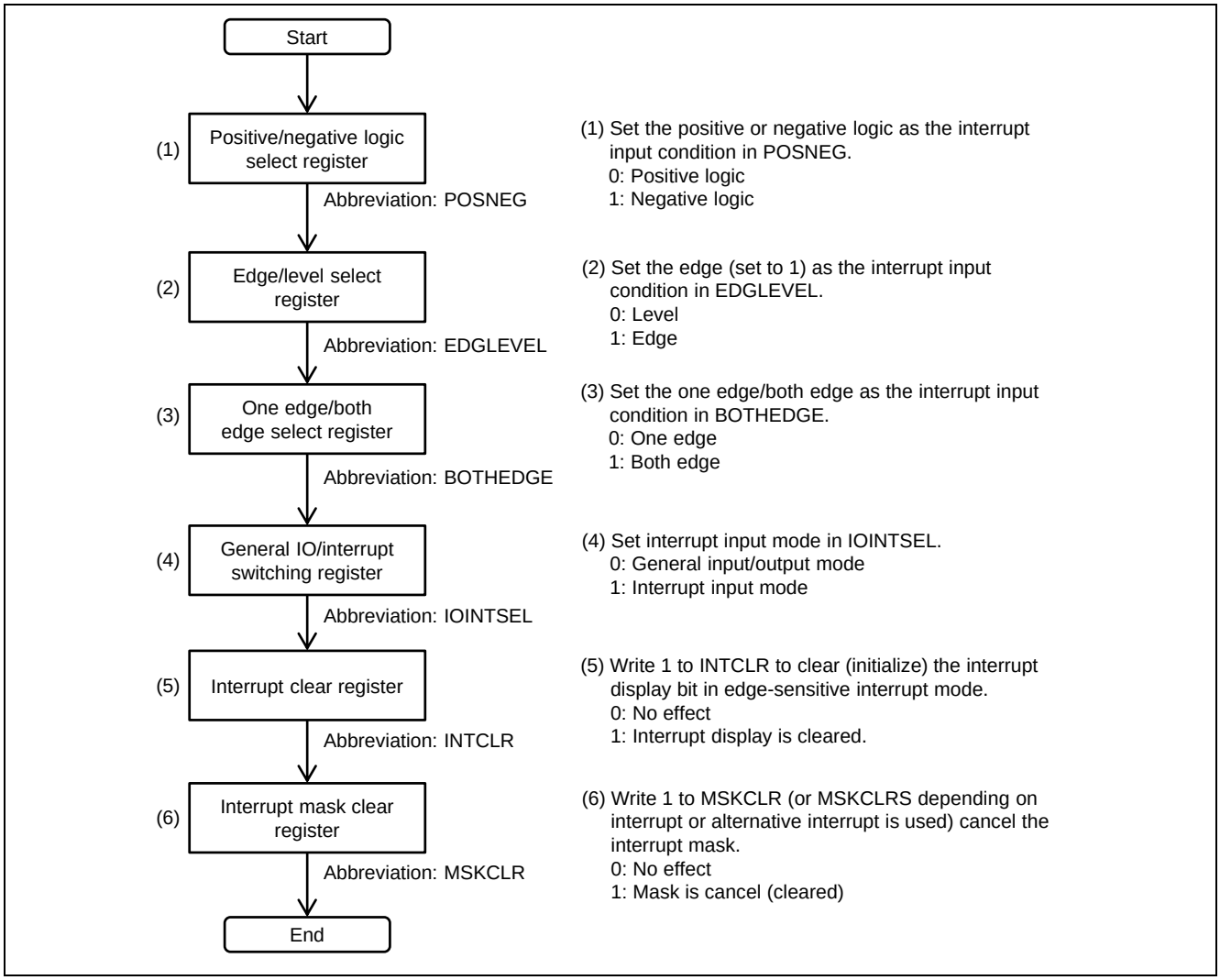



**Figure 7.13 Flowchart of Setting the GPIO to Edge-Sensitive Interrupt Input Mode**

### (4) Setting Level-Sensitive Interrupt Input Mode

For setting level-sensitive interrupt input mode, refer to the procedure shown in Figure 7.14.

Note that when an external level-sensitive interrupt input signal is stopped, the corresponding interrupt is canceled automatically. Inlevel-sensitive interrupt input mode, the interrupt clear register is invalid.

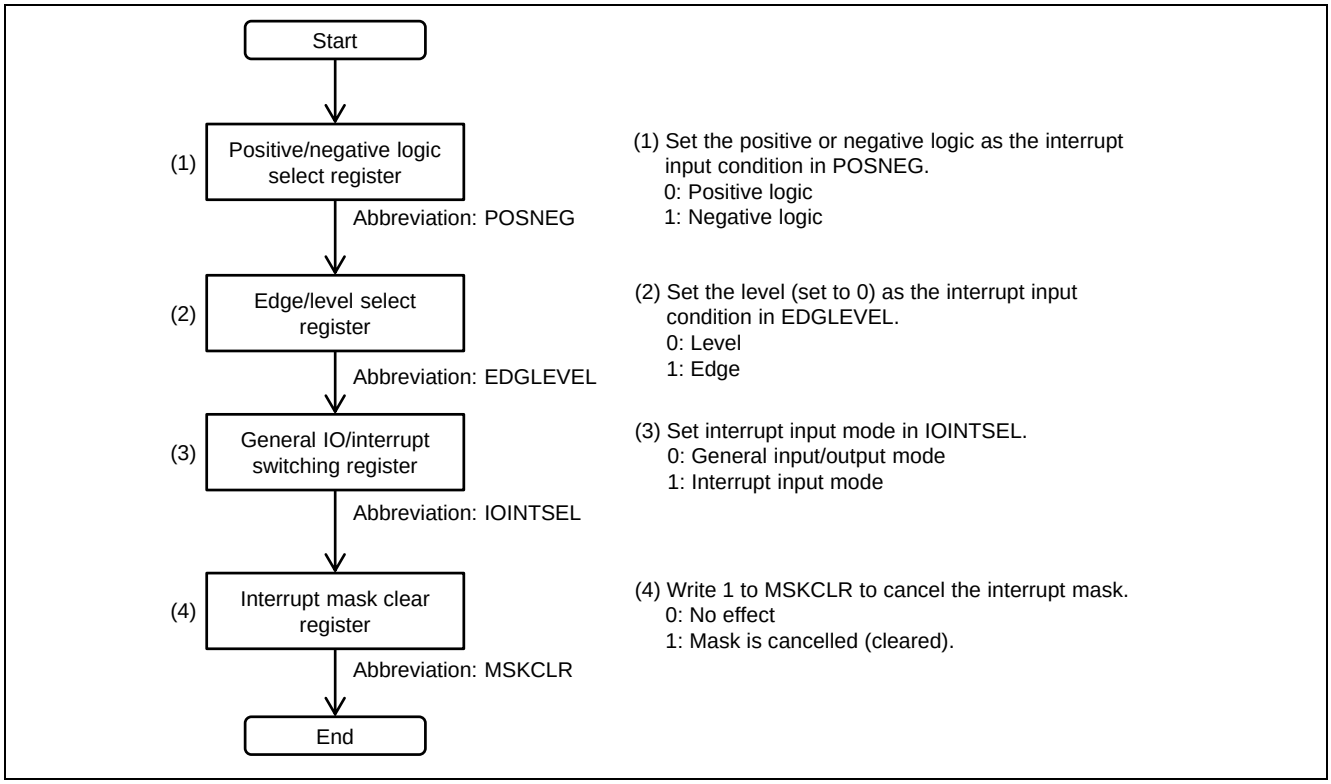



**Figure 7.14 Flowchart of Setting the GPIO to Level-Sensitive Interrupt Input Mode**

### (5) Setting General Output Mode

For setting general output mode, refer to the procedure shown in **Figure 7.15**.

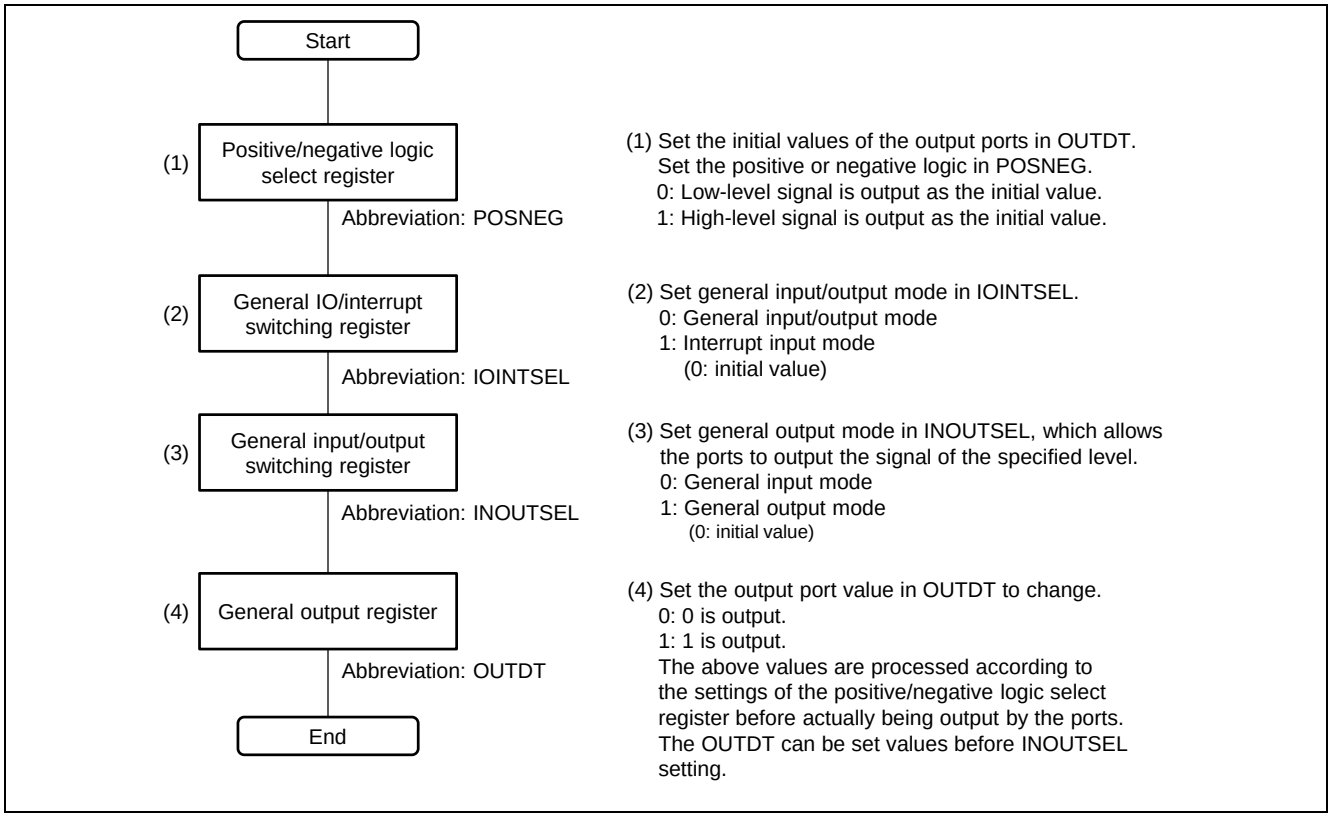



**Figure 7.15 Flowchart of Setting the GPIO to General Output Mode**

### (6) Setting Output data high / Output data low Mode

For setting output data high / output data low mode, refer to the procedure shown in Figure 7.16.

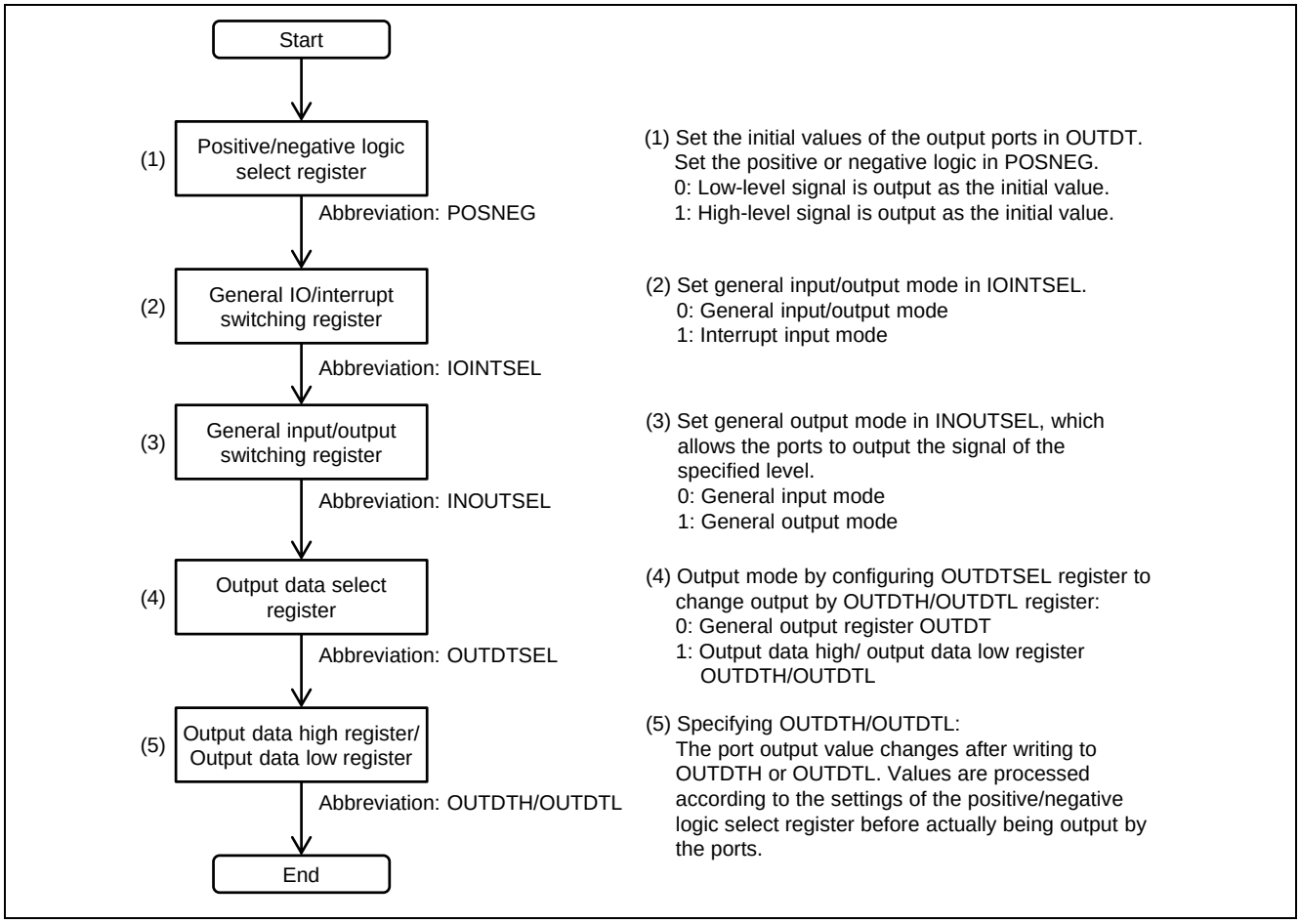



**Figure 7.16 Flowchart of Setting the GPIO to Output data high / Output data low Mode**

### (7) Setting General Input Mode

For setting general input mode, refer to the procedure shown in Figure 7.17.

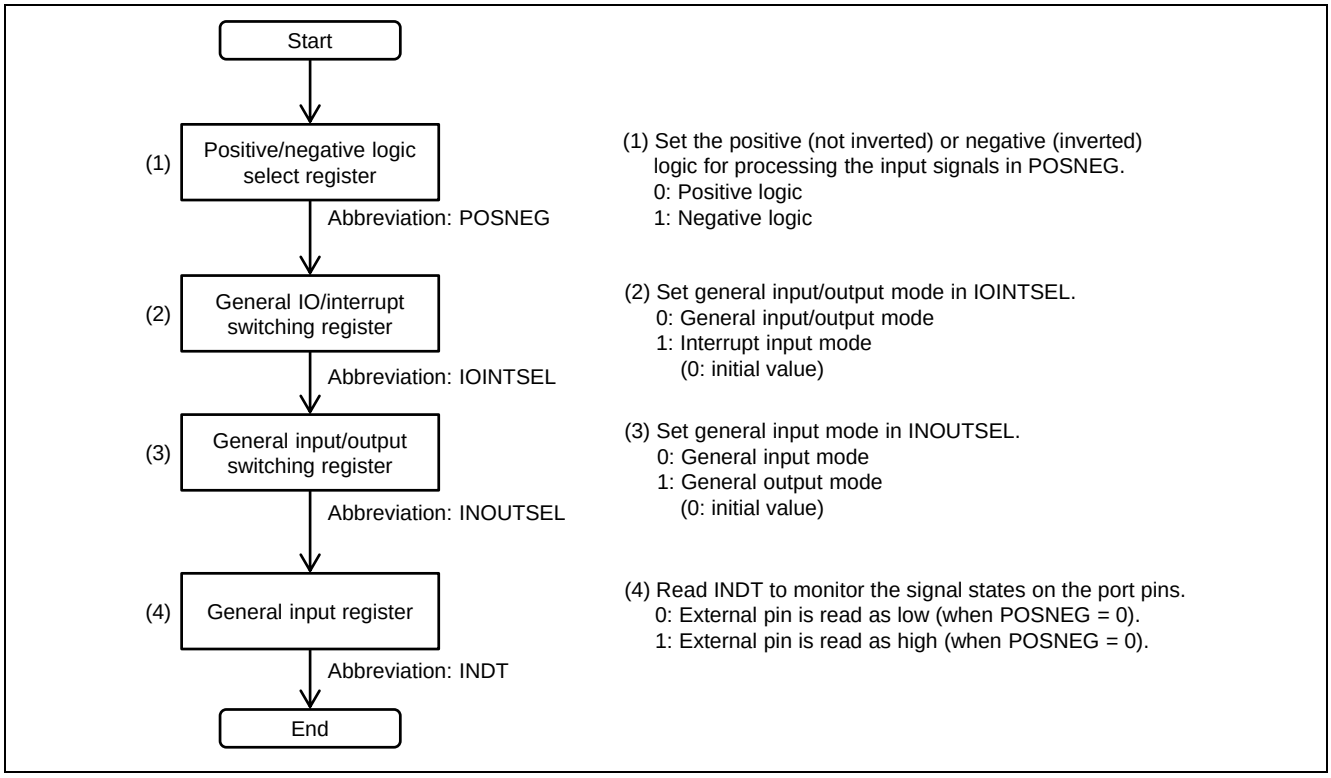



**Figure 7.17 Flowchart of Setting the GPIO to General Input Mode**

### 7.3.9 Port Safe State Function

To keep safe operating state (Port Safe State) in case of failure, and then stop it, Port Safe State operate to receive fault information from ECM and make Pin for each domain.

Output trouble information signal for each domain from ECM. This is received by the PFC, and it is controlled for each Pin so that it becomes Port Safe State.

Registers (PSSR 0, 1) that select 4 types of Port Safe State with 2 bits. Provide a register (PSER) to enable / disable this register.

**Table 7.51 Port Safe State function**

| INPUT | | | OUTPUT |
|---|---|---|---|
| PSERn | PS1SRn | | |
| bit[m*+16] | bit[m*×2+1] | bit[m*×2] | Port Safe State |
| 0 | * | * | Port Safe State is disable |
| 1 | 0 | 0 | Initial state = ON |
| 1 | 0 | 1 | HiZ = ON |
| 1 | 1 | 0 | Pull Down = ON |
| 1 | 1 | 1 | Pull Up = ON |

Note: * m = 0 to 15

| INPUT | | | OUTPUT |
|---|---|---|---|
| PSERn | PS0SRn | | |
| bit[m*] | bit[m*×2+1] | bit[m*×2] | Port Safe State |
| 0 | * | * | Port Safe State is disable |
| 1 | 0 | 0 | Initial state = ON |
| 1 | 0 | 1 | HiZ = ON |
| 1 | 1 | 0 | Pull Down = ON |
| 1 | 1 | 1 | Pull Up = ON |

Note: * m = 0 to 15

### 7.3.10 Switching signal level voltage

This is setting flow of POCn register to switch voltage 3.3/2.5V to 1.8V.
Initial setting value of POCn is available in “7.2.7 POWER Condition Control Register”.

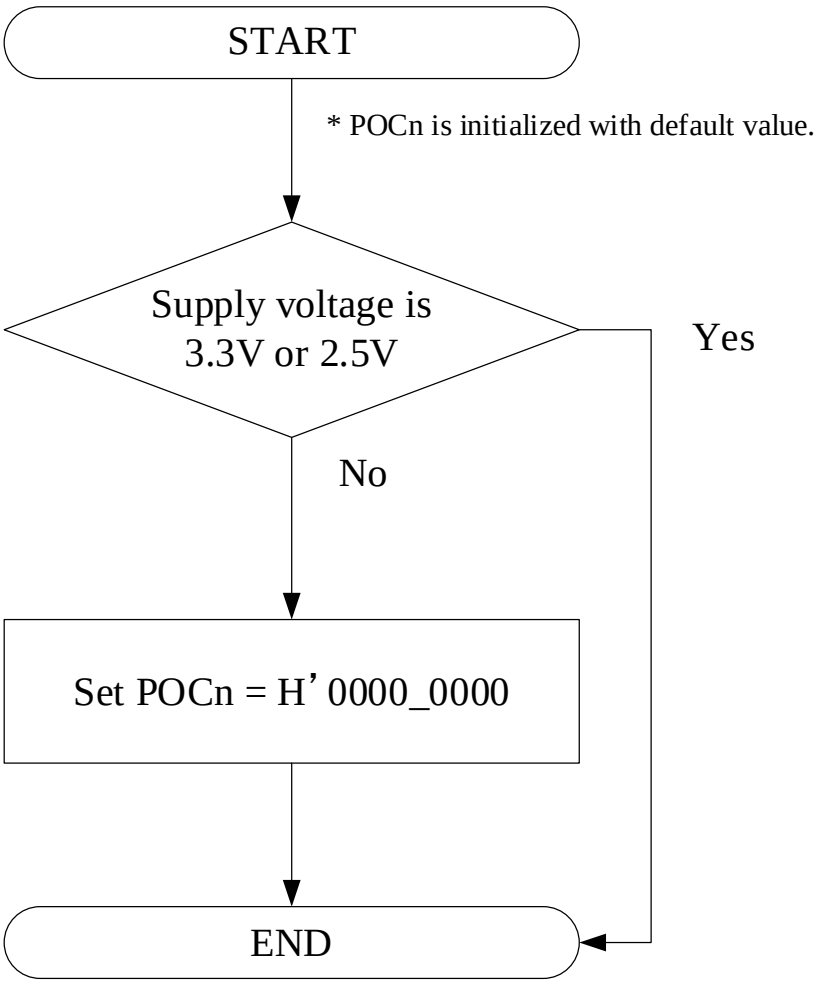



**Figure 7.18 Voltage switching flow**

## 7.4 Usage Notes

### 7.4.1 Notes for asserted PRESET#

When PRESET# is asserted, a small glitch may occur at the pin configured to the peripheral output function.
Please be aware of this behavior if the connected device expects to continue the stable state at the PRESET# asserted timing.
If the timing of the PRESET# can be determined in advance, please set the pin to GPIO mode upfront of the PRESET# to avoid the situation.

### 7.4.2 Notes for PS0SRn/PS1SRn/PSERn

The registers “PS0SRn/PS1SRn/PSERn” are only supporting condition stating when ECM find any error cause on supporting pins. These are not supporting countermeasure of any gridges on pins.

# 8. Clock Pulse Generator (CPG)

## 8.1 Overview

This LSI has a clock pulse generator (CPG). The CPG consists of oscillators, PLL circuit n (n = 1,2,3,4,5,6,7), clock dividers, and control circuit. The CPG generates various clocks used by this LSI.

### 8.1.1 Features

#### (1) Clock Generate

CPG generates various clocks for LSI internal operation. As shown in Figure 8.1.2, PLL circuit n (n = 1,2,3,4,5,6,7) and oscillators generate the clocks. Some PLLs support not only integer multiplication mode also SSCG (Down spread or Center spread), fractional multiplication mode, which is a technique to reduce EMI (Electromagnetic Interference). The detail specification is shown in 8.1.5.

**Table 8.1.1 PLL circuit n (n = 1,2,3,4,5,6,7) configuration**

| PLL circuit n | SSCG (Spread Spectrum Clock Generator) (*1) | Fractional Multiplication (*2) |
|---|---|---|
| PLL2, PLL6 | Support (Down spread) | Support |
| PLL4 | Support (Down spread) | Support |
| PLL1 | Support (Down spread, Center spread) | Support |
| PLL3 | Not support | Support |
| PLL5, PLL7 | Not support | Not support |

**Notes (1)**

SSCG = Spread Spectrum Clock Generator.
It modulates the output clock frequency over a small range, spreading the energy and reducing the energy peak.
Down spread: Down spread spectrum clocking modulates below the nominal clock frequency.
Center spread: Center spread spectrum clocking modulates evenly above and below the nominal clock frequency.

**Notes (2)**

The fractional multiplication divider divides the frequency of the input signal by a tunable fractional value.

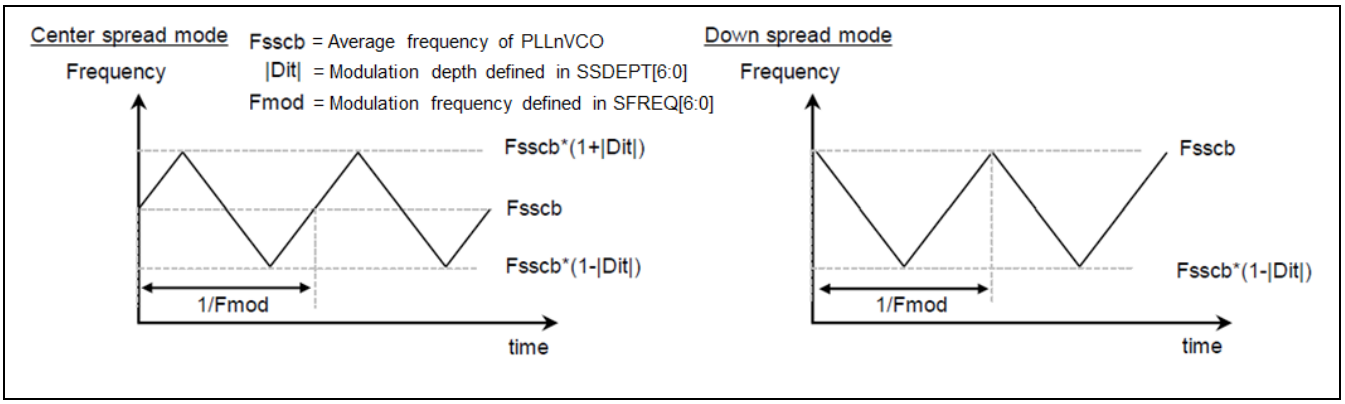



**Figure 8.1.1 Image diagram of SSCG (Down spread and Center spread)**

### (2) Clock Control

CPG controls the clock supply and frequency according to the module status (power supply status, etc.). There are clock dividers as shown in Figure 8.1.2.

### (3) Clock Monitor [Functional Safety]

To achieve ASIL-D requirement, Clock Monitor is located both on CPG internal and some domains (Slave Access Bus domain, Main Memory domain, RT (Real Time) domain, Peripheral domain, HSC (High Speed Communication) domain). The detail specification is shown in 8.5.2.

### (4) Bus Access Protection [Functional Safety]

Register access protection is applied for registers shown in 8.2. In addition, the function is a safety mechanism to guarantee “Freedom From Interference (FFI)”. The detail specification is described in 8.5.3.

### 8.1.2 Block Diagram

The block diagram is shown in Figure 8.1.2. See 8.1.4 for the frequency and target modules of each clock.

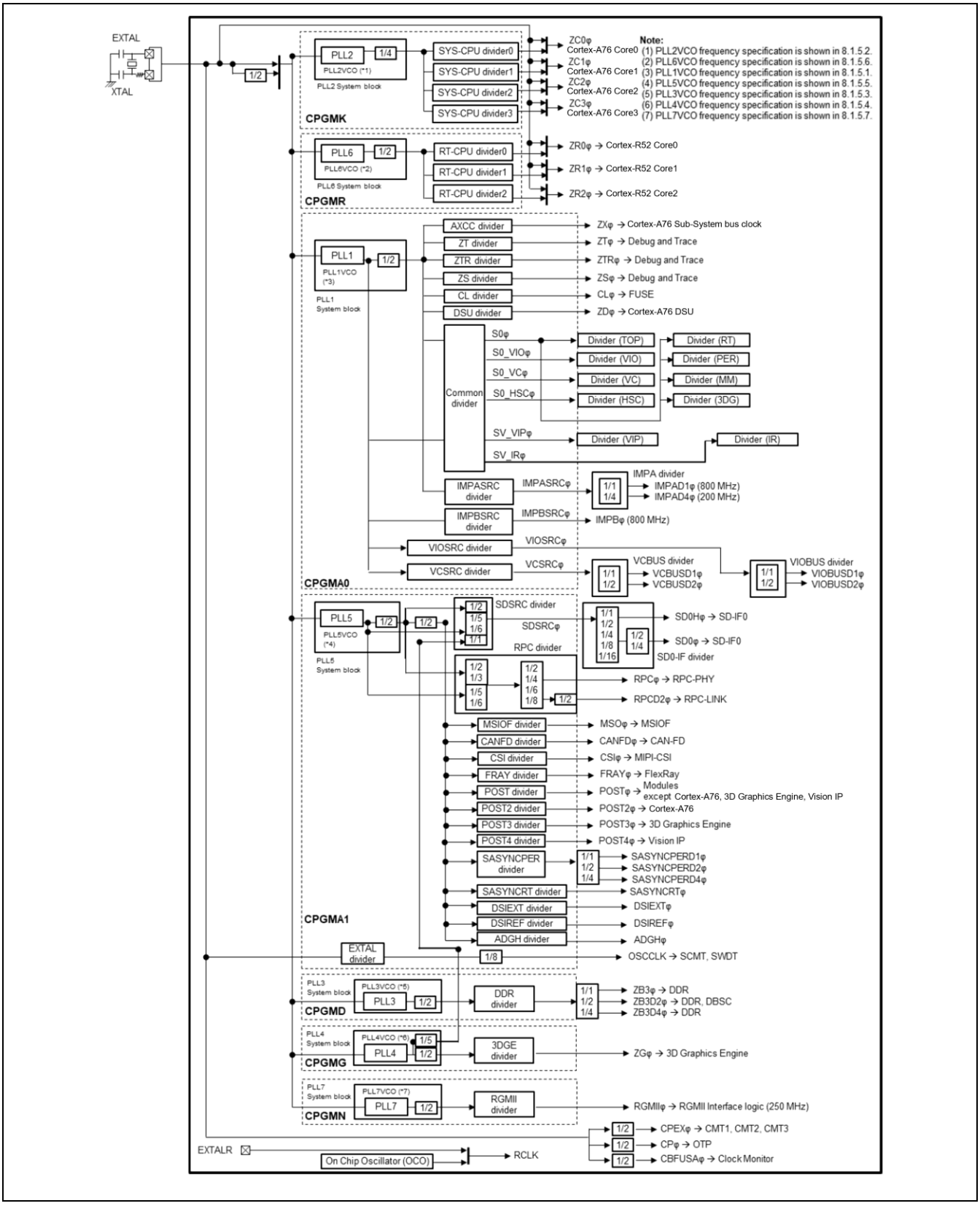



**Figure 8.1.2 Block Diagram of CPG**

**(1) PLL circuit 1 (PLL1)**

- PLL1 multiplies the input clock from EXTAL (EXTAL or EXTAL/2) by the multiplication ratio (See 8.1.5.1).
- SSCG (Center spread, Down spread) and Fractional Multiplication function are supported.

**(2) PLL circuit 2 (PLL2)**

- PLL2 multiplies the input clock from EXTAL (EXTAL or EXTAL/2) by the multiplication ratio (See 8.1.5.2).
- The multiplication ratio is set by the PLL2CR0, PLL2CR1 software register.
- SSCG (Down spread) and Fractional Multiplication function are supported.

**(3) PLL circuit 3 (PLL3)**

- PLL3 multiplies the input clock from EXTAL (EXTAL or EXTAL/2) by the multiplication ratio (See 8.1.5.3).
- The multiplication ratio is set by the PLL3CR0 software register.
- SSCG function is not supported. It is allowed to select a Fractional Multiplication mode.

**(4) PLL circuit 4 (PLL4)**

- PLL4 multiplies the input clock from EXTAL (EXTAL or EXTAL/2) by the multiplication ratio (See 8.1.5.4).
- The multiplication ratio is set by the PLL4CR0, PLL4CR1 software register.
- SSCG (Center spread) and Fractional Multiplication function are supported.

**(5) PLL circuit 5 (PLL5)**

- PLL5 multiplies the input clock from EXTAL (EXTAL or EXTAL/2) by the multiplication ratio (See 8.1.5.5).
- Both of SSCG and Fractional Multiplication function are not supported.

**(6) PLL circuit 6 (PLL6)**

- PLL6 multiplies the input clock from EXTAL (EXTAL or EXTAL/2) by the multiplication ratio (See 8.1.5.6).
- The multiplication ratio is set by the PLL6CR0, PLL6CR1 software register.
- SSCG (Down spread) and Fractional Multiplication function are supported.

**(7) PLL circuit 7 (PLL7)**

- PLL7 multiplies the input clock from EXTAL (EXTAL or EXTAL/2) by the multiplication ratio (See 8.1.5.7).
- Both of SSCG and Fractional Multiplication function are not supported.

**(8) SYS-CPU divider0**

- The SYS-CPU0 divider divides PLL2 output clock.
- This divider generates the core clock (ZC0φ) for Cortex-A76 Core0.
- The division ratio and stop control is set by the FRQCRC0 software register.

**(9) SYS-CPU divider1**

- The SYS-CPU divider divides PLL2 output clock.
- This divider generates the core clock (ZC1φ) for Cortex-A76 Core1.
- The division ratio and stop control is set by the FRQCRC0 software register.

**(10) SYS-CPU divider2**

- The SYS-CPU divider divides PLL2 output clock.
- This divider generates the core clock (ZC2φ) for Cortex-A76 Core2.
- The division ratio and stop control is set by the FRQCRC1 software register.

**(11) SYS-CPU divider3**

- The SYS-CPU divider divides PLL2 output clock.
- This divider generates the core clock (ZC3φ) for Cortex-A76 Core3.
- The division ratio and stop control is set by the FRQCRC1 software register.

**(12) RT-CPU divider 0**

- The RT-CPU divider divides PLL6 output clock.
- This divider generates the Cortex-R52 Core0 clock (ZR0φ).
- The division ratio is set by the FRQCRC0 software register.

**(13) RT-CPU divider 1**

- The RT-CPU divider divides PLL6 output clock.
- This divider generates the Cortex-R52 Core1clock (ZR1φ).
- The division ratio and stop control is set by the FRQCRC0 software register.

**(14) RT-CPU divider 2**

- The RT-CPU divider divides PLL6 output clock.
- This divider generates the Cortex-R52 Core2 clock (ZR2φ).
- The division ratio and stop control is set by the FRQCRC1 software register.

**(15) AXCC divider**

- The AXCC divider divides PLL1 output clock.
- The divider generates AXCC clock (ZXφ).
- The division ratio is fixed as shown in Table 8.1.4a. The stop control is set by CPGMA0CKSTPCR software register.

**(16) DSU divider**

- The DSU divider divides PLL1 output clock.
- The divider generates DSU clock (ZDφ).
- The division ratio is fixed as shown in Table 8.1.4a. The stop control is set by FRQCRB software register.

**(17) ZT / ZTR / ZS divider**

- The ZT / ZTR / ZS divider divides PLL1 output clock.
- The divider generates debug function clocks (ZTφ, ZTRφ, and ZSφ).
- The division ratio for ZTφ, ZTRφ, and ZSφ is set by the FRQCRB software register.

**(18) CL divider**

- The CL divider divides PLL1 output clock. The divider generates FUSE clock (CLφ). The division ratio is fixed as shown in Table 8.1.4a.

**(19) Common divider**

- The Common divider divides PLL1 output clock.
- The divider generates various system clocks (The frequency and the target modules are shown in Table 8.1.4a).

・ Divider (TOP) generates S0D2φ, S0D3φ, and S0D4φ by dividing S0φ.
・ Divider (VIO) generates S0D1_VIOφ, S0D2_VIOφ, S0D4_VIOφ, and S0D8_VIOφ by dividing S0_VIOφ.
・ Divider (VC) generates S0D1_VCφ, S0D2_VCφ, and S0D4_VCφ by dividing S0_VCφ.
・ Divider (HSC) generates S0D1_HSCφ, S0D2_HSCφ, S0D4_HSCφ, and S0D8_HSCφ by dividing S0_HSCφ.
・ Divider (RT) generates S0D2_RTφ, S0D3_RTφ, S0D4_RTφ, S0D6_RTφ and S0D24_RTφ by dividing S0φ.
・ Divider (PER) generates S0D2_PERφ, S0D3_PERφ, S0D4_PERφ, S0D6_PERφ, S0D12_PERφ and S0D24_PERφ by dividing S0φ.
・ Divider (MM) generates S0D2_MMφ and S0D4_MMφ by dividing S0φ.
・ Divider (3DG) generates S0D2_U3DGφ and S0D4_U3DGφ by dividing S0φ.
・ Divider (VIP) generates SVD1_VIPφ and SVD2_VIPφ by dividing SV_VIPφ.
・ Divider (IR) generates SVD1_IRφ and SVD2_IRφ by dividing SV_IRφ.

**(20) IMPASRC / IMPA divider**

- The IMPA divider divides IMPASRCφ and generates the IMPAD1φ clock for IMPAD4φ divider.
- The division ratio is fixed as shown in Table 8.1.4a.

**(21) IMPBSRC divider**

- The IMPBSRC divider divides PLL1 output clock and generates the IMPBφ clock for IMPBφ divider.
- The division ratio is fixed as shown in Table 8.1.4a.

**(22) VIOSRC / VIOBUS divider**

- The VIOBUS divider divides VIOSRCφ and generates the VIOBUSD1φ, VIOBUSD2φ clock.
- The division ratio is fixed as shown in Table 8.1.4a.

**(23) VCBUS divider**

- The VCBUS divider divides VCSRCφ and generates the VCBUSD1φ, VCBUSD2φ clock.
- The division ratio is fixed as shown in Table 8.1.4a.

**(24) SDSRC divider**

- The SDSRC divider divides PLL5 output clock and generates the SDSRCφ clock for SD0-IF divider.
- The division ratio is set by the SD0CKCR and CKSRCSELCR register (See 8.2.24, 8.2.34).

**(25) SD0-IF divider**

- The SD0-IF divider divides SDSRCφ and generates the SD0Hφ / SD0φ clocks for SDHI.
- The division ratio is set by the SD0CKCR and CKSRCSELCR register (See 8.2.24, 8.2.34).

**(26) RPC divider**

- The RPC divider divides PLL5 output clock, and generates the RPCφ, RPCD2φ clocks for RPC-IF.
- The division ratio is set by the RPCCKCR register (See 8.2.25).

**(27) MSIOF divider**

- The MSO divider divides PLL5 output clock and generates the MSOφ clock for MSIOF module.
- The division ratio is set by the MSOCKCR register (See 8.2.27).

**(28) CANFD divider**

- The CANFD divider divides PLL5 output clock and generates the CANFDφ clock for CAN-FD module.
- The division ratio is set by the CANFDCKCR register (See 8.2.26).

**(29) CSI divider**

- The CSI divider divides PLL5 output clock and generates the CSIφ clock for MIPI-CSI2 module.
- The division ratio is set by the CSICKCR register (See 8.2.28).

**(30) FRAY divider**

- The FRAY divider divides PLL5 output clock and generates the FRAYφ clock for FlexRay module.
- The division ratio is fixed as shown in Table 8.1.4e.

**(31) POST divider**

- The POST divider divides PLL5 output clock and generates the POSTφ clock during Power on Self Test (POST) executing for modules except Cortex-A76, 3D Graphics Engine, Vision IP.
- The division ratio is set by the POSTCKCR register (See 8.2.30).

**(32) POST2 divider**

- The POST2 divider divides PLL5 output clock and generates the POST2φ clock during Power on Self Test (POST) executing for Cortex-A76.
- The division ratio is set by the POST2CKCR register (See 8.2.31).

**(33) POST3 divider**

- The POST3 divider divides PLL5 output clock and generates the POST3φ clock during Power on Self Test (POST) executing for 3D Graphics Engine.
- The division ratio is set by the POST3CKCR register (See 8.2.32).

**(34) POST4 divider**

- The POST4 divider divides PLL5 output clock and generates the POST4φ clock during Power on Self Test (POST) executing for Vision IP.
- The division ratio is set by the POST4CKCR register (See 8.2.33).

**(35) SASYNCPER divider**

- The SASYNCPER divider divides PLL5 output clock and generates the SASYNCPERDnφ (n = 1,2,4), clocks for modules which cannot allow the clock (S0Dn_PER (n = 2,3,4,6,12,24)) with SSCG and Fractional Multiplication.
- The division ratio is fixed as shown in Table 8.1.4e.

**(36) SASYNCRT divider**

- The SASYNCRT divider divides PLL5 output clock and generates the SASYNCRTφ, clocks for modules which cannot allow the clock (S0Dn_RT (n = 2,3,4,6,24)) with SSCG and Fractional Multiplication.
- The division ratio is fixed as shown in Table 8.1.4e.

**(37) DSIEXT divider**

- The DSIEXT divider divides PLL5 output clock and generates the DSIEXTφ clock.
- The division ratio is set by the DSIEXTCKCR register (See 8.2.29).

**(38) DSIREF divider**

- The DSIREF divider divides PLL5 output clock and generates the DSIREFφ clock.
- The division ratio is fixed as shown in Table 8.1.4e.

**(39) ADGH divider**

- The ADGH divider divides PLL5 output clock and generates the ADGHφ clock.
- The division ratio is fixed as shown in Table 8.1.4e.

**(40) EXTAL divider**

- The EXTAL divider divides EXTAL (EXTAL or EXTAL/2) and generates the OSCCLK. The divider specification is shown in Table 8.1.4e (Note 5).

**(41) DDR divider**

- The DDR clock divider divides PLL3 output clock and generates DDR and DBSC clock (ZB3φ, ZB3D2φ, ZB3D4φ).
- The division ratio for ZB3φ, ZB3D2φ, ZB3D4φ is set by the FRQCRD register (See 8.2.11).

**(42) 3DGE divider**

- The 3DGE clock divider divides PLL4 output clock and generates ZGφ.
- The division ratio and the stop control are set by the FRQCRB register (See 8.2.8).

**(43) On Chip Oscillator**

- By controlling MD29, the clock generated from On Chip Oscillator (OCO) is output as RCLK. See the Table 8.1.5.8 about MD29 setting.

### 8.1.3 External Pins

Table 8.1.3 lists the CPG pin configuration.

**Table 8.1.3 Pin Configuration and Functions of CPG**

| Name | I/O | Function |
|---|---|---|
| EXTALR | Input | Clock input pin.<br>Used as an external clock input pin and as clock source of RCLK. |
| XTAL | Input | Clock input pin.<br>Outputs amplified negative feedback of EXTAL. |
| EXTAL | Input | Clock input pin.<br>Used as an external clock input pin. |
| MD0 | Input | Mode 0. Step-up mode |
| MD1 | Input | Mode 1. Boot Device select (*1) |
| MD2 | Input | Mode 2. Boot Device select (*1) |
| MD3 | Input | Mode 3. Boot Device select (*1) |
| MD4 | Input | Mode 4. Boot Device select (*1) |
| MD10 | Input | Mode 10. Debug Mode select (*2) |
| MD11 | Input | Mode 11. Debug Mode select (*2) |
| MD13 | Input | Mode 13. PLL multiplication ratio setting |
| MD14 | Input | Mode 14. PLL multiplication ratio setting |
| MD17 | Input | Mode 17. DDR clock frequency setting |
| MD18 | Input | Mode 18. SSCG enable / disable for PLL1 |
| MD20 | Input | Mode 20. Debug Mode select (*2) |
| MD21 | Input | Mode 21. Debug Mode select (*2) |
| MD23 | Input | Mode 23. POST target select (*3) |
| MD25 | Input | Mode 25. Field BIST enable / disable control |
| MD29 | Input | Mode 29. RCLK reference clock select |
| MD36 | Input | Mode 36. SSCG enable / disable for PLL1 |
| MD37 | Input | Mode 37. SSCG enable / disable for PLL1 |
| MDT0 | Input | MDT0. Debug Mode select (*2) |

Notes:
1) It is used by not only CPG also Boot function. See Table 31.1 Mode Pin Configuration (1).
2) It is used by not only CPG also Debug and Trace. See 170 Debug and Trace (Table JTAG selection table).
3) See Table 171.2.

### 8.1.4 List of Clock Outputs

**Table 8.1.4a Lists of CPG clocks generated from PLL1**

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum (*3) | Initial | |
| ZXφ | AXCC divider | Fixed | 800.00 MHz | 800.00 MHz | Cortex-A76 Sub-System |
| ZDφ | DSU divider | Fixed | 800.00 MHz | 800.00 MHz | Cortex-A76 Sub-System |
| ZSφ | ZS divider | Variable | 266.66 MHz | 266.66 MHz (*1) | Debug and Trace |
| ZTφ | ZTR divider | Variable | 533.33 MHz | 533.33 MHz (*1) | Debug and Trace |
| ZTRφ | ZS divider | Variable | 533.33 MHz | 533.33 MHz (*1) | Debug and Trace |
| S0φ | Common divider | Fixed | 800.00 MHz | 800.00 MHz | Source of any dividers for system clock shown in Figure 8.1.1. |
| S0D2φ | Common divider | Fixed | 400.00 MHz | 400.00 MHz | Slave Access Bus (*2). |
| S0D3φ | Common divider | Fixed | 266.66 MHz | 266.66 MHz | Slave Access Bus (*2). |
| S0D4φ | Common divider | Fixed | 200.00 MHz | 200.00 MHz | Slave Access Bus (*2). |
| S0_VIOφ | Common divider | Fixed | 800.00 MHz | 800.00 MHz | Source of any dividers for system clock shown in Figure 8.1.1 |
| S0D1_VIOφ | Common divider | Fixed | 800.00 MHz | 800.00 MHz | VIO (Video IO) domain bus (*2), VSPX, FCPVX, ISP |
| S0D2_VIOφ | Common divider | Fixed | 400.00 MHz | 400.00 MHz | VIO (Video IO) domain bus (*2), IPMMU, FCPVX, DSI, CSI-2, ISP |
| S0D4_VIOφ | Common divider | Fixed | 200.00 MHz | 200.00 MHz | VIO (Video IO) domain bus (*2), IPMMU, FCPVX, VIN, DSI, VSPX |
| S0D8_VIOφ | Common divider | Fixed | 100.00 MHz | 100.00 MHz | VIO (Video IO) domain bus (*2), DSI, CSI-2 |
| VIOSRCφ | VIOBUS divider | Fixed | 533.33 MHz | 533.33 MHz | Source of VIOBUSD1 and VIOBUSD2 shown in Figure 8.1.1 |
| VIOBUSD1φ | VIOBUS divider | Fixed | 533.33 MHz | 533.33 MHz | DU, DOC, VSPD, FCPVD, MIPI-DSI |
| VIOBUSD2φ | VIOBUS divider | Fixed | 266.66 MHz | 266.66 MHz | VIO (Video IO) domain bus (*2), DU, DOC, VSPD, FCPVD, MIPI-DSI |
| S0_VCφ | Common divider | Fixed | 800.00 MHz | 800.00 MHz | Source of any dividers for system clock shown in Figure 8.1.1 |
| S0D1_VCφ | Common divider | Fixed | 800.00 MHz | 800.00 MHz | |
| S0D2_VCφ | Common divider | Fixed | 400.00 MHz | 400.00 MHz | VC (Video Codec) domain bus (*2), IPMMU |
| S0D4_VCφ | Common divider | Fixed | 200.00 MHz | 200.00 MHz | VC (Video Codec) domain bus (*2), IPMMU |
| VCSRCφ | VCBUS divider | Fixed | 533.33 MHz | 533.33 MHz | Source of VCBUSD1 and VCBUSD2 shown in Figure 8.1.1 |
| VCBUSD1φ | VCBUS divider | Fixed | 533.33 MHz | 533.33 MHz | VC (Video Codec) domain bus (*2), iVCP1E, IMR-LX4, IPMMU |
| VCBUSD2φ | VCBUS divider | Fixed | 266.66 MHz | 266.66 MHz | VC (Video Codec) domain bus (*2), iVCP1E, IMR-LX4, IPMMU |
| S0D2_MMφ | Common divider | Fixed | 400.00 MHz | 400.00 MHz | Main Memory domain bus (*2), IPMMU, DBSC |
| S0D4_MMφ | Common divider | Fixed | 200.00 MHz | 200.00 MHz | Main Memory domain bus (*2), IPMMU |
| S0D2_U3DGφ | Common divider | Fixed | 400.00 MHz | 400.00 MHz | GSX (Graphics) domain bus (*2), IPMMU, 3D Graphics Engine |
| S0D4_U3DGφ | Common divider | Fixed | 200.00 MHz | 200.00 MHz | GSX (Graphics) domain bus (*2), IPMMU |
| S0D2_RTφ | Common divider | Fixed | 400.00 MHz | 400.00 MHz | RT (Real Time) domain bus (*2), Cortex-R52, CoreSight, RTDMAC, IPMMU, RT-SRAM, RT-VRAM, Life Cycle, CRC, ICUMX, WCRC3, WRC2, WRC1, WRC0 |
| S0D3_RTφ | Common divider | Fixed | 266.66 MHz | 266.66 MHz | RT (Real Time) domain bus (*2), INTC |
| S0D4_RTφ | Common divider | Fixed | 200 MHz | 200 MHz | RT (Real Time) domain bus (*2), Cortex-R52, CoreSight, RTDMAC, IPMMU, RT-SRAM, RT-VRAM, Life Cycle, CRC, ICUMX, WCRC3, WRC2, WRC1, WRC0, ECM |
| S0D6_RTφ | Common divider | Fixed | 133.33 MHz | 133.33 MHz | RT (Real Time) domain bus (*2), INTC, TMU |
| S0D24_RTφ | Common divider | Fixed | 33.33 MHz | 33.33 MHz | There are not any modules which use the clock. |
| S0D2_PERφ | Common divider | Fixed | 400.00 MHz | 400.00 MHz | Peripheral domain bus (*2) |
| S0D3_PERφ | Common divider | Fixed | 266.66 MHz | 266.66 MHz | Peripheral domain bus (*2), RPC-IF, SYS-DMAC, IPMMU, SDHI |

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum (*3) | Initial | |
| S0D4_PERφ | Common divider | Fixed | 200.00 MHz | 200.00 MHz | ADG, SYS-DMAC |
| S0D6_PERφ | Common divider | Fixed | 133.33 MHz | 133.33 MHz | Peripheral domain bus (*2), SYS-DMAC, IPMMU, I2C, ADG, FlexRay |
| S0D12_PERφ | Common divider | Fixed | 66.66 MHz | 66.66 MHz | Peripheral domain bus (*2), RPC-IF |
| S0D24_PERφ | Common divider | Fixed | 33.33 MHz | 33.33 MHz | RPC-IF |
| S0_HSCφ | Common divider | Fixed | 800.00 MHz | 800.00 MHz | Source of any dividers for system clock shown in Figure 8.1.1 |
| S0D1_HSCφ | Common divider | Fixed | 800 MHz | 800 MHz | Ethernet AVB, Ethernet TSN |
| S0D2_HSCφ | Common divider | Fixed | 400 MHz | 400 MHz | HC (High speed Communication) domain bus (*2), PCI Express, IPMMU |
| S0D4_HSCφ | Common divider | Fixed | 200 MHz | 200 MHz | HC (High speed Communication) domain bus (*2), Ethernet AVB, Ethernet TSN, PCI Express, IPMMU |
| S0D8_HSCφ | Common divider | Fixed | 100 MHz | 100 MHz | Ethernet AVB, Ethernet TSN |
| SV_IRφ | Common divider | Fixed | 640 MHz | 640 MHz | Source of any dividers for system clock shown in Figure 8.1.1 |
| SVD1_IRφ | Common divider | Fixed | 640 MHz | 640 MHz | IMP domain bus (*2), IPMMU, Image Processing (IMP) Unit |
| SVD2_IRφ | Common divider | Fixed | 320 MHz | 320 MHz | IMP domain bus (*2), IPMMU, Image Processing (IMP) Unit |
| IMPASRCφ | Common divider | Fixed | 800 MHz | 800 MHz | Source of any dividers for system clock shown in Figure 8.1.1 |
| IMPAD1φ | Common divider | Fixed | 800 MHz | 800 MHz | Image Processing (IMP) Unit. |
| IMPAD4φ | Common divider | Fixed | 200 MHz | 200 MHz | Image Processing (IMP) Unit. |
| IMPBSRCφ | Common divider | Fixed | 800 MHz | 800 MHz | Source of any dividers for system clock shown in Figure 8.1.1 |
| IMPBφ | Common divider | Fixed | 800 MHz | 800 MHz | Image Processing (IMP) Unit. |
| SV_VIPφ | Common divider | Fixed | 640 MHz | 640 MHz | Source of any dividers for system clock shown in Figure 8.1.1 |
| SVD1_VIPφ | Common divider | Fixed | 640 MHz | 640 MHz | VIP (Vision IP Processor) domain bus (*2), IPMMU, Vision IP |
| SVD2_VIPφ | Common divider | Fixed | 320 MHz | 320 MHz | VIP (Vision IP Processor) domain bus (*2), IPMMU, Vision IP |
| CLφ | CL divider | Fixed | 33.33 MHz | 33.33 MHz | FUSE. |
| CL16Mφ | Common divider | Fixed | 16.66 MHz | 16.66 MHz | CMT0, CMT1, CMT2, CMT3, SCMT, INTC-EX, APMU, Slave access bus (*2), PFC, SYSC, AVS, THS, CPG |
| CL16M_MMφ | Common divider | Fixed | 16.66 MHz | 16.66 MHz | DDR, DBSC |
| CL16M_RTφ | Common divider | Fixed | 16.66 MHz | 16.66 MHz | Life Cycle, SWDT, TMU, WDT, RCLK Watchdog Timer, Cortex-R52, WWDT |
| CL16M_PERφ | Common divider | Fixed | 16.66 MHz | 16.66 MHz | HSCIF, SCIF, SYS-DMAC, FlexRay |
| CL16M_HSCφ | Common divider | Fixed | 16.66 MHz | 16.66 MHz | Ether TSN |

Note: 1. These clocks are oscillated in the debug mode. Otherwise, they are stopped.

Note: 2. The bus structure is shown in Figure 18-1 Block diagram of AXI-Bus.

Note: 3. It shows a maximum frequency during PLL1 integer multiplication mode. The maximum frequency is increased by 1.0 % during PLL1 SSCG (Center spread) mode.

**Table 8.1.4b Lists of CPG clocks generated from PLL2**

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum | Initial | |
| ZC0φ | SYS-CPU divider0 | Variable | V4M-7: 1000 MHz<br>V4M-5: 800 MHz<br>V4M-3: 933 MHz<br>V4M-2: 933 MHz | 1000 MHz | Cortex-A76 Core0 |
| ZC1φ | SYS-CPU divider1 | Variable | V4M-7: 1000 MHz<br>V4M-5: 800 MHz<br>V4M-3: 933 MHz<br>V4M-2: 933 MHz | 1000 MHz | Cortex-A76 Core1 |
| ZC2φ | SYS-CPU divider2 | Variable | V4M-7: 1000 MHz<br>V4M-5: 800 MHz<br>V4M-3: 933 MHz<br>V4M-2: 933 MHz | 1000 MHz | Cortex-A76 Core2 |
| ZC3φ | SYS-CPU divider3 | Variable | V4M-7: 1000 MHz<br>V4M-5: 800 MHz<br>V4M-3: Not support<br>V4M-2: Not support | 1000 MHz | Cortex-A76 Core3 |

Note) The "multiplication ratio" must be set so as not to exceed the maximum frequency value of each product. Operation cannot be guaranteed if the maximum frequency is exceeded.

**Table 8.1.4c Lists of CPG clocks generated from PLL3**

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum * | Initial | |
| ZB3φ | DDR divider | Variable | V4M-7:<br>LP5: 800MHz<br>LP4: 1066.66MHz<br>V4M-5:<br>LP5: 625MHz<br>LP4: 833.33MHz<br>V4M-3:<br>LP5: 600MHz<br>LP4: 800MHz<br>V4M-2:<br>LP5: 600MHz<br>LP4: 800MHz | 1066.66 MHz (*4) | DDR |
| ZB3D2φ | DDR divider | Variable | V4M-7:<br>LP5: 800MHz<br>LP4: 533.33MHz<br>V4M-5:<br>LP5: 625MHz<br>LP4: 416.66MHz<br>V4M-3:<br>LP5: 600MHz<br>LP4: 400MHz<br>V4M-2:<br>LP5: 600MHz<br>LP4: 400MHz | 533.33 MHz (*4) | DDR, DBSC |
| ZB3D4φ | DDR divider | Variable | V4M-7:<br>LP5: 400MHz<br>LP4: 266.66MHz<br>V4M-5:<br>LP5: 312.5MHz<br>LP4: 208.33MHz<br>V4M-3:<br>LP5: 300MHz<br>LP4: 200MHz<br>V4M-2:<br>LP5: 300MHz<br>LP4: 200MHz | 266.66 MHz (*4) | DDR |

Note The "multiplication ratio" must be set so as not to exceed the maximum frequency value of each product. Operation cannot be guaranteed if the maximum frequency is exceeded.

4. The initial frequency depends on the setting of MD17 and MD19. See 8.1.5.3.

**Table 8.1.4d Lists of CPG clocks generated from PLL4**

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum * | Initial | |
| ZGφ | 3DGE divider | Variable | V4M-7:<br>1000 MHz<br>V4M-5:<br>750 MHz<br>V4M-3:<br>500 MHz<br>V4M-2:<br>Not support | 600 MHz | 3D Graphics Engine |

Note The “multiplication ratio” must be set so as not to exceed the maximum frequency value of each product. Operation cannot be guaranteed if the maximum frequency is exceeded.

**Table 8.1.4e Lists of CPG clocks generated from PLL5**

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum | Initial | |
| SDSRCφ | SDSRC divider | Variable | 800 MHz | 800 MHz | SD0-IF divider |
| SD0Hφ | SD0-IF divider | Variable | 800 MHz | Stopped | SDHI |
| SD0φ | SD0-IF divider | Variable | 200 MHz | 50 MHz | SDHI |
| RPCφ | RPC divider | Variable | 320 MHz | (*5) | RPC-IF |
| RPCD2φ | RPC divider | Variable | 160 MHz | (*5) | RPC-IF |
| MSOφ | MSO divider | Variable | 133.33 MHz | 12.5 MHz | MSIOF |
| CANFDφ | CANFD divider | Variable | 80 MHz | 12.5 MHz | CAN-FD |
| CSIφ | CSI divider | Variable | 25 MHz | Stopped | MIPI-CSI2 |
| FRAY | FRAY divider | Fixed | 80 MHz | 80 MHz | FlexRay |
| POST2φ | POST2 divider | Variable | 133.33 MHz | 133.33 MHz | Clock during Power on Self Test (POST) for Cortex-A76 |
| POST3φ | POST3 divider | Variable | 66.66 MHz | 66.66 MHz | Clock during Power on Self Test (POST) for 3D Graphics Engine |
| POST4φ | POST4 divider | Variable | 66.66 MHz | 66.66 MHz | Clock during Power on Self Test (POST) for Vision IP |
| POSTφ | POST divider | Variable | 66.66 MHz | 66.66 MHz | Clock during Power on Self Test (POST) for modules except Cortex-A76, 3D Graphics Engine, Vision IP |
| SASYNCRTφ | SASYNCRT divider | Fixed | 16.66 MHz | 16.66 MHz | RT (Real Time) domain bus, Life Cycle, SWDT, TMU, WDT, WWDT, RWDT |
| SASYNCPERD1φ | SASYNCPER divider | Fixed | 266.66 MHz | 266.66 MHz | HSCIF, SCIF, CAN-FD |
| SASYNCPERD2φ | SASYNCPER divider | Fixed | 133.33 MHz | 133.33 MHz | Peripheral domain bus, TMU, HSCIF, SYS-DMAC, CAN-FD |
| SASYNCPERD4φ | SASYNCPER divider | Fixed | 66.66 MHz | 66.66 MHz | Peripheral (PeriE/PeriW) domain bus, SCIF, SYS-DMAC, TPU, PWM |
| DSIEXTφ | DSIEXT divider | Variable | 800 MHz | 800 MHz | DSI |
| DSIREFφ | DSIREF divider | Fixed | 16.66 MHz | 16.66 MHz | DSI |
| ADGHφ | ADGH divider | Fixed | 800 MHz | 800 MHz | There are not any modules which use the clock. |
| OSCCLK | EXTAL divider | Fixed | 131.57 kHz | (*6) | SCMT, SWDT |

Note 5. The frequency of RPCφ depends on the value of MD4, MD3, MD2 and MD1.
When MD4 = H or L, MD3 = L, MD2 = H, MD1 = L, RPCφ = 320 MHz, RPCD2φ = 160MHz.
When MD4 = H or L, MD3 = L, MD2 = H, MD1 = H, RPCφ = 160 MHz, RPCD2φ = 80MHz.
When MD4 = L, MD3 = H, MD2 = L, MD1 = L, RPCφ = 80 MHz, RPCD2φ = 40MHz.
In the other cases of MD4, MD3, MD2 and MD1 setting, RPCφ = 80 MHz, RPCD2φ = 40MHz.

Note 6. The initial value depends on MD14, MD13.

| MD14 | MD13 | EXTAL (MHz) | EXTAL divider | | OSCCLK | |
|---|---|---|---|---|---|---|
| | | | Fixed division ratio | Frequency (MHz) | Fixed division ratio | Frequency (kHz) |
| 0 | 0 | 16.66 | × 1/16 | 1.04 | × 1/8 | 130.20 |
| 0 | 1 | 20.00 | × 1/19 | 1.05 | × 1/8 | 131.57 |
| 1 | 0 | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited |
| 1 | 1 | 33.33 | × 1/32 | 1.04 | × 1/8 | 130.20 |

**Table 8.1.4f Lists of CPG clocks generated from PLL6**

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum * | Initial | |
| ZR0φ | RT-CPU divider0 | Variable | V4M-7: 1400 MHz<br>V4M-5: 1100 MHz<br>V4M-3: 900 MHz<br>V4M-2: 900 MHz | 1400 MHz | Cortex-R52 Sub-System 0 |
| ZR1φ | RT-CPU divider0 | Variable | V4M-7: 1400 MHz<br>V4M-5: 1100 MHz<br>V4M-3: 900 MHz<br>V4M-2: 900 MHz | 1400 MHz | Cortex-R52 Sub-System 1 |
| ZR2φ | RT-CPU divider0 | Variable | V4M-7: 1400 MHz<br>V4M-5: 1100 MHz<br>V4M-3: 900 MHz<br>V4M-2: Not support | 1400 MHz | Cortex-R52 Sub-System 2 |

Note The "multiplication ratio" must be set so as not to exceed the maximum frequency value of each product. Operation cannot be guaranteed if the maximum frequency is exceeded.

**Table 8.1.4g Lists of CPG clocks generated from PLL7**

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum | Initial | |
| RGMIIφ | RGMII divider | Variable | 250 MHz | 250 MHz | RGMII interface logic |

**Table 8.1.4h Lists of CPG other clocks**

| Name | Clock Source | Frequency | | | Target modules |
|---|---|---|---|---|---|
| | | Dividing ratio | Maximum | Initial | |
| CPEXφ | EXTAL | Fixed | 16.66 MHz | (*7) | CMT1, CMT2, CMT3 |
| CPφ | EXTAL | Fixed | 16.66 MHz | (*7) | OTP |
| CBFUSAφ | EXTAL | Fixed | 16.66 MHz | (*7) | Clock Monitor |
| RCLK | EXTALR / OCO (*8) | Fixed | 32.8 KHz | 32.8 KHz | CMT0, CMT1, CMT2, CMT3, INTC-EX, Life Cycle, WDT, RCLK Watchdog Timer, ICUMX, WWDT, Clock Monitor, DOC, Image Processing (IMP) Unit |

Note 7. CPEXφ = EXTAL/2, CBFUSAφ = EXTAL/2. EXTAL value depends on MD14, MD13 as shown in Table 8.1.4e (Note 6).
Note 8. The initial value depends on MD29.
When MD29 = L, RCLK is EXTALR.
When MD29 = H, RCLK is OCO

### 8.1.5 Clock Operating Modes

#### 8.1.5.1 PLL1 Multiplication Ratio

- PLL1 initial multiplication ratio and initial state (SSCG mode or fractional multiplication mode) can be selected through MD pins as shown in Table 8.1.5.1.
- The PLL1 parameter cannot be changed by software registers (PLL1CR0 and PLL1CR1).

**Table 8.1.5.1 PLL1 Initial Parameter Setting**

| MD pins | | PLL1 reference clock (*1) [MHz] | PLL1 parameter settings (*2) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MD18:37:36 | MD14:13 (*4) | | SSMODE [2:0] | SSFREQ [6:0] | SSDEPT [6:0] | Mode (*3) | NI[7:0] | NF[24:0] | PLL1VCO [MHz] | PLL1VCO (min) [MHz] | PLL1VCO (max) [MHz] |
| B'000 | B'00 | 16.66 | B'000 | (*5) | (*5) | FRAC=OFF SSCG=OFF | D'95 | H'0000000 | 3200.000 | (*5) | (*5) |
| | B'01 | 20.00 | | (*5) | | | D'79 | H'0000000 | 3200.000 | (*5) | (*5) |
| | B'11 | 16.66 | | (*5) | | | D'95 | H'0000000 | 3200.000 | (*5) | (*5) |
| B'001 | B'00 | 16.66 | B'100 | (*5) | (*5) | FRAC=ON SSCG=OFF | D'94 | H'073BA34 | 3174.201 | (*5) | (*5) |
| | B'01 | 20.00 | | (*5) | | | D'78 | H'0B5C5D6 | 3174.201 | (*5) | (*5) |
| | B'11 | 16.66 | | (*5) | | | D'94 | H'073BA34 | 3174.201 | (*5) | (*5) |
| B'010 | B'00 | 16.66 | B'110 | B'0100000 | B'0010100 | FRAC=ON SSCG=ON (Down spread, -1.0%) | D'96 | H'1FDF8F4 | 3266.535 | 3233.869 | 3266.535 |
| | B'01 | 20.00 | | B'0101000 | | | D'80 | H'153A4C0 | 3266.535 | 3233.869 | 3266.535 |
| | B'11 | 16.66 | | B'0100000 | | | D'96 | H'1FDF8F4 | 3266.535 | 3233.869 | 3266.535 |
| B'011 | B'00 | 16.66 | B'110 | B'0100000 | B'0010100 | FRAC=ON SSCG=ON (Down spread, -1.0%) | D'95 | H'0000000 | 3200.000 | 3168.000 | 3200.000 |
| | B'01 | 20.00 | | B'0101000 | | | D'79 | H'0000000 | 3200.000 | 3168.000 | 3200.000 |
| | B'11 | 16.66 | | B'0100000 | | | D'95 | H'0000000 | 3200.000 | 3168.000 | 3200.000 |
| B'100 | B'00 | 16.66 | B'111 | B'0100000 | B'0010100 | FRAC=ON SSCG=ON (Center spread, ±0.5%) | D'96 | H'1FDF8F4 | 3266.535 | 3250.202 | 3282.867 |
| | B'01 | 20.00 | | B'0101000 | | | D'80 | H'153A4C0 | 3266.535 | 3250.202 | 3282.867 |
| | B'11 | 16.66 | | B'0100000 | | | D'96 | H'1FDF8F4 | 3266.535 | 3250.202 | 3282.867 |
| B'101 | B'00 | 16.66 | B'111 | B'0100000 | B'0101000 | FRAC=ON SSCG=ON (Center spread, ±1.0%) | D'96 | H'1FDF8F4 | 3266.535 | 3233.869 | 3299.200 |
| | B'01 | 20.00 | | B'0101000 | | | D'80 | H'153A4C0 | 3266.535 | 3233.869 | 3299.200 |
| | B'11 | 16,66 | | B'0100000 | | | D'96 | H'1FDF8F4 | 3266.535 | 3233.869 | 3299.200 |
| B'110 | B'00 | 16.66 | B'111 | B'0010011 | B'0010100 | FRAC=ON SSCG=ON (Center spread, ±0.5%) | D'96 | H'1FDF8F4 | 3266.535 | 3250.202 | 3282.867 |
| | B'01 | 20.00 | | B'0011001 | | | D'80 | H'153A4C0 | 3266.535 | 3250.202 | 3282.867 |
| | B'11 | 16.66 | | B'0010011 | | | D'96 | H'1FDF8F4 | 3266.535 | 3250.202 | 3282.867 |
| B'111 | B'00 | 16.66 | B'111 | B'0010011 | B'0101000 | FRAC=ON SSCG=ON (Center spread, ±1.0%) | D'96 | H'1FDF8F4 | 3266.535 | 3233.869 | 3299.200 |
| | B'01 | 20.00 | | B'0011001 | | | D'80 | H'153A4C0 | 3266.535 | 3233.869 | 3299.200 |
| | B'11 | 16.66 | | B'0010011 | | | D'96 | H'1FDF8F4 | 3266.535 | 3233.869 | 3299.200 |

Note 1. When "MD14=1 and MD13=1", the reference clock is EXTAL x 1/2 (= 33.33 x 1/2). When the other mode setting, it is EXTAL x 1/1.
Note 2 Detail calculation and parameter description are shown in 8.2 Register Description (PLL1CR0, PLL1CR1).
Note 3 FRAC = Fractional Multiplication mode, SSCG = SSCG mode. Ex) "FRAC=ON" means "Fractional Multiplication mode is enabled".
Note 4 "[MD14:MD13] = B'10" setting is prohibited because the system operation cannot be guaranteed.
Note 5 In case of the integer and fractional multiplication mode (SSMODE = 3'b000 and 3'b100), these values are not valid.

### 8.1.5.2 PLL2 Multiplication Ratio

- PLL2 initial multiplication ratio and initial state (integer mode) can are shown in Table 8.1.5.2. PLL2 is initially operated with the integer mode.

- The PLL2 parameter can be changed by software registers. The parameter setting must be executed under PLL standby mode (See 8.3.1). The PLL2 parameter setting of SSCG (Center spread) is prohibited as shown in Table 8.1.1.

- When the PLL2VCO is changed by the software registers, PLL2VCO frequency range must be set as the below.

  1. Integer multiplication mode → 2000.000 MHz ≦ PLL2VCO ≦ 4000.000 MHz
  2. Fractional multiplication mode → 2000.000 MHz ≦ PLL2VCO ≦ (Not fixed)
  3. SSCG mode (Down spread) → 2000.000 MHz ≦ PLL2VCO ≦ (Not fixed)

**Table 8.1.5.2 PLL2 Initial Parameter Setting (Initial parameters)**

| MD14:13 (*3) | PLL2 reference clock (*1) [MHz] | PLL2 parameter setting (*2) | | | | | |
|---|---|---|---|---|---|---|---|
| | | SSMODE [2:0] | SSFREQ [6:0] | SSDEPT [6:0] | PLL2VCO [MHz] | NI[7:0] | NF[24:0] |
| B'00 | 16.66 | B'000 | (*4) | (*4) | 4000.000 | D'119 | H'0000000 |
| B'01 | 20.00 | B'000 | (*4) | (*4) | 4000.000 | D'99 | H'0000000 |
| B'11 | 16.66 | B'000 | (*4) | (*4) | 4000.000 | D'119 | H'0000000 |

Note 1. When "MD14=1 and MD13=1", the reference clock is EXTAL x 1/2 (= 33.33 x 1/2). When the other mode setting, it is EXTAL x 1/1.
Note 2 Detail calculation and parameter description are shown in 8.2 Register Description (PLL2CR0, PLL2CR1).
Note 3 "[MD14:MD13] = B'10" setting is prohibited because the system operation cannot be guaranteed.
Note 4 In case of the integer multiplication mode (SSMODE = 3'b000), SSFREQ and SSDEPT value are not enable.

### 8.1.5.3 PLL3 Multiplication Ratio

- The integer multiplication and fractional multiplication mode are supported for PLL3. The SSCG mode setting is prohibited because DDR operation cannot be guaranteed.

- Table 8.1.5.3a shows an initial parameter setting of PLL3. The initial parameter is same in all cases of Mode Pins. After the chip is booted, the parameter can be changed by PLL3CR0 and FRQCRD software registers so that ZB3φ clock frequency corresponds with Table 8.1.5.3c. PLL3VCO frequency range is 2000.0 MHz ~ 4266.7 MHz.

**Table 8.1.5.3a PLL3 Parameter Setting (Initial parameters)**

| MD27:22:19:17 | MD14:13 (*3) | PLL3 reference clock (*1) [MHz] | PLL3 parameter setting (*2) | | | |
|---|---|---|---|---|---|---|
| | | | SSMODE [2:0] | PLL3VCO [MHz] | NI[7:0] | NF[24:0] |
| - (*4) | B'00 | 16.66 | B'000 | 3200.000 | D'95 | H'0000000 |
| | B'01 | 20.00 | B'000 | 3200.000 | D'79 | H'0000000 |
| | B'11 | 16.66 | B'000 | 3200.000 | D'95 | H'0000000 |

Note 1. When "MD14=1 and MD13=1", the reference clock is EXTAL x 1/2 (= 33.33 x 1/2). When the other mode setting, it is EXTAL x 1/1.
Note 2 Detail calculation and parameter description are shown in 8.2 Register Description (PLL3CR0, PLL3CR1).
Note 3 "[MD14:MD13] = B'10" setting is prohibited because the system operation cannot be guaranteed.
Note 4 Initial parameter is same in all cases of Mode Pins (MD27, MD22, MD19, MD17).

**Table 8.1.5.3b DDR Mode**

| MD27:22 | PLL3VCO [MHz] |
|---|---|
| B'00 | LPDDR5 |
| B'01 | LPDDR4X |
| B'10 | Setting prohibited |
| B'11 | Setting prohibited |

**Table 8.1.5.3c DDR Frequency**

| MD19:17 | LPDDR5 | LPDDR4X |
|---|---|---|
| B'00 | 6400 Mbps (ZB3φ = 800.00 MHz) (*1) | 4266 Mbps (ZB3φ = 1066.66 MHz) (*2) |
| B'01 | 6000 Mbps (ZB3φ = 750.00 MHz) (*1) | 3733 Mbps (ZB3φ = 933.33 MHz) (*2) |
| B'10 | 5500 Mbps (ZB3φ = 687.50 MHz) (*1) | 3200 Mbps (ZB3φ = 800.00 MHz) (*2) |
| B'11 | 4800 Mbps (ZB3φ = 600.00 MHz) (*1) | 2133 Mbps (ZB3φ = 533.33 MHz) (*2) |

Note 1 When selecting LPDDR5, clock frequency is increased by 8 times when the clock is provided for DDR.
Note 2 When selecting LPDDR4X, clock frequency is increased by 4 times when the clock is provided for DDR.

**Notification**
When PLL1 is under SSCG (Down spread) mode, PLL3VCO frequency should be change so that "ZB3D2_MMφ ≦ 2 × S0D2_MMφ" relationship can be kept. According to DBSC specification, the system operation cannot be guaranteed if the relationship is not kept.

### 8.1.5.4 PLL4 Multiplication Ratio

- PLL4 initial multiplication ratio and multiplication mode are shown in Table 8.1.5.4. The integer multiplication mode is set as the initial parameter setting.

- The PLL4 parameter can be changed by software registers. After the chip is booted, PLL4 parameter can be changed from the integer multiplication mode to fractional multiplication mode or SSCG mode (Down spread) by PLL4CR0 and PLL4CR1.

- When the PLL4VCO is changed by the software registers, PLL4VCO frequency range must be set as the below.

  1. Integer multiplication mode → 2000.000 MHz ≦ PLL4VCO ≦ 4000.000 MHz
  2. Fractional multiplication mode → 2000.000 MHz ≦ PLL4VCO ≦ (Not fixed)
  3. SSCG mode (Down spread) → 2000.000 MHz ≦ PLL4VCO ≦ (Not fixed)

**Table 8.1.5.4 PLL4 Initial Parameter Setting (Initial parameters)**

| MD14:13 (*3) | PLL4 reference clock (*1) [MHz] | PLL4 parameter setting (*2) | | | | | |
|---|---|---|---|---|---|---|---|
| | | SSMODE [2:0] | SSFREQ [6:0] | SSDEPT [6:0] | PLL4VCO [MHz] | NI[7:0] | NF[24:0] |
| B'00 | 16.66 | B'000 | (*4) | (*4) | 4000.000 | D'119 | H'0000000 |
| B'01 | 20.00 | B'000 | (*4) | (*4) | 4000.000 | D'99 | H'0000000 |
| B'11 | 16.66 | B'000 | (*4) | (*4) | 4000.000 | D'119 | H'0000000 |

Note 1 When "MD14=1 and MD13=1", the reference clock is EXTAL x 1/2 (= 33.33 x 1/2). When the other mode setting, it is EXTAL x 1/1.
Note 2 Detail calculation and parameter description are shown in 8.2.19 and 8.2.20.
Note 3 "[MD14:MD13] = B'10" setting is prohibited because the system operation cannot be guaranteed.
Note 4 In case of the integer multiplication mode (SSMODE = 3'b000), SSFREQ and SSDEPT value are not enable.

### 8.1.5.5 PLL5 Multiplication Ratio

- There are no control registers for PLL5. The PLL5VCO frequency must be always 3200 MHz because all the modules operation which are shown in Table 8.1.4e cannot be guaranteed.

- Only the integer multiplication mode is supported because the modules operation cannot be guaranteed during the fractional multiplication mode and SSCG mode.

**Table 8.1.5.5 PLL5VCO Initial frequency**

| MD14:13 (*2) | PLL5 reference clock (*1) [MHz] | PLL5VCO [MHz] |
|---|---|---|
| B'00 | 16.66 | 3200.000 |
| B'01 | 20.00 | 3200.000 |
| B'11 | 16.66 | 3200.000 |

Note 1. When "MD14=1 and MD13=1", the reference clock is EXTAL x 1/2 (= 33.33 x 1/2). When the other mode setting, it is EXTAL x 1/1.
Note 2 "[MD14:MD13] = B'10" setting is prohibited because the system operation cannot be guaranteed.

### 8.1.5.6 PLL6 Multiplication ratio

- PLL6 initial multiplication ratio and initial state are shown in Table 8.1.5.6. PLL6 is initially operated with the integer mode.

- The PLL6 parameter can be changed by software registers. The parameter setting must be executed under PLL standby mode (See 8.3.1). The PLL6 parameter setting of SSCG (Center spread) is prohibited as shown in Table 8.1.1.

- When the PLL6VCO is changed by the software registers, PLL6VCO frequency range must be set as the below.
  1. Integer multiplication mode → 2000.000 MHz ~ 2800.000 MHz
  2. Fractional multiplication mode → 2000.000 MHz ~ (Not fixed)
  3. SSCG mode (Down spread) → 2000.000 MHz ~ (Not fixed)

**Table 8.1.5.6 PLL6 Initial Parameter Setting**

| MD14:13 (*3) | PLL6 reference clock (*1) [MHz] | PLL6 parameter setting (*2) | | | | | |
|---|---|---|---|---|---|---|---|
| | | SSMODE [2:0] | SSFREQ [6:0] | SSDEPT [6:0] | PLL6VCO [MHz] | NI[7:0] | NF[24:0] |
| B'00 | 16.66 | B'000 | (*4) | (*4) | 2800.000 | D'83 | H'0000000 |
| B'01 | 20.00 | B'000 | (*4) | (*4) | 2800.000 | D'69 | H'0000000 |
| B'11 | 16.66 | B'000 | (*4) | (*4) | 2800.000 | D'83 | H'0000000 |

Note 1. When "MD14=1 and MD13=1", the reference clock is EXTAL x 1/2 (= 33.33 x 1/2). When the other mode setting, it is EXTAL x 1/1.
Note 2 Detail calculation and parameter description are shown in Sec.8.2.21 and Sec.8.2.22.
Note 3 "[MD14:MD13] = B'10" setting is prohibited because the system operation cannot be guaranteed.
Note 4 In case of the integer multiplication mode (SSMODE = 3'b000), SSFREQ and SSDEPT value are not enable.

### 8.1.5.7 PLL7 Multiplication Ratio

- There are no control registers for PLL7. The PLL7VCO frequency must be always 2000 MHz because RGMII operation which are shown in Table 8.1.4g cannot be guaranteed.

- Only the integer multiplication mode is supported because the modules operation cannot be guaranteed during the fractional multiplication mode and SSCG mode.

**Table 8.1.5.7 PLL7VCO Initial frequency**

| MD14:13 (*2) | PLL7 reference clock (*1) [MHz] | PLL7VCO [MHz] |
|---|---|---|
| B'00 | 16.66 | 2000.000 |
| B'01 | 20.00 | 2000.000 |
| B'11 | 16.66 | 2000.000 |

Note 1. When "MD14=1 and MD13=1", the reference clock is EXTAL x 1/2 (= 33.33 x 1/2). When the other mode setting, it is EXTAL x 1/1.
Note 2 "[MD14:MD13] = B'10" setting is prohibited because the system operation cannot be guaranteed.

### 8.1.5.8 EXTAL/XTAL Pin Settings

EXTAL/XTAL settings can be selected by MD9 setting as shown in Table 8.1.5.8.

**Table 8.1.5.8 EXTAL/XTAL Pin Settings**

| MD9 | MD9 Settings |
|---|---|
| 0 | Inputs an external clock to the EXTAL pin. |
| 1 | Connects the crystal resonator to the EXTAL and XTAL pins. |

### 8.1.5.9 Clock Source of RCLK

Clock source of RCLK can be selected by MD29 setting as shown in Table 8.1.5.9.

**Table 8.1.5.9 MD29 Settings**

| MD29 | MD29 Settings |
|---|---|
| 0 | EXTALR select. |
| 1 | On-Chip-Oscillator (OCO) select. |

### 8.1.5.10 Step-up Mode

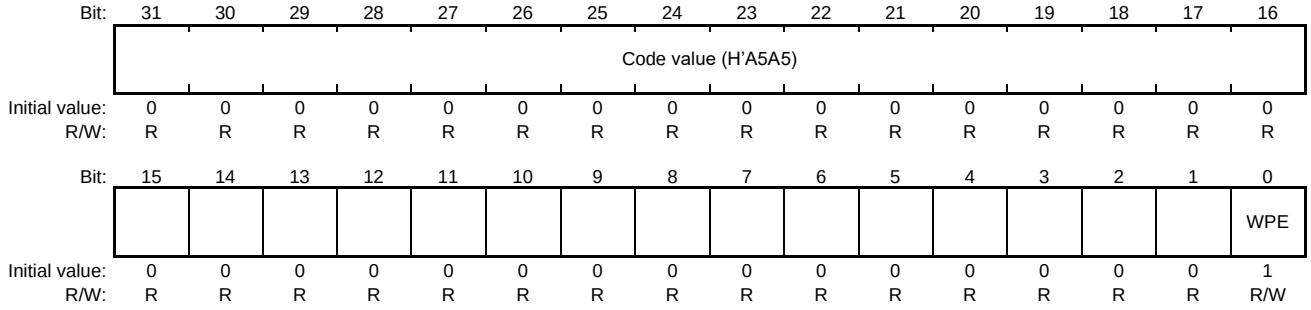
The MD0 pin selects either Free-running mode or Step-up mode. In Step-up mode, the frequency of the LSI-internal clocks is increased step by step at booting up of the LSI by the RESET# pin so that the power consumption increases gradually.

**Table 8.1.5.10 MD0 Settings**

| MD0 | MD0 Settings |
|---|---|
| 0 | Free-running mode |
| 1 | Step-up mode. |

### 8.1.1 CPG Write Protect Control Register (CPGWPCR)

CPGWPR is a 32-bit readable/writable register. See 8.4.1 in detail.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | - | All 0 | R | Code value (H'A5A5). When read, return 0. |
| 15 to 1 | - | All 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 0 | WPE | 1 | R/W | Write protect enable.<br>0: Disable the write protect.<br>1: Enable the write protect. |

### 8.1.2 Register Configuration

#### 8.1.2.1 Register Configuration for CPG

- The base-address of CPG is H'E615 0000
- CPG has Bus Access Protection function (DxWACRA (x = 0, 1, 2, 3)) for each register. See 8.4.2 in detail.

**Table 8.1.6.1a Register Configuration for CPG**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| CPGWPR | R/W | H'E615 0000 | H'00000000 | 32 | DxWACRA00[0] |
| CPGWPCR | R/W | H'E615 0004 | H'00000001 | 32 | DxWACRA00[1] |
| BKBAPR | R/W | H'E615 0280 | H'00000000 | 32 | DxWACRA05[0] |
| FSRCHKRA0 | R | H'E615 0400 | (*4) | 32 | (*1) |
| FSRCHKRA1 | R | H'E615 0404 | (*4) | 32 | (*1) |
| FSRCHKRA2 | R | H'E615 0408 | (*4) | 32 | (*1) |
| FSRCHKRA3 | R | H'E615 040C | (*4) | 32 | (*1) |
| FSRCHKRA4 | R | H'E615 0410 | (*4) | 32 | (*1) |
| FSRCHKRA5 | R | H'E615 0414 | (*4) | 32 | (*1) |
| FSRCHKRA6 | R | H'E615 0418 | (*4) | 32 | (*1) |
| FSRCHKRA7 | R | H'E615 041C | (*4) | 32 | (*1) |
| FSRCHKRA8 | R | H'E615 0420 | (*4) | 32 | (*1) |
| FSRCHKRA9 | R | H'E615 0424 | (*4) | 32 | (*1) |
| FSRCHKRA10 | R | H'E615 0428 | (*4) | 32 | (*1) |
| FSRCHKRA11 | R | H'E615 042C | (*4) | 32 | (*1) |
| FSRCHKRA12 | R | H'E615 0430 | (*4) | 32 | (*1) |
| FSRCHKRA13 | R | H'E615 0434 | (*4) | 32 | (*1) |
| FSRCHKRA14 | R | H'E615 0438 | (*4) | 32 | (*1) |
| FSRCHKRA15 | R | H'E615 043C | (*4) | 32 | (*1) |
| FSRCHKRA16 | R | H'E615 0440 | (*4) | 32 | (*1) |
| FSRCHKRA17 | R | H'E615 0444 | (*4) | 32 | (*1) |
| FSRCHKRA18 | R | H'E615 0448 | (*4) | 32 | (*1) |
| FSRCHKRA19 | R | H'E615 044C | (*4) | 32 | (*1) |
| FSRCHKRA20 | R | H'E615 0450 | (*4) | 32 | (*1) |
| FSRCHKRA21 | R | H'E615 0454 | (*4) | 32 | (*1) |
| FSRCHKRA22 | R | H'E615 0458 | (*4) | 32 | (*1) |
| FSRCHKRA23 | R | H'E615 045C | (*4) | 32 | (*1) |
| FSRCHKRA24 | R | H'E615 0460 | (*4) | 32 | (*1) |
| FSRCHKRA25 | R | H'E615 0464 | (*4) | 32 | (*1) |
| FSRCHKRA26 | R | H'E615 0468 | (*4) | 32 | (*1) |
| FSRCHKRA27 | R | H'E615 046C | (*4) | 32 | (*1) |
| FSRCHKRA28 | R | H'E615 0470 | (*4) | 32 | (*1) |
| FSRCHKRA29 | R | H'E615 0474 | (*4) | 32 | (*1) |
| FSRCHKRB0 | R | H'E615 0480 | (*4) | 32 | (*1) |
| FSRCHKRB1 | R | H'E615 0484 | (*4) | 32 | (*1) |
| FSRCHKRB2 | R | H'E615 0488 | (*4) | 32 | (*1) |
| FSRCHKRB3 | R | H'E615 048C | (*4) | 32 | (*1) |
| FSRCHKRB4 | R | H'E615 0490 | (*4) | 32 | (*1) |
| FSRCHKRB5 | R | H'E615 0494 | (*4) | 32 | (*1) |
| FSRCHKRB6 | R | H'E615 0498 | (*4) | 32 | (*1) |
| FSRCHKRB7 | R | H'E615 049C | (*4) | 32 | (*1) |
| FSRCHKRB8 | R | H'E615 04A0 | (*4) | 32 | (*1) |
| FSRCHKRB9 | R | H'E615 04A4 | (*4) | 32 | (*1) |
| FSRCHKRB10 | R | H'E615 04A8 | (*4) | 32 | (*1) |
| FSRCHKRB11 | R | H'E615 04AC | (*4) | 32 | (*1) |
| FSRCHKRB12 | R | H'E615 04B0 | (*4) | 32 | (*1) |
| FSRCHKRB13 | R | H'E615 04B4 | (*4) | 32 | (*1) |

| | | | | | |
|---|---|---|---|---|---|
| FSRCHKRB14 | R | H'E615 04B8 | (*4) | 32 | (*1) |
| FSRCHKRB15 | R | H'E615 04BC | (*4) | 32 | (*1) |
| FSRCHKRB16 | R | H'E615 04C0 | (*4) | 32 | (*1) |
| FSRCHKRB17 | R | H'E615 04C4 | (*4) | 32 | (*1) |
| FSRCHKRB18 | R | H'E615 04C8 | (*4) | 32 | (*1) |
| FSRCHKRB19 | R | H'E615 04CC | (*4) | 32 | (*1) |
| FSRCHKRB20 | R | H'E615 04D0 | (*4) | 32 | (*1) |
| FSRCHKRB21 | R | H'E615 04D4 | (*4) | 32 | (*1) |
| FSRCHKRB22 | R | H'E615 04D8 | (*4) | 32 | (*1) |
| FSRCHKRB23 | R | H'E615 04DC | (*4) | 32 | (*1) |
| FSRCHKRB24 | R | H'E615 04E0 | (*4) | 32 | (*1) |
| FSRCHKRB25 | R | H'E615 04E4 | (*4) | 32 | (*1) |
| FSRCHKRB26 | R | H'E615 04E8 | (*4) | 32 | (*1) |
| FSRCHKRB27 | R | H'E615 04EC | (*4) | 32 | (*1) |
| FSRCHKRB28 | R | H'E615 04F0 | (*4) | 32 | (*1) |
| FSRCHKRB29 | R | H'E615 04F4 | (*4) | 32 | (*1) |
| FSRCHKSETR0 | W | H'E615 0500 | H'00000000 | 32 | DxWACRA10[0] |
| FSRCHKSETR1 | W | H'E615 0504 | H'00000000 | 32 | DxWACRA10[1] |
| FSRCHKSETR2 | W | H'E615 0508 | H'00000000 | 32 | DxWACRA10[2] |
| FSRCHKSETR3 | W | H'E615 050C | H'00000000 | 32 | DxWACRA10[3] |
| FSRCHKSETR4 | W | H'E615 0510 | H'00000000 | 32 | DxWACRA10[4] |
| FSRCHKSETR5 | W | H'E615 0514 | H'00000000 | 32 | DxWACRA10[5] |
| FSRCHKSETR6 | W | H'E615 0518 | H'00000000 | 32 | DxWACRA10[6] |
| FSRCHKSETR7 | W | H'E615 051C | H'00000000 | 32 | DxWACRA10[7] |
| FSRCHKSETR8 | W | H'E615 0520 | H'00000000 | 32 | DxWACRA10[8] |
| FSRCHKSETR9 | W | H'E615 0524 | H'00000000 | 32 | DxWACRA10[9] |
| FSRCHKSETR10 | W | H'E615 0528 | H'00000000 | 32 | DxWACRA10[10] |
| FSRCHKSETR11 | W | H'E615 052C | H'00000000 | 32 | DxWACRA10[11] |
| FSRCHKSETR12 | W | H'E615 0530 | H'00000000 | 32 | DxWACRA10[12] |
| FSRCHKSETR13 | W | H'E615 0534 | H'00000000 | 32 | DxWACRA10[13] |
| FSRCHKSETR14 | W | H'E615 0538 | H'00000000 | 32 | DxWACRA10[14] |
| FSRCHKSETR15 | W | H'E615 053C | H'00000000 | 32 | DxWACRA10[15] |
| FSRCHKSETR16 | W | H'E615 0540 | H'00000000 | 32 | DxWACRA10[16] |
| FSRCHKSETR17 | W | H'E615 0544 | H'00000000 | 32 | DxWACRA10[17] |
| FSRCHKSETR18 | W | H'E615 0548 | H'00000000 | 32 | DxWACRA10[18] |
| FSRCHKSETR19 | W | H'E615 054C | H'00000000 | 32 | DxWACRA10[19] |
| FSRCHKSETR20 | W | H'E615 0550 | H'00000000 | 32 | DxWACRA10[20] |
| FSRCHKSETR21 | W | H'E615 0554 | H'00000000 | 32 | DxWACRA10[21] |
| FSRCHKSETR22 | W | H'E615 0558 | H'00000000 | 32 | DxWACRA10[22] |
| FSRCHKSETR23 | W | H'E615 055C | H'00000000 | 32 | DxWACRA10[23] |
| FSRCHKSETR24 | W | H'E615 0560 | H'00000000 | 32 | DxWACRA10[24] |
| FSRCHKSETR25 | W | H'E615 0564 | H'00000000 | 32 | DxWACRA10[25] |
| FSRCHKSETR26 | W | H'E615 0568 | H'00000000 | 32 | DxWACRA10[26] |
| FSRCHKSETR27 | W | H'E615 056C | H'00000000 | 32 | DxWACRA10[27] |
| FSRCHKSETR28 | W | H'E615 0570 | H'00000000 | 32 | DxWACRA10[28] |
| FSRCHKSETR29 | W | H'E615 0574 | H'00000000 | 32 | DxWACRA10[29] |
| FSRCHKCLRR0 | W | H'E615 0580 | H'00000000 | 32 | DxWACRA11[0] |
| FSRCHKCLRR1 | W | H'E615 0584 | H'00000000 | 32 | DxWACRA11[1] |
| FSRCHKCLRR2 | W | H'E615 0588 | H'00000000 | 32 | DxWACRA11[2] |
| FSRCHKCLRR3 | W | H'E615 058C | H'00000000 | 32 | DxWACRA11[3] |
| FSRCHKCLRR4 | W | H'E615 0590 | H'00000000 | 32 | DxWACRA11[4] |
| FSRCHKCLRR5 | W | H'E615 0594 | H'00000000 | 32 | DxWACRA11[5] |
| FSRCHKCLRR6 | W | H'E615 0598 | H'00000000 | 32 | DxWACRA11[6] |
| FSRCHKCLRR7 | W | H'E615 059C | H'00000000 | 32 | DxWACRA11[7] |
| FSRCHKCLRR8 | W | H'E615 05A0 | H'00000000 | 32 | DxWACRA11[8] |
| FSRCHKCLRR9 | W | H'E615 05A4 | H'00000000 | 32 | DxWACRA11[9] |
| FSRCHKCLRR10 | W | H'E615 05A8 | H'00000000 | 32 | DxWACRA11[10] |
| FSRCHKCLRR11 | W | H'E615 05AC | H'00000000 | 32 | DxWACRA11[11] |

| | | | | | |
|---|---|---|---|---|---|
| FSRCHKCLRR12 | W | H'E615 05B0 | H'00000000 | 32 | DxWACRA11[12] |
| FSRCHKCLRR13 | W | H'E615 05B4 | H'00000000 | 32 | DxWACRA11[13] |
| FSRCHKCLRR14 | W | H'E615 05B8 | H'00000000 | 32 | DxWACRA11[14] |
| FSRCHKCLRR15 | W | H'E615 05BC | H'00000000 | 32 | DxWACRA11[15] |
| FSRCHKCLRR16 | W | H'E615 05C0 | H'00000000 | 32 | DxWACRA11[16] |
| FSRCHKCLRR17 | W | H'E615 05C4 | H'00000000 | 32 | DxWACRA11[17] |
| FSRCHKCLRR18 | W | H'E615 05C8 | H'00000000 | 32 | DxWACRA11[18] |
| FSRCHKCLRR19 | W | H'E615 05CC | H'00000000 | 32 | DxWACRA11[19] |
| FSRCHKCLRR20 | W | H'E615 05D0 | H'00000000 | 32 | DxWACRA11[20] |
| FSRCHKCLRR21 | W | H'E615 05D4 | H'00000000 | 32 | DxWACRA11[21] |
| FSRCHKCLRR22 | W | H'E615 05D8 | H'00000000 | 32 | DxWACRA11[22] |
| FSRCHKCLRR23 | W | H'E615 05DC | H'00000000 | 32 | DxWACRA11[23] |
| FSRCHKCLRR24 | W | H'E615 05E0 | H'00000000 | 32 | DxWACRA11[24] |
| FSRCHKCLRR25 | W | H'E615 05E4 | H'00000000 | 32 | DxWACRA11[25] |
| FSRCHKCLRR26 | W | H'E615 05E8 | H'00000000 | 32 | DxWACRA11[26] |
| FSRCHKCLRR27 | W | H'E615 05EC | H'00000000 | 32 | DxWACRA11[27] |
| FSRCHKCLRR28 | W | H'E615 05F0 | H'00000000 | 32 | DxWACRA11[28] |
| FSRCHKCLRR29 | W | H'E615 05F4 | H'00000000 | 32 | DxWACRA11[29] |
| FRQCRB | R/W | H'E615 0804 | H'10113000 | 32 | DxWACRA16[1] |
| FRQCRC0 | R/W | H'E615 0808 | H'00000000 | 32 | DxWACRA16[2] |
| FRQCRD | R/W | H'E615 080C | H'00000000 | 32 | DxWACRA16[3] |
| PLLECR | R/W | H'E615 0820 | H'0000EBEB | 32 | DxWACRA16[8] |
| PLL1CR0 | R | H'E615 0830 | (*3) | 32 | DxWACRA16[12] |
| PLL2CR0 | R/W | H'E615 0834 | (*3) | 32 | DxWACRA16[13] |
| PLL3CR0 | R/W | H'E615 083C | (*3) | 32 | DxWACRA16[15] |
| PLL4CR0 | R/W | H'E615 0844 | (*3) | 32 | DxWACRA16[17] |
| PLL6CR0 | R/W | H'E615 084C | (*3) | 32 | DxWACRA16[19] |
| PLL1STPCR | R/W | H'E615 0850 | H'00000000 | 32 | DxWACRA16[20] |
| PLL2STPCR | R/W | H'E615 0854 | H'00000000 | 32 | DxWACRA16[21] |
| PLL3STPCR | R/W | H'E615 085C | H'00000000 | 32 | DxWACRA16[23] |
| PLL4STPCR | R/W | H'E615 0864 | H'00000000 | 32 | DxWACRA16[25] |
| PLL5STPCR | R/W | H'E615 0868 | H'00000000 | 32 | DxWACRA16[26] |
| PLL6STPCR | R/W | H'E615 086C | H'00000000 | 32 | DxWACRA16[27] |
| PLL7STPCR | R/W | H'E615 0904 | H'00000000 | 32 | DxWACRA18[1] |
| SD0CKCR | R/W | H'E615 0870 | H'00000209 | 32 | DxWACRA16[28] |
| RPCCKCR | R/W | H'E615 0874 | (*3) | 32 | DxWACRA16[29] |
| CANFDCKCR | R/W | H'E615 0878 | H'0000003F | 32 | DxWACRA16[30] |
| MSOCKCR | R/W | H'E615 087C | H'0000003F | 32 | DxWACRA16[31] |
| CSICKCR | R/W | H'E615 0880 | H'0000011F | 32 | DxWACRA17[0] |
| DSIEXTCKCR | R/W | H'E615 0884 | H'00000100 | 32 | DxWACRA17[1] |
| POSTCKCR | R/W | H'E615 0890 | (*3) | 32 | DxWACRA17[4] |
| POST2CKCR | R/W | H'E615 0894 | (*3) | 32 | DxWACRA17[5] |
| POST3CKCR | R/W | H'E615 0898 | (*3) | 32 | DxWACRA17[6] |
| POST4CKCR | R/W | H'E615 089C | (*3) | 32 | DxWACRA17[7] |
| PLL1CR1 | R | H'E615 08B0 | (*3) | 32 | DxWACRA17[12] |
| PLL2CR1 | R/W | H'E615 08B8 | (*3) | 32 | DxWACRA17[14] |
| PLL3CR1 | R | H'E615 08C0 | (*3) | 32 | DxWACRA17[16] |
| PLL4CR1 | R/W | H'E615 08C8 | (*3) | 32 | DxWACRA17[18] |
| PLL6CR1 | R/W | H'E615 08D8 | (*3) | 32 | DxWACRA17[22] |
| FRQCRC1 | R/W | H'E615 08E0 | H'00000000 | 32 | DxWACRA17[24] |
| CKSRCSELCR | R/W | H'E615 08A4 | H'0FFF0000 | 32 | DxWACRA17[9] |
| CPGMA0CKSTPCR | R/W | H'E615 0908 | H'00000000 | 32 | DxWACRA18[2] |
| ADGHCKCR | R/W | H'E615 090C | H'00000000 | 32 | DxWACRA18[3] |
| D0WACRA00 | R/W | H'E615 3800 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA01 | R/W | H'E615 3804 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA02 | R/W | H'E615 3808 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA03 | R/W | H'E615 380C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA04 | R/W | H'E615 3810 | H'FFFFFFFF | 32 | (*2) |

| | | | | | |
|---|---|---|---|---|---|
| D0WACRA05 | R/W | H'E615 3814 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA06 | R/W | H'E615 3818 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA07 | R/W | H'E615 381C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA08 | R/W | H'E615 3820 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA09 | R/W | H'E615 3824 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA10 | R/W | H'E615 3828 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA11 | R/W | H'E615 382C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA12 | R/W | H'E615 3830 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA13 | R/W | H'E615 3834 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA14 | R/W | H'E615 3838 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA15 | R/W | H'E615 383C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA16 | R/W | H'E615 3840 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA17 | R/W | H'E615 3844 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA18 | R/W | H'E615 3848 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA19 | R/W | H'E615 384C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA20 | R/W | H'E615 3850 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA21 | R/W | H'E615 3854 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA22 | R/W | H'E615 3858 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA23 | R/W | H'E615 385C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA24 | R/W | H'E615 3860 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA25 | R/W | H'E615 3864 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA26 | R/W | H'E615 3868 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA27 | R/W | H'E615 386C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA28 | R/W | H'E615 3870 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA29 | R/W | H'E615 3874 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA30 | R/W | H'E615 3878 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA31 | R/W | H'E615 387C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA32 | R/W | H'E615 3880 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA33 | R/W | H'E615 3884 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA34 | R/W | H'E615 3888 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA35 | R/W | H'E615 388C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA36 | R/W | H'E615 3890 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA37 | R/W | H'E615 3894 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA38 | R/W | H'E615 3898 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA39 | R/W | H'E615 389C | H'FFFFFFFF | 32 | (*2) |
| D0WACRA40 | R/W | H'E615 38A0 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA41 | R/W | H'E615 38A4 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA42 | R/W | H'E615 38A8 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA43 | R/W | H'E615 38AC | H'FFFFFFFF | 32 | (*2) |
| D0WACRA44 | R/W | H'E615 38B0 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA45 | R/W | H'E615 38B4 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA46 | R/W | H'E615 38B8 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA47 | R/W | H'E615 38BC | H'FFFFFFFF | 32 | (*2) |
| D0WACRA48 | R/W | H'E615 38C0 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA49 | R/W | H'E615 38C4 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA50 | R/W | H'E615 38C8 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA51 | R/W | H'E615 38CC | H'FFFFFFFF | 32 | (*2) |
| D0WACRA52 | R/W | H'E615 38D0 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA53 | R/W | H'E615 38D4 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA54 | R/W | H'E615 38D8 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA55 | R/W | H'E615 38DC | H'FFFFFFFF | 32 | (*2) |
| D0WACRA56 | R/W | H'E615 38E0 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA57 | R/W | H'E615 38E4 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA58 | R/W | H'E615 38E8 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA59 | R/W | H'E615 38EC | H'FFFFFFFF | 32 | (*2) |
| D0WACRA60 | R/W | H'E615 38F0 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA61 | R/W | H'E615 38F4 | H'FFFFFFFF | 32 | (*2) |
| D0WACRA62 | R/W | H'E615 38F8 | H'FFFFFFFF | 32 | (*2) |

| | | | | | |
|---|---|---|---|---|---|
| D0WACRA63 | R/W | H'E615 38FC | H'FFFFFFFF | 32 | (*2) |
| D1WACRA00 | R/W | H'E615 3900 | H'00000000 | 32 | (*2) |
| D1WACRA01 | R/W | H'E615 3904 | H'00000000 | 32 | (*2) |
| D1WACRA02 | R/W | H'E615 3908 | H'00000000 | 32 | (*2) |
| D1WACRA03 | R/W | H'E615 390C | H'00000000 | 32 | (*2) |
| D1WACRA04 | R/W | H'E615 3910 | H'00000000 | 32 | (*2) |
| D1WACRA05 | R/W | H'E615 3914 | H'00000000 | 32 | (*2) |
| D1WACRA06 | R/W | H'E615 3918 | H'00000000 | 32 | (*2) |
| D1WACRA07 | R/W | H'E615 391C | H'00000000 | 32 | (*2) |
| D1WACRA08 | R/W | H'E615 3920 | H'00000000 | 32 | (*2) |
| D1WACRA09 | R/W | H'E615 3924 | H'00000000 | 32 | (*2) |
| D1WACRA10 | R/W | H'E615 3928 | H'00000000 | 32 | (*2) |
| D1WACRA11 | R/W | H'E615 392C | H'00000000 | 32 | (*2) |
| D1WACRA12 | R/W | H'E615 3930 | H'00000000 | 32 | (*2) |
| D1WACRA13 | R/W | H'E615 3934 | H'00000000 | 32 | (*2) |
| D1WACRA14 | R/W | H'E615 3938 | H'00000000 | 32 | (*2) |
| D1WACRA15 | R/W | H'E615 393C | H'00000000 | 32 | (*2) |
| D1WACRA16 | R/W | H'E615 3940 | H'00000000 | 32 | (*2) |
| D1WACRA17 | R/W | H'E615 3944 | H'00000000 | 32 | (*2) |
| D1WACRA18 | R/W | H'E615 3948 | H'00000000 | 32 | (*2) |
| D1WACRA19 | R/W | H'E615 394C | H'00000000 | 32 | (*2) |
| D1WACRA20 | R/W | H'E615 3950 | H'00000000 | 32 | (*2) |
| D1WACRA21 | R/W | H'E615 3954 | H'00000000 | 32 | (*2) |
| D1WACRA22 | R/W | H'E615 3958 | H'00000000 | 32 | (*2) |
| D1WACRA23 | R/W | H'E615 395C | H'00000000 | 32 | (*2) |
| D1WACRA24 | R/W | H'E615 3960 | H'00000000 | 32 | (*2) |
| D1WACRA25 | R/W | H'E615 3964 | H'00000000 | 32 | (*2) |
| D1WACRA26 | R/W | H'E615 3968 | H'00000000 | 32 | (*2) |
| D1WACRA27 | R/W | H'E615 396C | H'00000000 | 32 | (*2) |
| D1WACRA28 | R/W | H'E615 3970 | H'00000000 | 32 | (*2) |
| D1WACRA29 | R/W | H'E615 3974 | H'00000000 | 32 | (*2) |
| D1WACRA30 | R/W | H'E615 3978 | H'00000000 | 32 | (*2) |
| D1WACRA31 | R/W | H'E615 397C | H'00000000 | 32 | (*2) |
| D1WACRA32 | R/W | H'E615 3980 | H'00000000 | 32 | (*2) |
| D1WACRA33 | R/W | H'E615 3984 | H'00000000 | 32 | (*2) |
| D1WACRA34 | R/W | H'E615 3988 | H'00000000 | 32 | (*2) |
| D1WACRA35 | R/W | H'E615 398C | H'00000000 | 32 | (*2) |
| D1WACRA36 | R/W | H'E615 3990 | H'00000000 | 32 | (*2) |
| D1WACRA37 | R/W | H'E615 3994 | H'00000000 | 32 | (*2) |
| D1WACRA38 | R/W | H'E615 3998 | H'00000000 | 32 | (*2) |
| D1WACRA39 | R/W | H'E615 399C | H'00000000 | 32 | (*2) |
| D1WACRA40 | R/W | H'E615 39A0 | H'00000000 | 32 | (*2) |
| D1WACRA41 | R/W | H'E615 39A4 | H'00000000 | 32 | (*2) |
| D1WACRA42 | R/W | H'E615 39A8 | H'00000000 | 32 | (*2) |
| D1WACRA43 | R/W | H'E615 39AC | H'00000000 | 32 | (*2) |
| D1WACRA44 | R/W | H'E615 39B0 | H'00000000 | 32 | (*2) |
| D1WACRA45 | R/W | H'E615 39B4 | H'00000000 | 32 | (*2) |
| D1WACRA46 | R/W | H'E615 39B8 | H'00000000 | 32 | (*2) |
| D1WACRA47 | R/W | H'E615 39BC | H'00000000 | 32 | (*2) |
| D1WACRA48 | R/W | H'E615 39C0 | H'00000000 | 32 | (*2) |
| D1WACRA49 | R/W | H'E615 39C4 | H'00000000 | 32 | (*2) |
| D1WACRA50 | R/W | H'E615 39C8 | H'00000000 | 32 | (*2) |
| D1WACRA51 | R/W | H'E615 39CC | H'00000000 | 32 | (*2) |
| D1WACRA52 | R/W | H'E615 39D0 | H'00000000 | 32 | (*2) |
| D1WACRA53 | R/W | H'E615 39D4 | H'00000000 | 32 | (*2) |
| D1WACRA54 | R/W | H'E615 39D8 | H'00000000 | 32 | (*2) |
| D1WACRA55 | R/W | H'E615 39DC | H'00000000 | 32 | (*2) |
| D1WACRA56 | R/W | H'E615 39E0 | H'00000000 | 32 | (*2) |

| | | | | | |
|---|---|---|---|---|---|
| D1WACRA57 | R/W | H'E615 39E4 | H'00000000 | 32 | (*2) |
| D1WACRA58 | R/W | H'E615 39E8 | H'00000000 | 32 | (*2) |
| D1WACRA59 | R/W | H'E615 39EC | H'00000000 | 32 | (*2) |
| D1WACRA60 | R/W | H'E615 39F0 | H'00000000 | 32 | (*2) |
| D1WACRA61 | R/W | H'E615 39F4 | H'00000000 | 32 | (*2) |
| D1WACRA62 | R/W | H'E615 39F8 | H'00000000 | 32 | (*2) |
| D1WACRA63 | R/W | H'E615 39FC | H'00000000 | 32 | (*2) |
| D2WACRA00 | R/W | H'E615 3A00 | H'00000000 | 32 | (*2) |
| D2WACRA01 | R/W | H'E615 3A04 | H'00000000 | 32 | (*2) |
| D2WACRA02 | R/W | H'E615 3A08 | H'00000000 | 32 | (*2) |
| D2WACRA03 | R/W | H'E615 3A0C | H'00000000 | 32 | (*2) |
| D2WACRA04 | R/W | H'E615 3A10 | H'00000000 | 32 | (*2) |
| D2WACRA05 | R/W | H'E615 3A14 | H'00000000 | 32 | (*2) |
| D2WACRA06 | R/W | H'E615 3A18 | H'00000000 | 32 | (*2) |
| D2WACRA07 | R/W | H'E615 3A1C | H'00000000 | 32 | (*2) |
| D2WACRA08 | R/W | H'E615 3A20 | H'00000000 | 32 | (*2) |
| D2WACRA09 | R/W | H'E615 3A24 | H'00000000 | 32 | (*2) |
| D2WACRA10 | R/W | H'E615 3A28 | H'00000000 | 32 | (*2) |
| D2WACRA11 | R/W | H'E615 3A2C | H'00000000 | 32 | (*2) |
| D2WACRA12 | R/W | H'E615 3A30 | H'00000000 | 32 | (*2) |
| D2WACRA13 | R/W | H'E615 3A34 | H'00000000 | 32 | (*2) |
| D2WACRA14 | R/W | H'E615 3A38 | H'00000000 | 32 | (*2) |
| D2WACRA15 | R/W | H'E615 3A3C | H'00000000 | 32 | (*2) |
| D2WACRA16 | R/W | H'E615 3A40 | H'00000000 | 32 | (*2) |
| D2WACRA17 | R/W | H'E615 3A44 | H'00000000 | 32 | (*2) |
| D2WACRA18 | R/W | H'E615 3A48 | H'00000000 | 32 | (*2) |
| D2WACRA19 | R/W | H'E615 3A4C | H'00000000 | 32 | (*2) |
| D2WACRA20 | R/W | H'E615 3A50 | H'00000000 | 32 | (*2) |
| D2WACRA21 | R/W | H'E615 3A54 | H'00000000 | 32 | (*2) |
| D2WACRA22 | R/W | H'E615 3A58 | H'00000000 | 32 | (*2) |
| D2WACRA23 | R/W | H'E615 3A5C | H'00000000 | 32 | (*2) |
| D2WACRA24 | R/W | H'E615 3A60 | H'00000000 | 32 | (*2) |
| D2WACRA25 | R/W | H'E615 3A64 | H'00000000 | 32 | (*2) |
| D2WACRA26 | R/W | H'E615 3A68 | H'00000000 | 32 | (*2) |
| D2WACRA27 | R/W | H'E615 3A6C | H'00000000 | 32 | (*2) |
| D2WACRA28 | R/W | H'E615 3A70 | H'00000000 | 32 | (*2) |
| D2WACRA29 | R/W | H'E615 3A74 | H'00000000 | 32 | (*2) |
| D2WACRA30 | R/W | H'E615 3A78 | H'00000000 | 32 | (*2) |
| D2WACRA31 | R/W | H'E615 3A7C | H'00000000 | 32 | (*2) |
| D2WACRA32 | R/W | H'E615 3A80 | H'00000000 | 32 | (*2) |
| D2WACRA33 | R/W | H'E615 3A84 | H'00000000 | 32 | (*2) |
| D2WACRA34 | R/W | H'E615 3A88 | H'00000000 | 32 | (*2) |
| D2WACRA35 | R/W | H'E615 3A8C | H'00000000 | 32 | (*2) |
| D2WACRA36 | R/W | H'E615 3A90 | H'00000000 | 32 | (*2) |
| D2WACRA37 | R/W | H'E615 3A94 | H'00000000 | 32 | (*2) |
| D2WACRA38 | R/W | H'E615 3A98 | H'00000000 | 32 | (*2) |
| D2WACRA39 | R/W | H'E615 3A9C | H'00000000 | 32 | (*2) |
| D2WACRA40 | R/W | H'E615 3AA0 | H'00000000 | 32 | (*2) |
| D2WACRA41 | R/W | H'E615 3AA4 | H'00000000 | 32 | (*2) |
| D2WACRA42 | R/W | H'E615 3AA8 | H'00000000 | 32 | (*2) |
| D2WACRA43 | R/W | H'E615 3AAC | H'00000000 | 32 | (*2) |
| D2WACRA44 | R/W | H'E615 3AB0 | H'00000000 | 32 | (*2) |
| D2WACRA45 | R/W | H'E615 3AB4 | H'00000000 | 32 | (*2) |
| D2WACRA46 | R/W | H'E615 3AB8 | H'00000000 | 32 | (*2) |
| D2WACRA47 | R/W | H'E615 3ABC | H'00000000 | 32 | (*2) |
| D2WACRA48 | R/W | H'E615 3AC0 | H'00000000 | 32 | (*2) |
| D2WACRA49 | R/W | H'E615 3AC4 | H'00000000 | 32 | (*2) |
| D2WACRA50 | R/W | H'E615 3AC8 | H'00000000 | 32 | (*2) |

| | | | | | |
|---|---|---|---|---|---|
| D2WACRA51 | R/W | H'E615 3ACC | H'00000000 | 32 | (*2) |
| D2WACRA52 | R/W | H'E615 3AD0 | H'00000000 | 32 | (*2) |
| D2WACRA53 | R/W | H'E615 3AD4 | H'00000000 | 32 | (*2) |
| D2WACRA54 | R/W | H'E615 3AD8 | H'00000000 | 32 | (*2) |
| D2WACRA55 | R/W | H'E615 3ADC | H'00000000 | 32 | (*2) |
| D2WACRA56 | R/W | H'E615 3AE0 | H'00000000 | 32 | (*2) |
| D2WACRA57 | R/W | H'E615 3AE4 | H'00000000 | 32 | (*2) |
| D2WACRA58 | R/W | H'E615 3AE8 | H'00000000 | 32 | (*2) |
| D2WACRA59 | R/W | H'E615 3AEC | H'00000000 | 32 | (*2) |
| D2WACRA60 | R/W | H'E615 3AF0 | H'00000000 | 32 | (*2) |
| D2WACRA61 | R/W | H'E615 3AF4 | H'00000000 | 32 | (*2) |
| D2WACRA62 | R/W | H'E615 3AF8 | H'00000000 | 32 | (*2) |
| D2WACRA63 | R/W | H'E615 3AFC | H'00000000 | 32 | (*2) |
| D3WACRA00 | R/W | H'E615 3B00 | H'00000000 | 32 | (*2) |
| D3WACRA01 | R/W | H'E615 3B04 | H'00000000 | 32 | (*2) |
| D3WACRA02 | R/W | H'E615 3B08 | H'00000000 | 32 | (*2) |
| D3WACRA03 | R/W | H'E615 3B0C | H'00000000 | 32 | (*2) |
| D3WACRA04 | R/W | H'E615 3B10 | H'00000000 | 32 | (*2) |
| D3WACRA05 | R/W | H'E615 3B14 | H'00000000 | 32 | (*2) |
| D3WACRA06 | R/W | H'E615 3B18 | H'00000000 | 32 | (*2) |
| D3WACRA07 | R/W | H'E615 3B1C | H'00000000 | 32 | (*2) |
| D3WACRA08 | R/W | H'E615 3B20 | H'00000000 | 32 | (*2) |
| D3WACRA09 | R/W | H'E615 3B24 | H'00000000 | 32 | (*2) |
| D3WACRA10 | R/W | H'E615 3B28 | H'00000000 | 32 | (*2) |
| D3WACRA11 | R/W | H'E615 3B2C | H'00000000 | 32 | (*2) |
| D3WACRA12 | R/W | H'E615 3B30 | H'00000000 | 32 | (*2) |
| D3WACRA13 | R/W | H'E615 3B34 | H'00000000 | 32 | (*2) |
| D3WACRA14 | R/W | H'E615 3B38 | H'00000000 | 32 | (*2) |
| D3WACRA15 | R/W | H'E615 3B3C | H'00000000 | 32 | (*2) |
| D3WACRA16 | R/W | H'E615 3B40 | H'00000000 | 32 | (*2) |
| D3WACRA17 | R/W | H'E615 3B44 | H'00000000 | 32 | (*2) |
| D3WACRA18 | R/W | H'E615 3B48 | H'00000000 | 32 | (*2) |
| D3WACRA19 | R/W | H'E615 3B4C | H'00000000 | 32 | (*2) |
| D3WACRA20 | R/W | H'E615 3B50 | H'00000000 | 32 | (*2) |
| D3WACRA21 | R/W | H'E615 3B54 | H'00000000 | 32 | (*2) |
| D3WACRA22 | R/W | H'E615 3B58 | H'00000000 | 32 | (*2) |
| D3WACRA23 | R/W | H'E615 3B5C | H'00000000 | 32 | (*2) |
| D3WACRA24 | R/W | H'E615 3B60 | H'00000000 | 32 | (*2) |
| D3WACRA25 | R/W | H'E615 3B64 | H'00000000 | 32 | (*2) |
| D3WACRA26 | R/W | H'E615 3B68 | H'00000000 | 32 | (*2) |
| D3WACRA27 | R/W | H'E615 3B6C | H'00000000 | 32 | (*2) |
| D3WACRA28 | R/W | H'E615 3B70 | H'00000000 | 32 | (*2) |
| D3WACRA29 | R/W | H'E615 3B74 | H'00000000 | 32 | (*2) |
| D3WACRA30 | R/W | H'E615 3B78 | H'00000000 | 32 | (*2) |
| D3WACRA31 | R/W | H'E615 3B7C | H'00000000 | 32 | (*2) |
| D3WACRA32 | R/W | H'E615 3B80 | H'00000000 | 32 | (*2) |
| D3WACRA33 | R/W | H'E615 3B84 | H'00000000 | 32 | (*2) |
| D3WACRA34 | R/W | H'E615 3B88 | H'00000000 | 32 | (*2) |
| D3WACRA35 | R/W | H'E615 3B8C | H'00000000 | 32 | (*2) |
| D3WACRA36 | R/W | H'E615 3B90 | H'00000000 | 32 | (*2) |
| D3WACRA37 | R/W | H'E615 3B94 | H'00000000 | 32 | (*2) |
| D3WACRA38 | R/W | H'E615 3B98 | H'00000000 | 32 | (*2) |
| D3WACRA39 | R/W | H'E615 3B9C | H'00000000 | 32 | (*2) |
| D3WACRA40 | R/W | H'E615 3BA0 | H'00000000 | 32 | (*2) |
| D3WACRA41 | R/W | H'E615 3BA4 | H'00000000 | 32 | (*2) |
| D3WACRA42 | R/W | H'E615 3BA8 | H'00000000 | 32 | (*2) |
| D3WACRA43 | R/W | H'E615 3BAC | H'00000000 | 32 | (*2) |
| D3WACRA44 | R/W | H'E615 3BB0 | H'00000000 | 32 | (*2) |

| | | | | | |
|---|---|---|---|---|---|
| D3WACRA45 | R/W | H'E615 3BB4 | H'00000000 | 32 | (*2) |
| D3WACRA46 | R/W | H'E615 3BB8 | H'00000000 | 32 | (*2) |
| D3WACRA47 | R/W | H'E615 3BBC | H'00000000 | 32 | (*2) |
| D3WACRA48 | R/W | H'E615 3BC0 | H'00000000 | 32 | (*2) |
| D3WACRA49 | R/W | H'E615 3BC4 | H'00000000 | 32 | (*2) |
| D3WACRA50 | R/W | H'E615 3BC8 | H'00000000 | 32 | (*2) |
| D3WACRA51 | R/W | H'E615 3BCC | H'00000000 | 32 | (*2) |
| D3WACRA52 | R/W | H'E615 3BD0 | H'00000000 | 32 | (*2) |
| D3WACRA53 | R/W | H'E615 3BD4 | H'00000000 | 32 | (*2) |
| D3WACRA54 | R/W | H'E615 3BD8 | H'00000000 | 32 | (*2) |
| D3WACRA55 | R/W | H'E615 3BDC | H'00000000 | 32 | (*2) |
| D3WACRA56 | R/W | H'E615 3BE0 | H'00000000 | 32 | (*2) |
| D3WACRA57 | R/W | H'E615 3BE4 | H'00000000 | 32 | (*2) |
| D3WACRA58 | R/W | H'E615 3BE8 | H'00000000 | 32 | (*2) |
| D3WACRA59 | R/W | H'E615 3BEC | H'00000000 | 32 | (*2) |
| D3WACRA60 | R/W | H'E615 3BF0 | H'00000000 | 32 | (*2) |
| D3WACRA61 | R/W | H'E615 3BF4 | H'00000000 | 32 | (*2) |
| D3WACRA62 | R/W | H'E615 3BF8 | H'00000000 | 32 | (*2) |
| D3WACRA63 | R/W | H'E615 3BFC | H'00000000 | 32 | (*2) |
| Write Access Protect Control/Status Register (CPGWPTCSR) | R/W | H'E615 0748 | H'00000002 | 32 | DxWACRA14[18] |

Note 1. There are no protect registers (DxWACRA (x = 0, 1, 2, 3)) for Read Only register.
Note 2 DxWACRA (x = 0, 1, 2, 3) registers can be accessed only by Domain 0.
Note 3 Initial values depend on MD pins. See each section in Sec.8.2 Register Description
Note 4 FSRCHKRAn and FSRCHKRAn (n = 0 to 29) → Initial value is “Undefined value”.

**Table 8.1.6.1b Register Configuration for CPG internal Clock Monitor**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| PLL1FBCKMCSR | R/W | H'E615 0C00 | H'----_---- | 32 | (*1) |
| PLL1FBCKMECR | R/W | H'E615 0C04 | H'00000000 | 32 | (*1) |
| PLL1FBCKMLCH | R/W | H'E615 0C08 | H'00000000 | 32 | (*1) |
| PLL1FBCKMLCL | R/W | H'E615 0C0C | H'00000000 | 32 | (*1) |
| PLL1FBCKMCNT | R | H'E615 0C10 | H'00000000 | 32 | (*1) |
| PLL1FBCKMCNTE | R | H'E615 0C14 | H'00000000 | 32 | (*1) |
| PLL2FBCKMCSR | R/W | H'E615 0C20 | H'----_---- | 32 | (*1) |
| PLL2FBCKMECR | R/W | H'E615 0C24 | H'00000000 | 32 | (*1) |
| PLL2FBCKMLCH | R/W | H'E615 0C28 | H'00000000 | 32 | (*1) |
| PLL2FBCKMLCL | R/W | H'E615 0C2C | H'00000000 | 32 | (*1) |
| PLL2FBCKMCNT | R | H'E615 0C30 | H'00000000 | 32 | (*1) |
| PLL2FBCKMCNTE | R | H'E615 0C34 | H'00000000 | 32 | (*1) |
| PLL3FBCKMCSR | R/W | H'E615 0C60 | H'----_---- | 32 | (*1) |
| PLL3FBCKMECR | R/W | H'E615 0C64 | H'00000000 | 32 | (*1) |
| PLL3FBCKMLCH | R/W | H'E615 0C68 | H'00000000 | 32 | (*1) |
| PLL3FBCKMLCL | R/W | H'E615 0C6C | H'00000000 | 32 | (*1) |
| PLL3FBCKMCNT | R | H'E615 0C70 | H'00000000 | 32 | (*1) |
| PLL3FBCKMCNTE | R | H'E615 0C74 | H'00000000 | 32 | (*1) |
| PLL4FBCKMCSR | R/W | H'E615 0CA0 | H'----_---- | 32 | (*1) |
| PLL4FBCKMECR | R/W | H'E615 0CA4 | H'00000000 | 32 | (*1) |
| PLL4FBCKMLCH | R/W | H'E615 0CA8 | H'00000000 | 32 | (*1) |
| PLL4FBCKMLCL | R/W | H'E615 0CAC | H'00000000 | 32 | (*1) |
| PLL4FBCKMCNT | R | H'E615 0CB0 | H'00000000 | 32 | (*1) |
| PLL4FBCKMCNTE | R | H'E615 0CB4 | H'00000000 | 32 | (*1) |
| PLL5FBCKMCSR | R/W | H'E615 0CC0 | H'----_---- | 32 | (*1) |
| PLL5FBCKMECR | R/W | H'E615 0CC4 | H'00000000 | 32 | (*1) |
| PLL5FBCKMLCH | R/W | H'E615 0CC8 | H'00000000 | 32 | (*1) |
| PLL5FBCKMLCL | R/W | H'E615 0CCC | H'00000000 | 32 | (*1) |
| PLL5FBCKMCNT | R | H'E615 0CD0 | H'00000000 | 32 | (*1) |
| PLL5FBCKMCNTE | R | H'E615 0CD4 | H'00000000 | 32 | (*1) |
| PLL6FBCKMCSR | R/W | H'E615 0E60 | H'----_---- | 32 | (*1) |
| PLL6FBCKMECR | R/W | H'E615 0E64 | H'00000000 | 32 | (*1) |
| PLL6FBCKMLCH | R/W | H'E615 0E68 | H'00000000 | 32 | (*1) |
| PLL6FBCKMLCL | R/W | H'E615 0E6C | H'00000000 | 32 | (*1) |
| PLL6FBCKMCNT | R | H'E615 0E70 | H'00000000 | 32 | (*1) |
| PLL6FBCKMCNTE | R | H'E615 0E74 | H'00000000 | 32 | (*1) |
| PLL7FBCKMCSR | R/W | H'E615 0EA0 | H'----_---- | 32 | (*1) |
| PLL7FBCKMECR | R/W | H'E615 0EA4 | H'00000000 | 32 | (*1) |
| PLL7FBCKMLCH | R/W | H'E615 0EA8 | H'00000000 | 32 | (*1) |
| PLL7FBCKMLCL | R/W | H'E615 0EAC | H'00000000 | 32 | (*1) |
| PLL7FBCKMCNT | R | H'E615 0EB0 | H'00000000 | 32 | (*1) |
| PLL7FBCKMCNTE | R | H'E615 0EB4 | H'00000000 | 32 | (*1) |
| CBFUSACKMCSR | R/W | H'E615 0D00 | H'----_---- | 32 | (*1) |
| CBFUSACKMECR | R/W | H'E615 0D04 | H'00000000 | 32 | (*1) |
| CBFUSACKMLCH | R/W | H'E615 0D08 | H'00000000 | 32 | (*1) |
| CBFUSACKMLCL | R/W | H'E615 0D0C | H'00000000 | 32 | (*1) |
| CBFUSACKMCNT | R | H'E615 0D10 | H'00000000 | 32 | (*1) |
| CBFUSACKMCNTE | R | H'E615 0D14 | H'00000000 | 32 | (*1) |
| RCLKCKMCSR | R/W | H'E615 0E80 | H'----_---- | 32 | (*1) |
| RCLKCKMECR | R/W | H'E615 0E84 | H'00000000 | 32 | (*1) |
| RCLKCKMLCH | R/W | H'E615 0E88 | H'00000000 | 32 | (*1) |
| RCLKCKMLCL | R/W | H'E615 0E8C | H'00000000 | 32 | (*1) |
| RCLKCKMCNT | R | H'E615 0E90 | H'00000000 | 32 | (*1) |
| RCLKCKMCNTE | R | H'E615 0E94 | H'00000000 | 32 | (*1) |
| CPGACKMSR | R | H'E615 0D1C | H'00000000 | 32 | (*1) |

Note 1 There are no protect registers (DxWACRA (x = 0, 1, 2, 3)) for registers related to Clock Monitor like registers shown in 8.1.6.2.

### 8.1.2.2 Register Configuration for Clock Monitor

#### (1) Clock Monitor in Slave Access Bus domain

- The base-address of CPG is H'E607 0000. The address map for clock monitor in Slave Access Bus domain is assigned from H'E607 0000 to H'E607 1FFF.
- There are no protect registers (DxWACRA (x = 0, 1, 2, 3)) for the registers.

**Table 8.1.6.2.1 Register Configuration**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|
| S0D2CKMCSR | R/W | H'E607 0240 | H'----_---- | 32 |
| S0D2CKMECR | R/W | H'E607 0244 | H'00000000 | 32 |
| S0D2CKMLCH | R/W | H'E607 0248 | H'00000000 | 32 |
| S0D2CKMLCL | R/W | H'E607 024C | H'00000000 | 32 |
| S0D2CKMCNT | R | H'E607 0250 | H'00000000 | 32 |
| S0D2CKMCNTE | R | H'E607 0254 | H'00000000 | 32 |
| S0D3CKMCSR | R/W | H'E607 0260 | H'----_---- | 32 |
| S0D3CKMECR | R/W | H'E607 0264 | H'00000000 | 32 |
| S0D3CKMLCH | R/W | H'E607 0268 | H'00000000 | 32 |
| S0D3CKMLCL | R/W | H'E607 026C | H'00000000 | 32 |
| S0D3CKMCNT | R | H'E607 0270 | H'00000000 | 32 |
| S0D3CKMCNTE | R | H'E607 0274 | H'00000000 | 32 |
| S0D4CKMCSR | R/W | H'E607 0280 | H'----_---- | 32 |
| S0D4CKMECR | R/W | H'E607 0284 | H'00000000 | 32 |
| S0D4CKMLCH | R/W | H'E607 0288 | H'00000000 | 32 |
| S0D4CKMLCL | R/W | H'E607 028C | H'00000000 | 32 |
| S0D4CKMCNT | R | H'E607 0290 | H'00000000 | 32 |
| S0D4CKMCNTE | R | H'E607 0294 | H'00000000 | 32 |
| CLCKMCSR | R/W | H'E607 00F0 | H'----_---- | 32 |
| CLCKMECR | R/W | H'E607 00F4 | H'00000000 | 32 |
| CLCKMLCH | R/W | H'E607 00F8 | H'00000000 | 32 |
| CLCKMLCL | R/W | H'E607 00FC | H'00000000 | 32 |
| CLCKMCNT | R | H'E607 0100 | H'00000000 | 32 |
| CLCKMCNTE | R | H'E607 0104 | H'00000000 | 32 |
| ZXCKMCSR | R/W | H'E607 0300 | H'---- ---- | 32 |
| ZXCKMECR | R/W | H'E607 0304 | H'00000000 | 32 |
| ZXCKMLCH | R/W | H'E607 0308 | H'00000000 | 32 |
| ZXCKMLCL | R/W | H'E607 030C | H'00000000 | 32 |
| ZXCKMCNT | R | H'E607 0310 | H'00000000 | 32 |
| ZXCKMCNTE | R | H'E607 0314 | H'00000000 | 32 |
| CL16MCKMCSR | R/W | H'E607 0320 | H'----_---- | 32 |
| CL16MCKMECR | R/W | H'E607 0324 | H'00000000 | 32 |
| CL16MCKMLCH | R/W | H'E607 0328 | H'00000000 | 32 |
| CL16MCKMLCL | R/W | H'E607 032C | H'00000000 | 32 |
| CL16MCKMCNT | R | H'E607 0330 | H'00000000 | 32 |
| CL16MCKMCNTE | R | H'E607 0334 | H'00000000 | 32 |
| CPCKMCSR | R/W | H'E607 0190 | H'----_---- | 32 |
| CPCKMECR | R/W | H'E607 0194 | H'00000000 | 32 |
| CPCKMLCH | R/W | H'E607 0198 | H'00000000 | 32 |
| CPCKMLCL | R/W | H'E607 019C | H'00000000 | 32 |
| CPCKMCNT | R | H'E607 01A0 | H'00000000 | 32 |
| CPCKMCNTE | R | H'E607 01A4 | H'00000000 | 32 |
| CPEXCKMCSR | R/W | H'E607 01D0 | H'---- ---- | 32 |
| CPEXCKMECR | R/W | H'E607 01D4 | H'00000000 | 32 |
| CPEXCKMLCH | R/W | H'E607 01D8 | H'00000000 | 32 |
| CPEXCKMLCL | R/W | H'E607 01DC | H'00000000 | 32 |
| CPEXCKMCNT | R | H'E607 01E0 | H'00000000 | 32 |
| CPEXCKMCNTE | R | H'E607 01E4 | H'00000000 | 32 |
| CBFUSATOPCKMCSR | R/W | H'E607 01F0 | H'----_---- | 32 |
| CBFUSATOPCKMECR | R/W | H'E607 01F4 | H'00000000 | 32 |
| CBFUSATOPCKMLCH | R/W | H'E607 01F8 | H'00000000 | 32 |
| CBFUSATOPCKMLCL | R/W | H'E607 01FC | H'00000000 | 32 |
| CBFUSATOPCKMCNT | R | H'E607 0200 | H'00000000 | 32 |
| CBFUSATOPCKMCNTE | R | H'E607 0204 | H'00000000 | 32 |
| RCLKTOPCKMCSR | R/W | H'E607 01B0 | H'----_---- | 32 |
| RCLKTOPCKMECR | R/W | H'E607 01B4 | H'00000000 | 32 |
| RCLKTOPCKMLCH | R/W | H'E607 01B8 | H'00000000 | 32 |

| | | | | |
|---|---|---|---|---|
| RCLKTOPCKMLCL | R/W | H'E607 01BC | H'00000000 | 32 |
| RCLKTOPCKMCNT | R | H'E607 01C0 | H'00000000 | 32 |
| RCLKTOPCKMCNTE | R | H'E607 01C4 | H'00000000 | 32 |
| TOPCKMWPR | R/W | H'E607 0000 | H'00000000 | 32 |
| TOPCKMWPCR | R/W | H'E607 0004 | H'00000001 | 32 |
| TOPACKMSR | R | H'E607 0210 | H'0000 0000 | 32 |

### (2) Clock Monitor in Main Memory domain

- The base-address of CPG is H'E67F A000. The address map for clock monitor in Main Memory domain is assigned from H'E67F_A000 to H'E67F_BFFF.
- There are no protect registers (DxWACRA (x = 0, 1, 2, 3)) for the registers.

**Table 8.1.6.2.2 Register Configuration**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|
| S0D2MMCKMCSR | R/W | H'E67F A0D0 | H'---- ---- | 32 |
| S0D2MMCKMECR | R/W | H'E67F A0D4 | H'00000000 | 32 |
| S0D2MMCKMLCH | R/W | H'E67F A0D8 | H'00000000 | 32 |
| S0D2MMCKMLCL | R/W | H'E67F A0DC | H'00000000 | 32 |
| S0D2MMCKMCNT | R | H'E67F A0E0 | H'00000000 | 32 |
| S0D2MMCKMCNTE | R | H'E67F A0E4 | H'00000000 | 32 |
| S0D4MMCKMCSR | R/W | H'E67F A110 | H'---- ---- | 32 |
| S0D4MMCKMECR | R/W | H'E67F A114 | H'00000000 | 32 |
| S0D4MMCKMLCH | R/W | H'E67F A118 | H'00000000 | 32 |
| S0D4MMCKMLCL | R/W | H'E67F A11C | H'00000000 | 32 |
| S0D4MMCKMCNT | R | H'E67F A120 | H'00000000 | 32 |
| S0D4MMCKMCNTE | R | H'E67F A124 | H'00000000 | 32 |
| CL16MMMCKMCSR | R/W | H'E67F A090 | H'---- ---- | 32 |
| CL16MMMCKMECR | R/W | H'E67F A094 | H'00000000 | 32 |
| CL16MMMCKMLCH | R/W | H'E67F A098 | H'00000000 | 32 |
| CL16MMMCKMLCL | R/W | H'E67F A09C | H'00000000 | 32 |
| CL16MMMCKMCNT | R | H'E67F A0A0 | H'00000000 | 32 |
| CL16MMMCKMCNTE | R | H'E67F A0A4 | H'00000000 | 32 |
| ZB3CKMCSR | R/W | H'E67F A130 | H'---- ---- | 32 |
| ZB3CKMECR | R/W | H'E67F A134 | H'00000000 | 32 |
| ZB3CKMLCH | R/W | H'E67F A138 | H'00000000 | 32 |
| ZB3CKMLCL | R/W | H'E67F A13C | H'00000000 | 32 |
| ZB3CKMCNT | R | H'E67F A140 | H'00000000 | 32 |
| ZB3CKMCNTE | R | H'E67F A144 | H'00000000 | 32 |
| ZB3D2CKMCSR | R/W | H'E67F A070 | H'---- ---- | 32 |
| ZB3D2CKMECR | R/W | H'E67F A074 | H'00000000 | 32 |
| ZB3D2CKMLCH | R/W | H'E67F A078 | H'00000000 | 32 |
| ZB3D2CKMLCL | R/W | H'E67F A07C | H'00000000 | 32 |
| ZB3D2CKMCNT | R | H'E67F A080 | H'00000000 | 32 |
| ZB3D2CKMCNTE | R | H'E67F A084 | H'00000000 | 32 |
| ZB3D4CKMCSR | R/W | H'E67F A150 | H'---- ---- | 32 |
| ZB3D4CKMECR | R/W | H'E67F A154 | H'00000000 | 32 |
| ZB3D4CKMLCH | R/W | H'E67F A158 | H'00000000 | 32 |
| ZB3D4CKMLCL | R/W | H'E67F A15C | H'00000000 | 32 |
| ZB3D4CKMCNT | R | H'E67F A160 | H'00000000 | 32 |
| ZB3D4CKMCNTE | R | H'E67F A164 | H'00000000 | 32 |
| CBFUSAMMCKMCSR | R/W | H'E67F A170 | H'---- ---- | 32 |
| CBFUSAMMCKMECR | R/W | H'E67F A174 | H'00000000 | 32 |
| CBFUSAMMCKMLCH | R/W | H'E67F A178 | H'00000000 | 32 |
| CBFUSAMMCKMLCL | R/W | H'E67F A17C | H'00000000 | 32 |
| CBFUSAMMCKMCNT | R | H'E67F A180 | H'00000000 | 32 |
| CBFUSAMMCKMCNTE | R | H'E67F A184 | H'00000000 | 32 |
| RCLKMMCKMCSR | R/W | H'E67F A190 | H'---- ---- | 32 |
| RCLKMMCKMECR | R/W | H'E67F A194 | H'00000000 | 32 |
| RCLKMMCKMLCH | R/W | H'E67F A198 | H'00000000 | 32 |
| RCLKMMCKMLCL | R/W | H'E67F A19C | H'00000000 | 32 |
| RCLKMMCKMCNT | R | H'E67F A1A0 | H'00000000 | 32 |
| RCLKMMCKMCNTE | R | H'E67F A1A4 | H'00000000 | 32 |
| MMCKMWPR | R/W | H'E67F A000 | H'00000000 | 32 |
| MMCKMWPCR | R/W | H'E67F A004 | H'00000001 | 32 |
| MMACKMSR | R | H'E67F A0AC | H'00000000 | 32 |

### (3) Clock Monitor in RT (Real Time) domain

- The base-address of CPG is H'E61F 0000. The address map for clock monitor in RT (Real Time) domain is assigned from H'E61F_0000 to H'E61F_3FFF.
- There are no protect registers (DxWACRA (x = 0, 1, 2, 3)) for the registers.

**Table 8.1.6.2.3a Register Configuration (ckmrt)**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|
| S0D2RTCKMCSR | R/W | H'E61F 01D0 | H'----_---- | 32 |
| S0D2RTCKMECR | R/W | H'E61F 01D4 | H'00000000 | 32 |
| S0D2RTCKMLCH | R/W | H'E61F 01D8 | H'00000000 | 32 |
| S0D2RTCKMLCL | R/W | H'E61F 01DC | H'00000000 | 32 |
| S0D2RTCKMCNT | R | H'E61F 01E0 | H'00000000 | 32 |
| S0D2RTCKMCNTE | R | H'E61F 01E4 | H'00000000 | 32 |
| S0D3RTCKMCSR | R/W | H'E61F 01F0 | H'----_---- | 32 |
| S0D3RTCKMECR | R/W | H'E61F 01F4 | H'00000000 | 32 |
| S0D3RTCKMLCH | R/W | H'E61F 01F8 | H'00000000 | 32 |
| S0D3RTCKMLCL | R/W | H'E61F 01FC | H'00000000 | 32 |
| S0D3RTCKMCNT | R | H'E61F 0200 | H'00000000 | 32 |
| S0D3RTCKMCNTE | R | H'E61F 0204 | H'00000000 | 32 |
| S0D4RTCKMCSR | R/W | H'E61F 0210 | H'----_---- | 32 |
| S0D4RTCKMECR | R/W | H'E61F 0214 | H'00000000 | 32 |
| S0D4RTCKMLCH | R/W | H'E61F 0218 | H'00000000 | 32 |
| S0D4RTCKMLCL | R/W | H'E61F 021C | H'00000000 | 32 |
| S0D4RTCKMCNT | R | H'E61F 0220 | H'00000000 | 32 |
| S0D4RTCKMCNTE | R | H'E61F 0224 | H'00000000 | 32 |
| S0D6RTCKMCSR | R/W | H'E61F 0230 | H'----_---- | 32 |
| S0D6RTCKMECR | R/W | H'E61F 0234 | H'00000000 | 32 |
| S0D6RTCKMLCH | R/W | H'E61F 0238 | H'00000000 | 32 |
| S0D6RTCKMLCL | R/W | H'E61F 023C | H'00000000 | 32 |
| S0D6RTCKMCNT | R | H'E61F 0240 | H'00000000 | 32 |
| S0D6RTCKMCNTE | R | H'E61F 0244 | H'00000000 | 32 |
| S0D24RTCKMCSR | R/W | H'E61F 0270 | H'----_---- | 32 |
| S0D24RTCKMECR | R/W | H'E61F 0274 | H'00000000 | 32 |
| S0D24RTCKMLCH | R/W | H'E61F 0278 | H'00000000 | 32 |
| S0D24RTCKMLCL | R/W | H'E61F 027C | H'00000000 | 32 |
| S0D24RTCKMCNT | R | H'E61F 0280 | H'00000000 | 32 |
| S0D24RTCKMCNTE | R | H'E61F 0284 | H'00000000 | 32 |
| CL16MRTCKMCSR | R/W | H'E61F 0190 | H'----_---- | 32 |
| CL16MRTCKMECR | R/W | H'E61F 0194 | H'00000000 | 32 |
| CL16MRTCKMLCH | R/W | H'E61F 0198 | H'00000000 | 32 |
| CL16MRTCKMLCL | R/W | H'E61F 019C | H'00000000 | 32 |
| CL16MRTCKMCNT | R | H'E61F 01A0 | H'00000000 | 32 |
| CL16MRTCKMCNTE | R | H'E61F 01A4 | H'00000000 | 32 |
| SASYNCRTCKMCSR | R/W | H'E61F 02D0 | H'----_---- | 32 |
| SASYNCRTCKMECR | R/W | H'E61F 02D4 | H'00000000 | 32 |
| SASYNCRTCKMLCH | R/W | H'E61F 02D8 | H'00000000 | 32 |
| SASYNCRTCKMLCL | R/W | H'E61F 02DC | H'00000000 | 32 |
| SASYNCRTCKMCNT | R | H'E61F 02E0 | H'00000000 | 32 |
| SASYNCRTCKMCNTE | R | H'E61F 02E4 | H'00000000 | 32 |
| CBFUSARTCKMCSR | R/W | H'E61F 02B0 | H'----_---- | 32 |
| CBFUSARTCKMECR | R/W | H'E61F 02B4 | H'00000000 | 32 |
| CBFUSARTCKMLCH | R/W | H'E61F 02B8 | H'00000000 | 32 |
| CBFUSARTCKMLCL | R/W | H'E61F 02BC | H'00000000 | 32 |
| CBFUSARTCKMCNT | R | H'E61F 02C0 | H'00000000 | 32 |
| CBFUSARTCKMCNTE | R | H'E61F 02C4 | H'00000000 | 32 |
| RCLKRTCKMCSR | R/W | H'E61F 0170 | H'----_---- | 32 |
| RCLKRTCKMECR | R/W | H'E61F 0174 | H'00000000 | 32 |
| RCLKRTCKMLCH | R/W | H'E61F 0178 | H'00000000 | 32 |
| RCLKRTCKMLCL | R/W | H'E61F 017C | H'00000000 | 32 |
| RCLKRTCKMCNT | R | H'E61F 0180 | H'00000000 | 32 |
| RCLKRTCKMCNTE | R | H'E61F 0184 | H'00000000 | 32 |
| RTCKMWPR | R/W | H'E61F 0000 | H'00000000 | 32 |
| RTCKMWPCR | R/W | H'E61F 0004 | H'00000001 | 32 |
| RTACKMSR | R | H'E61F 01AC | H'00000000 | 32 |

**Table 8.1.6.2.3b Register Configuration (ckmcr52)**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|
| S0D2RTCR52CKMCSR | R/W | H'E61F 20C0 | H'----_---- | 32 |
| S0D2RTCR52CKMECR | R/W | H'E61F 20C4 | H'00000000 | 32 |
| S0D2RTCR52CKMLCH | R/W | H'E61F 20C8 | H'00000000 | 32 |
| S0D2RTCR52CKMLCL | R/W | H'E61F 20CC | H'00000000 | 32 |
| S0D2RTCR52CKMCNT | R | H'E61F 20D0 | H'00000000 | 32 |
| S0D2RTCR52CKMCNTE | R | H'E61F 20D4 | H'00000000 | 32 |
| ZR0CR52CKMCSR | R/W | H'E61F 2010 | H'----_---- | 32 |
| ZR0CR52CKMECR | R/W | H'E61F 2014 | H'00000000 | 32 |
| ZR0CR52CKMLCH | R/W | H'E61F 2018 | H'00000000 | 32 |
| ZR0CR52CKMLCL | R/W | H'E61F 201C | H'00000000 | 32 |
| ZR0CR52CKMCNT | R | H'E61F 2020 | H'00000000 | 32 |
| ZR0CR52CKMCNTE | R | H'E61F 2024 | H'00000000 | 32 |
| ZR1CR52CKMCSR | R/W | H'E61F 2100 | H'----_---- | 32 |
| ZR1CR52CKMECR | R/W | H'E61F 2104 | H'00000000 | 32 |
| ZR1CR52CKMLCH | R/W | H'E61F 2108 | H'00000000 | 32 |
| ZR1CR52CKMLCL | R/W | H'E61F 210C | H'00000000 | 32 |
| ZR1CR52CKMCNT | R | H'E61F 2110 | H'00000000 | 32 |
| ZR1CR52CKMCNTE | R | H'E61F 2114 | H'00000000 | 32 |
| ZR2CR52CKMCSR | R/W | H'E61F 2120 | H'----_---- | 32 |
| ZR2CR52CKMECR | R/W | H'E61F 2124 | H'00000000 | 32 |
| ZR2CR52CKMLCH | R/W | H'E61F 2128 | H'00000000 | 32 |
| ZR2CR52CKMLCL | R/W | H'E61F 212C | H'00000000 | 32 |
| ZR2CR52CKMCNT | R | H'E61F 2130 | H'00000000 | 32 |
| ZR2CR52CKMCNTE | R | H'E61F 2134 | H'00000000 | 32 |
| CBFUSARTCR52CKMCSR | R/W | H'E61F 2090 | H'----_---- | 32 |
| CBFUSARTCR52CKMECR | R/W | H'E61F 2094 | H'00000000 | 32 |
| CBFUSARTCR52CKMLCH | R/W | H'E61F 2098 | H'00000000 | 32 |
| CBFUSARTCR52CKMLCL | R/W | H'E61F 209C | H'00000000 | 32 |
| CBFUSARTCR52CKMCNT | R | H'E61F 20A0 | H'00000000 | 32 |
| CBFUSARTCR52CKMCNTE | R | H'E61F 20A4 | H'00000000 | 32 |
| RCLKRTCR52CKMCSR | R/W | H'E61F 2070 | H'----_---- | 32 |
| RCLKRTCR52CKMECR | R/W | H'E61F 2074 | H'00000000 | 32 |
| RCLKRTCR52CKMLCH | R/W | H'E61F 2078 | H'00000000 | 32 |
| RCLKRTCR52CKMLCL | R/W | H'E61F 207C | H'00000000 | 32 |
| RCLKRTCR52CKMCNT | R | H'E61F 2080 | H'00000000 | 32 |
| RCLKRTCR52CKMCNTE | R | H'E61F 2084 | H'00000000 | 32 |
| RTCR52CKMWPR | R/W | H'E61F 2000 | H'00000000 | 32 |
| RTCR52CKMWPCR | R/W | H'E61F 2004 | H'00000001 | 32 |
| RTCR52ACKMSR | R | H'E61F 20B0 | H'00000000 | 32 |

### (4) Clock Monitor in Peripheral domain

- The base-address of CPG is H'E677 E000. The address map for clock monitor in Peripheral domain is assigned from H'E677 E000 to H'E677 FFFF.
- There are no protect registers (DxWACRA (x = 0, 1, 2, 3)) for the registers.

**Table 8.1.6.2.4 Register Configuration**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|
| S0D2PERCKMCSR | R/W | H'E677 E280 | H'----_---- | 32 |
| S0D2PERCKMECR | R/W | H'E677 E284 | H'00000000 | 32 |
| S0D2PERCKMLCH | R/W | H'E677 E288 | H'00000000 | 32 |
| S0D2PERCKMLCL | R/W | H'E677 E28C | H'00000000 | 32 |
| S0D2PERCKMCNT | R | H'E677 E290 | H'00000000 | 32 |
| S0D2PERCKMCNTE | R | H'E677 E294 | H'00000000 | 32 |
| S0D3PERCKMCSR | R/W | H'E677 E2A0 | H'----_---- | 32 |
| S0D3PERCKMECR | R/W | H'E677 E2A4 | H'00000000 | 32 |
| S0D3PERCKMLCH | R/W | H'E677 E2A8 | H'00000000 | 32 |
| S0D3PERCKMLCL | R/W | H'E677 E2AC | H'00000000 | 32 |
| S0D3PERCKMCNT | R | H'E677 E2B0 | H'00000000 | 32 |
| S0D3PERCKMCNTE | R | H'E677 E2B4 | H'00000000 | 32 |
| S0D4PERCKMCSR | R/W | H'E677 E2C0 | H'----_---- | 32 |
| S0D4PERCKMECR | R/W | H'E677 E2C4 | H'00000000 | 32 |
| S0D4PERCKMLCH | R/W | H'E677 E2C8 | H'00000000 | 32 |
| S0D4PERCKMLCL | R/W | H'E677 E2CC | H'00000000 | 32 |
| S0D4PERCKMCNT | R | H'E677 E2D0 | H'00000000 | 32 |
| S0D4PERCKMCNTE | R | H'E677 E2D4 | H'00000000 | 32 |
| S0D6PERCKMCSR | R/W | H'E677 E2E0 | H'----_---- | 32 |
| S0D6PERCKMECR | R/W | H'E677 E2E4 | H'00000000 | 32 |
| S0D6PERCKMLCH | R/W | H'E677 E2E8 | H'00000000 | 32 |
| S0D6PERCKMLCL | R/W | H'E677 E2EC | H'00000000 | 32 |
| S0D6PERCKMCNT | R | H'E677 E2F0 | H'00000000 | 32 |
| S0D6PERCKMCNTE | R | H'E677 E2F4 | H'00000000 | 32 |
| S0D12PERCKMCSR | R/W | H'E677 E300 | H'----_---- | 32 |
| S0D12PERCKMECR | R/W | H'E677 E304 | H'00000000 | 32 |
| S0D12PERCKMLCH | R/W | H'E677 E308 | H'00000000 | 32 |
| S0D12PERCKMLCL | R/W | H'E677 E30C | H'00000000 | 32 |
| S0D12PERCKMCNT | R | H'E677 E310 | H'00000000 | 32 |
| S0D12PERCKMCNTE | R | H'E677 E314 | H'00000000 | 32 |
| S0D24PERCKMCSR | R/W | H'E677 E340 | H'----_---- | 32 |
| S0D24PERCKMECR | R/W | H'E677 E344 | H'00000000 | 32 |
| S0D24PERCKMLCH | R/W | H'E677 E348 | H'00000000 | 32 |
| S0D24PERCKMLCL | R/W | H'E677 E34C | H'00000000 | 32 |
| S0D24PERCKMCNT | R | H'E677 E350 | H'00000000 | 32 |
| S0D24PERCKMCNTE | R | H'E677 E354 | H'00000000 | 32 |
| CL16MPERCKMCSR | R/W | H'E677 E1B0 | H'----_---- | 32 |
| CL16MPERCKMECR | R/W | H'E677 E1B4 | H'00000000 | 32 |
| CL16MPERCKMLCH | R/W | H'E677 E1B8 | H'00000000 | 32 |
| CL16MPERCKMLCL | R/W | H'E677 E1BC | H'00000000 | 32 |
| CL16MPERCKMCNT | R | H'E677 E1C0 | H'00000000 | 32 |
| CL16MPERCKMCNTE | R | H'E677 E1C4 | H'00000000 | 32 |
| SD0HCKMCSR | R/W | H'E677 E090 | H'----_---- | 32 |
| SD0HCKMECR | R/W | H'E677 E094 | H'00000000 | 32 |
| SD0HCKMLCH | R/W | H'E677 E098 | H'00000000 | 32 |
| SD0HCKMLCL | R/W | H'E677 E09C | H'00000000 | 32 |
| SD0HCKMCNT | R | H'E677 E0A0 | H'00000000 | 32 |
| SD0HCKMCNTE | R | H'E677 E0A4 | H'00000000 | 32 |
| SD0CKMCSR | R/W | H'E677 E0B0 | H'----_---- | 32 |
| SD0CKMECR | R/W | H'E677 E0B4 | H'00000000 | 32 |
| SD0CKMLCH | R/W | H'E677 E0B8 | H'00000000 | 32 |
| SD0CKMLCL | R/W | H'E677 E0BC | H'00000000 | 32 |
| SD0CKMCNT | R | H'E677 E0C0 | H'00000000 | 32 |
| SD0CKMCNTE | R | H'E677 E0C4 | H'00000000 | 32 |
| RPCCKMCSR | R/W | H'E677 E1F0 | H'----_---- | 32 |
| RPCCKMECR | R/W | H'E677 E1F4 | H'00000000 | 32 |
| RPCCKMLCH | R/W | H'E677 E1F8 | H'00000000 | 32 |
| RPCCKMLCL | R/W | H'E677 E1FC | H'00000000 | 32 |
| RPCCKMCNT | R | H'E677 E200 | H'00000000 | 32 |
| RPCCKMCNTE | R | H'E677 E204 | H'00000000 | 32 |

| | | | | |
|---|---|---|---|---|
| RPCD2CKMCSR | R/W | H'E677 E210 | H'----_---- | 32 |
| RPCD2CKMECR | R/W | H'E677 E214 | H'00000000 | 32 |
| RPCD2CKMLCH | R/W | H'E677 E218 | H'00000000 | 32 |
| RPCD2CKMLCL | R/W | H'E677 E21C | H'00000000 | 32 |
| RPCD2CKMCNT | R | H'E677 E220 | H'00000000 | 32 |
| RPCD2CKMCNTE | R | H'E677 E224 | H'00000000 | 32 |
| CANFDCKMCSR | R/W | H'E677 E1D0 | H'----_---- | 32 |
| CANFDCKMECR | R/W | H'E677 E1D4 | H'00000000 | 32 |
| CANFDCKMLCH | R/W | H'E677 E1D8 | H'00000000 | 32 |
| CANFDCKMLCL | R/W | H'E677 E1DC | H'00000000 | 32 |
| CANFDCKMCNT | R | H'E677 E1E0 | H'00000000 | 32 |
| CANFDCKMCNTE | R | H'E677 E1E4 | H'00000000 | 32 |
| MSOCKMCSR | R/W | H'E677 E130 | H'----_---- | 32 |
| MSOCKMECR | R/W | H'E677 E134 | H'00000000 | 32 |
| MSOCKMLCH | R/W | H'E677 E138 | H'00000000 | 32 |
| MSOCKMLCL | R/W | H'E677 E13C | H'00000000 | 32 |
| MSOCKMCNT | R | H'E677 E140 | H'00000000 | 32 |
| MSOCKMCNTE | R | H'E677 E144 | H'00000000 | 32 |
| FRAYCKMCSR | R/W | H'E677 E170 | H'----_---- | 32 |
| FRAYCKMECR | R/W | H'E677 E174 | H'00000000 | 32 |
| FRAYCKMLCH | R/W | H'E677 E178 | H'00000000 | 32 |
| FRAYCKMLCL | R/W | H'E677 E17C | H'00000000 | 32 |
| FRAYCKMCNT | R | H'E677 E180 | H'00000000 | 32 |
| FRAYCKMCNTE | R | H'E677 E184 | H'00000000 | 32 |
| ADGHCKMCSR | R/W | H'E677 E360 | H'----_---- | 32 |
| ADGHCKMECR | R/W | H'E677 E364 | H'00000000 | 32 |
| ADGHCKMLCH | R/W | H'E677 E368 | H'00000000 | 32 |
| ADGHCKMLCL | R/W | H'E677 E36C | H'00000000 | 32 |
| ADGHCKMCNT | R | H'E677 E370 | H'00000000 | 32 |
| ADGHCKMCNTE | R | H'E677 E374 | H'00000000 | 32 |
| SASYNCPERD1CKMCSR | R/W | H'E677 E3A0 | H'----_---- | 32 |
| SASYNCPERD1CKMECR | R/W | H'E677 E3A4 | H'00000000 | 32 |
| SASYNCPERD1CKMLCH | R/W | H'E677 E3A8 | H'00000000 | 32 |
| SASYNCPERD1CKMLCL | R/W | H'E677 E3AC | H'00000000 | 32 |
| SASYNCPERD1CKMCNT | R | H'E677 E3B0 | H'00000000 | 32 |
| SASYNCPERD1CKMCNTE | R | H'E677 E3B4 | H'00000000 | 32 |
| SASYNCPERD2CKMCSR | R/W | H'E677 E3C0 | H'----_---- | 32 |
| SASYNCPERD2CKMECR | R/W | H'E677 E3C4 | H'00000000 | 32 |
| SASYNCPERD2CKMLCH | R/W | H'E677 E3C8 | H'00000000 | 32 |
| SASYNCPERD2CKMLCL | R/W | H'E677 E3CC | H'00000000 | 32 |
| SASYNCPERD2CKMCNT | R | H'E677 E3D0 | H'00000000 | 32 |
| SASYNCPERD2CKMCNTE | R | H'E677 E3D4 | H'00000000 | 32 |
| SASYNCPERD4CKMCSR | R/W | H'E677 E3E0 | H'----_---- | 32 |
| SASYNCPERD4CKMECR | R/W | H'E677 E3E4 | H'00000000 | 32 |
| SASYNCPERD4CKMLCH | R/W | H'E677 E3E8 | H'00000000 | 32 |
| SASYNCPERD4CKMLCL | R/W | H'E677 E3EC | H'00000000 | 32 |
| SASYNCPERD4CKMCNT | R | H'E677 E3F0 | H'00000000 | 32 |
| SASYNCPERD4CKMCNTE | R | H'E677 E3F4 | H'00000000 | 32 |
| CBFUSAPERCKMCSR | R/W | H'E677 E380 | H'----_---- | 32 |
| CBFUSAPERCKMECR | R/W | H'E677 E384 | H'00000000 | 32 |
| CBFUSAPERCKMLCH | R/W | H'E677 E388 | H'00000000 | 32 |
| CBFUSAPERCKMLCL | R/W | H'E677 E38C | H'00000000 | 32 |
| CBFUSAPERCKMCNT | R | H'E677 E390 | H'00000000 | 32 |
| CBFUSAPERCKMCNTE | R | H'E677 E394 | H'00000000 | 32 |
| RCLKPERCKMCSR | R/W | H'E677 E150 | H'----_---- | 32 |
| RCLKPERCKMECR | R/W | H'E677 E154 | H'00000000 | 32 |
| RCLKPERCKMLCH | R/W | H'E677 E158 | H'00000000 | 32 |
| RCLKPERCKMLCL | R/W | H'E677 E15C | H'00000000 | 32 |
| RCLKPERCKMCNT | R | H'E677 E160 | H'00000000 | 32 |
| RCLKPERCKMCNTE | R | H'E677 E164 | H'00000000 | 32 |
| PERCKMWPR | R/W | H'E677 E000 | H'00000000 | 32 |
| PERCKMWPCR | R/W | H'E677 E004 | H'00000001 | 32 |
| PERACKMSR | R | H'E677 E250 | H'00000000 | 32 |

### (5) Clock Monitor in HSC (High speed Communication) domain

- The base-address of CPG is H'E644 E000. The address map for clock monitor in HSC (High speed Communication) domain is assigned from H'E644 E000 to H'E644 FFFF.
- There are no protect registers (DxWACRA (x = 0, 1, 2, 3)) for the registers.

**Table 8.1.6.2.5 Register Configuration**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|
| S0D1HSCCKMCSR | R/W | H'E644 E150 | H'----_---- | 32 |
| S0D1HSCCKMECR | R/W | H'E644 E154 | H'00000000 | 32 |
| S0D1HSCCKMLCH | R/W | H'E644 E158 | H'00000000 | 32 |
| S0D1HSCCKMLCL | R/W | H'E644 E15C | H'00000000 | 32 |
| S0D1HSCCKMCNT | R | H'E644 E160 | H'00000000 | 32 |
| S0D1HSCCKMCNTE | R | H'E644 E164 | H'00000000 | 32 |
| S0D2HSCCKMCSR | R/W | H'E644 E170 | H'----_---- | 32 |
| S0D2HSCCKMECR | R/W | H'E644 E174 | H'00000000 | 32 |
| S0D2HSCCKMLCH | R/W | H'E644 E178 | H'00000000 | 32 |
| S0D2HSCCKMLCL | R/W | H'E644 E17C | H'00000000 | 32 |
| S0D2HSCCKMCNT | R | H'E644 E180 | H'00000000 | 32 |
| S0D2HSCCKMCNTE | R | H'E644 E184 | H'00000000 | 32 |
| S0D4HSCCKMCSR | R/W | H'E644 E1B0 | H'----_---- | 32 |
| S0D4HSCCKMECR | R/W | H'E644 E1B4 | H'00000000 | 32 |
| S0D4HSCCKMLCH | R/W | H'E644 E1B8 | H'00000000 | 32 |
| S0D4HSCCKMLCL | R/W | H'E644 E1BC | H'00000000 | 32 |
| S0D4HSCCKMCNT | R | H'E644 E1C0 | H'00000000 | 32 |
| S0D4HSCCKMCNTE | R | H'E644 E1C4 | H'00000000 | 32 |
| S0D8HSCCKMCSR | R/W | H'E644 E1D0 | H'----_---- | 32 |
| S0D8HSCCKMECR | R/W | H'E644 E1D4 | H'00000000 | 32 |
| S0D8HSCCKMLCH | R/W | H'E644 E1D8 | H'00000000 | 32 |
| S0D8HSCCKMLCL | R/W | H'E644 E1DC | H'00000000 | 32 |
| S0D8HSCCKMCNT | R | H'E644 E1E0 | H'00000000 | 32 |
| S0D8HSCCKMCNTE | R | H'E644 E1E4 | H'00000000 | 32 |
| CL16MHSCCKMCSR | R/W | H'E644 E130 | H'----_---- | 32 |
| CL16MHSCCKMECR | R/W | H'E644 E134 | H'00000000 | 32 |
| CL16MHSCCKMLCH | R/W | H'E644 E138 | H'00000000 | 32 |
| CL16MHSCCKMLCL | R/W | H'E644 E13C | H'00000000 | 32 |
| CL16MHSCCKMCNT | R | H'E644 E140 | H'00000000 | 32 |
| CL16MHSCCKMCNTE | R | H'E644 E144 | H'00000000 | 32 |
| CBFUSAHSCCKMCSR | R/W | H'E644 E2F0 | H'----_---- | 32 |
| CBFUSAHSCCKMECR | R/W | H'E644 E2F4 | H'00000000 | 32 |
| CBFUSAHSCCKMLCH | R/W | H'E644 E2F8 | H'00000000 | 32 |
| CBFUSAHSCCKMLCL | R/W | H'E644 E2FC | H'00000000 | 32 |
| CBFUSAHSCCKMCNT | R | H'E644 E300 | H'00000000 | 32 |
| CBFUSAHSCCKMCNTE | R | H'E644 E304 | H'00000000 | 32 |
| RCLKHSCCKMCSR | R/W | H'E644 E0B0 | H'----_---- | 32 |
| RCLKHSCCKMECR | R/W | H'E644 E0B4 | H'00000000 | 32 |
| RCLKHSCCKMLCH | R/W | H'E644 E0B8 | H'00000000 | 32 |
| RCLKHSCCKMLCL | R/W | H'E644 E0BC | H'00000000 | 32 |
| RCLKHSCCKMCNT | R | H'E644 E0C0 | H'00000000 | 32 |
| RCLKHSCCKMCNTE | R | H'E644 E0C4 | H'00000000 | 32 |
| HSCCKMWPR | R/W | H'E644 E000 | H'00000000 | 32 |
| HSCCKMWPCR | R/W | H'E644 E004 | H'00000001 | 32 |
| HSCACKMSR | R | H'E644 E14C | H'00000000 | 32 |

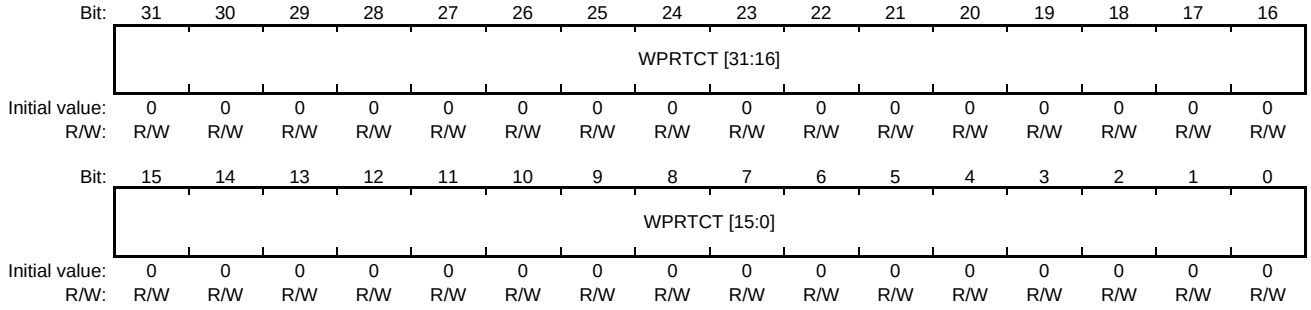
## 8.2 Register Description

Legend for Register Description
Initial value : Register value after a reset
– : Undefined value
R/W : Readable/writable. The written value can be read.
R : Read-only.
W : Write-only. The read value is undefined.

(Under construction)

### 8.2.1 CPG Write Protect Register (CPGWPR)

CPGWPR is a 32-bit readable/writable register.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | WPRTCT [31:0] | All 0 | R/W | Writing a value to CPG registers is enabled by writing the inverse of the value to this register. |

### 8.2.2 Backup Buffer Access Protect Register (BKBAPR)

BKBAPR is a 32-bit readable/writable register. The usage method is described in 37.3.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | BKB APR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | - | All 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 0 | BKBAPR | 0 | R/W | Enables or disables protection of the data storage in the Backup Buffer.<br><br>0: Enables protection of the data storage in the Backup Buffer.<br>1: Disables protection of the data storage in the Backup Buffer.<br>Note) Don't access the Backup Buffer while protection of the stored data is enabled (BKBAPR = 1'b0). |

### 8.2.3 Functional Safety Reset Check Register A (FSRCHKRAn (n = 0 to 29))

FSRCHKRAn is a 32-bit readable register. Each bit of these registers can be set by writing 1 to the corresponding bit of FSRCHKSETRn. Each bit is cleared by the reset signal of the module assigned to each bit. Each bit can be also cleared by writing 1 to the corresponding bit of FSRCHKCLRRn. Unknown value is read from reserved bits. These registers can be used to check whether the reset signal of each module is issued or not for the purpose of functional safety on the reset function.

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

### 8.2.4 Functional Safety Reset Check Register B (FSRCHKRBn (n= 0 to 29))

FSRCHKRBn is a 32-bit readable register. Each bit of these registers can be set by writing 1 to the corresponding bit of FSRCHKSETRn. Each bit is cleared by the reset signal of the module assigned to each bit. Each bit can be also cleared by writing 1 to the corresponding bit of FSRCHKCLRRn. Unknown value is read from reserved bits. These registers can be used to check whether the reset signal of each module is issued or not for the purpose of functional safety on the reset function.

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

### 8.2.5 Functional Safety Reset Check Set Register (FSRCHKSETRn (n = 0 to 29))

FSRCHKSETRn is a 32-bit writable register. When writing 1 to the designated bit of FSRCHKSETRn, the corresponding bits of both FSRCHKRAn and FSRCHKRBn are set to 1. Writing 0 to each bit of FSRCHKSETRn does not change the values of FSRCHKRAn and FSRCHKRBn.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSET 31 | RSET 30 | RSET 29 | RSET 28 | RSET 27 | RSET 26 | RSET 25 | RSET 24 | RSET 23 | RSET 22 | RSET 21 | RSET 20 | RSET 19 | RSET 18 | RSET 17 | RSET 16 |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSET 15 | RSET 14 | RSET 13 | RSET 12 | RSET 11 | RSET 10 | RSET 09 | RSET 08 | RSET 07 | RSET 06 | RSET 05 | RSET 04 | RSET 03 | RSET 02 | RSET 01 | RSET 00 |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | RSETm (31 to 0) | - | W | When writing 1 to RSETm of FSRCHKSETRn, RCHKm bits of both FSRCHKRAn and FSRCHKRBn are set to 1. Writing 0 to each bit of FSRCHKSETRn does not change the values of FSRCHKRAn and FSRCHKRBn. |

### 8.2.6 Functional Safety Reset Check Clear Register (FSRCHKCLRRn (n = 0 to 29))

- FSRCHKCLRRn is a 32-bit writable register. When writing 1 to the designated bit of FSRCHKCLRRn, the corresponding bits of both FSRCHKRAn and FSRCHKRBn are cleared to 0. Writing 0 to each bit of FSRCHKCLRRn does not change the values of FSRCHKRAn and FSRCHKRBn. Some modules may have two or more reset signals. In such cases, the reset signals correspond to multiple bits of the FSRCHKRAn and FSRCHKRBn registers.

- Some modules have two or more reset signals. In such cases, the reset signals correspond with multiple bits of the FSRCHKRAn and FSCHKRBn.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RCLR 31 | RCLR 30 | RCLR 29 | RCLR 28 | RCLR 27 | RCLR 26 | RCLR 25 | RCLR 24 | RCLR 23 | RCLR 22 | RCLR 21 | RCLR 20 | RCLR 19 | RCLR 18 | RCLR 17 | RCLR 16 |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RCLR 15 | RCLR 14 | RCLR 13 | RCLR 12 | RCLR 11 | RCLR 10 | RCLR 09 | RCLR 08 | RCLR 07 | RCLR 06 | RCLR 05 | RCLR 04 | RCLR 03 | RCLR 02 | RCLR 01 | RCLR 00 |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | RCLRm (31 to 0) | - | W | When writing 1 to RCLRm of FSRCHKCLRRn, RCHKm bits of both FSRCHKRAn and FSRCHKRBn are cleared to 0. Writing 0 to each bit of FSRCHKCLRRn does not change the values of FSRCHKRAn and FSRCHKRBn. |

### 8.2.7 Frequency Control Register B (FRQCRB)

FRQCRB is a 32-bit readable/writable register. This register specifies the frequency division ratios of Debug Trace port clock (ZTRφ), Debug Trace bus clock (ZTφ), and Debug clock (ZSφ), 3D Graphics Engine clock (ZGφ), Cortex-A76 Sub System clock (ZDφ).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KICK | - | ZGSTP | ZGFC [4:0] | | | | | ZTRFC [3:0] | | | | ZTFC [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| R/W: | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ZSFC [3:0] | | | | - | - | - | - | - | - | - | ZDSTP | - | - | - | - |
| Initial value: | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | KICK | 0 | R/W | KICK bit. Setting 1 to this register activates the FRQCRB, FRQCRC0, FRQCRC1 setting.<br><br>- 1'b0: Does not activate the FRQCRB, FRQCRC0, FRQCRC1 settings.<br>- 1'b1: Activates the FRQCRB, FRQCRC0, FRQCRC1 settings.<br>This bit is automatically cleared to 0 when the frequency division setting is completed after 1 is written to this bit. |
| 30 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 29 | ZGSTP | 0 | R/W | Stop control bit for 3D Graphics Engine clock (ZGφ).<br><br>- 1'b0: Supplies ZGφ<br>- 1'b1: Stops ZGφ. |
| 28 to 24 | ZGFC [4:0] | 5'b10000 | R/W | 3D Graphics Engine clock (ZGφ) Frequency division ratio. As shown in Table 8.1.4d, the Maximum clock frequency is 1000 MHz. ZGFC[4:0] shall be set so that ZGφ frequency doesn't exceed 1000 MHz.<br><br>5'b00000 → ZGφ = (PLL4VCO x 1/2) x 32/32<br>5'b00001 → ZGφ = (PLL4VCO x 1/2) x 31/32<br>-----<br>5'b10000 → ZGφ = (PLL4VCO x 1/2) x 16/32<br>-----<br>-----<br>5'b11110 → ZGφ = (PLL4VCO x 1/2) x 2/32<br>5'b11111 → ZGφ = (PLL4VCO x 1/2) x 1/32 |
| 23 to 20 | ZTRFC [3:0] | 4'b0001 | R/W | Debug Trace port clock (ZTRφ) Frequency division ratio. As shown in Table 8.1.4a, the Maximum clock frequency is 533.33 MHz. ZTRFC[3:0] should be set so that ZTRφ frequency doesn't exceed 533.33 MHz.<br>ZTRFC[3:0] = 4'b0000: ZTRφ frequency = (PLL1VCO x 1/2) x 1/2<br>ZTRFC[3:0] = 4'b0001: ZTRφ frequency = (PLL1VCO x 1/2) x 1/3<br>ZTRFC[3:0] = 4'b0010: ZTRφ frequency = (PLL1VCO x 1/2) x 1/4<br>ZTRFC[3:0] = 4'b0011: ZTRφ frequency = (PLL1VCO x 1/2) x 1/6<br>ZTRFC[3:0] = 4'b0100: ZTRφ frequency = (PLL1VCO x 1/2) x 1/8<br>ZTRFC[3:0] = 4'b0101: ZTRφ frequency = (PLL1VCO x 1/2) x 1/12<br>ZTRFC[3:0] = 4'b0110: ZTRφ frequency = (PLL1VCO x 1/2) x 1/16<br>Other setting is prohibited. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 19 to 16 | ZTFC [3:0] | 4'b0001 | R/W | Debug Trace bus clock (ZTφ) Frequency division ratio. As shown in Table 8.1.4a, the Maximum clock frequency is 533.33 MHz. ZTFC[3:0] should be set so that ZTφ frequency doesn't exceed 533.33 MHz.<br>ZTFC[3:0] = 4'b0000: ZTφ frequency = (PLL1VCO x 1/2) x 1/2<br>ZTFC[3:0] = 4'b0001: ZTφ frequency = (PLL1VCO x 1/2) x 1/3<br>ZTFC[3:0] = 4'b0010: ZTφ frequency = (PLL1VCO x 1/2) x 1/4<br>ZTFC[3:0] = 4'b0011: ZTφ frequency = (PLL1VCO x 1/2) x 1/6<br>ZTFC[3:0] = 4'b0100: ZTφ frequency = (PLL1VCO x 1/2) x 1/8<br>ZTFC[3:0] = 4'b0101: ZTφ frequency = (PLL1VCO x 1/2) x 1/12<br>ZTFC[3:0] = 4'b0110: ZTφ frequency = (PLL1VCO x 1/2) x 1/16<br>Other setting is prohibited. |
| 15 to 12 | ZSFC [3:0] | 4'b0011 | R/W | Debug Trace bus clock (ZSφ) Frequency division ratio. As shown in Table 8.1.4a, the Maximum clock frequency is 266.66 MHz. ZSFC[3:0] should be set so that ZSφ frequency doesn't exceed 266.66 MHz.<br>ZSFC[3:0] = 4'b0000: ZSφ frequency = (PLL1VCO x 1/2) x 1/2<br>ZSFC[3:0] = 4'b0001: ZSφ frequency = (PLL1VCO x 1/2) x 1/3<br>ZSFC[3:0] = 4'b0010: ZSφ frequency = (PLL1VCO x 1/2) x 1/4<br>ZSFC[3:0] = 4'b0011: ZSφ frequency = (PLL1VCO x 1/2) x 1/6<br>ZSFC[3:0] = 4'b0100: ZSφ frequency = (PLL1VCO x 1/2) x 1/8<br>ZSFC[3:0] = 4'b0101: ZSφ frequency = (PLL1VCO x 1/2) x 1/12<br>ZSFC[3:0] = 4'b0110: ZSφ frequency = (PLL1VCO x 1/2) x 1/16<br>Other setting is prohibited. |
| 11 to 5 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 4 | ZDSTP | 0 | R/W | Stop control bit for Cortex-A76 Sub system clock (ZDφ).<br>- 1'b0: Supplies ZDφ<br>- 1'b1: Stops ZDφ. |
| 3 to 0 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### 8.2.8 Frequency Control Register C0 (FRQCRC0)

FRQCRC0 is a 32-bit readable/writable register. This register specifies the frequency division ratios of Cortex-R52 clock (ZR0φ, ZR1φ), Cortex-A76 clock (ZC0φ, ZC1φ). When division ratio or stop control of these clocks is changed, any other clock division ratio or stop control should be changed by FRQCRB register before FRQCRB.KICK bit is set to 1.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | ZR1STP | ZR1FC [4:0] | | | | | - | - | - | ZR0FC [4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

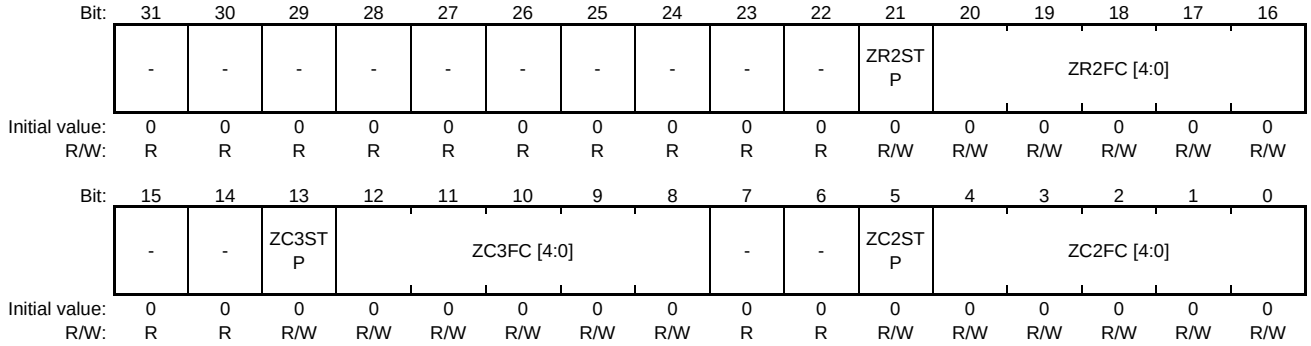
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | ZC1STP | ZC1FC [4:0] | | | | | - | - | ZC0STP | ZC0FC [4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 29 | ZR1STP | 0 | R/W | Clock Stop for ZR1φ.<br>0: Supplies clock to Cortex-R52 Core1.<br>1: Stops clock to Cortex-R52 Core1. |
| 28 to 24 | ZR1FC [4:0] | 5'b00000 | R/W | Cortex-R52 Clock (ZR1φ) frequency division ratio. As shown in Table 8.1.4f, the Maximum clock frequency is 1400 MHz. ZR1FC[4:0] shall be set so that ZR1φ frequency doesn't exceed 1400 MHz.<br><br>5'b00000 → ZR1φ = (PLL6VCO x 1/2) x 32/32<br>5'b00001 → ZR1φ = (PLL6VCO x 1/2) x 31/32<br>-----<br>5'b10000 → ZR1φ = (PLL6VCO x 1/2) x 16/32<br>-----<br>-----<br>5'b11110 → ZR1φ = (PLL6VCO x 1/2) x 2/32<br>5'b11111 → ZR1φ = (PLL6VCO x 1/2) x 1/32 |
| 23 to 21 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 to 16 | ZR0FC [4:0] | 5'b00000 | R/W | Cortex-R52 Clock (ZR0φ) frequency division ratio. As shown in Table 8.1.4f, the Maximum clock frequency is 1400 MHz. ZR0FC[4:0] shall be set so that ZR0φ frequency doesn't exceed 1400 MHz.<br><br>5'b00000 → ZR0φ = (PLL6VCO x 1/2) x 32/32<br>5'b00001 → ZR0φ = (PLL6VCO x 1/2) x 31/32<br>-----<br>5'b10000 → ZR0φ = (PLL6VCO x 1/2) x 16/32<br>-----<br>-----<br>5'b11110 → ZR0φ = (PLL6VCO x 1/2) x 2/32<br>5'b11111 → ZR0φ = (PLL6VCO x 1/2) x 1/32 |
| 15 to 14 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 13 | ZC1STP | 0 | R/W | Clock Stop for ZC1φ.<br>0: Supplies clock to Cortex-A76 Core1.<br>1: Stops clock to Cortex-A76 Core1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 8 | ZC1FC [4:0] | 5'b00000 | R/W | Cortex-A76 Clock (ZC1φ) frequency division ratio. As shown in Table 8.1.4b, the Maximum clock frequency is 1000 MHz. ZC1FC[4:0] shall be set so that ZC1φ frequency doesn't exceed 1000 MHz.<br><br>5'b00000 → ZC1φ = (PLL2VCO x 1/4) x 32/32<br>5'b00001 → ZC1φ = (PLL2VCO x 1/4) x 31/32<br>-----<br>5'b10000 → ZC1φ = (PLL2VCO x 1/4) x 16/32<br>-----<br>-----<br>5'b11110 → ZC1φ = (PLL2VCO x 1/4) x 2/32<br>5'b11111 → ZC1φ = (PLL2VCO x 1/4) x 1/32 |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 | ZC0STP | 0 | R/W | Clock Stop for ZC0φ.<br>0: Supplies clock to Cortex-A76 Core0.<br>1: Stops clock to Cortex-A76 Core0. |
| 4 to 0 | ZC0FC [4:0] | 5'b00000 | R/W | Cortex-A76 Clock (ZC0φ) frequency division ratio. As shown in Table 8.1.4b, the Maximum clock frequency is 1000 MHz. ZC0FC[4:0] shall be set so that ZC0φ frequency doesn't exceed 1000 MHz.<br><br>5'b00000 → ZC0φ = (PLL2VCO x 1/4) x 32/32<br>5'b00001 → ZC0φ = (PLL2VCO x 1/4) x 31/32<br>-----<br>5'b10000 → ZC0φ = (PLL2VCO x 1/4) x 16/32<br>-----<br>-----<br>5'b11110 → ZC0φ = (PLL2VCO x 1/4) x 2/32<br>5'b11111 → ZC0φ = (PLL2VCO x 1/4) x 1/32 |

### 8.2.9 Frequency Control Register C1 (FRQCRC1)

FRQCRC1 is a 32-bit readable/writable register. This register specifies the frequency division ratios of Cortex-R52 clock (ZR2φ), Cortex-A76 clock (ZC2φ, ZC3φ). When division ratio or stop control of these clocks is changed, any other clock division ratio or stop control should be changed by FRQCRB register before FRQCRB.KICK bit is set to 1.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 22 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 21 | ZR2STP | 0 | R/W | Clock Stop for ZR2φ.<br>0: Supplies clock to Cortex-R52 Core2.<br>1: Stops clock to Cortex-R52 Core2. |
| 20 to 16 | ZR2FC [4:0] | 5'b00000 | R/W | Cortex-R52 Clock (ZR2φ) frequency division ratio. As shown in Table 8.1.4f, the Maximum clock frequency is 1400 MHz. ZR2FC[4:0] shall be set so that ZR2φ frequency doesn't exceed 1400 MHz<br><br>5'b00000 → ZR2φ = (PLL6VCO x 1/2) x 32/32<br>5'b00001 → ZR2φ = (PLL6VCO x 1/2) x 31/32<br>-----<br>5'b10000 → ZR2φ = (PLL6VCO x 1/2) x 16/32<br>-----<br>-----<br>5'b11110 → ZR2φ = (PLL6VCO x 1/2) x 2/32<br>5'b11111 → ZR2φ = (PLL6VCO x 1/2) x 1/32 |
| 15 to 14 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 13 | ZC3STP | 0 | R/W | Clock Stop for ZC3φ.<br>0: Supplies clock to Cortex-A76 Core3.<br>1: Stops clock to Cortex-A76 Core3. |
| 12 to 8 | ZC3FC [4:0] | 5'b00000 | R/W | Cortex-A76 Clock (ZC3φ) frequency division ratio. As shown in Table 8.1.4b, the Maximum clock frequency is 1000 MHz. ZC3FC[4:0] shall be set so that ZC3φ frequency doesn't exceed 1000 MHz.<br><br>5'b00000 → ZC3φ = (PLL2VCO x 1/4) x 32/32<br>5'b00001 → ZC3φ = (PLL2VCO x 1/4) x 31/32<br>-----<br>5'b10000 → ZC3φ = (PLL2VCO x 1/4) x 16/32<br>-----<br>-----<br>5'b11110 → ZC3φ = (PLL2VCO x 1/4) x 2/32<br>5'b11111 → ZC3φ = (PLL2VCO x 1/4) x 1/32 |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 | ZC2STP | 0 | R/W | Clock Stop for ZC2φ.<br>0: Supplies clock to Cortex-A76 Core2.<br>1: Stops clock to Cortex-A76 Core2. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 to 0 | ZC2FC [4:0] | 5'b00000 | R/W | Cortex-A76 Clock (ZC2φ) frequency division ratio. As shown in Table 8.1.4b, the Maximum clock frequency is 1000 MHz. ZC2FC[4:0] shall be set so that ZC2φ frequency doesn't exceed 1000 MHz.<br><br>5'b00000 → ZC2φ = (PLL2VCO x 1/4) x 32/32<br>5'b00001 → ZC2φ = (PLL2VCO x 1/4) x 31/32<br>-----<br>5'b10000 → ZC2φ = (PLL2VCO x 1/4) x 16/32<br>-----<br>-----<br>5'b11110 → ZC2φ = (PLL2VCO x 1/4) x 2/32<br>5'b11111 → ZC2φ = (PLL2VCO x 1/4) x 1/32 |

### 8.2.10 Frequency Control Register D (FRQCRD)

FRQCRD0 is a 32-bit readable/writable register. This register specifies the DDR and DBSC clock (ZB3φ, ZB3D2φ, ZB3D4φ) frequency.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KICK | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | ZB3FC [4:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | KICK | 0 | R/W | KICK bit. Setting 1 to this register activates the FRQCRD setting.<br><br>- 1'b0: Does not activate the FRQCRD settings.<br>- 1'b1: Activates the FRQCRD settings.<br>This bit is automatically cleared to 0 when the frequency division setting is completed after 1 is written to this bit. |
| 30 to 4 | - | All 0 | R/W | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 3 to 0 | ZB3FC [4:0] | 4'b0010 | R/W | ZB3φ, ZB3D2φ, ZB3D4φ clock for DDR/DBSC frequency division ratio is set (*1). |

| ZB3FC [4:0] | ZB3φ [MHz] | ZB3D2φ [MHz] | ZB3D4φ [MHz] |
|---|---|---|---|
| 0000 | (PLL3VCO x 1/2) x 1/2 | (PLL3VCO x 1/2) x 1/4 | (PLL3VCO x 1/2) x 1/8 |
| 0010 | (PLL3VCO x 1/2) x 1/4 | (PLL3VCO x 1/2) x 1/8 | (PLL3VCO x 1/2) x 1/16 |
| 0100 | (PLL3VCO x 1/2) x 1/8 | (PLL3VCO x 1/2) x 1/16 | (PLL3VCO x 1/2) x 1/32 |
| 0110 | (PLL3VCO x 1/2) x 1/16 | (PLL3VCO x 1/2) x 1/32 | (PLL3VCO x 1/2) x 1/64 |
| 1001 | (PLL3VCO x 1/2) x 1/36 | (PLL3VCO x 1/2) x 1/64 | (PLL3VCO x 1/2) x 1/128 |
| 1100 | (PLL3VCO x 1/2) x 1/64 | (PLL3VCO x 1/2) x 1/128 | (PLL3VCO x 1/2) x 1/256 |
| Others | Setting prohibited | | |

Note 1. The frequency depends on PLL3VCO frequency. See 8.1.5.3 about PLL3VCO frequency specification.

### 8.2.11 PLL Enable Control Register (PLLECR)

PLLECR is a 32-bit readable/writable register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | PLL7ST | - | - | - | - | - | - | - | PLL7E |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PLL6ST | PLL5ST | PLL4ST | - | PLL3ST | - | PLL2ST | PLL1ST | PLL6E | PLL5E | PLL4E | - | PLL3E | - | PLL2E | PLL1E |
| Initial value: | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 24 | PLL7ST | 1 | R | PLL circuit 7 status. Displays PLL circuit 7 status (on or off).<br>0: PLL circuit 7 is turned off. The main clock is supplied.<br>1: PLL circuit 7 is turned on. The output from PLL circuit 7 is supplied. |
| 23 to 17 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 16 | PLL7E | 1 | R/W | PLL circuit 7 enable. Turns PLL circuit 7 on or off. Actual status of PLL7 is shown in PLL7ST bit.<br>0: Turns off PLL circuit 7<br>1: Turns on PLL circuit 7 |
| 15 | PLL6ST | 1 | R | PLL circuit 6 status. Displays PLL circuit 6 status (on or off).<br>0: PLL circuit 6 is turned off. The main clock is supplied.<br>1: PLL circuit 6 is turned on. The output from PLL circuit 6 is supplied. |
| 14 | PLL5ST | 1 | R | PLL circuit 5 status. Displays PLL circuit 5 status (on or off).<br>0: PLL circuit 5 is turned off. The main clock is supplied.<br>1: PLL circuit 5 is turned on. The output from PLL circuit 5 is supplied. |
| 13 | PLL4ST | 1 | R | PLL circuit 4 status. Displays PLL circuit 4 status (on or off).<br>0: PLL circuit 4 is turned off. The main clock is supplied.<br>1: PLL circuit 4 is turned on. The output from PLL circuit 4 is supplied. |
| 12 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 11 | PLL3ST | 1 | R | PLL circuit 3 status. Displays PLL circuit 3 status (on or off).<br>0: PLL circuit 3 is turned off. The main clock is supplied.<br>1: PLL circuit 3 is turned on. The output from PLL circuit 3 is supplied. |
| 10 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 9 | PLL2ST | 1 | R | PLL circuit 2 status. Displays PLL circuit 2 status (on or off).<br>0: PLL circuit 2 is turned off. The main clock is supplied.<br>1: PLL circuit 2 is turned on. The output from PLL circuit 2 is supplied. |
| 8 | PLL1ST | 1 | R | PLL circuit 1 status. Displays PLL circuit 1 status (on or off).<br>0: PLL circuit 1 is turned off. The main clock is supplied.<br>1: PLL circuit 1 is turned on. The output from PLL circuit 1 is supplied. |
| 7 | PLL6E | 1 | R/W | PLL circuit 6 enable. Turns PLL circuit 6 on or off. Actual status of PLL6 is shown in PLL6ST bit.<br>0: Turns off PLL circuit 6<br>1: Turns on PLL circuit 6 |
| 6 | PLL5E | 1 | R/W | PLL circuit 5 enable. Turns PLL circuit 5 on or off. Actual status of PLL5 is shown in PLL5ST bit.<br>0: Turns off PLL circuit 5<br>1: Turns on PLL circuit 5 |
| 5 | PLL4E | 1 | R/W | PLL circuit 4 enable. Turns PLL circuit 4 on or off. Actual status of PLL4 is shown in PLL4ST bit.<br>0: Turns off PLL circuit 4<br>1: Turns on PLL circuit 4 |
| 4 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 3 | PLL3E | 1 | R/W | PLL circuit 3 enable. Turns PLL circuit 3 on or off. Actual status of PLL3 is shown in PLL3ST bit.<br>0: Turns off PLL circuit 3<br>1: Turns on PLL circuit 3 |
| 2 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | PLL2E | 1 | R/W | PLL circuit 2 enable. Turns PLL circuit 2 on or off. Actual status of PLL2 is shown in PLL2ST bit.<br>0: Turns off PLL circuit 2<br>1: Turns on PLL circuit 2 |
| 0 | PLL1E | 1 | R/W | PLL circuit 1 enable. Turns PLL circuit 1 on or off. Actual status of PLL1 is shown in PLL1ST bit.<br>0: Turns off PLL circuit 1<br>1: Turns on PLL circuit 1 |

### 8.2.12 PLL1 Control Register 0 (PLL1CR0)

PLL1CR0 is a 32-bit readable/writable register. This register specifies the integer multiplication ratio and SSCG mode of PLL circuit 1. Mode pin control for PLL parameter setting is applied for PLL1 because the parameter cannot be changed by software registers as described in 8.1.5.1 (PLL1 Multiplication Ratio).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | NI [7:0] | | | | | | | | - | SSMODE [2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | 0 | (*1) | (*1) | (*1) |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | SSFREQ [6:0] | | | | | | | - | SSDEPT [6:0] | | | | | | |
| Initial value: | 0 | (*1) | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 20 | NI[7:0] | *(1) | R | PLL1 multiplication ratio control parameter.<br><br>SSMODE[2:0] = 3'b000:<br>・ PLL1 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL1VCO = (PLL1 reference clock) x (PLL1 multiplication ratio)<br>SSMODE[2:0] = 3'b100, 3'b110, 3'b111:<br>・ PLL1 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL1VCO = (PLL1 reference clock) x (PLL1 multiplication ratio) |
| 19 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 18 to 16 | SSMODE [2:0] | *(1) | R | PLL1 mode and output frequency dither mode are shown in the value of SSMODE[2:0].<br><br>B'000:<br>Integer multiplication PLL mode. Output frequency is fixed (0% dither).<br>B'100:<br>Fractional multiplication PLL mode. Output frequency is fixed (0% dither).<br>B'110:<br>Fractional multiplication PLL mode. Output frequency is dithered (Down spread).<br>B'111:<br>Fractional multiplication PLL mode. Output frequency is dithered (Center spread).<br>Others:<br>Setting prohibited. |
| 15 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 14 to 8 | SSFREQ [6:0] | *(1) | R | $F_{mod}$ is always 20.833 KHz.<br>The parameter is described in Figure 8.2.15.1. |
| 7 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 to 0 | SSDEPT [6:0] | *(1) | R | PLL1 modulation depth parameter described in Figure 8.2.15.1.<br><br>1) SSMODE[2:0] = B'110 (Down spread)<br>B'0000000: the modulation depth = 0 %<br>B'0010100: the modulation depth = -1.0 %<br>B'0101000: the modulation depth = -2.0 %<br>B'0111100: the modulation depth = -3.0 %<br>B'1010000: the modulation depth = -4.0 %<br>B'1100100: the modulation depth = -5.0 %<br>B'1111000: the modulation depth = -6.0 %<br>Others: Setting prohibited<br>2) SSMODE[2:0] = B'111 (Center spread)<br>B'0000000: the modulation depth = 0 %<br>B'0010100: the modulation depth = ±0.5 %<br>B'0101000: the modulation depth = ±1.0 %<br>B'0111100: the modulation depth = ±1.5 %<br>B'1010000: the modulation depth = ±2.0 %<br>B'1100100: the modulation depth = ±2.5 %<br>B'1111000: the modulation depth = ±3.0 %<br>Others: Setting prohibited |

Note 1. See Table 8.1.5.1 (PLL1 Initial Parameter Setting).

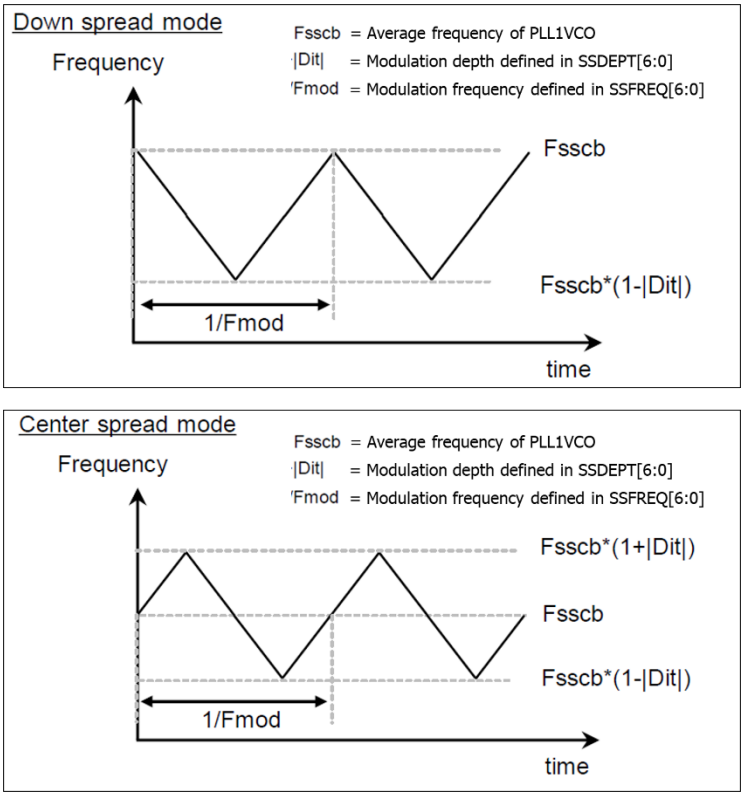
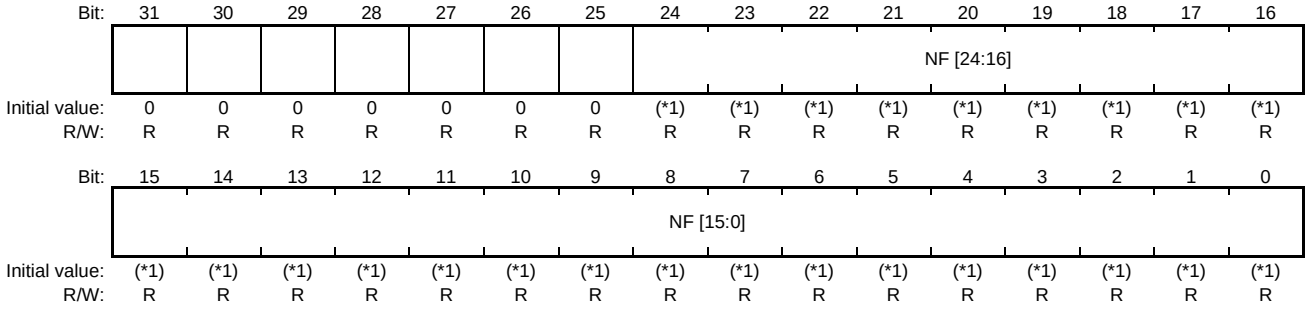



**Figure 8.2.15.1 Image diagram of SSCG (Down spread and Center spread)**

### 8.2.13 PLL1 Control Register 1 (PLL1CR1)

PLL1CR1 is a 32-bit readable/writable register. This register specifies the fractional multiplication.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 24 to 0 | NF[24:0] | *(1) | R | PLL1 multiplication ratio control parameter.<br><br>SSMODE[2:0] = 3'b000<br>・ PLL1 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL1VCO = (PLL1 reference clock) x (PLL1 multiplication ratio)<br>SSMODE[2:0] = 3'b100, 3'b110, 3'b111<br>・ PLL1 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL1VCO = (PLL1 reference clock) x (PLL1 multiplication ratio) |

Note 1. See Table 8.1.5.1 (PLL1 Initial Parameter Setting).

### 8.2.14 PLL2 Control Register 0 (PLL2CR0)

PLL2CR0 is a 32-bit readable/writable register. This register specifies the integer multiplication ratio and SSCG mode of PLL circuit 2. The parameter setting of Center spread is prohibited because Cortex-A76 operation cannot be guaranteed. The PLL2 parameter can be changed by software registers (PLL2CR0, PLL2CR1) as shown in Sec.8.1.5.2 (PLL2 Multiplication Ratio).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KICK | | | | NI [7:0] | | | | | | | | | SSMODE [2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | 0 | (*1) | (*1) | (*1) |
| R/W: | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SSFREQ [6:0] | | | | | | | | SSDEPT [6:0] | | | | | | |
| Initial value: | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | KICK | 0 | R/W | KICK bit. Setting 1 to this register activates the PLL2CRn (n = 0,1) setting.<br><br>- 1'b0: Does not activate the PLL2CRn (n = 0,1) settings.<br>- 1'b1: Activate the PLL2CRn (n = 0,1) settings.<br>This bit is automatically cleared to 0 when the PLL setting is completed after 1 is written to this bit. |
| 30 to 28 | - | 000 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 20 | NI[7:0] | *(1) | R/W | PLL2 multiplication ratio control parameter.<br>One example of the parameter setting is shown in Table 8.1.5.2.<br>SSMODE[2:0] = 3'b000<br>・ PLL2 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL2VCO = (PLL2 reference clock) x (PLL2 multiplication ratio)<br>SSMODE[2:0] = 3'b100, 3'b110<br>・ PLL2 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL2VCO = (PLL2 reference clock) x (PLL2 multiplication ratio) |
| 19 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 18 to 16 | SSMODE [2:0] | *(1) | R/W | PLL2 mode and output frequency dither mode are shown in the value of SSMODE[2:0].<br><br>B'000:<br>Integer multiplication PLL mode. Output frequency is fixed (0% dither).<br>B'100:<br>Fractional multiplication PLL mode. Output frequency is fixed (0% dither).<br>B'110:<br>Fractional multiplication PLL mode. Output frequency is dithered (Down spread).<br>Others:<br>Setting prohibited. |
| 15 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 14 to 8 | SSFREQ [6:0] | All 0 | R/W | $F_{mod}$ = (PLL reference clock) / (SSFREQ_value x 20 + 160).<br>The parameter is described in Figure 8.2.17.1.<br>1) SSFREQ[6:0] setting<br>B'0000000: SSFREQ_value = 0<br>-----<br>B'1111111: SSFREQ_value = 127<br>2) $F_{mod}$ range<br>SSDEPT[6:0] = B'0000000 → 20K < $F_{mod}$ < (Not fix) due to layout trials not finish.<br>SSDEPT[6:0] = B'0010100 → 20K < $F_{mod}$ < (Not fix) due to layout trials not finish.<br>SSDEPT[6:0] = B'0101000 → 20K < $F_{mod}$ < (Not fix) due to layout trials not finish.<br>SSDEPT[6:0] = B'0111100 → 20K < $F_{mod}$ < (Not fix) due to layout trials not finish.<br>SSDEPT[6:0] = B'1010000 → 20K < $F_{mod}$ < (Not fix) due to layout trials not finish.<br>SSDEPT[6:0] = B'1100100 → 20K < $F_{mod}$ < (Not fix) due to layout trials not finish.<br>SSDEPT[6:0] = B'1111000 → 20K < $F_{mod}$ < (Not fix) due to layout trials not finish. |
| 7 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 6 to 0 | SSDEPT [6:0] | *(1) | R | PLL2 modulation depth parameter described in Figure 8.2.17.1.<br><br>B'0000000: the modulation depth = 0 %<br>B'0010100: the modulation depth = -1.0 %<br>B'0101000: the modulation depth = -2.0 %<br>B'0111100: the modulation depth = -3.0 %<br>B'1010000: the modulation depth = -4.0 %<br>B'1100100: the modulation depth = -5.0 %<br>B'1111000: the modulation depth = -6.0 %<br>Others: Setting prohibited |

Note 1. See Table 8.1.5.2 (PLL2 Initial Parameter Setting).

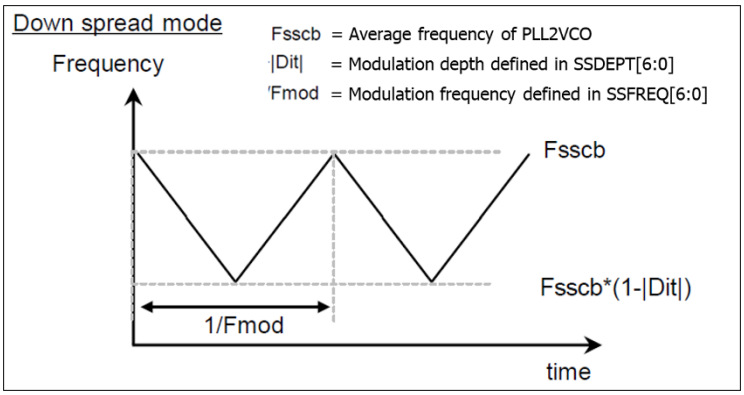
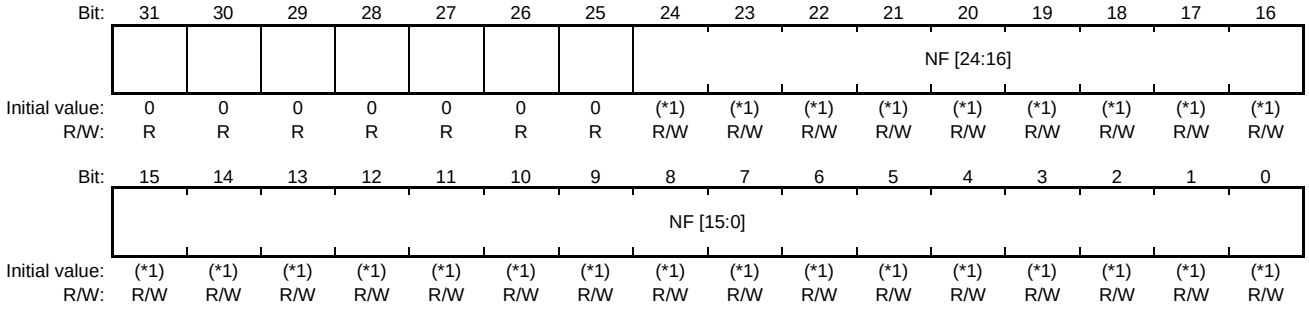



**Figure 8.2.17.1 Image diagram of SSCG (Down spread)**

### 8.2.15 PLL2 Control Register 1 (PLL2CR1))

PLL2CR1 is a 32-bit readable/writable register. This register specifies the fractional multiplication.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 25 to 0 | NF[24:0] | *(1) | R/W | PLL2 multiplication ratio control parameter.<br>One example of the parameter setting is shown in Table 8.1.5.2.<br>SSMODE[2:0] = 3'b000<br>・ PLL2 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL2VCO = (PLL2 reference clock) x (PLL2 multiplication ratio)<br>SSMODE[2:0] = 3'b100, 3'b110<br>・ PLL2 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL2VCO = (PLL2 reference clock) x (PLL2 multiplication ratio) |

Note 1. See Table 8.1.5.2 (PLL2 Initial Parameter Setting).

### 8.2.16 PLL3 Control Register 0 (PLL3CR0)

PLL3CR0 is a 32-bit readable/writable register. The parameter setting of SSCG mode is prohibited because DDR operation cannot be guaranteed. The PLL3 parameter can be changed by software registers (PLL3CR0, PLL3CR1) as shown in 8.1.5.3 (PLL3 Multiplication Ratio).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KICK | | | | NI [7:0] | | | | | | | | | SSMODE [2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

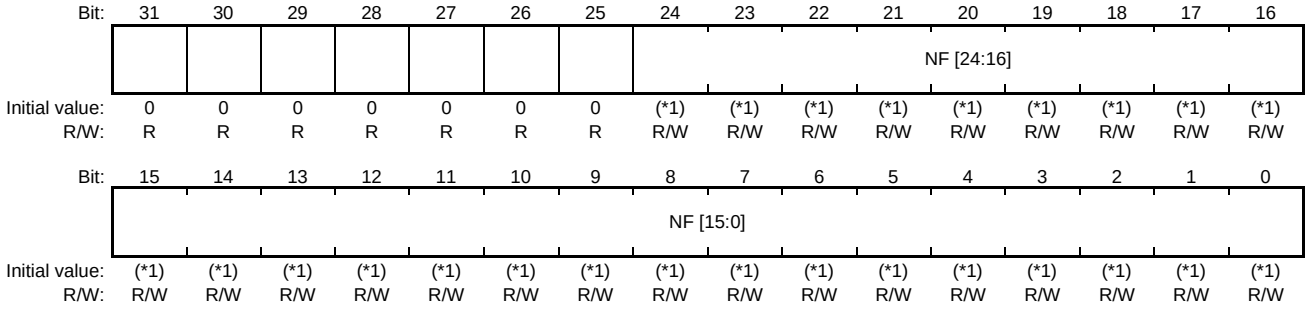
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | KICK | 0 | R/W | KICK bit. Setting 1 to this register activates the PLL3CRn (n = 0,1) setting.<br><br>- 1'b0: Does not activate the PLL3CRn (n = 0,1) settings.<br>- 1'b1: Activate the PLL3CRn (n = 0,1) settings.<br>This bit is automatically cleared to 0 when the PLL setting is completed after 1 is written to this bit. |
| 30 to 28 | - | 00 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 20 | NI[7:0] | *(1) | R/W | PLL3 multiplication ratio control parameter.<br><br>・ PLL3 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL3VCO = (PLL3 reference clock) x (PLL3 multiplication ratio) |
| 19 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 18 to 16 | SSMODE [2:0] | *(1) | R | As described in 8.1.5.3, DDR operation can be guaranteed during the integer and fractional multiplication PLL mode.<br><br>SSMODE [2:0] / PLL mode and output frequency dither mode<br>B'000 / Integer multiplication PLL mode. Output frequency is fixed (0% dither)<br>B'100 / Fractional multiplication PLL mode. Output frequency is fixed (0% dither).<br>Others / Setting prohibited. |
| 15 to 0 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

Note 1. See Table 8.1.5.3a (PLL3 Initial Parameter Setting).

### 8.2.17 PLL3 Control Register 1 (PLL3CR1)

PLL3CR1 is a 32-bit readable/writable register. This register specifies the fractional multiplication.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 24 to 0 | NF[24:0] | *(1) | R/W | PLL3 multiplication ratio control parameter. Before SSMODE[2:0] is changed from 3'B100 (fractional multiplication mode) to 3'B000 (integer multiplication mode), write 25'H000 to NF[24:0]. Any other values except 25'H000 are prohibited for NF[24:0] in integer multiplication mode.<br><br>SSMODE[2:0] = 3'b000<br>・ PLL3 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL3VCO = (PLL3 reference clock) x (PLL3 multiplication ratio)<br>SSMODE[2:0] = 3'b100<br>・ PLL3 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL3VCO = (PLL3 reference clock) x (PLL3 multiplication ratio)<br>SSMODE[2:0] = Others<br>・ Setting prohibited. |

Note 1. See Table 8.1.5.3a (PLL3 Initial Parameter Setting). The value is always All 0.

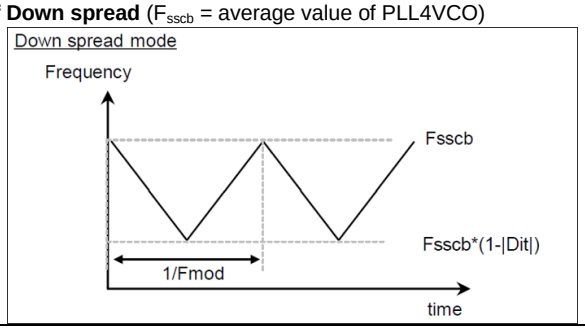
### 8.2.18 PLL4 Control Register 0 (PLL4CR0)

PLL4CR0 is a 32-bit readable/writable register. This register specifies the integer multiplication ratio and SSCG mode of PLL circuit 2. The PLL4 parameter can be changed by software registers (PLL4CR0, PLL4CR1) as shown in 8.1.5.4 (PLL4 Multiplication Ratio).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KICK | | | | NI [7:0] | | | | | | | | | SSMODE [2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | 0 | (*1) | (*1) | (*1) |
| R/W: | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SSFREQ [6:0] | | | | | | | | SSDEPT [6:0] | | | | | | |
| Initial value: | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

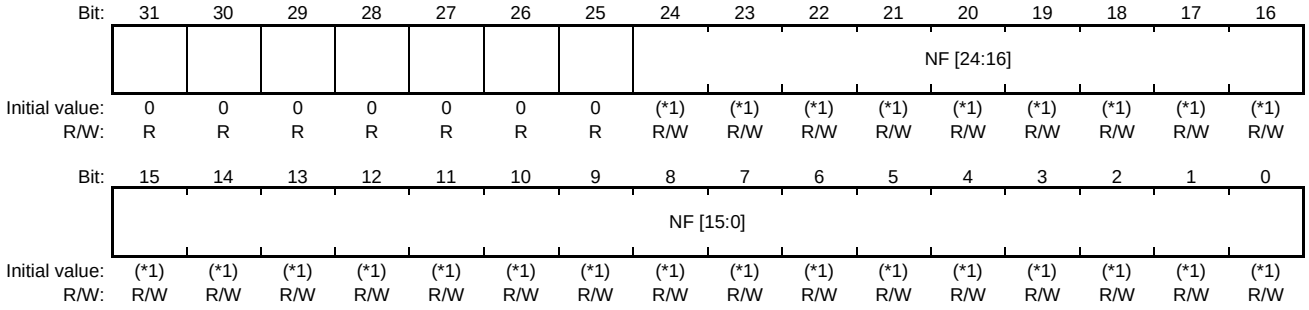
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | KICK | 0 | R/W | KICK bit. Setting 1 to this register activates the PLL4CRn (n = 0,1) setting.<br><br>- 1'b0: Does not activate the PLL4CRn (n = 0,1) settings.<br>- 1'b1: Activate the PLL4CRn (n = 0,1) settings.<br>This bit is automatically cleared to 0 when the PLL setting is completed after 1 is written to this bit. |
| 30 to 28 | - | 000 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 20 | NI[7:0] | *(1) | R/W | PLL4 multiplication ratio control parameter.<br>One example of the parameter setting is shown in Table 8.1.5.4.<br>SSMODE[2:0] = 3'b000<br>・ PLL4 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL4VCO = (PLL4 reference clock) x (PLL4 multiplication ratio)<br>SSMODE[2:0] = 3'b100, 3'b110<br>・ PLL4 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL4VCO = (PLL4 reference clock) x (PLL4 multiplication ratio) |
| 19 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 18 to 16 | SSMODE [2:0] | *(1) | R/W | SSMODE [2:0] / PLL mode and output frequency dither mode<br>B'000 / Integer multiplication PLL mode. Output frequency is fixed (0% dither)<br>B'100 / Fractional multiplication PLL mode. Output frequency is fixed (0% dither).<br>B'110 / Fractional multiplication PLL mode. Output frequency is dithered (Down spread).<br>Others / Setting prohibited. |
| 15 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

<table>
<tr><th>Bit</th><th>Bit Name</th><th>Initial Value</th><th>R/W</th><th>Description</th></tr>
<tr><td>14 to 8</td><td>SSFREQ [6:0]</td><td>All 0</td><td>R/W</td><td>The SSCG modulation frequency ($F_{mod}$) is calculated by following.<br>$F_{mod}$ range is same for each SSDEPT[6:0].<br><br>$F_{mod}$ = (PLL reference clock) / (SSFREQ_value x 20 + 160)<br><br>SSFREQ [6:0] bit setting
<table>
<tr><th>SSFREQ[6:0]</th><th>SSFREQ_value</th></tr>
<tr><td>B’0000000</td><td>0</td></tr>
<tr><td>:</td><td>:</td></tr>
<tr><td>B’1111111</td><td>127</td></tr>
</table>
$F_{mod}$ range
<table>
<tr><th>SSDEPT [6:0]</th><th>Down spread (modulation depth)</th><th>$F_{mod}$ range [Hz]</th></tr>
<tr><td>B’0000000</td><td>0</td><td>-</td></tr>
<tr><td>B’0010100</td><td>-1.0 %</td><td>20K < $F_{mod}$ < 60K</td></tr>
<tr><td>B’0101000</td><td>-2.0 %</td><td>20K < $F_{mod}$ < 60K</td></tr>
<tr><td>B’0111100</td><td>-3.0 %</td><td>20K < $F_{mod}$ < 60K</td></tr>
<tr><td>B’1010000</td><td>-4.0 %</td><td>20K < $F_{mod}$ < 60K</td></tr>
<tr><td>B’1100100</td><td>-5.0 %</td><td>20K < $F_{mod}$ < 60K</td></tr>
<tr><td>B’1111000</td><td>-6.0 %</td><td>20K < $F_{mod}$ < 60K</td></tr>
<tr><td>Others</td><td colspan="2">Setting prohibited.</td></tr>
</table>
Diagram of Down spread ($F_{sscb}$ = average value of PLL4VCO)<br>

</td></tr>
<tr><td>7</td><td>-</td><td>0</td><td>R</td><td>Reserved.<br>These bits are always read as 0. The write value should always be 0.</td></tr>
<tr><td>6 to 0</td><td></td><td>*(1)</td><td>R</td><td>
<table>
<tr><th>SSDEPT [6:0]</th><th>SSDEPT_value</th><th>Down spread (modulation depth)</th></tr>
<tr><td>B’0000000</td><td>0</td><td>0</td></tr>
<tr><td>B’0010100</td><td>20</td><td>-1.0 %</td></tr>
<tr><td>B’0101000</td><td>40</td><td>-2.0 %</td></tr>
<tr><td>B’0111100</td><td>60</td><td>-3.0 %</td></tr>
<tr><td>B’1010000</td><td>80</td><td>-4.0 %</td></tr>
<tr><td>B’1100100</td><td>100</td><td>-5.0 %</td></tr>
<tr><td>B’1111000</td><td>120</td><td>-6.0 %</td></tr>
<tr><td>Others</td><td colspan="2">Setting prohibited.</td></tr>
</table>
</td></tr>
</table>

Note 1. See Table 8.1.5.4 (PLL4 Initial Parameter Setting).

### 8.2.19 PLL4 Control Register 1 (PLL4CR1)

PLL4CR1 is a 32-bit readable/writable register. This register specifies the fractional multiplication.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 25 to 0 | NF[24:0] | *(1) | R/W | PLL4 multiplication ratio control parameter.<br>One example of the parameter setting is shown in Table 8.1.5.4.<br>SSMODE[2:0] = 3'b000<br>・ PLL4 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL4VCO = (PLL4 reference clock) x (PLL4 multiplication ratio)<br>SSMODE[2:0] = 3'b100, 3'b111<br>・ PLL4 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL4VCO = (PLL4 reference clock) x (PLL4 multiplication ratio) |

Note 1. See Table 8.1.5.4 (PLL4 Initial Parameter Setting).

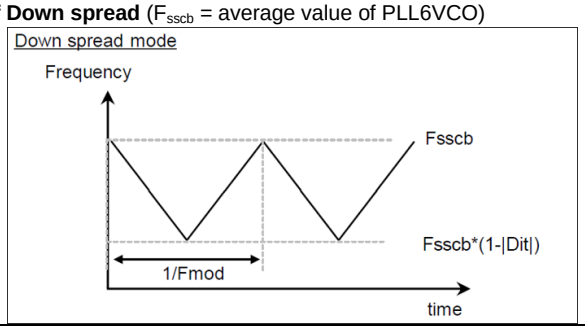
### 8.2.20 PLL6 Control Register 0 (PLL6CR0)

PLL6CR0 is a 32-bit readable/writable register. This register specifies the integer multiplication ratio and SSCG mode of PLL circuit 6. The parameter setting of Center spread is prohibited because Cortex-R52 operation cannot be guaranteed. The PLL6 parameter can be changed by software registers (PLL6CR0, PLL6CR1) as shown in 8.1.5.6 (PLL6 Multiplication Ratio).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KICK | | | | NI [7:0] | | | | | | | | | SSMODE [2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | 0 | (*1) | (*1) | (*1) |
| R/W: | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SSFREQ [6:0] | | | | | | | | SSDEPT [6:0] | | | | | | |
| Initial value: | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | KICK | 0 | R/W | KICK bit. Setting 1 to this register activates the PLL6CRn (n = 0,1) setting.<br><br>- 1'b0: Does not activate the PLL6CRn (n = 0,1) settings.<br>- 1'b1: Activate the PLL6CRn (n = 0,1) settings.<br>This bit is automatically cleared to 0 when the PLL setting is completed after 1 is written to this bit. |
| 30 to 28 | - | 000 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 20 | NI[7:0] | *(1) | R/W | PLL6 multiplication ratio control parameter.<br>One example of the parameter setting is shown in Table 8.1.5.6.<br>SSMODE[2:0] = 3'b000<br>・ PLL6 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL6VCO = (PLL6 reference clock) x (PLL6 multiplication ratio)<br>SSMODE[2:0] = 3'b100, 3'b110<br>・ PLL6 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL6VCO = (PLL6 reference clock) x (PLL6 multiplication ratio) |
| 19 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 18 to 16 | SSMODE [2:0] | *(1) | R/W | SSMODE [2:0] / PLL mode and output frequency dither mode<br>B'000 / Integer multiplication PLL mode. Output frequency is fixed (0% dither)<br>B'100 / Fractional multiplication PLL mode. Output frequency is fixed (0% dither).<br>B'110 / Fractional multiplication PLL mode. Output frequency is dithered (Down spread).<br>Others / Setting prohibited. |
| 15 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

<table>
<tr><th>Bit</th><th>Bit Name</th><th>Initial Value</th><th>R/W</th><th>Description</th></tr>
<tr><td>14 to 8</td><td>SSFREQ [6:0]</td><td>All 0</td><td>R/W</td><td>The SSCG modulation frequency ($F_{mod}$) is calculated by following.<br>$F_{mod}$ range is same for each SSDEPT[6:0].<br><br>$F_{mod}$ = (PLL reference clock) / (SSFREQ_value x 20 + 160)<br><br>SSFREQ [6:0] bit setting
<table>
<tr><th>SSFREQ[6:0]</th><th>SSFREQ_value</th></tr>
<tr><td>B’0000000</td><td>0</td></tr>
<tr><td>:</td><td>:</td></tr>
<tr><td>B’1111111</td><td>127</td></tr>
</table>
$F_{mod}$ range
<table>
<tr><th>SSDEPT [6:0]</th><th>Down spread (modulation depth)</th><th>$F_{mod}$ range [Hz]</th></tr>
<tr><td>B’0000000</td><td>0</td><td>-</td></tr>
<tr><td>B’0010100</td><td>-1.0 %</td><td>20K < $F_{mod}$ < 60K</td></tr>
<tr><td>B’0101000</td><td>-2.0 %</td><td>20K < $F_{mod}$ < 60K</td></tr>
<tr><td>B’0111100</td><td>-3.0 %</td><td>20K < $F_{mod}$ < 52.5K</td></tr>
<tr><td>B’1010000</td><td>-4.0 %</td><td>20K < $F_{mod}$ < 39.4K</td></tr>
<tr><td>B’1100100</td><td>-5.0 %</td><td>20K < $F_{mod}$ < 31.5K</td></tr>
<tr><td>B’1111000</td><td>-6.0 %</td><td>20K < $F_{mod}$ < 26.2K</td></tr>
<tr><td>Others</td><td colspan="2">Setting prohibited.</td></tr>
</table>
Diagram of Down spread ($F_{sscb}$ = average value of PLL6VCO)<br> 
</td></tr>
<tr><td>7</td><td>-</td><td>0</td><td>R</td><td>Reserved.<br>These bits are always read as 0. The write value should always be 0.</td></tr>
<tr><td>6 to 0</td><td></td><td>*(1)</td><td>R</td><td>
<table>
<tr><th>SSDEPT [6:0]</th><th>SSDEPT_value</th><th>Down spread (modulation depth)</th></tr>
<tr><td>B’0000000</td><td>0</td><td>0</td></tr>
<tr><td>B’0010100</td><td>20</td><td>-1.0 %</td></tr>
<tr><td>B’0101000</td><td>40</td><td>-2.0 %</td></tr>
<tr><td>B’0111100</td><td>60</td><td>-3.0 %</td></tr>
<tr><td>B’1010000</td><td>80</td><td>-4.0 %</td></tr>
<tr><td>B’1100100</td><td>100</td><td>-5.0 %</td></tr>
<tr><td>B’1111000</td><td>120</td><td>-6.0 %</td></tr>
<tr><td>Others</td><td colspan="2">Setting prohibited.</td></tr>
</table>
</td></tr>
</table>

Note 1. See Table 8.1.5.6 (PLL6 Initial Parameter Setting).

### 8.2.21 PLL6 Control Register 1 (PLL6CR1))

PLL6CR1 is a 32-bit readable/writable register. This register specifies the fractional multiplication.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | NF [24:16] | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NF [15:0] | | | | | | | | | | | | | | | |
| Initial value: | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) | (*1) |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 25 to 0 | NF[24:0] | *(1) | R/W | PLL6 multiplication ratio control parameter.<br>One example of the parameter setting is shown in Table 8.1.5.6.<br>SSMODE[2:0] = 3'b000<br>・ PLL6 multiplication ratio = (NI[7:0] + 1) x 2<br>・ PLL6VCO = (PLL6 reference clock) x (PLL6 multiplication ratio)<br>SSMODE[2:0] = 3'b100, 3'b110<br>・ PLL6 multiplication ratio = (NI[7:0] + 1) x 2 + NF[24:0] / 2^24<br>・ PLL6VCO = (PLL6 reference clock) x (PLL6 multiplication ratio) |

Note 1. See Table 8.1.5.6 (PLL6 Initial Parameter Setting).

### 8.2.22 PLLn Stop Control Register (PLLnSTPCR (n = 1,2,3,4,6,7))

- PLLnSTPCR is a 32-bit readable/writable register. This register specifies the stop condition of PLL circuit n when PLLECR.PLLnE is set to 1.

- If the power supply status meets the stop condition of PLL circuit specified by this register (all of the power domains specified by this register is power off), the CPG cancels the oscillation request to PLL circuit n even if the PLLECR.PLLnE is set to 1. PLL circuit oscillates when this register is set to the initial value, meaning that the oscillation request from the application part is always enabled.

### (a) PLL1STPCR

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | A3IRSTP | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | 3DGBSTP | A3ISP1STP | A3ISP0STP | - | - | - | - | A2E0D1STP | A2E0D0STP | - | A3DULSTP | - | A3VIP2STP | A3VIP1STP | A3VIP0STP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | A3IRSTP | 0 | R/W | PLL1 circuit Stop Condition by A3IR Power Status<br>0: PLL circuit is not turned off when the power supply of the A3IR area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3IR area is turned off. |
| 19 to 15 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 14 | 3DGBSTP | 0 | R/W | PLL1 circuit Stop Condition by 3DGB Power Status<br>0: PLL circuit is not turned off when the power supply of the 3DGB area is turned off.<br>1: PLL circuit is turned off when the power supply of the 3DGB area is turned off. |
| 13 | A3ISP1STP | 0 | R/W | PLL1 circuit Stop Condition by A3ISP1STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP1STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP1STP area is turned off. |
| 12 | A3ISP0STP | 0 | R/W | PLL1 circuit Stop Condition by A3ISP0STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP0STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP0STP area is turned off. |
| 11 to 8 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 7 | A2E0D1STP | 0 | R/W | PLL1 circuit Stop Condition by A2E0D1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D1 area is turned off. |
| 6 | A2E0D0STP | 0 | R/W | PLL1 circuit Stop Condition by A2E0D0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D0 area is turned off. |
| 5 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 4 | A3DULSTP | 0 | R/W | PLL1 circuit Stop Condition by A3DUL Power Status<br>0: PLL circuit is not turned off when the power supply of the A3DUL area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3DUL area is turned off. |
| 3 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 2 | A3VIP2STP | 0 | R/W | PLL1 circuit Stop Condition by A3VIP2 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP2 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP2 area is turned off. |
| 1 | A3VIP1STP | 0 | R/W | PLL1 circuit Stop Condition by A3VIP1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP1 area is turned off. |
| 0 | A3VIP0STP | 0 | R/W | PLL1 circuit Stop Condition by A3VIP0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP0 area is turned off. |

### (b) PLL2STPCR

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | A3IRSTP | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | 3DGBSTP | A3ISP1STP | A3ISP0STP | - | - | - | - | - | - | - | A3DULSTP | - | A3VIP2STP | A3VIP1STP | A3VIP0STP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | A3IRSTP | 0 | R/W | PLL2 circuit Stop Condition by A3IR Power Status<br>0: PLL circuit is not turned off when the power supply of the A3IR area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3IR area is turned off. |
| 19 to 15 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 14 | 3DGBSTP | 0 | R/W | PLL2 circuit Stop Condition by 3DGB Power Status<br>0: PLL circuit is not turned off when the power supply of the 3DGB area is turned off.<br>1: PLL circuit is turned off when the power supply of the 3DGB area is turned off. |
| 13 | A3ISP1STP | 0 | R/W | PLL2 circuit Stop Condition by A3ISP1STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP1STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP1STP area is turned off. |
| 12 | A3ISP0STP | 0 | R/W | PLL2 circuit Stop Condition by A3ISP0STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP0STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP0STP area is turned off. |
| 11 to 5 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 4 | A3DULSTP | 0 | R/W | PLL2 circuit Stop Condition by A3DUL Power Status<br>0: PLL circuit is not turned off when the power supply of the A3DUL area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3DUL area is turned off. |
| 3 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 2 | A3VIP2STP | 0 | R/W | PLL2 circuit Stop Condition by A3VIP2 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP2 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP2 area is turned off. |
| 1 | A3VIP1STP | 0 | R/W | PLL2 circuit Stop Condition by A3VIP1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP1 area is turned off. |
| 0 | A3VIP0STP | 0 | R/W | PLL2 circuit Stop Condition by A3VIP0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP0 area is turned off. |

### (c) PLL3STPCR

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | A3IRSTP | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | 3DGBSTP | A3ISP1STP | A3ISP0STP | - | - | - | - | A2E0D1STP | A2E0D0STP | - | A3DULSTP | - | A3VIP2STP | A3VIP1STP | A3VIP0STP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | A3IRSTP | 0 | R/W | PLL3 circuit Stop Condition by A3IR Power Status<br>0: PLL circuit is not turned off when the power supply of the A3IR area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3IR area is turned off. |
| 19 to 15 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 14 | 3DGBSTP | 0 | R/W | PLL3 circuit Stop Condition by 3DGB Power Status<br>0: PLL circuit is not turned off when the power supply of the 3DGB area is turned off.<br>1: PLL circuit is turned off when the power supply of the 3DGB area is turned off. |
| 13 | A3ISP1STP | 0 | R/W | PLL3 circuit Stop Condition by A3ISP1STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP1STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP1STP area is turned off. |
| 12 | A3ISP0STP | 0 | R/W | PLL3 circuit Stop Condition by A3ISP0STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP0STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP0STP area is turned off. |
| 11 to 8 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 7 | A2E0D1STP | 0 | R/W | PLL3 circuit Stop Condition by A2E0D1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D1 area is turned off. |
| 6 | A2E0D0STP | 0 | R/W | PLL3 circuit Stop Condition by A2E0D0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D0 area is turned off. |
| 5 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 4 | A3DULSTP | 0 | R/W | PLL3 circuit Stop Condition by A3DUL Power Status<br>0: PLL circuit is not turned off when the power supply of the A3DUL area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3DUL area is turned off. |
| 3 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 2 | A3VIP2STP | 0 | R/W | PLL3 circuit Stop Condition by A3VIP2 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP2 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP2 area is turned off. |
| 1 | A3VIP1STP | 0 | R/W | PLL3 circuit Stop Condition by A3VIP1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP1 area is turned off. |
| 0 | A3VIP0STP | 0 | R/W | PLL3 circuit Stop Condition by A3VIP0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP0 area is turned off. |

### (d) PLL4STPCR

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | A3IRSTP | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | 3DGBSTP | A3ISP1STP | A3ISP0STP | - | - | - | - | A2E0D1STP | A2E0D0STP | - | A3DULSTP | - | A3VIP2STP | A3VIP1STP | A3VIP0STP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | A3IRSTP | 0 | R/W | PLL4 circuit Stop Condition by A3IR Power Status<br>0: PLL circuit is not turned off when the power supply of the A3IR area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3IR area is turned off. |
| 19 to 15 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 14 | 3DGBSTP | 0 | R/W | PLL4 circuit Stop Condition by 3DGB Power Status<br>0: PLL circuit is not turned off when the power supply of the 3DGB area is turned off.<br>1: PLL circuit is turned off when the power supply of the 3DGB area is turned off. |
| 13 | A3ISP1STP | 0 | R/W | PLL4 circuit Stop Condition by A3ISP1STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP1STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP1STP area is turned off. |
| 12 | A3ISP0STP | 0 | R/W | PLL4 circuit Stop Condition by A3ISP0STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP0STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP0STP area is turned off. |
| 11 to 8 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 7 | A2E0D1STP | 0 | R/W | PLL4 circuit Stop Condition by A2E0D1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D1 area is turned off. |
| 6 | A2E0D0STP | 0 | R/W | PLL4 circuit Stop Condition by A2E0D0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D0 area is turned off. |
| 5 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 4 | A3DULSTP | 0 | R/W | PLL4 circuit Stop Condition by A3DUL Power Status<br>0: PLL circuit is not turned off when the power supply of the A3DUL area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3DUL area is turned off. |
| 3 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 2 | A3VIP2STP | 0 | R/W | PLL4 circuit Stop Condition by A3VIP2 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP2 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP2 area is turned off. |
| 1 | A3VIP1STP | 0 | R/W | PLL4 circuit Stop Condition by A3VIP1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP1 area is turned off. |
| 0 | A3VIP0STP | 0 | R/W | PLL4 circuit Stop Condition by A3VIP0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP0 area is turned off. |

### (e) PLL6STPCR

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | A3IRSTP | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | 3DGBSTP | A3ISP1STP | A3ISP0STP | - | - | - | - | A2E0D1STP | A2E0D0STP | - | A3DULSTP | - | A3VIP2STP | A3VIP1STP | A3VIP0STP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | A3IRSTP | 0 | R/W | PLL6 circuit Stop Condition by A3IR Power Status<br>0: PLL circuit is not turned off when the power supply of the A3IR area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3IR area is turned off. |
| 19 to 15 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 14 | 3DGBSTP | 0 | R/W | PLL6 circuit Stop Condition by 3DGB Power Status<br>0: PLL circuit is not turned off when the power supply of the 3DGB area is turned off.<br>1: PLL circuit is turned off when the power supply of the 3DGB area is turned off. |
| 13 | A3ISP1STP | 0 | R/W | PLL6 circuit Stop Condition by A3ISP1STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP1STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP1STP area is turned off. |
| 12 | A3ISP0STP | 0 | R/W | PLL6 circuit Stop Condition by A3ISP0STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP0STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP0STP area is turned off. |
| 11 to 8 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 7 | A2E0D1STP | 0 | R/W | PLL6 circuit Stop Condition by A2E0D1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D1 area is turned off. |
| 6 | A2E0D0STP | 0 | R/W | PLL6 circuit Stop Condition by A2E0D0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D0 area is turned off. |
| 5 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 4 | A3DULSTP | 0 | R/W | PLL6 circuit Stop Condition by A3DUL Power Status<br>0: PLL circuit is not turned off when the power supply of the A3DUL area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3DUL area is turned off. |
| 3 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 2 | A3VIP2STP | 0 | R/W | PLL6 circuit Stop Condition by A3VIP2 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP2 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP2 area is turned off. |
| 1 | A3VIP1STP | 0 | R/W | PLL6 circuit Stop Condition by A3VIP1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP1 area is turned off. |
| 0 | A3VIP0STP | 0 | R/W | PLL6 circuit Stop Condition by A3VIP0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP0 area is turned off. |

### (f) PLL7STPCR

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | A3IRSTP | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | 3DGBSTP | A3ISP1STP | A3ISP0STP | - | - | - | - | A2E0D1STP | A2E0D0STP | - | A3DULSTP | - | A3VIP2STP | A3VIP1STP | A3VIP0STP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | A3IRSTP | 0 | R/W | PLL7 circuit Stop Condition by A3IR Power Status<br>0: PLL circuit is not turned off when the power supply of the A3IR area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3IR area is turned off. |
| 19 to 15 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 14 | 3DGBSTP | 0 | R/W | PLL7 circuit Stop Condition by 3DGB Power Status<br>0: PLL circuit is not turned off when the power supply of the 3DGB area is turned off.<br>1: PLL circuit is turned off when the power supply of the 3DGB area is turned off. |
| 13 | A3ISP1STP | 0 | R/W | PLL7 circuit Stop Condition by A3ISP1STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP1STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP1STP area is turned off. |
| 12 | A3ISP0STP | 0 | R/W | PLL7 circuit Stop Condition by A3ISP0STP Power Status<br>0: PLL circuit is not turned off when the power supply of the A3ISP0STP area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3ISP0STP area is turned off. |
| 11 to 8 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 7 | A2E0D1STP | 0 | R/W | PLL7 circuit Stop Condition by A2E0D1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D1 area is turned off. |
| 6 | A2E0D0STP | 0 | R/W | PLL7 circuit Stop Condition by A2E0D0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A2E0D0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A2E0D0 area is turned off. |
| 5 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 4 | A3DULSTP | 0 | R/W | PLL7 circuit Stop Condition by A3DUL Power Status<br>0: PLL circuit is not turned off when the power supply of the A3DUL area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3DUL area is turned off. |
| 3 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 2 | A3VIP2STP | 0 | R/W | PLL7 circuit Stop Condition by A3VIP2 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP2 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP2 area is turned off. |
| 1 | A3VIP1STP | 0 | R/W | PLL7 circuit Stop Condition by A3VIP1 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP1 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP1 area is turned off. |
| 0 | A3VIP0STP | 0 | R/W | PLL7 circuit Stop Condition by A3VIP0 Power Status<br>0: PLL circuit is not turned off when the power supply of the A3VIP0 area is turned off.<br>1: PLL circuit is turned off when the power supply of the A3VIP0 area is turned off. |

### 8.2.23 SD-IF0 Clock Frequency Control Register (SD0CKCR)

This register should be set before SD-IF0 module (SDHI) is operated. Do not access SD-IF0 module during changing the clock frequency of SD0Hφ, and SD0φ, or stopping these clocks.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | STP0 HCK | STP0 CK | - | - | - | SDSRCFC [2:0] | | | SD0FC [1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | — | All 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 9 | STP0HCK | 1 | R/W | Clock Stop<br>0: Supplies SDnH clock<br>1: Stops SDnH clock |
| 8 | STP0CK | 0 | R/W | Clock Stop<br>0: Supplies SDn clock<br>1: Stops SDn clock |
| 7 to 5 | — | All 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 4 to 2 | SDSRCFC [2:0] | 010 | R/W | SD0Hφ clock Frequency Division Ratio<br><br>**(1) CKSRCSELCR.SDSRCSEL[1:0] = 00:**<br><br>SDSRCφ = 800 MHz<br>000: SD0Hφ = SDSRCφ × 1/1<br>SD0φ = SDSRCφ × 1/4 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = 800, SD0φ = 200 MHz)<br>001: SD0Hφ = SDSRCφ × 1/2,<br>SD0φ = SDSRCφ × 1/8 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = 400, SD0φ = 100 MHz)<br>SD0φ = SDSRCφ × 1/4 (when SD0FC[1:0] = 00)<br>(e.g. SD0Hφ = 400, SD0φ = 200 MHz)<br>010: SD0Hφ = SDSRCφ × 1/4 (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/16 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 50 MHz)<br>011: SD0Hφ = should be stopped (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/32 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 25 MHz)<br>100: SD0Hφ = should be stopped (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/64 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 12.5 MHz)<br>101: Setting prohibited<br>110: Setting prohibited<br>111: Setting prohibited<br>**(2) CKSRCSELCR.SDSRCSEL[1:0] = 01:**<br>SDSRCφ = 640 MHz<br>000: SD0Hφ = SDSRCφ × 1/1<br>SD0φ = SD0SRCφ × 1/4 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = 640, SD0φ = 160 MHz)<br>001: SD0Hφ = SDSRCφ × 1/2,<br>SD0φ = SDSRCφ × 1/8 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = 320, SD0φ = 80 MHz)<br>SD0φ = SDSRCφ × 1/4 (when SD0FC[1:0] = 00)<br>(e.g. SD0Hφ = 320, SD0φ = 160 MHz)<br>010: SD0Hφ = SDSRCφ × 1/4 (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/16 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 40 MHz)<br>011: SD0Hφ = should be stopped (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/32 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 20 MHz)<br>100: SD0Hφ = should be stopped (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/64 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 10 MHz)<br>101: Setting prohibited<br>110: Setting prohibited<br>111: Setting prohibited<br>**(3) CKSRCSELCR.SDSRCSEL[1:0] = 10:**<br>SDSRCϕ = 533.33 MHz<br>000: SD0Hφ = SDSRCφ × 1/1<br>SD0φ = SD0SRCφ × 1/4 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = 533.33, SD0φ = 133.33 MHz)<br>001: SD0Hφ = SDSRCφ × 1/2,<br>SD0φ = SDSRCφ × 1/8 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hϕ = 266.66, SD0ϕ = 66.66 MHz)<br>SD0φ = SDSRCφ × 1/4 (when SD0FC[1:0] = 00)<br>(e.g. SD0Hφ = 266.66, SD0φ = 133.33 MHz)<br>010: SD0Hφ = SDSRCφ × 1/4 (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/16 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 33.33 MHz)<br>011: SD0Hφ = should be stopped (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/32 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 16.66 MHz)<br>100: SD0Hφ = should be stopped (should be stopped by STPnHCK)<br>SD0φ = SDSRCφ × 1/64 (when SD0FC[1:0] = 01)<br>(e.g. SD0Hφ = should be stopped, SD0φ = 8.88 MHz)<br>101: Setting prohibited<br>110: Setting prohibited<br>111: Setting prohibited<br>**(4) CKSRCSELCR.SDSRCSEL[1:0] = 11:**<br>000: SD0Hφ = SDSRCφ × 1/1<br>001: SD0Hφ = SDSRCφ × 1/2,<br>010: SD0Hφ = SDSRCφ × 1/4 (should be stopped by STPnHCK) |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| | | | | 011: SD0Hφ = SDSRCφ × 1/8 (should be stopped by STPnHCK)<br>100: SD0Hφ = SDSRCφ × 1/16 (should be stopped by STPnHCK)<br>101: Setting prohibited<br>110: Setting prohibited<br>111: Setting prohibited |
| 1 to 0 | SD0FC[1:0] | 01 | R/W | Select SD0φ division Ratio<br><br>00: SD0φ = SD0Hφ × 1/2<br>01: SD0φ = SD0Hφ × 1/4<br>10: Setting prohibited<br>11: Setting prohibited<br>Example 1<br>CKSRCSELCR.SDSRCSEL[1:0] = 00, SDSRCFC[2:0] = 000, SD0FC[1:0] = 01,<br>SD0Hφ = 800 MHz, SD0φ = 200 MHz<br>Example 2<br>CKSRCSELCR.SDSRCSEL[1:0] = 00, SDSRCFC[2:0] = 001, SD0FC[1:0] = 00,<br>SD0Hφ = 400 MHz, SD0φ = 200 MHz<br>Refer to SD0SRCFC[2:0] explanation. |

### 8.2.24 RPC Clock Frequency Control Register (RPCCKCR)

RPCCKCR is a 32-bit readable/writable register. This register controls the RPC clock (RPCφ). This register should be set before RPC module is operated. Do not access RPC module during changing the clock frequency of RPCφ and RPCD2φ.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | CKSTP2 | CKSTP | - | - | - | RPCFC [4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | (*1) | (*1) | 1 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 9 | CKSTP2 | 0 | R/W | RPCD2φ Clock<br>0: Supplies clock to RPC<br>1: Stops clock to RPC |
| 8 | CKSTP | 0 | R/W | RPCφ Clock Stop.<br>0: Supplies clock to RPC<br>1: Stops clock to RPC |
| 7 to 5 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 to 0 | RPCFC [4:0] | [4:3] = 10<br>[2:1] = *(1)<br>[0] = 1 | R/W | RPC clock (RPCφ, RPCD2φ) Frequency Division Ratio.<br>When PLL5 is turned off, RPC clock can't be oscillated (please set CKSTP1 and CKSTP2 before turning off PLL5).<br><br>**RPCFC[4:3]**<br>RPCFC [4:3] / RPCSRCφ [MHz]<br>00 / (PLL5VCO) x 1/2 x 1/2<br>01 / (PLL5VCO) x 1/2 x 1/3<br>10 / (PLL5VCO) x 1/5<br>11 / (PLL5VCO) x 1/6<br><br>**RPCFC[2:0]**<br>RPCϕ = RPCSRCϕ / ( DIV[2:0] +1 )<br>RPCD2ϕ = RPCSRCϕ / { ( DIV[2:0] +1 ) x 2 }<br>The setting DIV[2:0]= 000 is prohibited.<br>The setting DIV[0] = 0 is also prohibited, because duty = 50% is required.<br><br>**Example**<br>RPCFC [4:3][2:0] / RPCφ [MHz] / RPCD2φ [MHz]<br>10_000 / Prohibited<br>10_001 / PLL5VCO(3200MHz) x 1/5 x 1/2 = 320 (RPC max, initial value when PRCφ = 320MHz) / 160<br>10_010 / Prohibited<br>10_011 / PLL5VCO(3200MHz) x 1/5 x 1/4 = 160 (QSPI max, initial value when RPCφ = 160MHz) / 80<br>10_100 / Prohibited<br>10_101 / PLL5VCO(3200MHz) x 1/5 x 1/6 = 106.66 / 53.33<br>10_110 / Prohibited<br>10_111 / PLL5VCO(3200MHz) x 1/5 x 1/8 = 80 (Initial value when RPCφ = 80MHz or stopped) / 40<br>11_000 / Prohibited<br>11_001 / PLL5VCO(3200MHz) x 1/6 x 1/2 = 266.66 / 133.33<br>11_010 / Prohibited<br>11_011 / PLL5VCO(3200MHz) x 1/6 x 1/4 = 133.33 / 66.66<br>11_100 / Prohibited<br>11_101 / PLL5VCO(3200MHz) x 1/6 x 1/6 = 88.88 / 44.44<br>11_110 / Prohibited<br>11_111 / PLL5VCO(3200MHz) x 1/6 x 1/8 = 66.66 / 33.33 |

Note 1. Depend on MD4, MD3, MD2, and MD1 values
When (MD4, MD3, MD2, MD1) = (Don't care, 0, 1, 0), DIV[2:0] = 001, DIV[4:3] = 10.
When (MD4, MD3, MD2, MD1) = (Don't care, 0, 1, 1), DIV[2:0] = 011, DIV[4:3] = 10.
When (MD4, MD3, MD2, MD1) = (Others), DIV[2:0] = 111, DIV[4:3] = 10.

### 8.2.25 CAN-FD Clock Frequency Control Register (CANFDCKCR)

CANFDCKCR is a 32-bit readable/writable register. This register controls the CAN-FD clock (CANFDφ). This register should be set before CAN-FD module is operated. Do not access CAN-FD module during changing the clock frequency of CANFDφ.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV [5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | 0 | R/W | Clock Stop.<br>0: Supplies clock to CAN-FD module<br>1: Stops clock to CAN-FD module |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV [5:0] | All 1 | R/W | Division Ratio.<br>These bits set the frequency division ratio of CAN-FD clock. CAN-FD clock source is divided by the division ratio of 1 / (setting + 1). DIV[5:0] should be set so that CANFDφ frequency doesn't exceed the max frequency shown in Sec. 8.1.4.<br>CANFDφ frequency [MHz] = PLL5VCO x 1/2 x 1/2 x 1 / (setting + 1). |
| | | | | |

### 8.2.26 MSIOF Clock Frequency Control Register (MSOCKCR)

MSOCKCR is a 32-bit readable/writable register. This register controls the MSIOF clock (MSOφ). This register should be set before MSIOF module is operated. Do not access MSIOF module during changing the clock frequency of MSOφ.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV [5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP (*1) | 0 | R/W | Clock Stop.<br>0: Supplies clock to MSIOF module<br>1: Stops clock to MSIOF module |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV [5:0] | All 1 | R/W | Division Ratio.<br>These bits set the frequency division ratio of MSIOF clock. MSIOF clock source is divided by the division ratio of 1 / (setting + 1). DIV[5:0] should be set so that MSOφ frequency doesn't exceed the max frequency shown in Sec. 8.1.4.<br>MSOφ frequency [MHz] = PLL5VCO x 1/2 x 1/2 x 1 / (setting + 1). |

Note 1:
When accessing any registers defined in 109.2 Register Description ( for Clock-Synchronized Serial Interface with FIFO (MSIOF)), the MSIOF clock (MSOφ) shall not be stopped. Before accessing the registers, the bit value of CKSTP shall be checked. The value shall be 0 (Supplies clock to MSIOF module).

### 8.2.27 CSI Clock Frequency Control Register (CSICKCR)

CSICKCR is a 32-bit readable/writable register. This register controls the CSI clock (CSIφ). This register should be set before CSI module is operated. Do not access MIPI-CSI2 module during changing the clock frequency of CSIφ.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV [5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | 1 | R/W | Clock Stop.<br>0: Supplies clock to CSI module<br>1: Stops clock to CSI module |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV [5:0] | 6'b011111 | R/W | Division Ratio.<br>These bits set the frequency division ratio of CSI clock. CSI clock source is divided by the division ratio of 1 / (setting + 1). DIV[5:0] should be set so that CSIφ frequency doesn't exceed the max frequency shown in Sec. 8.1.4.<br>CSIφ frequency [MHz] = PLL5VCO x 1/2 x 1/2 x 1 / (setting + 1). |

### 8.2.28 DSI Clock Frequency Control Register (DSIEXTCKCR)

DSIEXTCKCR is a 32-bit readable/writable register. This register controls the DSI clock (DSIEXTφ). This register should be set before DSI module is operated. Do not access MIPI-DSI module during changing the clock frequency of DSIEXTφ.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV [5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | 1 | R/W | Clock Stop.<br>0: Supplies clock to DSI module<br>1: Stops clock to DSI module |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV [5:0] | All 0 | R/W | Division Ratio.<br>These bits set the frequency division ratio of DSI clock. DSI clock source is divided by the division ratio of 1 / (setting + 1). DIV[5:0] should be set so that DSIEXTφ frequency doesn't exceed the max frequency shown in Sec. 8.1.4.<br>DSIEXTφ frequency [MHz] = PLL5VCO x 1/2 x 1/2 x 1 / (setting + 1). |

### 8.2.29 Power on Self Test Clock Frequency Control Register (POSTCKCR)

POSTCKCR is a 32-bit readable/writable register. This register controls the Power-On Self Test clock (POSTφ). POST frequency is 66.66 MHz, so the initial frequency is also 66.66MHz. Run-Time Test is done at 66.66MHz. POSTφ target is any others except Cortex-A76, 3D Graphics Engine, Vision IP.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV [5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ~MD25 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | ~MD25 | R/W | Clock Stop.<br>0: Supplies clock with POSTCK target modules.<br>1: Stops POSTφ. |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV [5:0] | 6'b001011 | R/W | Division Ratio.<br>These bits set the frequency division ratio of clock. POST clock source is divided by the division ratio of 1 / (setting + 1).<br>Initial value:<br>PLL5VCO (3200MHz) x 1/2 x 1/2 x 1 / (setting + 1) = 66.66 MHz.<br>"setting" is equal to 6'b001011. |

### 8.2.30 Power on Self Test 2 Clock Frequency Control Register (POST2CKCR)

POST2CKCR is a 32-bit readable/writable register. This register controls the Power-On Self Test 2 clock (POST2φ), which is used for Cortex-A76 during POST executing.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV [5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ~MD25 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | ~MD25 | R/W | Clock Stop.<br>0: Supplies clock with POST2CK target modules.<br>1: Stops POST2φ. |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV [5:0] | 6'b000101 | R/W | Division Ratio.<br>These bits set the frequency division ratio of clock. POST clock source is divided by the division ratio of 1 / (setting + 1).<br>Initial value:<br>PLL5VCO (3200MHz) x 1/2 x 1/2 x 1 / (setting + 1) = 133.33 MHz.<br>"setting" is equal to 6'b000101. |

### 8.2.31 Power on Self Test 3 Clock Frequency Control Register (POST3CKCR)

POST3CKCR is a 32-bit readable/writable register. This register controls the Power-On Self Test clock (POST3φ). POST frequency is 66.66 MHz, so the initial frequency is also 66.66MHz. Run-Time Test is done at 66.66MHz. POST3φ target is 3D Graphics Engine.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV [5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ~MD25 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | ~MD25 | R/W | Clock Stop.<br>0: Supplies clock with POST3CK target modules.<br>1: Stops POST3CK. |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV [5:0] | 6'b001011 | R/W | Division Ratio.<br>These bits set the frequency division ratio of clock. POST3 clock source is divided by the division ratio of 1 / (setting + 1).<br>Initial value:<br>PLL5VCO (3200MHz) x 1/2 x 1/2 x 1 / (setting + 1) = 66.66 MHz.<br>"setting" is equal to 6'b001011. |

### 8.2.32 Power on Self Test 4 Clock Frequency Control Register (POST4CKCR)

POST4CKCR is a 32-bit readable/writable register. This register controls the Power-On Self Test clock (POST4φ). POST frequency is 66.66 MHz, so the initial frequency is also 66.66MHz. Run-Time Test is done at 66.66MHz. POST4φ target is Vision IP.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV [5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ~MD25 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | ~MD25 | R/W | Clock Stop.<br>0: Supplies clock with POST4CK target modules.<br>1: Stops POST4CK. |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV [5:0] | 6'b001011 | R/W | Division Ratio.<br>These bits set the frequency division ratio of clock. POST4 clock source is divided by the division ratio of 1 / (setting + 1).<br>Initial value:<br>PLL5VCO (3200MHz) x 1/2 x 1/2 x 1 / (setting + 1) = 66.66 MHz.<br>"setting" is equal to 6'b001011. |

### 8.2.33 Clock Source Select Control Register (CKSRCSELCR)

CKSRCSELCR is a 32-bit readable/writable register. The register selects the clock frequency select of SDSRCφ.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | SDSRC SEL | | - | - | - | - | - | - | - | - | - | - | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R/W | R/W | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | - | 0 | R | Reserved. This bit is always read as 0. The write value should always be 0. |
| 30 to 29 | SDSRCSEL [1:0] | All 0 | R/W | SDSRCφ clock frequency select.<br><br>The bit should be set before SD-IF0 module (SDHI) is operated. Do not access SDHI during changing the clock frequency of SD0Hφ, and SD0φ, or stopping these clocks. The bit setting is necessary when SD0CKCR register is set.<br>2'b00: 800.00 MHz (PLL5VCO (always 3200 MHz) x 1/2 x 1/2)<br>2'b01: 640.00 MHz (PLL5VCO (always 3200 MHz) x 1/5)<br>2'b10: 533.33 MHz (PLL5VCO (always 3200 MHz) x 1/6)<br>2'b11: Setting prohibited |
| 28 | - | 0 | R | Reserved. These bits are always read as 0. The write value should always be 0. |
| 27 to 20 | - | All 1 | R | Reserved. This bit is always read as 1. The write value should always be 1. |
| 19 to 16 | - | All 1 | R | Reserved. This bit is always read as 1. The write value should always be 1. |
| 15 to 4 | - | All 0 | R | Reserved. This bit is always read as 0. The write value should always be 0. |
| 3 to 0 | - | All 0 | R | Reserved. This bit is always read as 0. The write value should always be 0. |

### 8.2.34 Domain 0 Write Access Control Register A n (D0WACRAn (n = 0 to 63))

D0WACRAn (n = 0 to 63) is a 32-bit readable/writable register. Only Domain 0 can access the register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D0WACRAn31 | D0WACRAn30 | D0WACRAn29 | D0WACRAn28 | D0WACRAn27 | D0WACRAn26 | D0WACRAn25 | D0WACRAn24 | D0WACRAn23 | D0WACRAn22 | D0WACRAn21 | D0WACRAn20 | D0WACRAn19 | D0WACRAn18 | D0WACRAn17 | D0WACRAn16 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D0WACRAn15 | D0WACRAn14 | D0WACRAn13 | D0WACRAn12 | D0WACRAn11 | D0WACRAn10 | D0WACRAn9 | D0WACRAn8 | D0WACRAn7 | D0WACRAn6 | D0WACRAn5 | D0WACRAn4 | D0WACRAn3 | D0WACRAn2 | D0WACRAn1 | D0WACRAn0 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | D0WACRAnm (n = 0 to 63) (m = 0 to 31) | All 1 | R/W | See Sec.8.4.2. |

### 8.2.35 Domain 1 Write Access Control Register A n (D1WACRAn (n = 0 to 63))

D1WACRAn (n = 0 to 63) is a 32-bit readable/writable register. Only Domain 0 can access the register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D1WACRAn31 | D1WACRAn30 | D1WACRAn29 | D1WACRAn28 | D1WACRAn27 | D1WACRAn26 | D1WACRAn25 | D1WACRAn24 | D1WACRAn23 | D1WACRAn22 | D1WACRAn21 | D1WACRAn20 | D1WACRAn19 | D1WACRAn18 | D1WACRAn17 | D1WACRAn16 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D1WACRAn15 | D1WACRAn14 | D1WACRAn13 | D1WACRAn12 | D1WACRAn11 | D1WACRAn10 | D1WACRAn9 | D1WACRAn8 | D1WACRAn7 | D1WACRAn6 | D1WACRAn5 | D1WACRAn4 | D1WACRAn3 | D1WACRAn2 | D1WACRAn1 | D1WACRAn0 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | D1WACRAnm (n = 0 to 63) (m = 0 to 31) | All 1 | R/W | See Sec.8.4.2. |

### 8.2.36 Domain 2 Write Access Control Register A n (D2WACRAn (n = 0 to 63))

D2WACRAn (n = 0 to 63) is a 32-bit readable/writable register. Only Domain 0 can access the register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D2WACRAn31 | D2WACRAn30 | D2WACRAn29 | D2WACRAn28 | D2WACRAn27 | D2WACRAn26 | D2WACRAn25 | D2WACRAn24 | D2WACRAn23 | D2WACRAn22 | D2WACRAn21 | D2WACRAn20 | D2WACRAn19 | D2WACRAn18 | D2WACRAn17 | D2WACRAn16 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D2WACRAn15 | D2WACRAn14 | D2WACRAn13 | D2WACRAn12 | D2WACRAn11 | D2WACRAn10 | D2WACRAn9 | D2WACRAn8 | D2WACRAn7 | D2WACRAn6 | D2WACRAn5 | D2WACRAn4 | D2WACRAn3 | D2WACRAn2 | D2WACRAn1 | D2WACRAn0 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | D2WACRAnm (n = 0 to 63) (m = 0 to 31) | All 1 | R/W | See Sec.8.4.2. |

### 8.2.37 Domain 3 Write Access Control Register A n (D3WACRAn (n = 0 to 63))

D3WACRAn (n = 0 to 63) is a 32-bit readable/writable register. Only Domain 0 can access the register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D3WACRAn31 | D3WACRAn30 | D3WACRAn29 | D3WACRAn28 | D3WACRAn27 | D3WACRAn26 | D3WACRAn25 | D3WACRAn24 | D3WACRAn23 | D3WACRAn22 | D3WACRAn21 | D3WACRAn20 | D3WACRAn19 | D3WACRAn18 | D3WACRAn17 | D3WACRAn16 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D3WACRAn15 | D3WACRAn14 | D3WACRAn13 | D3WACRAn12 | D3WACRAn11 | D3WACRAn10 | D3WACRAn9 | D3WACRAn8 | D3WACRAn7 | D3WACRAn6 | D3WACRAn5 | D3WACRAn4 | D3WACRAn3 | D3WACRAn2 | D3WACRAn1 | D3WACRAn0 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | D3WACRAnm (n = 0 to 63) (m = 0 to 31) | All 1 | R/W | See Sec.8.4.2. |

### 8.2.38 XXX Clock Monitor Control / Status Register (XXXCKMCSR)

XXXCKMCSR is a 32-bit readable/writable register.

| Bit: | 31 | 30 | 29 | 28 | | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHK RES | - | - | - | | - | - | - | CLR MON | - | - | - | - | - | RSTM MON | RSTM SMP | EN MON |
| Initial value: | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | | - | - | - | - | - | - | - | - | - | - | INT EN | CHK EN |
| Initial value: | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CHKRES | 0 | R | Check result.<br>0: Clock oscillates normally.<br>1: Detect unexpected clock oscillation.<br>Even if INTEN is 0, this bit shows check result.<br>This bit is cleared when XXXCKMECR.CLRRES is 1. |
| 30 to 25 | - | All 0 | R | Reserved.<br>These bits are always read as unknown value. The write value should always be 0. |
| 24 | CLRMON | All 0 | R | Reference for clear command signal - XXXCKMECR.CLRRES<br>0: Clear command signal is released.<br>1: Clear command signal is issued. |
| 23 to 19 | - | All 0 | R | Reserved.<br>These bits are always read as unknown value. The write value should always be 0. |
| 18 | RSTMMON | 0 | R | Reset status for monitor clock domain of clock monitor module.<br>0: Reset signal is issued.<br>1: Reset signal is released. |
| 17 | RSTMSMP | 0 | R | Reset status for sampling clock domain of clock monitor module.<br>0: Reset signal is issued.<br>1: Reset signal is released. |
| 16 | ENMON | 0 | R | Operating status.<br>0: Clock monitor is stopped.<br>1: Clock monitor is operating. |
| 15 to 2 | - | All 0 | R | Reserved.<br>These bits are always read as unknown value. The write value should always be 0. |
| 1 | INTEN | 0 | R/W | Interrupt mask bit.<br>0: Mask an interrupt.<br>1: Permit to issue an interrupt when clock monitor detects an error. |
| 0 | CHKEN | 0 | R/W | Enable for checking clock oscillation.<br>0: Stop clock monitor checking.<br>1: Run clock monitor checking. |

### 8.2.39 XXX Clock Monitor Error Clear Register (XXXCKMECR)

XXXCKMECR is a 32-bit readable/writable register.

| Bit: | 31 | 30 | 29 | 28 | | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | CLRRES |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | - | All 0 | R | Reserved.<br>These bits are always read as unknown value. The write value should always be 0. |
| 0 | CLRRES | 0 | R/W | Reference for clear command signal - XXXCKMECR.CLRRES<br><br>0: Release clear signal.<br>1: Issue clear signal to clock monitor, when clock monitor accepts it.<br>XXXCKMCSR.CHKRES and XXXCKMCNTE are cleared to all 0.<br>After confirming that they are cleared, release this clear signal.<br>Don't issue this clear signal while XXXCKMCSR.CHKEN or XXXCKMCSR.ENMON is 1. |

### 8.2.40 XXX Clock Monitor Limit Count H Register (XXXCKMLCH)

XXXCKMLCH is a 32-bit readable/writable register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | CMPH[23:16] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPH[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | - | All 0 | R | Reserved.<br>These bits are always read as unknown value. The write value should always be 0. |
| 23 to 0 | CMPH[23:0] | All 0 | R/W | The upper limit to judge as expected oscillation.<br>This value is compared with XXXCKMCNT.MONCNT.<br>If XXXCKMCNT.MONCNT is greater than or equal to CMPH, clock monitor judges it as an error.<br>Don't modify this value while XXXCKMCSR.CHKEN or XXXCKMCSR.ENMON is 1. |

### 8.2.41 XXX Clock Monitor Limit Count L Register (XXXCKMLCL)

XXXCKMLCH is a 32-bit readable/writable register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | CMPL[23:16] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPL[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | - | All 0 | R | Reserved.<br>These bits are always read as unknown value. The write value should always be 0. |
| 23 to 0 | CMPL[23:0] | All 0 | R/W | The lower limit to judge as expected oscillation.<br>This value is compared with XXXCKMCNT.MONCNT.<br>If XXXCKMCNT.MONCNT is less than CMPL, clock monitor judges it as an error.<br>Don't modify this value while XXXCKMCSR.CHKEN or XXXCKMCSR.ENMON is 1. |

### 8.2.42 XXX Clock Monitor Count Register (XXXCKMCNT)

XXXCKMCNT is a 32-bit readable/writable register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | UPDCNT[6:0] | | | | | | | MONCNT[23:16] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MONCNT[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

### 8.2.43 XXX Clock Monitor Count on Error Register (XXXCKMCNTE)

XXXCKMCNTE is a 32-bit readable/writable register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VLD | UPDCNTE [6:0] | | | | | | | MONCNTE [23:16] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MONCNTE [15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | VLD | All 0 | R | Valid bit.<br>0: UPDCNTE & MONCNTE are showing invalid value.<br>1: UPDCNTE & MONCNTE are showing valid value. |
| 30 to 24 | UPDCNTE[6:0] | All 0 | R | XXXCKMCNT.UPDCNT value when the first error occurred. |
| 23 to 0 | MONCNTE [23:0] | All 0 | R | The counter value of clock monitor when the first error occurred.<br>If monitor clock is stopped, there is a possibility that this register may show the value within the normal range. |

### 8.2.44 YYY Clock Monitor Write Protect Register (YYYCKMWPR)

The specification is same as CPG Write Protect Register (CPGWPR) specification in 8.2.1.
YYY = TOP, VIO, VC, VIP, MM, IR, IRCN, IRDSP, U3DG, RT, RTCR52, PER, HSC.

### 8.2.45 ZZZ Clock Monitor Write Protect Control Register (ZZZCKMWPCR)

The specification is same as CPG Write Protect Control Register (CPGWPCR) specification in 8.2.2.
ZZZ = TOP, VIO, VC, VIP, MM, IR, IRCN, IRDSP, U3DG, RT, RTCR52, PER, HSC.

### 8.2.46 CPG All Clock Monitor Status Register (CPGACKMSR)

CPGACKMSR is a 32-bit readable register.

The registers are applied for Clock Monitor for clocks generated from PLL (Sec.8.5.2.1). The monitor result of each clock monitor is stored on the corresponding bit.

0: Clock oscillates normally.
1: Clock doesn't oscillate normally (Unexpected clock oscillation).
—: Reserved. The bit is always read as 0. The write value should always be 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CHKRES011 | CHKRES010 | CHKRES009 | CHKRES008 | — | CHKRES006 | CHKRES005 | — | CHKRES003 | — | CHKRES001 | CHKRES000 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description (Monitor Clock) |
|---|---|---|---|---|
| 31 | — | 0 | R | — |
| 30 | — | 0 | R | — |
| 29 | — | 0 | R | — |
| 28 | — | 0 | R | — |
| 27 | — | 0 | R | — |
| 26 | — | 0 | R | — |
| 25 | — | 0 | R | — |
| 24 | — | 0 | R | — |
| 23 | — | 0 | R | — |
| 22 | — | 0 | R | — |
| 21 | — | 0 | R | — |
| 20 | — | 0 | R | — |
| 19 | — | 0 | R | — |
| 18 | — | 0 | R | — |
| 17 | — | 0 | R | — |
| 16 | — | 0 | R | — |
| 15 | — | 0 | R | — |
| 14 | — | 0 | R | — |
| 13 | — | 0 | R | — |
| 12 | — | 0 | R | — |
| 11 | CHKRES011 | 0 | R | PLL7FBCK |
| 10 | CHKRES010 | 0 | R | RCLK |
| 9 | CHKRES009 | 0 | R | PLL6FBCK |
| 8 | CHKRES008 | 0 | R | CBFUSA |
| 7 | — | 0 | R | — |
| 6 | CHKRES006 | 0 | R | PLL5FBCK |
| 5 | CHKRES005 | 0 | R | PLL4FBCK |
| 4 | — | 0 | R | — |
| 3 | CHKRES003 | 0 | R | PLL3FBCK |
| 2 | — | 0 | R | — |
| 1 | CHKRES001 | 0 | R | PLL2FBCK |
| 0 | CHKRES000 | 0 | R | PLL1FBCK |

### 8.2.47 TOP All Clock Monitor Status Register (TOPACKMSR)

TOPACKMSR is a 32-bit readable register.

The registers are applied for Clock Monitor in Slave Access Bus domain (Sec.8.5.2.2). The monitor result of each clock monitor is stored on the corresponding bit.

0: Clock oscillates normally.
1: Clock doesn't oscillate normally (Unexpected clock oscillation).
—: Reserved. The bit is always read as 0. The write value should always be 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | CHKRES029 | — | — | — | — | CHKRES024 | — | — | — | — | CHKRES019 | CHKRES018 | CHKRES017 | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHKRES015 | CHKRES014 | CHKRES013 | CHKRES012 | — | — | — | — | CHKRES007 | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description (Monitor Clock) |
|---|---|---|---|---|
| 31 | — | 0 | R | — |
| 30 | — | 0 | R | — |
| 29 | CHKRES029 | 0 | R | RGMII |
| 28 | — | 0 | R | — |
| 27 | — | 0 | R | — |
| 26 | — | 0 | R | — |
| 25 | — | 0 | R | — |
| 24 | CHKRES024 | 0 | R | CL16M |
| 23 | — | 0 | R | — |
| 22 | — | 0 | R | — |
| 21 | — | 0 | R | — |
| 20 | — | 0 | R | — |
| 19 | CHKRES019 | 0 | R | S0D4 |
| 18 | CHKRES018 | 0 | R | S0D3 |
| 17 | CHKRES017 | 0 | R | S0D2 |
| 16 | — | 0 | R | — |
| 15 | CHKRES015 | 0 | R | CBFUSA |
| 14 | CHKRES014 | 0 | R | CPEX |
| 13 | CHKRES013 | 0 | R | RCLK |
| 12 | CHKRES012 | 0 | R | CP |
| 11 | — | 0 | R | — |
| 10 | — | 0 | R | — |
| 9 | — | 0 | R | — |
| 8 | — | 0 | R | — |
| 7 | CHKRES007 | 0 | R | CL |
| 6 | — | 0 | R | — |
| 5 | — | 0 | R | — |
| 4 | — | 0 | R | — |
| 3 | — | 0 | R | — |
| 2 | — | 0 | R | — |
| 1 | — | 0 | R | — |
| 0 | — | 0 | R | — |

### 8.2.48 MM All Clock Monitor Status Register (MMACKMSR)

MMACKMSR is a 32-bit readable register.

The registers are applied for Clock Monitor in Main Memory domain (Sec.8.5.2.3). The monitor result of each clock monitor is stored on the corresponding bit.

0: Clock oscillates normally.
1: Clock doesn't oscillate normally (Unexpected clock oscillation).
—: Reserved. The bit is always read as 0. The write value should always be 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | CHKRES013 | CHKRES012 | CHKRES011 | CHKRES010 | CHKRES009 | — | CHKRES007 | — | CHKRES005 | CHKRES004 | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description (Monitor Clock) |
|---|---|---|---|---|
| 31 | — | 0 | R | — |
| 30 | — | 0 | R | — |
| 29 | — | 0 | R | — |
| 28 | — | 0 | R | — |
| 27 | — | 0 | R | — |
| 26 | — | 0 | R | — |
| 25 | — | 0 | R | — |
| 24 | — | 0 | R | — |
| 23 | — | 0 | R | — |
| 22 | — | 0 | R | — |
| 21 | — | 0 | R | — |
| 20 | — | 0 | R | — |
| 19 | — | 0 | R | — |
| 18 | — | 0 | R | — |
| 17 | — | 0 | R | — |
| 16 | — | 0 | R | — |
| 15 | — | 0 | R | — |
| 14 | — | 0 | R | — |
| 13 | CHKRES013 | 0 | R | RCLK |
| 12 | CHKRES012 | 0 | R | CBFUSA |
| 11 | CHKRES011 | 0 | R | ZB3D4 |
| 10 | CHKRES010 | 0 | R | ZB3D1 |
| 9 | CHKRES009 | 0 | R | S0D4_MM |
| 8 | — | 0 | R | — |
| 7 | CHKRES007 | 0 | R | S0D2_MM |
| 6 | — | 0 | R | — |
| 5 | CHKRES005 | 0 | R | CL16M_MM |
| 4 | CHKRES004 | 0 | R | ZB3D2 |
| 3 | — | 0 | R | — |
| 2 | — | 0 | R | — |
| 1 | — | 0 | R | — |
| 0 | — | 0 | R | — |

### 8.2.49 RT All Clock Monitor Status Register (RTACKMSR)

RTACKMSR is a 32-bit readable register.

The registers are applied for Clock Monitor (ckmrt) in RT (Real Time) domain (Sec.8.5.2.4). The monitor result of each clock monitor is stored on the corresponding bit.

0: Clock oscillates normally.
1: Clock doesn't oscillate normally (Unexpected clock oscillation).
—: Reserved. The bit is always read as 0. The write value should always be 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CHKRES021 | CHKRES020 | — | CHKRES018 | — | CHKRES016 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHKRES015 | CHKRES014 | CHKRES013 | — | CHKRES011 | CHKRES010 | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description (Monitor Clock) |
|---|---|---|---|---|
| 31 | — | 0 | R | — |
| 30 | — | 0 | R | — |
| 29 | — | 0 | R | — |
| 28 | — | 0 | R | — |
| 27 | — | 0 | R | — |
| 26 | — | 0 | R | — |
| 25 | — | 0 | R | — |
| 24 | — | 0 | R | — |
| 23 | — | 0 | R | — |
| 22 | — | 0 | R | — |
| 21 | CHKRES021 | 0 | R | SASYNCRT |
| 20 | CHKRES020 | 0 | R | CBFUSA |
| 19 | — | 0 | R | — |
| 18 | CHKRES018 | 0 | R | S0D24_RT |
| 17 | — | 0 | R | — |
| 16 | CHKRES016 | 0 | R | S0D6_RT |
| 15 | CHKRES015 | 0 | R | S0D4_RT |
| 14 | CHKRES014 | 0 | R | S0D3_RT |
| 13 | CHKRES013 | 0 | R | S0D2_RT |
| 12 | — | 0 | R | — |
| 11 | CHKRES011 | 0 | R | CL16M_RT |
| 10 | CHKRES010 | 0 | R | RCLK |
| 9 | — | 0 | R | — |
| 8 | — | 0 | R | — |
| 7 | — | 0 | R | — |
| 6 | — | 0 | R | — |
| 5 | — | 0 | R | — |
| 4 | — | 0 | R | — |
| 3 | — | 0 | R | — |
| 2 | — | 0 | R | — |
| 1 | — | 0 | R | — |
| 0 | — | 0 | R | — |

### 8.2.50 RT (CR52) All Clock Monitor Status Register (RTCR52ACKMSR)

RTCR52ACKMSR is a 32-bit readable register.

The registers are applied for Clock Monitor (ckmcr52) in RT (Real Time) domain (Sec.8.5.2.4). The monitor result of each clock monitor is stored on the corresponding bit.

0: Clock oscillates normally.
1: Clock doesn't oscillate normally (Unexpected clock oscillation).
—: Reserved. The bit is always read as 0. The write value should always be 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CHKRES008 | CHKRES007 | — | CHKRES005 | CHKRES004 | CHKRES003 | — | — | CHKRES000 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description (Monitor Clock) |
|---|---|---|---|---|
| 31 | — | 0 | R | — |
| 30 | — | 0 | R | — |
| 29 | — | 0 | R | — |
| 28 | — | 0 | R | — |
| 27 | — | 0 | R | — |
| 26 | — | 0 | R | — |
| 25 | — | 0 | R | — |
| 24 | — | 0 | R | — |
| 23 | — | 0 | R | — |
| 22 | — | 0 | R | — |
| 21 | — | 0 | R | — |
| 20 | — | 0 | R | — |
| 19 | — | 0 | R | — |
| 18 | — | 0 | R | — |
| 17 | — | 0 | R | — |
| 16 | — | 0 | R | — |
| 15 | — | 0 | R | — |
| 14 | — | 0 | R | — |
| 13 | — | 0 | R | — |
| 12 | — | 0 | R | — |
| 11 | — | 0 | R | — |
| 10 | — | 0 | R | — |
| 9 | — | 0 | R | — |
| 8 | CHKRES008 | 0 | R | ZR2 |
| 7 | CHKRES007 | 0 | R | ZR1 |
| 6 | — | 0 | R | — |
| 5 | CHKRES005 | 0 | R | S0D2_RT |
| 4 | CHKRES004 | 0 | R | CBFUSA |
| 3 | CHKRES003 | 0 | R | RCLK |
| 2 | — | 0 | R | — |
| 1 | — | 0 | R | — |
| 0 | CHKRES000 | 0 | R | ZR0 |

### 8.2.51 PER All Clock Monitor Status Register (PERACKMSR)

PERACKMSR is a 32-bit readable register.

The registers are applied for Clock Monitor in Peripheral domain (Sec.8.5.2.5). The monitor result of each clock monitor is stored on the corresponding bit.

0: Clock oscillates normally.
1: Clock doesn't oscillate normally (Unexpected clock oscillation).
—: Reserved. The bit is always read as 0. The write value should always be 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | CHKRES030 | CHKRES029 | CHKRES028 | CHKRES027 | CHKRES026 | CHKRES025 | — | CHKRES023 | CHKRES022 | CHKRES021 | CHKRES020 | CHKRES019 | — | — | CHKRES016 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHKRES015 | CHKRES014 | CHKRES013 | — | CHKRES011 | CHKRES010 | CHKRES009 | — | — | — | CHKRES005 | CHKRES004 | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description (Monitor Clock) |
|---|---|---|---|---|
| 31 | — | 0 | R | — |
| 30 | CHKRES030 | 0 | R | SASYNCPERD4 |
| 29 | CHKRES029 | 0 | R | SASYNCPERD2 |
| 28 | CHKRES028 | 0 | R | SASYNCPERD1 |
| 27 | CHKRES027 | 0 | R | CBFUSA |
| 26 | CHKRES026 | 0 | R | ADGH |
| 25 | CHKRES025 | 0 | R | S0D24_PER |
| 24 | — | 0 | R | — |
| 23 | CHKRES023 | 0 | R | S0D12_PER |
| 22 | CHKRES022 | 0 | R | S0D6_PER |
| 21 | CHKRES021 | 0 | R | S0D4_PER |
| 20 | CHKRES020 | 0 | R | S0D3_PER |
| 19 | CHKRES019 | 0 | R | S0D2_PER |
| 18 | — | 0 | R | — |
| 17 | — | 0 | R | — |
| 16 | CHKRES016 | 0 | R | RPCD2 |
| 15 | CHKRES015 | 0 | R | RPC |
| 14 | CHKRES014 | 0 | R | CANFD |
| 13 | CHKRES013 | 0 | R | CL16M_PER |
| 12 | — | 0 | R | — |
| 11 | CHKRES011 | 0 | R | FRAY |
| 10 | CHKRES010 | 0 | R | RCLK |
| 9 | CHKRES009 | 0 | R | MSO |
| 8 | — | 0 | R | — |
| 7 | — | 0 | R | — |
| 6 | — | 0 | R | — |
| 5 | CHKRES005 | 0 | R | SD0 |
| 4 | CHKRES004 | 0 | R | SD0H |
| 3 | — | 0 | R | — |
| 2 | — | 0 | R | — |
| 1 | — | 0 | R | — |
| 0 | — | 0 | R | — |

### 8.2.52 HSC All Clock Monitor Status Register (HSCACKMSR)

HSCACKMSR is a 32-bit readable register.

The registers are applied for Clock Monitor in HSC (High speed Communication) domain (Sec.8.5.2.6). The monitor result of each clock monitor is stored on the corresponding bit.

0: Clock oscillates normally.
1: Clock doesn't oscillate normally (Unexpected clock oscillation).
—: Reserved. The bit is always read as 0. The write value should always be 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CHKRES021 | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | CHKRES012 | CHKRES011 | — | CHKRES009 | CHKRES008 | CHKRES007 | — | CHKRES005 | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description (Monitor Clock) |
|---|---|---|---|---|
| 31 | — | 0 | R | — |
| 30 | — | 0 | R | — |
| 29 | — | 0 | R | — |
| 28 | — | 0 | R | — |
| 27 | — | 0 | R | — |
| 26 | — | 0 | R | — |
| 25 | — | 0 | R | — |
| 24 | — | 0 | R | — |
| 23 | — | 0 | R | — |
| 22 | — | 0 | R | — |
| 21 | CHKRES021 | 0 | R | CBFUSA |
| 20 | — | 0 | R | — |
| 19 | — | 0 | R | — |
| 18 | — | 0 | R | — |
| 17 | — | 0 | R | — |
| 16 | — | 0 | R | — |
| 15 | — | 0 | R | — |
| 14 | — | 0 | R | — |
| 13 | — | 0 | R | — |
| 12 | CHKRES012 | 0 | R | S0D8_HSC |
| 11 | CHKRES011 | 0 | R | S0D4_HSC |
| 10 | — | 0 | R | — |
| 9 | CHKRES009 | 0 | R | S0D2_HSC |
| 8 | CHKRES008 | 0 | R | S0D1_HSC |
| 7 | CHKRES007 | 0 | R | CL16M_HSC |
| 6 | — | 0 | R | — |
| 5 | CHKRES005 | 0 | R | RCLK |
| 4 | — | 0 | R | — |
| 3 | — | 0 | R | — |
| 2 | — | 0 | R | — |
| 1 | — | 0 | R | — |
| 0 | — | 0 | R | — |

### 8.2.53 CPGMA0 Clock Stop Control Register (CPGMA0CKSTPCR)

CPGMA0CKSTPCR is a 32-bit readable register. The register is applied for stopping clocks generated from PLL1 when the corresponding modules are not operated.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | ZXFLG | S0VIOFLG | S0VCFLG | SVVIPFLG | SVIRFLG | IMPAFLG | IMPBFLG | VIOFLG | VCFLG |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | ZXSTP | S0VIOSTP | S0VCSTP | SVVIPSTP | SVIRSTP | IMPASTP | IMPBSTP | VIOSTP | VCSTP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 24 | ZXFLG | 0 | R | Check bit for ZXSTP bit.<br>- 1'b0: Does not activate the ZXSTP setting.<br>- 1'b1: Activates the ZXSTP setting. |
| 23 | S0VIOFLG | 0 | R | Check bit for S0VIOSTP bit.<br>- 1'b0: Does not activate the S0VIOSTP setting.<br>- 1'b1: Activates the S0VIOSTP setting. |
| 22 | S0VCFLG | 0 | R | Check bit for S0VCSTP bit.<br>- 1'b0: Does not activate the S0VCSTP setting.<br>- 1'b1: Activates the S0VCSTP setting. |
| 21 | SVVIPFLG | 0 | R | Check bit for SVVIPSTP bit.<br>- 1'b0: Does not activate the SVVIPSTP setting.<br>- 1'b1: Activates the SVVIPSTP setting. |
| 20 | SVIRFLG | 0 | R | Check bit for SVIRSTP bit.<br>- 1'b0: Does not activate the SVIRSTP setting.<br>- 1'b1: Activates the SVIRSTP setting. |
| 19 | IMPAFLG | 0 | R | Check bit for IMPASTP bit.<br>- 1'b0: Does not activate the IMPASTP setting.<br>- 1'b1: Activates the IMPASTP setting. |
| 18 | IMPBFLG | 0 | R | Check bit for IMPBSTP bit.<br>- 1'b0: Does not activate the IMPBSTP setting.<br>- 1'b1: Activates the IMPBSTP setting. |
| 17 | VIOFLG | 0 | R | Check bit for VIOSTP bit.<br>- 1'b0: Does not activate the VIOSTP setting.<br>- 1'b1: Activates the VIOSTP setting. |
| 16 | VCFLG | 0 | R | Check bit for VCSTP bit.<br>- 1'b0: Does not activate the VCSTP setting.<br>- 1'b1: Activates the VCSTP setting. |
| 15 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | ZXSTP | 0 | R/W | Stop control bit for ZXφ.<br>- 1'b0: Supplies ZXφ<br>- 1'b1: Stops ZXφ. |
| 7 | S0VIOSTP | 0 | R/W | Stop control bit for S0_VIOφ.<br>- 1'b0: Supplies S0_VIOφ.<br>- 1'b1: Stops S0_VIOφ. |
| 6 | S0VCSTP | 0 | R/W | Stop control bit for S0_VCφ.<br>- 1'b0: Supplies S0_VCφ.<br>- 1'b1: Stops S0_VCφ. |
| 5 | SVVIPSTP | 0 | R/W | Stop control bit for SV_VIPφ.<br>- 1'b0: Supplies SV_VIPφ.<br>- 1'b1: Stops SV_VIPφ. |
| 4 | SVIRSTP | 0 | R/W | Stop control bit for SV_IRφ.<br>- 1'b0: Supplies SV_IRφ.<br>- 1'b1: Stops SV_IRφ. |

| | | | | |
|---|---|---|---|---|
| 3 | IMPASTP | 0 | R/W | Stop control bit for IMPASRCφ<br>- 1'b0: Supplies IMPASRCφ.<br>- 1'b1: Stops IMPASRCφ. |
| 2 | IMPBSTP | 0 | R/W | Stop control bit for IMPBSRCφ<br>- 1'b0: Supplies IMPBSRCφ.<br>- 1'b1: Stops IMPBSRCφ. |
| 1 | VIOSTP | 0 | R/W | Stop control bit for VIOSRCφ.<br>- 1'b0: Supplies VIOSRCφ.<br>- 1'b1: Stops VIOSRCφ. |
| 0 | VCSTP | 0 | R/W | Stop control bit for VCSRCφ.<br>- 1'b0: Supplies VCSRCφ.<br>- 1'b1: Stops VCSRCφ. |

### 8.2.54 ADGH Clock Frequency Control Register (ADGHCKCR)

ADGHCKCR is a 32-bit readable/writable register. This register controls the ADGH clock (ADGHφ). This register should be set before ADGH module is operated. Do not access the module during ADGH module is operated.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | - | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | 0 | R/W | Clock Stop.<br>0: Supplies clock to ADGH module<br>1: Stops clock to ADGH module |
| 7 to 0 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### 8.2.55 Write Access Protect Control/Status Register (CPGWPTCSR)

CPGWPTCSR is a 32-bit readable register.

The register is used for a function described in 8.1.2 (4) Bus Access Protection [Functional Safety]. If there are illegal write accesses for CPG and MSTPRST registers (*1), an error notification is reported to ECM (ECMERRxxxR00 Bit29). In addition, the error notification is cleared by using the register (See the description of Bit 0 (ERR)).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | EIE | ERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description (Monitor Clock) |
|---|---|---|---|---|
| 31 to 2 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 1 | EIE | 1 | R/W | Error notification mask bit.<br><br>0:<br>An error notification is masked.<br>Even if there are illegal write accesses for CPG and MSTPRST registers, an error notification is not reported to ECM (ECMERRxxxR00 Bit29).<br>1:<br>An error notification is enable.<br>If there are illegal write accesses for CPG and MSTPRST registers, an error notification is reported to ECM (ECMERRxxxR00 Bit29). |
| 0 | ERR | 0 | R/W | Check result bit.<br>When EIE (Bit 1) is equal to 1'b0, the bit value is always 1'b0.<br>0:<br>An error notification is not reported because there are not any illegal write accesses for CPG and MSTPRST registers.<br>1:<br>An error notification is reported because there are illegal write accesses for CPG and MSTPRST registers. When 1'b1 is written to the bit after the bit value is 1'b1 (the check result shows there are the illegal write accesses for CPG and MSTPRST registers), the error notification is cleared. |

Note 1. Regarding MSTPRST registers, see 9.1.1 (5). In this section, it is described which registers are target of the write access protection function.

## 8.3 Operation

### 8.3.1 Changing Multiplication Ratio of PLL circuit

- PLL multiplication ratio and operation mode (integer mode, fractional mode, SSCG mode) can be changed by the PLL control register. The methods are described in the section below. Figure 8.3.1 shows the procedure for changing the multiplication ratio and operation mode.

- PLL parameter setting is executed by PLLnCR0, PLLnCR1 registers setting. The modules operation must be stopped before setting the PLL parameter setting because the PLL is automatically under standby mode when writing access to PLLnCR0, PLLnCR1. The clock source is changed from PLL to Main OSC when PLL is under standby mode. Therefore, the modules operation cannot be guaranteed when the Figure 8.3.1 procedure is not kept.

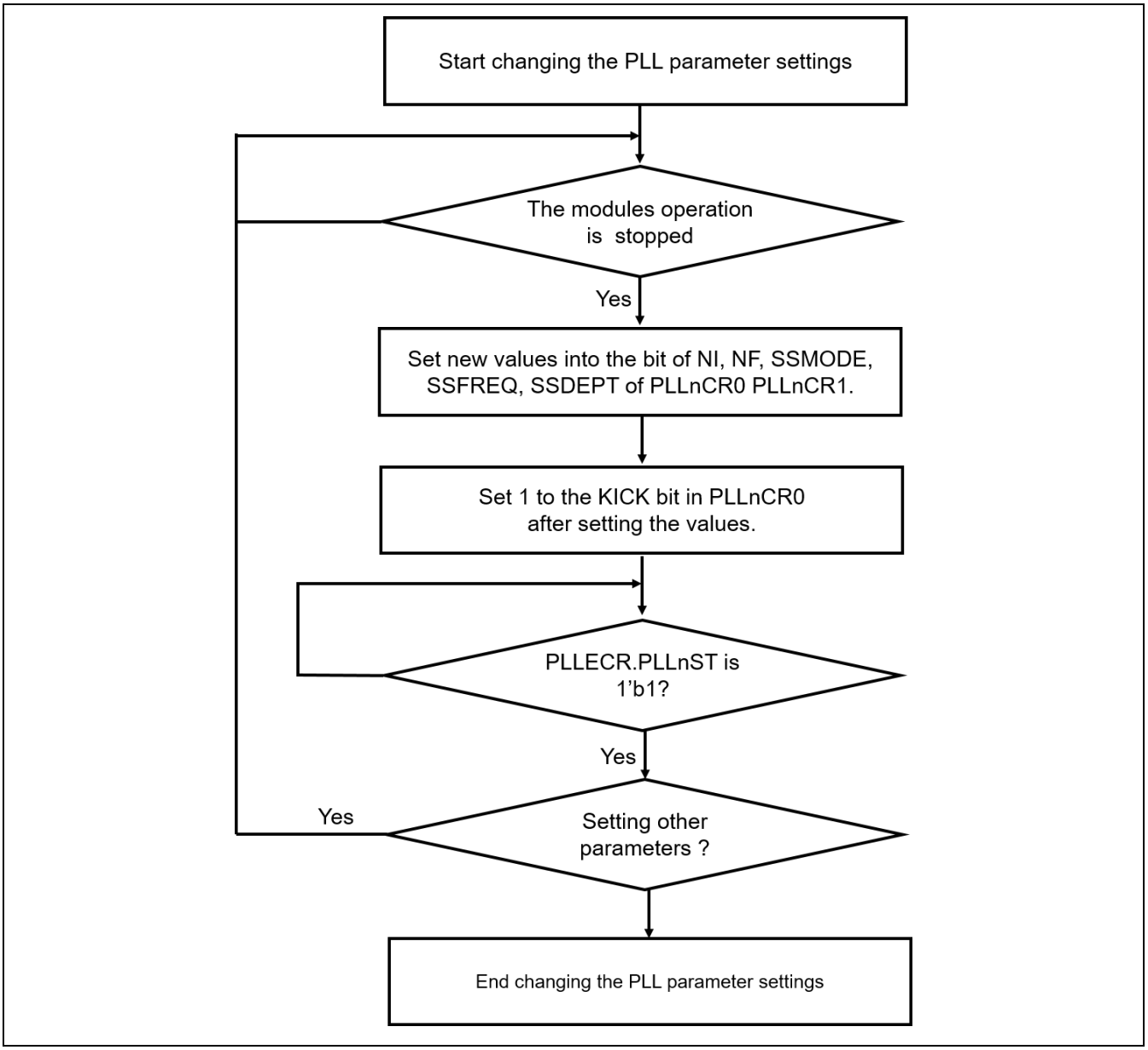



**Figure 8.3.1 PLLn Parameter Setting Procedure**

#### 8.3.1.1 PLL Circuit 1

- PLL1 parameter setting is prohibited by software registers because the system operation cannot be guaranteed. Therefore, although there are control registers (PLL1CR0, PLL1CR1), the write access to the registers is prohibited (only the read access can be accepted).

- Initial parameter setting can be executed by MD pins (MD13, MD14, MD36, MD37). The detail information for the parameter setting is shown in Table 8.1.5.1.

#### 8.3.1.2 PLL Circuit 2

- PLL2 parameter setting is executed by writing access to PLL2CR0, PLL2CR1. The initial PLL2 operation mode is the integer mode. The detail parameter information is shown in Table 8.1.5.2.

- Table 8.1.5.2 shows one example of the PLL2 parameter setting. When the parameter setting is necessary so that PLL2 can be operated with the fractional mode or the SSCG (Down spread) mode, see the Figure 8.3.1.

#### 8.3.1.3 PLL Circuit 3

- PLL3 parameter setting is executed by writing access to PLL3CR0, PLL3CR1. The parameter setting is necessary as described in Sec. 8.1.5.3 because PLL3 can be operated only with the integer mode. When the multiplication of PLL3 is changed, see the Figure 8.3.1.

#### 8.3.1.4 PLL Circuit 4

- PLL4 parameter setting is executed by writing access to PLL4CR0, PLL4CR1. The initial PLL4 operation mode is the integer mode. The detail parameter information is shown in Table 8.1.5.4.

- Table 8.1.5.4 shows one example of the PLL4 parameter setting. When the parameter setting is necessary so that PLL4 can be operated with the fractional mode or the SSCG (Down spread) mode, see the Figure 8.3.1.

#### 8.3.1.5 PLL Circuit 5

- As shown in Sec.8.1.5.5, there are no control registers for PLL5 parameter setting. Table 8.1.5.5 shows the initial PLL5 parameter setting which depends on MD pins (MD13, MD14).

#### 8.3.1.6 PLL Circuit 6

- PLL6 parameter setting is executed by writing access to PLL6CR0, PLL6CR1. The initial PLL6 operation mode is the integer mode. The detail parameter information is shown in Table 8.1.5.6.

- Table 8.1.5.6 shows one example of the PLL6 parameter setting. When the parameter setting is necessary so that PLL6 can be operated with the fractional mode or the SSCG (Down spread) mode, see the Figure 8.3.1.

#### 8.3.1.7 PLL Circuit 7

- As shown in Sec.8.1.5.7, there are no control registers for PLL7 parameter setting. Table 8.1.5.7 shows the initial PLL7 parameter setting which depends on MD pins (MD13, MD14).

### 8.3.2 Changing Division Ratio of Cortex-A76, Cortex-R52, Debug Trace, 3DG Clock

Changing the frequency of Z2φ, ZRφ, ZSφ, ZTφ, ZTRφ, and ZGφ can be done by modifying each set of bits for setting the division ratio in FRQCRB and FRQCRC0 and FRQCRC1. After setting new ratio, write 1 to the KICK bit in FRQCRB to start the division ratio change. Before changing the division ratio, the KICK bit must be checked for 0. After 1 is written to the KICK bit in FRQCRB, 0 can be read when the change of new division ratio is completed. While the setting is being changed with the KICK bit being 1, do not modify FRQCRB and FRQCRC0 and FRQCRC1. When the KICK bit is read as 1, do not write 0 into the KICK bit.

### 8.3.3 Changing Division Ratio for DDR, DBSC clock

Changing the frequency of ZBφ, ZB3D2φ, ZB3D4φ can be done by modifying ZB3FC bits for setting the division ratio in FRQCRD. After setting new ratio, write 1 to the KICK bit in FRQCRD to start the division ratio change. Before changing the division ratio, the KICK bit must be checked for 0. After 1 is written to the KICK bit in FRQCRD, 0 can be read when the change of new division ratio is completed. While the setting is being changed with the KICK bit being 1, do not modify FRQCRD. When the KICK bit is read as 1, do not write 0 into the KICK bit.

### 8.3.4 Changing Division Ratio for any other clocks

If “Variable” is filled on “Dividing ratio” column in Table 8.1.4a ~ Table 8.1.4g, the corresponding clock division ratio can be changed by the clock control register. See Sec.8.2 Register Description which corresponds with the target clock when the clock division ratio is changed.

## 8.4 Usage Notes

### 8.4.1 Write Protect Function for CPG registers

**(1) Protected Registers**

All registers in CPG (registers described in the section 8 CPG and section 9 Module Standby, Software Reset) except for CPGWPR can be protected from unexpected write operation when CPGWPCR.WPE = 1. When CPGWPCR.WPE = 1, CPG registers can't be written without using CPGWPR before the write access to the target register.

**(2) How to Write to CPG Registers When CPGWPCR.WPE = 1**

When CPGWPCR.WPE = 1, writing to CPRWPR is required before writing to the target register. The sample procedure is shown below (the case in which you want to write H'AAAA 5555 to the target register).

1. Write H'5555 AAAA (= inverted value of H'AAAA 5555) to CPGWPR
2. Write H'AAAA 5555 to the target register

Then, the target register is written by H'AAAA 5555.
When you skip the first step above, you can't write H'AAAA 5555 to the target register.

**(3) How to Clear CPGWPCR.WPE**

When CPGWPCR.WPE = 1, the bit can be cleared by the following procedure.

0. CPGWPCR.WPE is 1
1. Write H'5A5A FFFF to CPGWPR
2. Write H'A5A5 0000 to CPGWPCR
3. CPGWPCR.WPE becomes 0

When CPGWPCR.WPE = 0, the write protect function is disabled. Registers can be written directly without using CPGWPR.

### 8.4.2 Bus Access Protection for CPG registers

#### (1) Overview

R-Car V4M system allows to access CPG registers from multiple domains. To avoid the access confliction when there are multiple accesses for different domains, APB Bus is available to control the domain access by each 16 KB register address area as shown in Figure 8.4.1. The purpose is to enhance the security protect and safety protect of CPG register accesses.

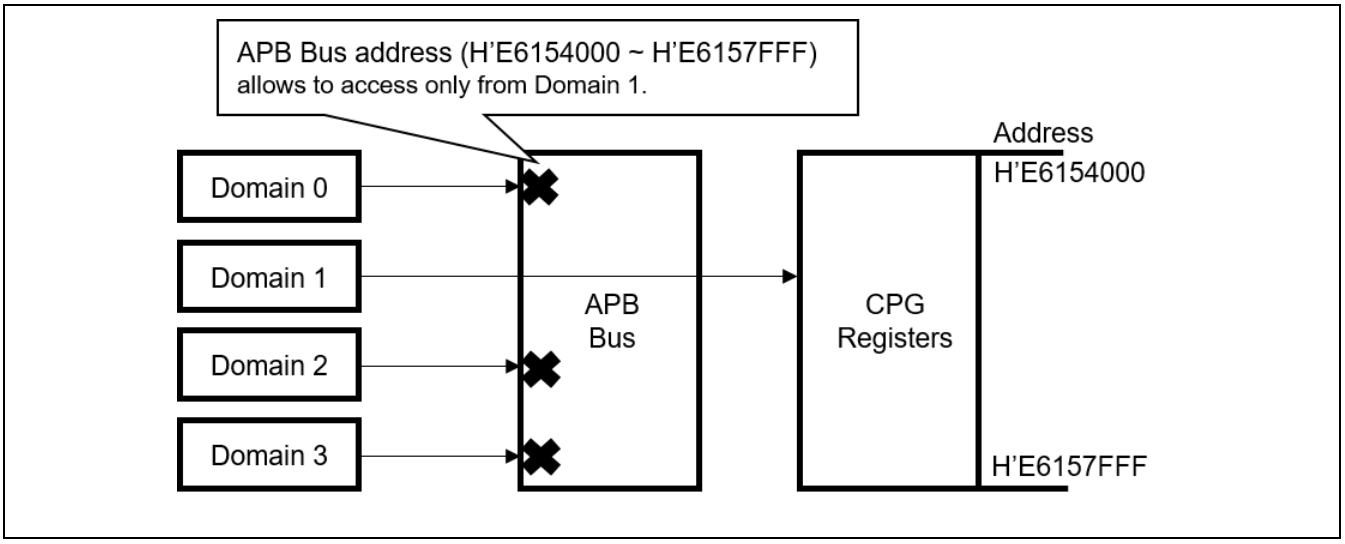



**Figure 8.4.1 Bus Access Protection for CPG registers**

CPG register address map is assigned from H'E6150000 to H'E615FFFF. The address map is separated into 4 groups which is accessed by each domain (Domain 0, Domain 1, Domain 2, Domain 3). The Physical address of CPG register body from 0x0000_0000 to 0x0000_3FFF.

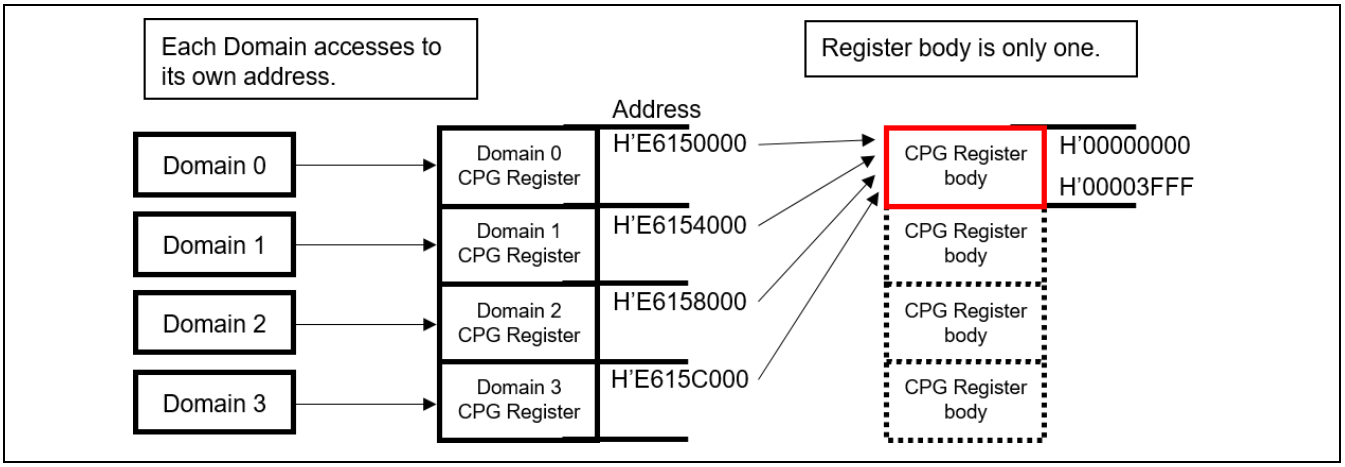



**Figure 8.4.2 Address Map Structures**

### (2) Domain x Write Access Control Register (DxWACR) (x = 0, 1, 2, 3)

Each register in “CPG Register body” is protected by “Domain x Write Address Control Register (DxWACR (x = 0, 1, 2, 3)). For example of the case below (Figure 8.4.3), D2WACR.bit1 = 1, D{0, 1, 3}WACR.bit1 = 0. Only Domain 2 can write access to “Register 1”.

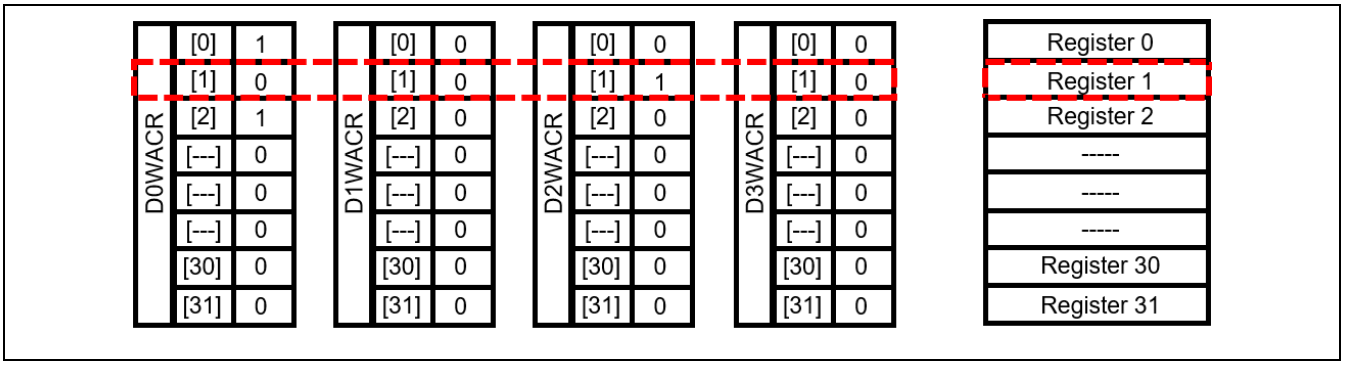



**Figure 8.4.3 Usage of D0WACR, D1WACR, D2WACR, D3WACR**

The register corresponds with D0WACRAn, D1WACRAn, D2WACRAn, D3WACRAn (n = 0 to 63) in Sec. 8.2.35, 8.2.36, 8.2.37, 8.2.38.

## 8.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 8.5.1 Overview

- CPG has Clock Monitor (a safety mechanism) in CPG itself to detect incorrect frequency of the clock generated from PLLn (n = 1, 2, 3, 4, 5, 6, 7). In addition, RCLK and CBFUSAφ is monitored by the Clock Monitor. It is described in 8.5.2.1.
- In each bus domain, the Clock Monitor is located to detect incorrect frequency of any clocks. It is described in 8.5.2.2 ~ 8.5.2.6.

### 8.5.2 Clock Monitor

Clock Monitor counts the rising edge of the monitor clock for N cycles of sampling clock (N value depends on the type of the monitor clock as shown in Table 8.5.2.1.1, Table 8.5.2.2.1, Table 8.5.2.3.1, Table 8.5.2.4.1a, Table 8.5.2.4.1b, Table 8.5.2.5.1, Table 8.5.2.6.1). When the counted value is out of upper/lower limit or monitor clock is stopped for a certain period, clock Monitor reports an error to ECM.

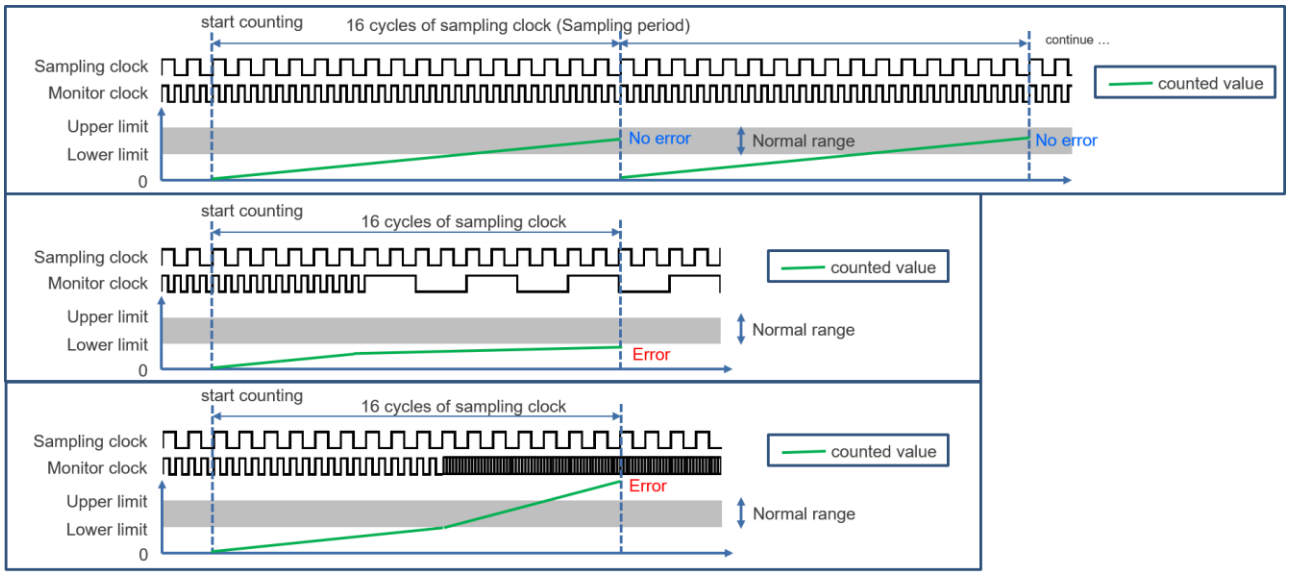



**Figure 8.5.2.1 Clock monitor counter Timing Chart (Ex. 16 cycles (N=4) of sampling clock)**

### 8.5.2.1 Clock Monitor for clocks generated from PLL

#### (1) Overview

Clock Monitor Counter exists in CPG module as shown in Figure 8.5.2.2. The clock monitor counter monitor whether the clock frequency for PLLnFBCK (n = 1, 2, 3, 4, 5, 6, 7), CBFUSAφ, RCLK is normal or not.

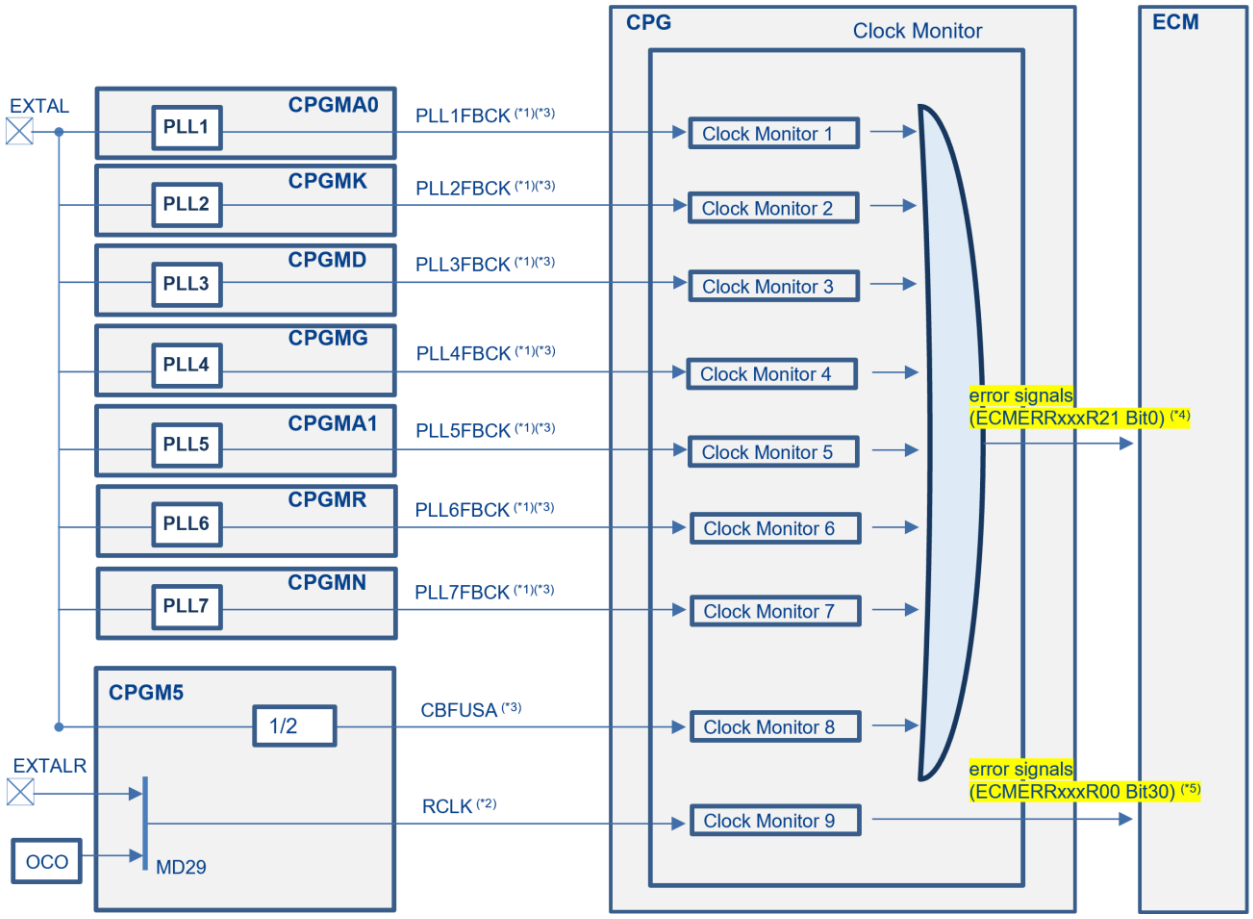



**Figure 8.5.2.2 Block diagram of Clock Monitor Counter**

Note *1: Frequency of Monitor clock (PLL1FBCK, PLL2FBCK, PLL3FBCK, PLL4FBCK, PLL5FBCK, PLL6FBCK, PLL7FBCK) is calculated as the following.

| Monitor clock | Frequency [MHz] |
|---|---|
| PLL1FBCK | (PLL1VCO x 1/2) x 1/96 |
| PLL2FBCK | (PLL2VCO x 1/4) x 1/12 |
| PLL3FBCK | (PLL3VCO x 1/2) x 1/64 |
| PLL4FBCK | (PLL4VCO x 1/2) x 1/12 |
| PLL5FBCK | (PLL5VCO x 1/2) x 1/96 |
| PLL6FBCK | (PLL6VCO x 1/2) x 1/12 |
| PLL7FBCK | (PLL7VCO x 1/2) x 1/24 |

Note *2: In case that RCLK is monitored, the sampling clock is CBFUSAφ.
Note *3: In case of monitoring clocks except RCLK, the sampling clock is RCLK.
Note *4: When PLLnFBCK (n = 1, 2, 3, 4, 5, 6, 7) and CBFUSAφ are incorrect frequency, the error signal is reported to ECM.
Note *5: When RCLK is incorrect frequency, the error signal is reported to ECM.

**(2) Register Configuration**

See Table 8.1.6.1b.

**(3) Register Description**

XXX = PLL1FB, PLL2FB, PLL3FB, PLL4FB, PLL5FB, PLL6FB, PLL7FB, CBFUSA, RCLK.

**(a) XXX Clock Monitor Control / Status Register (XXXCKMCSR)**

See 8.2.39.

**(b) XXX Clock Monitor Error Clear Register (XXXCKMECR)**

See 8.2.40.

**(c) XXX Clock Monitor Limit Count H Register (XXXCKMLCH)**

See 8.2.41.

**(d) XXX Clock Monitor Limit Count L Register (XXXCKMLCL)**

See 8.2.42.

**(e) XXX Clock Monitor Count Register (XXXCKMCNT)**

See 8.2.43.

**(f) XXX Clock Monitor Count on Error Register (XXXCKMCNTE)**

See 8.2.44.

**(g) CPG All Clock Monitor Status Register (CPGACKMSR)**

See 8.2.47.

### (4) Operation

All registers in Clock Monitor except for CPGWPR can be protected from unexpected write operation when CPGWPCR.WPE = 1. When CPGWPCR.WPE = 1, CPG registers cannot be written without using CPGWPR before the write access to the target register. In detail, see the Sec. 8.4.1. The write access usage is same as it.

[Initial Check]
1. Check if the reset for Clock Monitor is negated.
   I. Wait until XXXCKMCSR.RSTMSMP == 1 and XXXCKMCSR.RSTMMON == 1

[Run Clock Monitor]
1. Set limit count value (*1).
   I. Write value to XXXCKMLCH.CMPH & XXXCKMLCL.CMPL.

2. Run Clock Monitor.
   I. Set XXXCKMCSR.CHKEN = 1.
      Set also XXXCKMCSR.INTEN = 1 if you want to report error to MFIS.
   II. Wait until AAACKMSSR.CHKEN = 1 and AAACKMCSR.INTEN = 1

3. Check the status of Clock Monitor.
   I. Read check XXXCKMCSR.ENMON == 1.
   II. Read check XXXCKMCSR.CLRMON == 0.
   III. Wait until AAACKMCSR.ENMON == 1 and AAACKMCSR.CLRMON == 0.

4. Check the result (*2).
   I. Read out XXXCKMCSR.CHKRES status

Note *1: Modifying limit count value is not allowed while Clock Monitor is running.
Note *2: When INTEN=1 (MFIS notification function is enable), skipping the step 4 is allowed.

[Stop clock monitor & Clear error]
1. Stop clock monitor.
   I. Set XXXCKMCSR.CHKEN = 0.
   II. Wait until XXXCKMCSR.ENMON == 0.

2. Clear error
   I. Set XXXCKMECR.CLRRES = 1
   II. Wait until XXXCKMCSR.CHKRES == 0
   III. Set XXXCKMECR.CLRRES = 0

Note: Setting XXXCKMECR.CLRRES = 1 is not allowed while Clock Monitor is running.

[Error Injection – Clock Monitor Test]

1. Check if the reset for Clock Monitor is negated.
   I. Wait until XXXCKMCSR.RSTMSMP == 1 and XXXCKMCSR.RSTMMON == 1

2. Set limit count value.
   I. Write test value to XXXCKMLCH.CMPH & XXXCKMLCL.CMPL.
   (See 8A in the calculation example for the test settings.)
   II. Confirm that the setting has been written to those registers.

3. Run Clock Monitor.
   I. Set XXXCKMCSR.CHKEN = 1
   Set also XXXCKMCSR.INTEN = 1 if you want to report error to MFIS.

4. Check the status of Clock Monitor
   I. Read check XXXCKMCSR.ENMON == 1.
   II. Read check XXXCKMCSR.CLRMON == 0.

5. Check the result.
   I. Wait for the clock monitor to detect error. (Wait 26 cycles of SMPCK).
   II. Read out XXXCKMCSR.CHKRES status.
   III. Go back to the step i while the status is 0 up to 3 times.

   The test is passed when the status is 1 since the clock monitor detects the pseudo error.
   If the status read is 0 for the third check, quit all the tests and follow the fault control process.

6. Stop Clock Monitor
   I. Set XXXCKMCSR.CHKEN = 0
   II. Wait until XXXCKMCSR.ENMON == 0

7. Clear error
   I. Set XXXCKMECR.CLRRES = 1
   II. Wait until XXXCKMCSR.CHKRES == 0
   III. Set XXXCKMECR.CLRRES = 0

8. Go back to the step 2 unless all the tests have finished.

8A.
[Example of calculating MONCNT value]
fs = Frequency of sampling clock = 16.6MHz
fm = Frequency of monitor clock = 133MHz (N=4)

Expected MONCNT value = fm x ($2^N$ / fs) = 133 x ($2^N$ / 16.6) = 128

**Table 8.5.2.1.1 N value for Monitor clock during calculating MONCNT value**

| Monitor clock | N value |
|---|---|
| PLL1FB, PLL2FB, PLL3FB, PLL4FB, PLL5FB, PLL6FB, PLL7FB, CBFUSA | 4 |
| - | 8 |
| RCLK | 16 |

To generate the pseudo error, the limit values are calculated as follows:

[For Upper Limit Test]
- The upper limit is set at half of the expected value.
- The lower limit is set at H'0000 0000.

[For Lower Limit Test]
- The lower limit is set at twice of the expected value.
- The upper limit is set at H'00FF FFFF.

### 8.5.2.2 Clock Monitor in Slave Access Bus domain

#### (1) Overview

A Clock Monitor Counter exists in Slave Access Bus domain as shown in Figure 8.5.2.3 to monitor whether the clock frequency is normal or not. One clock is monitored by one clock monitor counter inside Clock Monitor.

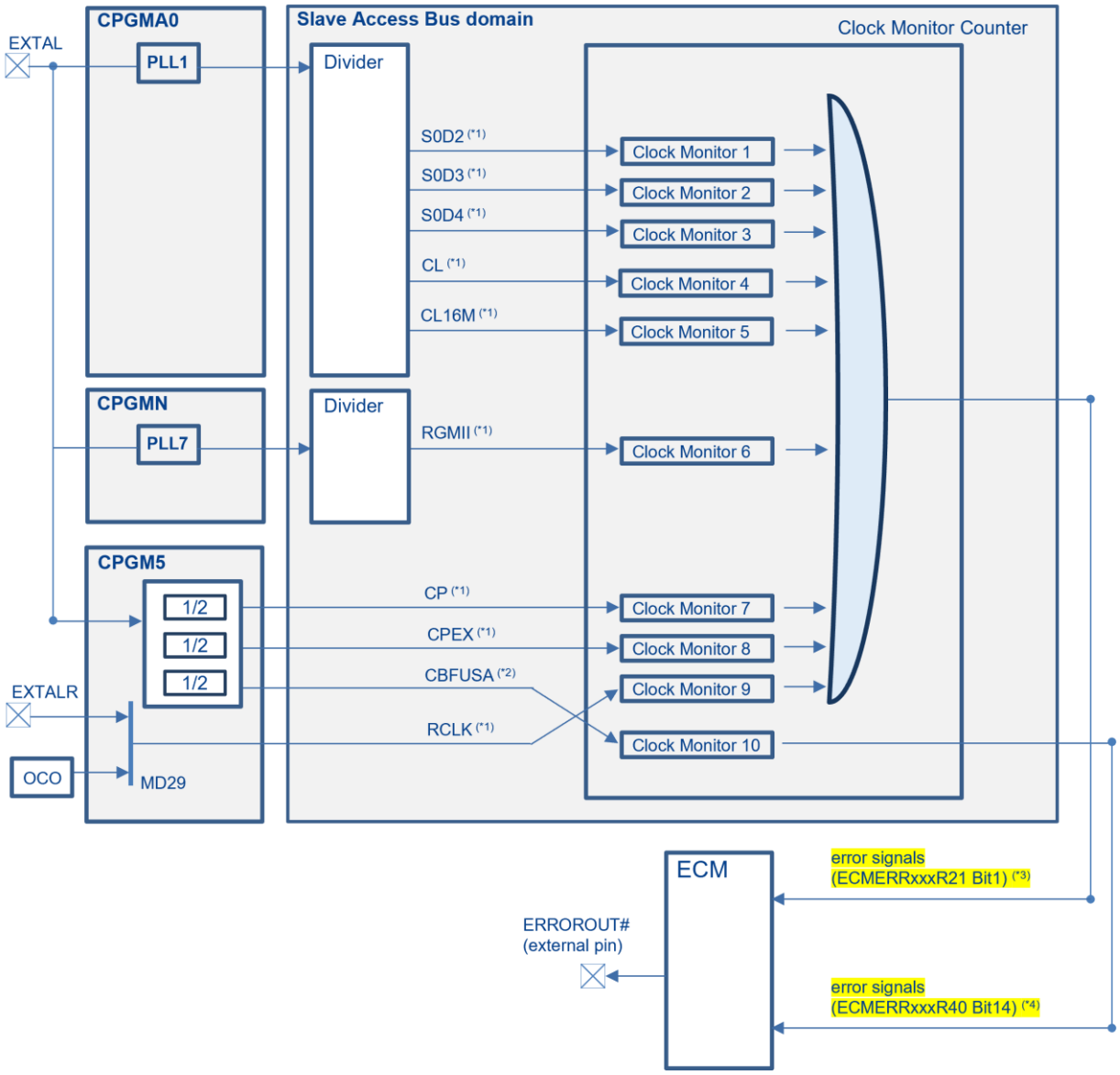



**Figure 8.5.2.3 Block diagram of Clock Monitor Counter**

Note *1 In case that CBFUSAφ is monitored, the sampling clock is RCLK. CBFUSAφ = EXTAL/2.
Note *2 In case of monitoring clocks except CBFUSAφ, the sampling clock is CBFUSAφ.
Note *3 Even if one of all clocks except CBFUSA has incorrect frequency, the error signal is reported to ECM.
Note *4 When CBFUSAφ is incorrect frequency, the error signal is reported to ECM via different notification path from any other clocks.

**(2) Register Configuration**

See Table 8.1.6.2.1.

### (3) Register Description

XXX = S0D2, S0D3, S0D4, CL, CL16M, RGMII, CP, CPEX, CBFUSATOP, RCLKTOP.

#### (a) XXX Clock Monitor Control / Status Register (XXXCKMCSR)

See 8.2.39.

#### (b) XXX Clock Monitor Error Clear Register (XXXCKMECR)

See 8.2.40.

#### (c) XXX Clock Monitor Limit Count H Register (XXXCKMLCH)

See 8.2.41.

#### (d) XXX Clock Monitor Limit Count L Register (XXXCKMLCL)

See 8.2.42.

#### (e) XXX Clock Monitor Count Register (XXXCKMCNT)

See 8.2.43.

#### (f) XXX Clock Monitor Count on Error Register (XXXCKMCNTE)

See 8.2.44.

#### (g) TOP Clock Monitor Write Protect Register (TOPCKMWPR)

The specification is same as CPG Write Protect Register (CPGWPR) specification in Sec.8.2.1. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.2 (3).

#### (h) TOP Clock Monitor Write Protect Control Register (TOPCKMWPCR)

The specification is same as CPG Write Protect Control Register (CPGWPCR) specification in Sec.8.2.2. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.2 (3).

#### (i) TOP All Clock Monitor Status Register (TOPACKMSR)

See 8.2.48.

**(4) Operation**

- All registers in Clock Monitor except TOPCKMWPR can be protected from unexpected write operation when TOPCKMWPCR.WPE = 1. When TOPCKMWPCR.WPE = 1, CPG registers cannot be written without using TOPCKMWPR before the write access to the target register. In detail, see the Sec. 8.4.1. The write access usage is same as it.

- The other operation procedure is same as that of 8.5.2.1 (4).

- See Table 8.5.2.2.1 about N cycles information of the sampling clock.

**Table 8.5.2.2.1 N value for Monitor clock during calculating MONCNT value**

| Monitor clock | N value |
|---|---|
| S0D2, S0D3, S0D4, RGMII, CBFUSA | 4 |
| CL, CL16M, CP, CPEX | 8 |
| RCLK | 16 |

### 8.5.2.3 Clock Monitor in Main Memory domain

#### (1) Overview

A Clock Monitor Counter exists in Main Memory domain as shown in Figure 8.5.2.4 to monitor whether the clock frequency is normal or not. One clock is monitored by one clock monitor counter inside Clock Monitor.

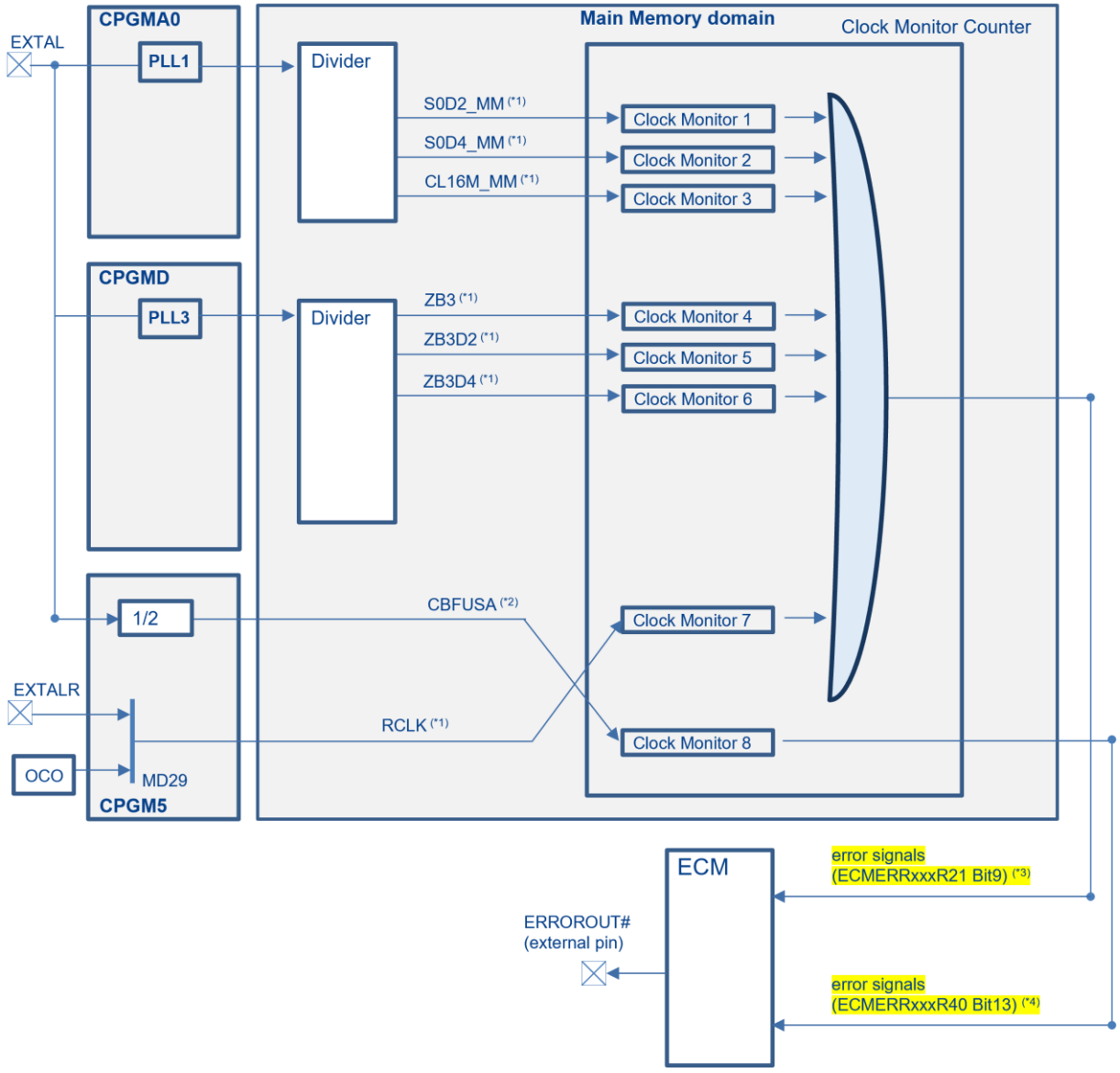



**Figure 8.5.2.4 Block diagram of Clock Monitor Counter**

Note *1 In case that CBFUSAφ is monitored, the sampling clock is RCLK. CBFUSAφ = EXTAL/2.
Note *2 In case of monitoring clocks except CBFUSAφ, the sampling clock is CBFUSAφ.
Note *3 Even if one of all clocks except CBFUSA has incorrect frequency, the error signal is reported to ECM.
Note *4 When CBFUSAφ is incorrect frequency, the error signal is reported to ECM via different notification path from any other clocks.

**(2) Register Configuration**

See Table 8.1.6.2.2.

**(3) Register Description**

XXX = S0D2MM, S0D4MM, CL16MMM, ZB3, ZB3D2, ZB3D4, CBFUSAMM, RCLKMM.

**(a) XXX Clock Monitor Control / Status Register (XXXCKMCSR)**

See 8.2.39.

**(b) XXX Clock Monitor Error Clear Register (XXXCKMECR)**

See 8.2.40.

**(c) XXX Clock Monitor Limit Count H Register (XXXCKMLCH)**

See 8.2.41.

**(d) XXX Clock Monitor Limit Count L Register (XXXCKMLCL)**

See 8.2.42.

**(e) XXX Clock Monitor Count Register (XXXCKMCNT)**

See 8.2.43.

**(f) XXX Clock Monitor Count on Error Register (XXXCKMCNTE)**

See 8.2.44.

**(g) MM Clock Monitor Write Protect Register (MMCKMWPR)**

The specification is same as CPG Write Protect Register (CPGWPR) specification in Sec.8.2.1. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.3 (3).

**(h) MM Clock Monitor Write Protect Control Register (MMCKMWPCR)**

The specification is same as CPG Write Protect Control Register (CPGWPCR) specification in Sec.8.2.2. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.3 (3).

**(i) MM All Clock Monitor Status Register (MMACKMSR)**

See 8.2.49.

**(4) Operation**

- All registers in Clock Monitor except MMCKMWPR can be protected from unexpected write operation when MMCKMWPCR.WPE = 1. When MMCKMWPCR.WPE = 1, CPG registers cannot be written without using MMCKMWPR before the write access to the target register. In detail, see the Sec. 8.4.1. The write access usage is same as it.

- The other operation procedure is same as that of 8.5.2.1 (4).

- See Table 8.5.2.3.1 about N cycles information of the sampling clock.

**Table 8.5.2.3.1 N value for Monitor clock during calculating MONCNT value**

| Monitor clock | N value |
|---|---|
| S0D2_MM, S0D4_MM, CBFUSA | 4 |
| CL16M_MM, ZB3, ZB3D2, ZB3D4 | 8 |
| RCLK | 16 |

### 8.5.2.4 Clock Monitor in RT (Real Time) domain

#### (1) Overview

Clock Monitor Counter exists in RT (Real Time) domain as shown in Figure 8.5.2.5 to monitor whether the clock frequency is normal or not. One clock is monitored by one clock monitor counter inside Clock Monitor. Clock Monitor (ckmcr52) monitors the clock which is provided for Cortex-R52 related modules. The clock for any other related modules is monitored by Clock Monitor (ckmrt).

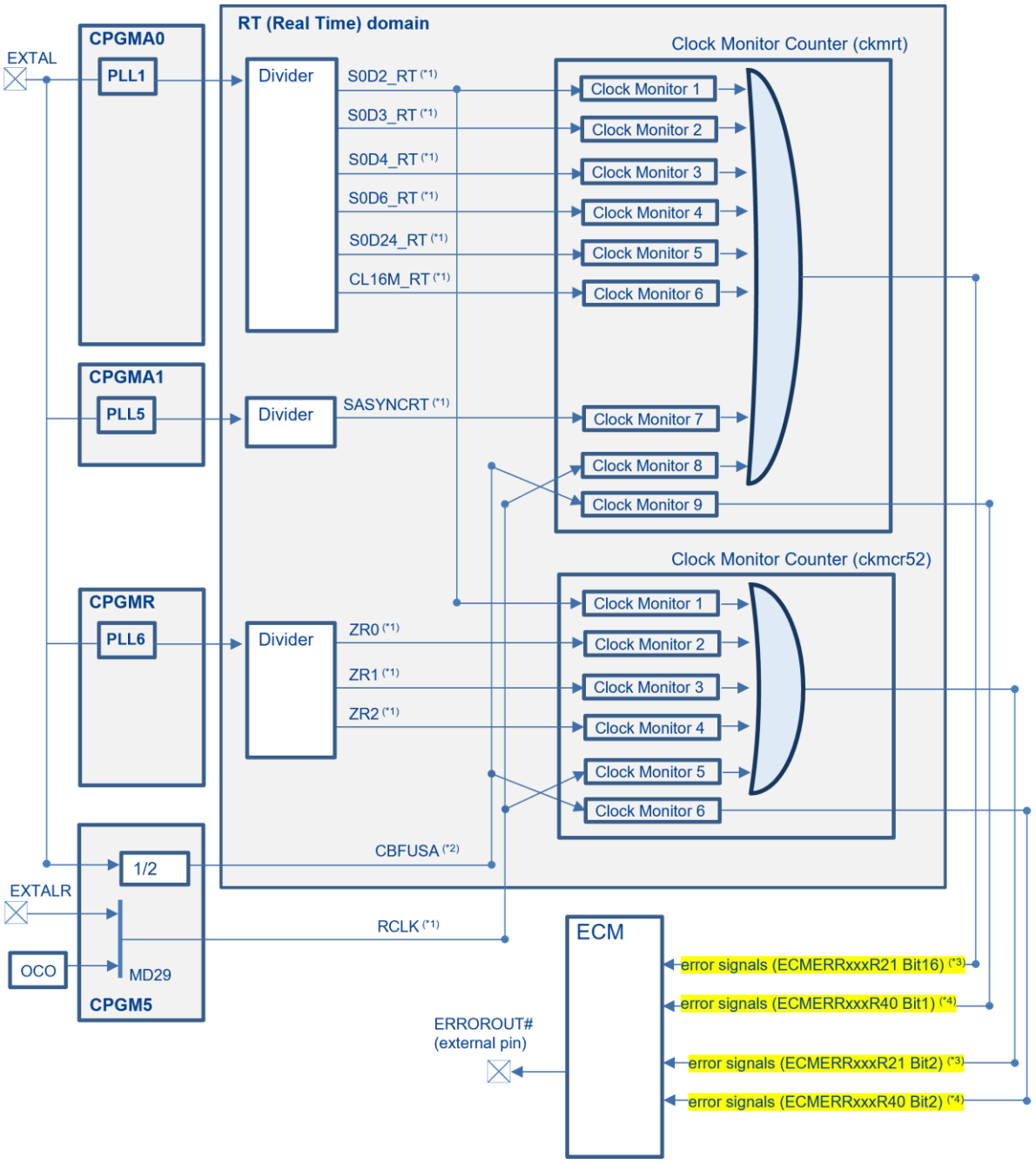



**Figure 8.5.2.5 Block diagram of Clock Monitor Counter in RT (Real Time) domain**

Note *1 In case that CBFUSAφ is monitored, the sampling clock is RCLK. CBFUSAφ = EXTAL/2.
In addition, S0D24_RTφ monitor is not mandatory because the clock is not used by any modules.
Note *2 In case of monitoring clocks except CBFUSAφ, the sampling clock is CBFUSAφ.
Note *3 Even if one of all clocks except CBFUSA has incorrect frequency, the error signal is reported to ECM.
Note *4 When CBFUSAφ is incorrect frequency, the error signal is reported to ECM via different notification path from any other clocks.

**(2) Register Configuration**

See Table 8.1.6.2.3a, 8.1.6.2.3b.

### (3) Register Description

#### 1. Clock Monitor Counter (ckmrt)

XXX = S0D2RT, S0D3RT, S0D4RT, S0D6RT, S0D24RT, CL16MRT, SASYNCRT, CBFUSART, RCLKRT.

**(a) XXX Clock Monitor Control / Status Register (XXXCKMCSR)**

See 8.2.39.

**(b) XXX Clock Monitor Error Clear Register (XXXCKMECR)**

See 8.2.40.

**(c) XXX Clock Monitor Limit Count H Register (XXXCKMLCH)**

See 8.2.41.

**(d) XXX Clock Monitor Limit Count L Register (XXXCKMLCL)**

See 8.2.42.

**(e) XXX Clock Monitor Count Register (XXXCKMCNT)**

See 8.2.43.

**(f) XXX Clock Monitor Count on Error Register (XXXCKMCNTE)**

See 8.2.44.

**(g) RT Clock Monitor Write Protect Register (RTCKMWPR)**

The specification is same as CPG Write Protect Register (CPGWPR) specification in Sec.8.2.1. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.4 (3) 1.

**(h) RT Clock Monitor Write Protect Control Register (RTCKMWPCR)**

The specification is same as CPG Write Protect Control Register (CPGWPCR) specification in Sec.8.2.2. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.4 (3) 1.

**(i) RT All Clock Monitor Status Register (RTACKMSR)**

See 8.2.50.

**2. Clock Monitor Counter (ckmcr52)**

YYY = S0D2RTCR52, ZR0CR52, ZR1CR52, ZR2CR52, CBFUSARTCR52, RCLKRTCR52.

**(j) YYY Clock Monitor Control / Status Register (YYYCKMCSR)**

See 8.2.39.

**(k) YYY Clock Monitor Error Clear Register (YYYCKMECR)**

See 8.2.40.

**(l) YYY Clock Monitor Limit Count H Register (YYYCKMLCH)**

See 8.2.41.

**(m) YYY Clock Monitor Limit Count L Register (YYYCKMLCL)**

See 8.2.42.

**(n) YYY Clock Monitor Count Register (YYYCKMCNT)**

See 8.2.43.

**(o) YYY Clock Monitor Count on Error Register (YYYCKMCNTE)**

See 8.2.44.

**(p) RTCR52 Clock Monitor Write Protect Register (RTCR52CKMWPR)**

The specification is same as CPG Write Protect Register (CPGWPR) specification in Sec.8.2.1. The register is effective for (j), (k), (l), (m), (n), (o) registers in 8.5.2.4 (3) 2.

**(q) RTCR52 Clock Monitor Write Protect Control Register (RTCR52CKMWPCR)**

The specification is same as CPG Write Protect Control Register (CPGWPCR) specification in Sec.8.2.2. The register is effective for (j), (k), (l), (m), (n), (o) registers in 8.5.2.4 (3) 2.

**(r) RTCR52 All Clock Monitor Status Register (RTCR52ACKMSR)**

See 8.2.51.

### (4) Operation

**1. Clock Monitor Counter (ckmrt)**

- All registers in Clock Monitor except RTCKMRTWPR can be protected from unexpected write operation when RTCKMRTWPCR.WPE = 1. When RTCKMRTWPCR.WPE = 1, CPG registers cannot be written without using RTCKMRTWPR before the write access to the target register. In detail, see the Sec. 8.4.1. The write access usage is same as it.

- The other operation procedure is same as that of 8.5.2.1 (4).

- See Table 8.5.2.4.1a about N cycles information of the sampling clock.

**Table 8.5.2.4.1a N value for Monitor clock during calculating MONCNT value**

| Monitor clock | N value |
|---|---|
| S0D2_RT, S0D3_RT, S0D4_RT, S0D6_RT, CBFUSA | 4 |
| S0D24_RT, CL16M_RT, SASYNCRT | 8 |
| RCLK | 16 |

**2. Clock Monitor Counter (ckmcr52)**

- All registers in Clock Monitor except RTCKMCR52WPR can be protected from unexpected write operation when RTCKMCR52WPCR.WPE = 1. When RTCKMCR52WPCR.WPE = 1, CPG registers cannot be written without using RTCKMCR52WPR before the write access to the target register. In detail, see the Sec. 8.4.1. The write access usage is same as it.

- The other operation procedure is same as that of 8.5.2.1 (4).

- See Table 8.5.2.4.1b about N cycles information of the sampling clock.

**Table 8.5.2.4.1b N value for Monitor clock during calculating MONCNT value**

| Monitor clock | N value |
|---|---|
| S0D2_RT, CBFUSA | 4 |
| ZR0, ZR1, ZR2 | 8 |
| RCLK | 16 |

### 8.5.2.5 Clock Monitor in Peripheral domain

#### (1) Overview

A Clock Monitor Counter exists in Peripheral domain as shown in Figure 8.5.2.6 to monitor whether the clock frequency is normal or not. One clock is monitored by one clock monitor counter inside Clock Monitor.

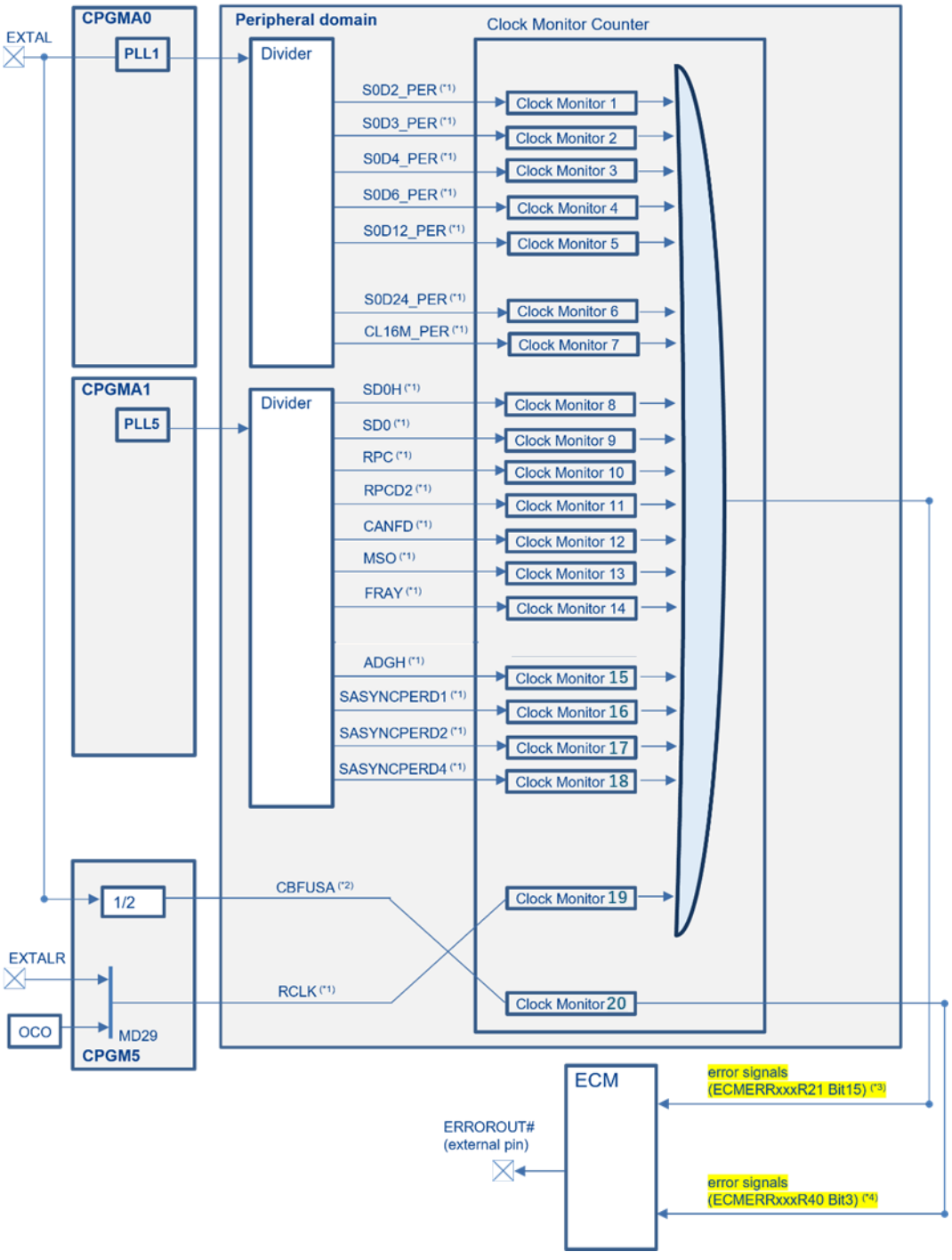



**Figure 8.5.2.6 Block diagram of Clock Monitor Counter**

Note *1 In case that CBFUSAφ is monitored, the sampling clock is RCLK. CBFUSAφ = EXTAL/2.
In addition, ADGHφ monitor is not mandatory because the clock is not used by any modules.
Note *2 In case of monitoring clocks except CBFUSAφ, the sampling clock is CBFUSAφ.
Note *3 Even if one of all clocks except CBFUSA has incorrect frequency, the error signal is reported to ECM.
Note *4 When CBFUSAφ is incorrect frequency, the error signal is reported to ECM via different notification path from any other clocks.

**(2) Register Configuration**

See Table 8.1.6.2.4.

**(3) Register Description**

XXX = S0D2PER, S0D3PER, S0D4PER, S0D6PER, S0D12PER, S0D24PER, CL16MPER, SD0H, SD0, RPC, RPCD2, CANFD, MSO, FRAY, ADGH, SASYNCPERD1, SASYNCPERD2, SASYNCPERD4, CBFUSAPER, RCLKPER.

**(a) XXX Clock Monitor Control / Status Register (XXXCKMCSR)**

See 8.2.39.

**(b) XXX Clock Monitor Error Clear Register (XXXCKMECR)**

See 8.2.40.

**(c) XXX Clock Monitor Limit Count H Register (XXXCKMLCH)**

See 8.2.41.

**(d) XXX Clock Monitor Limit Count L Register (XXXCKMLCL)**

See 8.2.42.

**(e) XXX Clock Monitor Count Register (XXXCKMCNT)**

See 8.2.43.

**(f) XXX Clock Monitor Count on Error Register (XXXCKMCNTE)**

See 8.2.44.

**(g) PER Clock Monitor Write Protect Register (PERCKMWPR)**

The specification is same as CPG Write Protect Register (CPGWPR) specification in Sec.8.2.1. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.5 (3).

**(h) PER Clock Monitor Write Protect Control Register (PERCKMWPCR)**

The specification is same as CPG Write Protect Control Register (CPGWPCR) specification in Sec.8.2.2. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.5 (3).

**(i) PER All Clock Monitor Status Register (PERACKMSR)**

See 8.2.52.

**(4) Operation**

- All registers in Clock Monitor except PERCKMWPR can be protected from unexpected write operation when PERCKMWPCR.WPE = 1. When PERCKMWPCR.WPE = 1, CPG registers cannot be written without using PERCKMWPR before the write access to the target register. In detail, see the Sec. 8.4.1. The write access usage is same as it.

- The other operation procedure is same as that of 8.5.2.1 (4).

- See Table 8.5.2.5.1 about N cycles information of the sampling clock.

**Table 8.5.2.5.1 N value for Monitor clock during calculating MONCNT value**

| Monitor clock | N value |
|---|---|
| S0D2_PER, S0D3_PER, S0D4_PER, S0D6_PER, SD0H, ADGH, SASYNCPERD1, SASYNCPERD2, CBFUSA | 4 |
| S0D12_PER, S0D24_PER, CL16M_PER, SD0, RPC, RPCD2, CANFD, MSO, FRAY, SASYNCPERD4 | 8 |
| RCLK | 16 |

### 8.5.2.6 Clock Monitor in HSC (High speed Communication) domain

#### (1) Overview

A Clock Monitor Counter exists in HSC (High speed Communication) domain as shown in Figure 8.5.2.7 to monitor whether the clock frequency is normal or not. One clock is monitored by one clock monitor counter inside Clock Monitor.

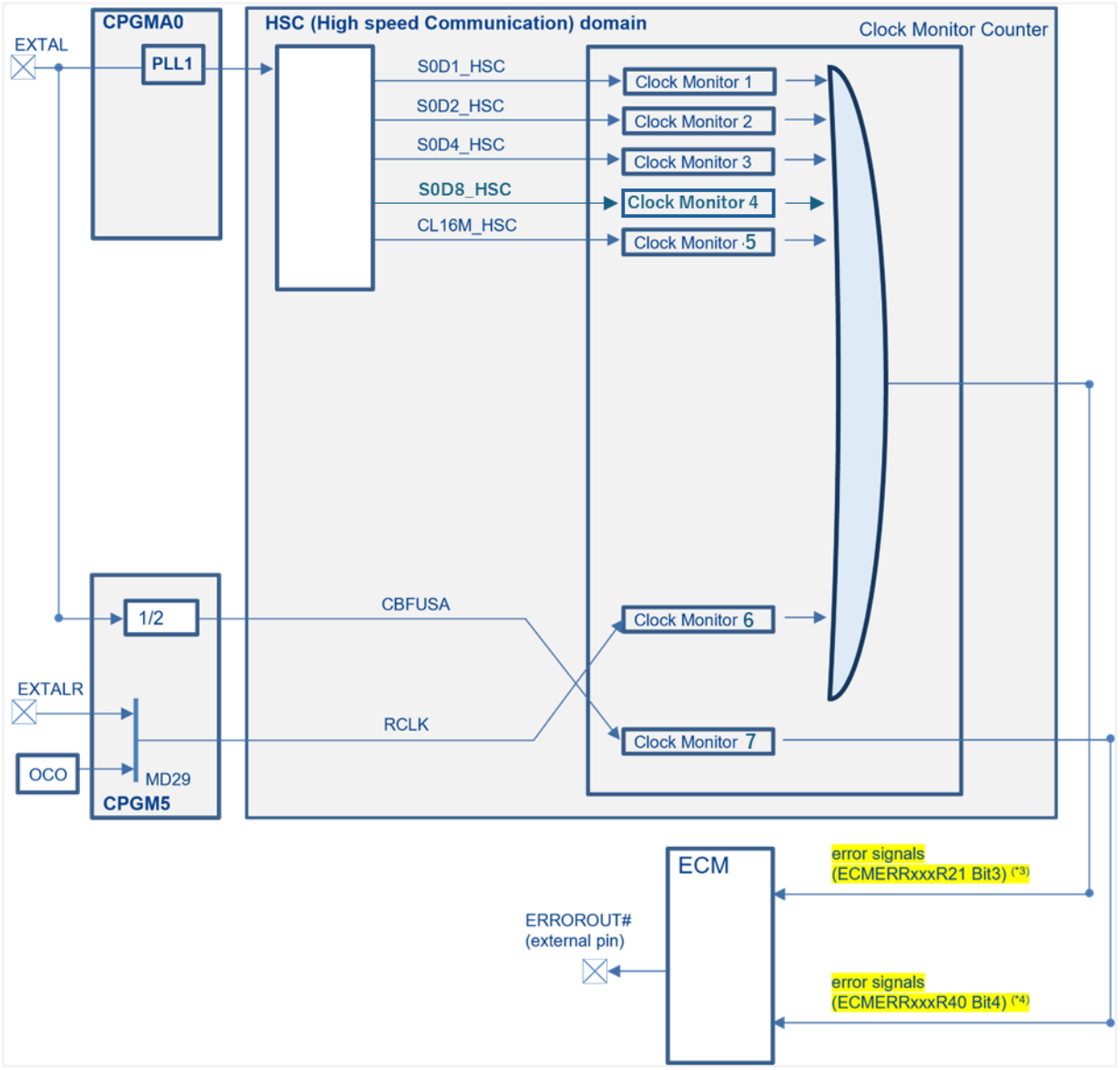



**Figure 8.5.2.7 Block diagram of Clock Monitor Counter**

Note *1 In case that CBFUSAφ is monitored, the sampling clock is RCLK. CBFUSAφ = EXTAL/2.
Note *2 In case of monitoring clocks except CBFUSAφ, the sampling clock is CBFUSAφ.
Note *3 Even if one of all clocks except CBFUSA has incorrect frequency, the error signal is reported to ECM.
Note *4 When CBFUSAφ is incorrect frequency, the error signal is reported to ECM via different notification path from any other clocks.

**(2) Register Configuration**

See Table 8.1.6.2.5.

### (3) Register Description

XXX = S0D1HSC, S0D2HSC S0D4HSC, CL16MHSC, CBFUSAHSC, RCLKHSC.

#### (a) XXX Clock Monitor Control / Status Register (XXXCKMCSR)

See 8.2.39.

#### (b) XXX Clock Monitor Error Clear Register (XXXCKMECR)

See 8.2.40.

#### (c) XXX Clock Monitor Limit Count H Register (XXXCKMLCH)

See 8.2.41.

#### (d) XXX Clock Monitor Limit Count L Register (XXXCKMLCL)

See 8.2.42.

#### (e) XXX Clock Monitor Count Register (XXXCKMCNT)

See 8.2.43.

#### (f) XXX Clock Monitor Count on Error Register (XXXCKMCNTE)

See 8.2.44.

#### (g) HSC Clock Monitor Write Protect Register (HSCCKMWPR)

The specification is same as CPG Write Protect Register (CPGWPR) specification in Sec.8.2.1. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.6 (3).

#### (h) HSC Clock Monitor Write Protect Control Register (HSCCKMWPCR)

The specification is same as CPG Write Protect Control Register (CPGWPCR) specification in Sec.8.2.2. The register is effective for (a), (b), (c), (d), (e), (f) registers in 8.5.2.6 (3).

#### (i) HSC All Clock Monitor Status Register (HSCACKMSR)

See 8.2.53.

**(4) Operation**

- All registers in Clock Monitor except HSCCKMWPR can be protected from unexpected write operation when HSCCKMWPCR.WPE = 1. When HSCCKMWPCR.WPE = 1, CPG registers cannot be written without using HSCCKMWPR before the write access to the target register. In detail, see the Sec. 8.4.1. The write access usage is same as it.

- The other operation procedure is same as that of 8.5.2.1 (4).

- See Table 8.5.2.6.1 about N cycles information of the sampling clock.

**Table 8.5.2.6.1 N value for Monitor clock during calculating MONCNT value**

| Monitor clock | N value |
|---|---|
| S0D1_HSC, S0D2_HSC, S0D4_HSC, S0D8_HSC, CBFUSA | 4 |
| CL16M_HSC | 8 |
| RCLK | 16 |

### 8.5.3 Safety Access Protection

#### (1) Overview

CPG provides a safety access protection mechanism to guarantee “Freedom From Interference (FFI)” when accessing the registers belonging to the CPG module. Several masters (e.g. Cortex-R52, Cortex-A76) take control of different portions and modules of the chip. The access protection mechanism shall enable sharing and splitting the functionality among cores.

The access protection mechanism for the control registers belonging to CPG module shall be aligned with the Region ID concept. In detail, see Sec.8.4.2. CPG has critical system control functions such as the clock enable, the clock distribution, the module reset. Therefore, the strong access restrictions are necessary for CPG.

#### (2) Register Description

As shown in Sec. 8.4.2 (2), the register corresponds with D0WACRAn, D1WACRAn, D2WACRAn, D3WACRAn (n = 0 to 63) in Sec. 8.2.35, 8.2.36, 8.2.37, 8.2.38.

#### (3) Operation

See Sec. 8.4.2 (1) and 8.4.2 (2).

# 9. Module Standby, Software Reset

## 9.1 Overview

The CPG functions related to module control are explained in this section. Under software control, the CPG is capable of turning the supply of clock signals to individual modules on or off and of resetting individual modules.

### 9.1.1 Features

**(1) Module Standby**

Clock supply to specified modules is stopped by setting the module stop control register bits. Two registers for one module, and supply of the clock signal to a module is stopped when all these registers bits are set to 'stop'. The register specification is described in 9.2.3, 9.2.4.

- Module Stop Control Register (MSTPCRn (n = 0 to 29))
- Module Stop Status Register (MSTPSRn (n = 0 to 29))

**(2) Software Reset**

Initialize the specified module by setting the software reset register bit. The register specification is described in 9.2.1, 9.2.2.

- Software Reset Control Register (SRCRn (n = 0 to 29))
- Software Reset Clearing Register (SRSTCLRn (n = 0 to 29))

**(3) Reset Status Monitor [Functional Safety]**

Reset status monitor is a safety mechanism that confirms if an intended reset is asserted and an unintended reset is asserted. The related register specification is described in Section 8.2.10, 8.2.11, 8.2.12, 8.2.13.

- Functional Safety Reset Check Register A (FSRCHKRAn (n = 0 to 29))
- Functional Safety Reset Check Register B (FSRCHKRBn (n = 0 to 29))
- Functional Safety Reset Check Set Register (FSRCHKSETRn (n = 0 to 29))
- Functional Safety Reset Check Clear Register (FSRCHKCLRRn (n = 0 to 29))

**(4) Force Reset**

Software Reset, which is shown in Section 9.1.1 (2), is asserted after the request/acknowledge handshake between CPG and the target modules is completed to guarantee the system operation in the chip. On the other hand, when writing 0 to the corresponding bit of RAHSRn (Request/Acknowledge Handshake Select Register), CPG is forced to assert the software reset to the target modules without completing the handshake. The related register specification is described in 9.2.5.

- Request/Acknowledge Handshake Select Register (RAHSRn (n = 0 to 29))

**(5) Bus Access Protection [Functional Safety]**

Register access protection is applied for registers of Module Standby, Software Reset, Forced Reset. The detail specification is described in 9.4.2. In addition, the function is a safety mechanism to guarantee "Freedom From Interference (FFI)". In detail, see 9.5.3. The register specification is described in 9.2.6.

- Domain x Write Access Control Register of MSTPCRn (DxWACR_MSTPCRn (x = 0 to 3))
- Domain x Write Access Control Register of SRCRn (DxWACR_SRCRn (x = 0 to 3))
- Domain x Write Access Control Register of SRSTCLRn (DxWACR_SRSTCLRn (x = 0 to 3))
- Domain x Write Access Control Register of RAHSRn (DxWACR_RAHSRn (x = 0 to 3))

### 9.1.2 Block Diagram

A block diagram of CPG is shown in Figure 9.1.1.

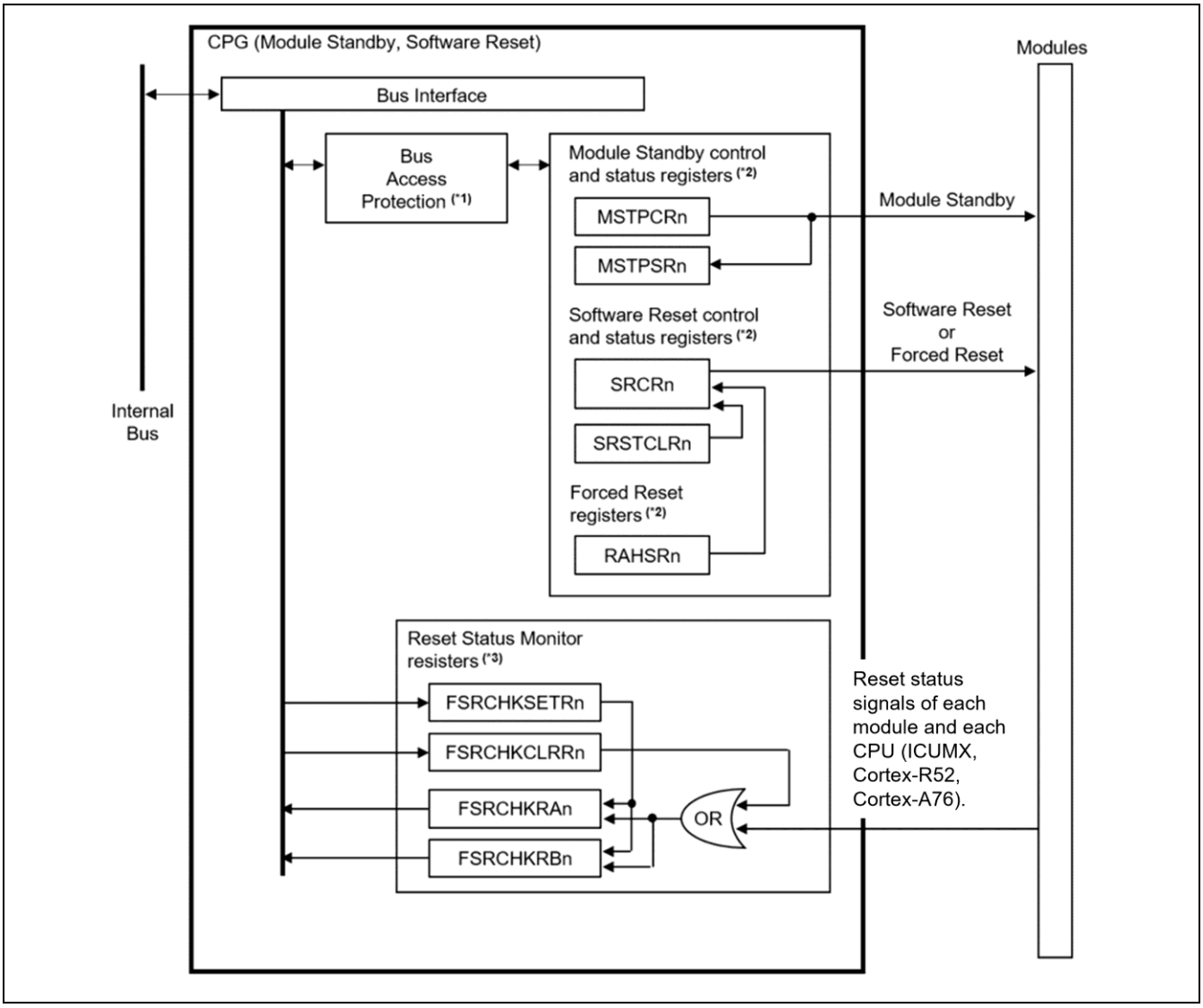



**Figure 9.1.1 Block Diagram of Module Standby, Software Reset**

Note *1: See 9.4.2, 9.5.3 about the detailed description.
Note *2: See 9.2.1, 9.2.2, 9.2.3, 9.2.4, 9.2.5 about the register specification.
Note *3: See 8.2.10, 8.2.11, 8.2.12, 8.2.13 about the register specification.

### 9.1.3 External Pins

There is no input/output pin related with this function.

### 9.1.4 Register Configuration

- The base-address of Module Standby, Software Reset is H'E615 0000
- CPG has Bus Access Protection registers (DxWACRA) as shown in Section 9.1.1 (5). See Section 9.4.2 in detail.

**Table 9.1.4.1 Register Configuration**

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| SRCR0 | R/W | H'E615 2C00 | H'0000 0000 | 32 | DxWACR_SRCR0 |
| SRCR1 | R/W | H'E615 2C04 | H'0000 0000 | 32 | DxWACR_SRCR1 |
| SRCR2 | R/W | H'E615 2C08 | H'0000 0000 | 32 | DxWACR_SRCR2 |
| SRCR3 | R/W | H'E615 2C0C | H'0000 0000 | 32 | DxWACR_SRCR3 |
| SRCR4 | R/W | H'E615 2C10 | H'0000 0000 | 32 | DxWACR_SRCR4 |
| SRCR5 | R/W | H'E615 2C14 | H'8000 0000 | 32 | DxWACR_SRCR5 |
| SRCR6 | R/W | H'E615 2C18 | H'0100 0000 | 32 | DxWACR_SRCR6 |
| SRCR7 | R/W | H'E615 2C1C | H'0000 0000 | 32 | DxWACR_SRCR7 |
| SRCR8 | R/W | H'E615 2C20 | H'0000 0000 | 32 | DxWACR_SRCR8 |
| SRCR9 | R/W | H'E615 2C24 | H'0000 0000 | 32 | DxWACR_SRCR9 |
| SRCR10 | R/W | H'E615 2C28 | H'0000 0000 | 32 | DxWACR_SRCR10 |
| SRCR11 | R/W | H'E615 2C2C | H'0008 0000 | 32 | DxWACR_SRCR11 |
| SRCR12 | R/W | H'E615 2C30 | H'0000 0000 | 32 | DxWACR_SRCR12 |
| SRCR13 | R/W | H'E615 2C34 | H'0000 0000 | 32 | DxWACR_SRCR13 |
| SRCR14 | R/W | H'E615 2C38 | H'0000 0000 | 32 | DxWACR_SRCR14 |
| SRCR15 | R/W | H'E615 2C3C | H'0000 0000 | 32 | DxWACR_SRCR15 |
| SRCR16 | R/W | H'E615 2C40 | H'0000 0000 | 32 | DxWACR_SRCR16 |
| SRCR17 | R/W | H'E615 2C44 | H'0000 0000 | 32 | DxWACR_SRCR17 |
| SRCR18 | R/W | H'E615 2C48 | H'0000 0000 | 32 | DxWACR_SRCR18 |
| SRCR19 | R/W | H'E615 2C4C | H'0000 0000 | 32 | DxWACR_SRCR19 |
| SRCR20 | R/W | H'E615 2C50 | H'0000 0000 | 32 | DxWACR_SRCR20 |
| SRCR21 | R/W | H'E615 2C54 | H'0000 0000 | 32 | DxWACR_SRCR21 |
| SRCR22 | R/W | H'E615 2C58 | H'0000 0000 | 32 | DxWACR_SRCR22 |
| SRCR23 | R/W | H'E615 2C5C | H'0000 0000 | 32 | DxWACR_SRCR23 |
| SRCR24 | R/W | H'E615 2C60 | H'0000 0000 | 32 | DxWACR_SRCR24 |
| SRCR25 | R/W | H'E615 2C64 | H'0000 0000 | 32 | DxWACR_SRCR25 |
| SRCR26 | R/W | H'E615 2C68 | H'0000 0000 | 32 | DxWACR_SRCR26 |
| SRCR27 | R/W | H'E615 2C6C | H'0000 0000 | 32 | DxWACR_SRCR27 |
| SRCR28 | R/W | H'E615 2C70 | H'CF60 0006 | 32 | DxWACR_SRCR28 |
| SRCR29 | R/W | H'E615 2C74 | H'0000 001E | 32 | DxWACR_SRCR29 |
| SRSTCLR0 | W | H'E615 2C80 | H'0000 0000 | 32 | DxWACR_SRSTCLR0 |
| SRSTCLR1 | W | H'E615 2C84 | H'0000 0000 | 32 | DxWACR_SRSTCLR1 |
| SRSTCLR2 | W | H'E615 2C88 | H'0000 0000 | 32 | DxWACR_SRSTCLR2 |
| SRSTCLR3 | W | H'E615 2C8C | H'0000 0000 | 32 | DxWACR_SRSTCLR3 |
| SRSTCLR4 | W | H'E615 2C90 | H'0000 0000 | 32 | DxWACR_SRSTCLR4 |
| SRSTCLR5 | W | H'E615 2C94 | H'0000 0000 | 32 | DxWACR_SRSTCLR5 |
| SRSTCLR6 | W | H'E615 2C98 | H'0000 0000 | 32 | DxWACR_SRSTCLR6 |
| SRSTCLR7 | W | H'E615 2C9C | H'0000 0000 | 32 | DxWACR_SRSTCLR7 |
| SRSTCLR8 | W | H'E615 2CA0 | H'0000 0000 | 32 | DxWACR_SRSTCLR8 |
| SRSTCLR9 | W | H'E615 2CA4 | H'0000 0000 | 32 | DxWACR_SRSTCLR9 |
| SRSTCLR10 | W | H'E615 2CA8 | H'0000 0000 | 32 | DxWACR_SRSTCLR10 |
| SRSTCLR11 | W | H'E615 2CAC | H'0000 0000 | 32 | DxWACR_SRSTCLR11 |
| SRSTCLR12 | W | H'E615 2CB0 | H'0000 0000 | 32 | DxWACR_SRSTCLR12 |
| SRSTCLR13 | W | H'E615 2CB4 | H'0000 0000 | 32 | DxWACR_SRSTCLR13 |
| SRSTCLR14 | W | H'E615 2CB8 | H'0000 0000 | 32 | DxWACR_SRSTCLR14 |
| SRSTCLR15 | W | H'E615 2CBC | H'0000 0000 | 32 | DxWACR_SRSTCLR15 |
| SRSTCLR16 | W | H'E615 2CC0 | H'0000 0000 | 32 | DxWACR_SRSTCLR16 |
| SRSTCLR17 | W | H'E615 2CC4 | H'0000 0000 | 32 | DxWACR_SRSTCLR17 |
| SRSTCLR18 | W | H'E615 2CC8 | H'0000 0000 | 32 | DxWACR_SRSTCLR18 |
| SRSTCLR19 | W | H'E615 2CCC | H'0000 0000 | 32 | DxWACR_SRSTCLR19 |
| SRSTCLR20 | W | H'E615 2CD0 | H'0000 0000 | 32 | DxWACR_SRSTCLR20 |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| SRSTCLR21 | W | H'E615 2CD4 | H'0000 0000 | 32 | DxWACR_SRSTCLR21 |
| SRSTCLR22 | W | H'E615 2CD8 | H'0000 0000 | 32 | DxWACR_SRSTCLR22 |
| SRSTCLR23 | W | H'E615 2CDC | H'0000 0000 | 32 | DxWACR_SRSTCLR23 |
| SRSTCLR24 | W | H'E615 2CE0 | H'0000 0000 | 32 | DxWACR_SRSTCLR24 |
| SRSTCLR25 | W | H'E615 2CE4 | H'0000 0000 | 32 | DxWACR_SRSTCLR25 |
| SRSTCLR26 | W | H'E615 2CE8 | H'0000 0000 | 32 | DxWACR_SRSTCLR26 |
| SRSTCLR27 | W | H'E615 2CEC | H'0000 0000 | 32 | DxWACR_SRSTCLR27 |
| SRSTCLR28 | W | H'E615 2CF0 | H'0000 0000 | 32 | DxWACR_SRSTCLR28 |
| SRSTCLR29 | W | H'E615 2CF4 | H'0000 0000 | 32 | DxWACR_SRSTCLR29 |
| MSTPCR0 | R/W | H'E615 2D00 | H'0003 7AA0 | 32 | DxWACR_MSTPCR0 |
| MSTPCR1 | R/W | H'E615 2D04 | H'00C0 0000 | 32 | DxWACR_MSTPCR1 |
| MSTPCR2 | R/W | H'E615 2D08 | H'0000 3800 | 32 | DxWACR_MSTPCR2 |
| MSTPCR3 | R/W | H'E615 2D0C | H'9000 0000 | 32 | DxWACR_MSTPCR3 |
| MSTPCR4 | R/W | H'E615 2D10 | H'0001 C801 | 32 | DxWACR_MSTPCR4 |
| MSTPCR5 | R/W | H'E615 2D14 | H'EE03 C380 | 32 | DxWACR_MSTPCR5 |
| MSTPCR6 | R/W | H'E615 2D18 | H'1301 3001 | 32 | DxWACR_MSTPCR6 |
| MSTPCR7 | R/W | H'E615 2D1C | H'FFF7 E040 | 32 | DxWACR_MSTPCR7 |
| MSTPCR8 | R/W | H'E615 2D20 | H'C000 3FFF | 32 | DxWACR_MSTPCR8 |
| MSTPCR9 | R/W | H'E615 2D24 | H'0000 3C78 | 32 | DxWACR_MSTPCR9 |
| MSTPCR10 | R/W | H'E615 2D28 | H'3000 0000 | 32 | DxWACR_MSTPCR10 |
| MSTPCR11 | R/W | H'E615 2D2C | H'0000 0003 | 32 | DxWACR_MSTPCR11 |
| MSTPCR12 | R/W | H'E615 2D30 | H'9E80 0000 | 32 | DxWACR_MSTPCR12 |
| MSTPCR13 | R/W | H'E615 2D34 | H'0000 0037 | 32 | DxWACR_MSTPCR13 |
| MSTPCR14 | R/W | H'E615 2D38 | H'0000 0000 | 32 | DxWACR_MSTPCR14 |
| MSTPCR15 | R/W | H'E615 2D3C | H'0000 0000 | 32 | DxWACR_MSTPCR15 |
| MSTPCR16 | R/W | H'E615 2D40 | H'0000 0000 | 32 | DxWACR_MSTPCR16 |
| MSTPCR17 | R/W | H'E615 2D44 | H'0000 0000 | 32 | DxWACR_MSTPCR17 |
| MSTPCR18 | R/W | H'E615 2D48 | H'0000 0000 | 32 | DxWACR_MSTPCR18 |
| MSTPCR19 | R/W | H'E615 2D4C | H'0000 0000 | 32 | DxWACR_MSTPCR19 |
| MSTPCR20 | R/W | H'E615 2D50 | H'0000 0000 | 32 | DxWACR_MSTPCR20 |
| MSTPCR21 | R/W | H'E615 2D54 | H'0000 0000 | 32 | DxWACR_MSTPCR21 |
| MSTPCR22 | R/W | H'E615 2D58 | H'0000 0000 | 32 | DxWACR_MSTPCR22 |
| MSTPCR23 | R/W | H'E615 2D5C | H'0000 0000 | 32 | DxWACR_MSTPCR23 |
| MSTPCR24 | R/W | H'E615 2D60 | H'0000 0000 | 32 | DxWACR_MSTPCR24 |
| MSTPCR25 | R/W | H'E615 2D64 | H'0000 0000 | 32 | DxWACR_MSTPCR25 |
| MSTPCR26 | R/W | H'E615 2D68 | H'0000 0000 | 32 | DxWACR_MSTPCR26 |
| MSTPCR27 | R/W | H'E615 2D6C | H'0080 0000 | 32 | DxWACR_MSTPCR27 |
| MSTPCR28 | R/W | H'E615 2D70 | H'000A 0040 | 32 | DxWACR_MSTPCR28 |
| MSTPCR29 | R/W | H'E615 2D74 | H'0C00 0000 | 32 | DxWACR_MSTPCR29 |
| RAHSR0 | R/W | H'E615 2D80 | H'0FC3 FAA1 | 32 | DxWACR_RAHSR0 |
| RAHSR1 | R/W | H'E615 2D84 | H'F71D 2C38 | 32 | DxWACR_RAHSR1 |
| RAHSR2 | R/W | H'E615 2D88 | H'8A1A 38AF | 32 | DxWACR_RAHSR2 |
| RAHSR3 | R/W | H'E615 2D8C | H'6482 1012 | 32 | DxWACR_RAHSR3 |
| RAHSR4 | R/W | H'E615 2D90 | H'F001 B060 | 32 | DxWACR_RAHSR4 |
| RAHSR5 | R/W | H'E615 2D94 | H'EE00 23FB | 32 | DxWACR_RAHSR5 |
| RAHSR6 | R/W | H'E615 2D98 | H'E301 06FB | 32 | DxWACR_RAHSR6 |
| RAHSR7 | R/W | H'E615 2D9C | H'DFF0 0FC3 | 32 | DxWACR_RAHSR7 |
| RAHSR8 | R/W | H'E615 2DA0 | H'0000 3FFF | 32 | DxWACR_RAHSR8 |
| RAHSR9 | R/W | H'E615 2DA4 | H'0000 0178 | 32 | DxWACR_RAHSR9 |
| RAHSR10 | R/W | H'E615 2DA8 | H'000C 0000 | 32 | DxWACR_RAHSR10 |
| RAHSR11 | R/W | H'E615 2DAC | H'0010 1F03 | 32 | DxWACR_RAHSR11 |
| RAHSR12 | R/W | H'E615 2DB0 | H'0100 0000 | 32 | DxWACR_RAHSR12 |
| RAHSR13 | R/W | H'E615 2DB4 | H'0000 0000 | 32 | DxWACR_RAHSR13 |
| RAHSR14 | R/W | H'E615 2DB8 | H'0000 0000 | 32 | DxWACR_RAHSR14 |
| RAHSR15 | R/W | H'E615 2DBC | H'FF50 018C | 32 | DxWACR_RAHSR15 |
| RAHSR16 | R/W | H'E615 2DC0 | H'0000 0000 | 32 | DxWACR_RAHSR16 |
| RAHSR17 | R/W | H'E615 2DC4 | H'0000 0000 | 32 | DxWACR_RAHSR17 |
| RAHSR18 | R/W | H'E615 2DC8 | H'0000 0000 | 32 | DxWACR_RAHSR18 |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| RAHSR19 | R/W | H'E615 2DCC | H'0000 0000 | 32 | DxWACR_RAHSR19 |
| RAHSR20 | R/W | H'E615 2DD0 | H'0000 0000 | 32 | DxWACR_RAHSR20 |
| RAHSR21 | R/W | H'E615 2DD4 | H'0000 0000 | 32 | DxWACR_RAHSR21 |
| RAHSR22 | R/W | H'E615 2DD8 | H'0000 0000 | 32 | DxWACR_RAHSR22 |
| RAHSR23 | R/W | H'E615 2DDC | H'0000 0000 | 32 | DxWACR_RAHSR23 |
| RAHSR24 | R/W | H'E615 2DE0 | H'0000 0000 | 32 | DxWACR_RAHSR24 |
| RAHSR25 | R/W | H'E615 2DE4 | H'0000 0000 | 32 | DxWACR_RAHSR25 |
| RAHSR26 | R/W | H'E615 2DE8 | H'0000 0000 | 32 | DxWACR_RAHSR26 |
| RAHSR27 | R/W | H'E615 2DEC | H'0080 33C0 | 32 | DxWACR_RAHSR27 |
| RAHSR28 | R/W | H'E615 2DF0 | H'4020 00FF | 32 | DxWACR_RAHSR28 |
| RAHSR29 | R/W | H'E615 2DF4 | H'0001 0080 | 32 | DxWACR_RAHSR29 |
| MSTPSR0 | R | H'E615 2E00 | H'0003 7AA0 | 32 | (*1) |
| MSTPSR1 | R | H'E615 2E04 | H'00C0 0000 | 32 | (*1) |
| MSTPSR2 | R | H'E615 2E08 | H'0000 3800 | 32 | (*1) |
| MSTPSR3 | R | H'E615 2E0C | H'9000 0000 | 32 | (*1) |
| MSTPSR4 | R | H'E615 2E10 | H'0001 C801 | 32 | (*1) |
| MSTPSR5 | R | H'E615 2E14 | H'6E03 C380 | 32 | (*1) |
| MSTPSR6 | R | H'E615 2E18 | H'1301 3001 | 32 | (*1) |
| MSTPSR7 | R | H'E615 2E1C | H'FFF7 E040 | 32 | (*1) |
| MSTPSR8 | R | H'E615 2E20 | H'C000 3FFF | 32 | (*1) |
| MSTPSR9 | R | H'E615 2E24 | H'0000 3C78 | 32 | (*1) |
| MSTPSR10 | R | H'E615 2E28 | H'3000 0000 | 32 | (*1) |
| MSTPSR11 | R | H'E615 2E2C | H'0000 0003 | 32 | (*1) |
| MSTPSR12 | R | H'E615 2E30 | H'9E80 0000 | 32 | (*1) |
| MSTPSR13 | R | H'E615 2E34 | H'0000 0037 | 32 | (*1) |
| MSTPSR14 | R | H'E615 2E38 | H'0000 0000 | 32 | (*1) |
| MSTPSR15 | R | H'E615 2E3C | H'0000 0000 | 32 | (*1) |
| MSTPSR16 | R | H'E615 2E40 | H'0000 0000 | 32 | (*1) |
| MSTPSR17 | R | H'E615 2E44 | H'0000 0000 | 32 | (*1) |
| MSTPSR18 | R | H'E615 2E48 | H'0000 0000 | 32 | (*1) |
| MSTPSR19 | R | H'E615 2E4C | H'0000 0000 | 32 | (*1) |
| MSTPSR20 | R | H'E615 2E50 | H'0000 0000 | 32 | (*1) |
| MSTPSR21 | R | H'E615 2E54 | H'0000 0000 | 32 | (*1) |
| MSTPSR22 | R | H'E615 2E58 | H'0000 0000 | 32 | (*1) |
| MSTPSR23 | R | H'E615 2E5C | H'0000 0000 | 32 | (*1) |
| MSTPSR24 | R | H'E615 2E60 | H'0000 0000 | 32 | (*1) |
| MSTPSR25 | R | H'E615 2E64 | H'0000 0000 | 32 | (*1) |
| MSTPSR26 | R | H'E615 2E68 | H'0000 0000 | 32 | (*1) |
| MSTPSR27 | R | H'E615 2E6C | H'0080 0000 | 32 | (*1) |
| MSTPSR28 | R | H'E615 2E70 | H'000A 0040 | 32 | (*1) |
| MSTPSR29 | R | H'E615 2E74 | H'0C00 0000 | 32 | (*1) |
| D0WACR_MSTPCR0 | R/W | H'E615 3000 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR1 | R/W | H'E615 3004 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR2 | R/W | H'E615 3008 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR3 | R/W | H'E615 300C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR4 | R/W | H'E615 3010 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR5 | R/W | H'E615 3014 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR6 | R/W | H'E615 3018 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR7 | R/W | H'E615 301C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR8 | R/W | H'E615 3020 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR9 | R/W | H'E615 3024 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR10 | R/W | H'E615 3028 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR11 | R/W | H'E615 302C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR12 | R/W | H'E615 3030 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR13 | R/W | H'E615 3034 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR14 | R/W | H'E615 3038 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR15 | R/W | H'E615 303C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR16 | R/W | H'E615 3040 | H'FFFF FFFF | 32 | (*2) |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| D0WACR_MSTPCR17 | R/W | H'E615 3044 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR18 | R/W | H'E615 3048 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR19 | R/W | H'E615 304C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR20 | R/W | H'E615 3050 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR21 | R/W | H'E615 3054 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR22 | R/W | H'E615 3058 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR23 | R/W | H'E615 305C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR24 | R/W | H'E615 3060 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR25 | R/W | H'E615 3064 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR26 | R/W | H'E615 3068 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR27 | R/W | H'E615 306C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR28 | R/W | H'E615 3070 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_MSTPCR29 | R/W | H'E615 3074 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR0 | R/W | H'E615 3080 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR1 | R/W | H'E615 3084 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR2 | R/W | H'E615 3088 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR3 | R/W | H'E615 308C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR4 | R/W | H'E615 3090 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR5 | R/W | H'E615 3094 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR6 | R/W | H'E615 3098 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR7 | R/W | H'E615 309C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR8 | R/W | H'E615 30A0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR9 | R/W | H'E615 30A4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR10 | R/W | H'E615 30A8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR11 | R/W | H'E615 30AC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR12 | R/W | H'E615 30B0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR13 | R/W | H'E615 30B4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR14 | R/W | H'E615 30B8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR15 | R/W | H'E615 30BC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR16 | R/W | H'E615 30C0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR17 | R/W | H'E615 30C4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR18 | R/W | H'E615 30C8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR19 | R/W | H'E615 30CC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR20 | R/W | H'E615 30D0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR21 | R/W | H'E615 30D4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR22 | R/W | H'E615 30D8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR23 | R/W | H'E615 30DC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR24 | R/W | H'E615 30E0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR25 | R/W | H'E615 30E4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR26 | R/W | H'E615 30E8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR27 | R/W | H'E615 30EC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR28 | R/W | H'E615 30F0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_RAHSR29 | R/W | H'E615 30F4 | H'FFFF FFFF | 32 | (*2) |
| D1WACR_MSTPCR0 | R/W | H'E615 3100 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR1 | R/W | H'E615 3104 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR2 | R/W | H'E615 3108 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR3 | R/W | H'E615 310C | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR4 | R/W | H'E615 3110 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR5 | R/W | H'E615 3114 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR6 | R/W | H'E615 3118 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR7 | R/W | H'E615 311C | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR8 | R/W | H'E615 3120 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR9 | R/W | H'E615 3124 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR10 | R/W | H'E615 3128 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR11 | R/W | H'E615 312C | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR12 | R/W | H'E615 3130 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR13 | R/W | H'E615 3134 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR14 | R/W | H'E615 3138 | H'0000 0000 | 32 | (*2) |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| D1WACR_MSTPCR15 | R/W | H'E615 313C | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR16 | R/W | H'E615 3140 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR17 | R/W | H'E615 3144 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR18 | R/W | H'E615 3148 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR19 | R/W | H'E615 314C | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR20 | R/W | H'E615 3150 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR21 | R/W | H'E615 3154 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR22 | R/W | H'E615 3158 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR23 | R/W | H'E615 315C | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR24 | R/W | H'E615 3160 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR25 | R/W | H'E615 3164 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR26 | R/W | H'E615 3168 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR27 | R/W | H'E615 316C | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR28 | R/W | H'E615 3170 | H'0000 0000 | 32 | (*2) |
| D1WACR_MSTPCR29 | R/W | H'E615 3174 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR0 | R/W | H'E615 3180 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR1 | R/W | H'E615 3184 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR2 | R/W | H'E615 3188 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR3 | R/W | H'E615 318C | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR4 | R/W | H'E615 3190 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR5 | R/W | H'E615 3194 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR6 | R/W | H'E615 3198 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR7 | R/W | H'E615 319C | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR8 | R/W | H'E615 31A0 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR9 | R/W | H'E615 31A4 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR10 | R/W | H'E615 31A8 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR11 | R/W | H'E615 31AC | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR12 | R/W | H'E615 31B0 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR13 | R/W | H'E615 31B4 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR14 | R/W | H'E615 31B8 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR15 | R/W | H'E615 31BC | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR16 | R/W | H'E615 31C0 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR17 | R/W | H'E615 31C4 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR18 | R/W | H'E615 31C8 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR19 | R/W | H'E615 31CC | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR20 | R/W | H'E615 31D0 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR21 | R/W | H'E615 31D4 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR22 | R/W | H'E615 31D8 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR23 | R/W | H'E615 31DC | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR24 | R/W | H'E615 31E0 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR25 | R/W | H'E615 31E4 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR26 | R/W | H'E615 31E8 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR27 | R/W | H'E615 31EC | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR28 | R/W | H'E615 31F0 | H'0000 0000 | 32 | (*2) |
| D1WACR_RAHSR29 | R/W | H'E615 31F4 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR0 | R/W | H'E615 3200 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR1 | R/W | H'E615 3204 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR2 | R/W | H'E615 3208 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR3 | R/W | H'E615 320C | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR4 | R/W | H'E615 3210 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR5 | R/W | H'E615 3214 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR6 | R/W | H'E615 3218 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR7 | R/W | H'E615 321C | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR8 | R/W | H'E615 3220 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR9 | R/W | H'E615 3224 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR10 | R/W | H'E615 3228 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR11 | R/W | H'E615 322C | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR12 | R/W | H'E615 3230 | H'0000 0000 | 32 | (*2) |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| D2WACR_MSTPCR13 | R/W | H'E615 3234 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR14 | R/W | H'E615 3238 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR15 | R/W | H'E615 323C | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR16 | R/W | H'E615 3240 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR17 | R/W | H'E615 3244 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR18 | R/W | H'E615 3248 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR19 | R/W | H'E615 324C | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR20 | R/W | H'E615 3250 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR21 | R/W | H'E615 3254 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR22 | R/W | H'E615 3258 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR23 | R/W | H'E615 325C | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR24 | R/W | H'E615 3260 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR25 | R/W | H'E615 3264 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR26 | R/W | H'E615 3268 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR27 | R/W | H'E615 326C | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR28 | R/W | H'E615 3270 | H'0000 0000 | 32 | (*2) |
| D2WACR_MSTPCR29 | R/W | H'E615 3274 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR0 | R/W | H'E615 3280 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR1 | R/W | H'E615 3284 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR2 | R/W | H'E615 3288 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR3 | R/W | H'E615 328C | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR4 | R/W | H'E615 3290 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR5 | R/W | H'E615 3294 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR6 | R/W | H'E615 3298 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR7 | R/W | H'E615 329C | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR8 | R/W | H'E615 32A0 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR9 | R/W | H'E615 32A4 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR10 | R/W | H'E615 32A8 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR11 | R/W | H'E615 32AC | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR12 | R/W | H'E615 32B0 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR13 | R/W | H'E615 32B4 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR14 | R/W | H'E615 32B8 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR15 | R/W | H'E615 32BC | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR16 | R/W | H'E615 32C0 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR17 | R/W | H'E615 32C4 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR18 | R/W | H'E615 32C8 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR19 | R/W | H'E615 32CC | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR20 | R/W | H'E615 32D0 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR21 | R/W | H'E615 32D4 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR22 | R/W | H'E615 32D8 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR23 | R/W | H'E615 32DC | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR24 | R/W | H'E615 32E0 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR25 | R/W | H'E615 32E4 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR26 | R/W | H'E615 32E8 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR27 | R/W | H'E615 32EC | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR28 | R/W | H'E615 32F0 | H'0000 0000 | 32 | (*2) |
| D2WACR_RAHSR29 | R/W | H'E615 32F4 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR0 | R/W | H'E615 3300 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR1 | R/W | H'E615 3304 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR2 | R/W | H'E615 3308 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR3 | R/W | H'E615 330C | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR4 | R/W | H'E615 3310 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR5 | R/W | H'E615 3314 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR6 | R/W | H'E615 3318 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR7 | R/W | H'E615 331C | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR8 | R/W | H'E615 3320 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR9 | R/W | H'E615 3324 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR10 | R/W | H'E615 3328 | H'0000 0000 | 32 | (*2) |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| D3WACR_MSTPCR11 | R/W | H'E615 332C | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR12 | R/W | H'E615 3330 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR13 | R/W | H'E615 3334 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR14 | R/W | H'E615 3338 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR15 | R/W | H'E615 333C | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR16 | R/W | H'E615 3340 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR17 | R/W | H'E615 3344 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR18 | R/W | H'E615 3348 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR19 | R/W | H'E615 334C | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR20 | R/W | H'E615 3350 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR21 | R/W | H'E615 3354 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR22 | R/W | H'E615 3358 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR23 | R/W | H'E615 335C | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR24 | R/W | H'E615 3360 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR25 | R/W | H'E615 3364 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR26 | R/W | H'E615 3368 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR27 | R/W | H'E615 336C | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR28 | R/W | H'E615 3370 | H'0000 0000 | 32 | (*2) |
| D3WACR_MSTPCR29 | R/W | H'E615 3374 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR0 | R/W | H'E615 3380 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR1 | R/W | H'E615 3384 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR2 | R/W | H'E615 3388 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR3 | R/W | H'E615 338C | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR4 | R/W | H'E615 3390 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR5 | R/W | H'E615 3394 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR6 | R/W | H'E615 3398 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR7 | R/W | H'E615 339C | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR8 | R/W | H'E615 33A0 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR9 | R/W | H'E615 33A4 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR10 | R/W | H'E615 33A8 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR11 | R/W | H'E615 33AC | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR12 | R/W | H'E615 33B0 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR13 | R/W | H'E615 33B4 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR14 | R/W | H'E615 33B8 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR15 | R/W | H'E615 33BC | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR16 | R/W | H'E615 33C0 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR17 | R/W | H'E615 33C4 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR18 | R/W | H'E615 33C8 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR19 | R/W | H'E615 33CC | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR20 | R/W | H'E615 33D0 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR21 | R/W | H'E615 33D4 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR22 | R/W | H'E615 33D8 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR23 | R/W | H'E615 33DC | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR24 | R/W | H'E615 33E0 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR25 | R/W | H'E615 33E4 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR26 | R/W | H'E615 33E8 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR27 | R/W | H'E615 33EC | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR28 | R/W | H'E615 33F0 | H'0000 0000 | 32 | (*2) |
| D3WACR_RAHSR29 | R/W | H'E615 33F4 | H'0000 0000 | 32 | (*2) |
| D0WACR_SRCR0 | R/W | H'E615 3400 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR1 | R/W | H'E615 3404 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR2 | R/W | H'E615 3408 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR3 | R/W | H'E615 340C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR4 | R/W | H'E615 3410 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR5 | R/W | H'E615 3414 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR6 | R/W | H'E615 3418 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR7 | R/W | H'E615 341C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR8 | R/W | H'E615 3420 | H'FFFF FFFF | 32 | (*2) |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| D0WACR_SRCR9 | R/W | H'E615 3424 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR10 | R/W | H'E615 3428 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR11 | R/W | H'E615 342C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR12 | R/W | H'E615 3430 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR13 | R/W | H'E615 3434 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR14 | R/W | H'E615 3438 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR15 | R/W | H'E615 343C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR16 | R/W | H'E615 3440 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR17 | R/W | H'E615 3444 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR18 | R/W | H'E615 3448 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR19 | R/W | H'E615 344C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR20 | R/W | H'E615 3450 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR21 | R/W | H'E615 3454 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR22 | R/W | H'E615 3458 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR23 | R/W | H'E615 345C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR24 | R/W | H'E615 3460 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR25 | R/W | H'E615 3464 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR26 | R/W | H'E615 3468 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR27 | R/W | H'E615 346C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR28 | R/W | H'E615 3470 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRCR29 | R/W | H'E615 3474 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR0 | R/W | H'E615 3480 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR1 | R/W | H'E615 3484 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR2 | R/W | H'E615 3488 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR3 | R/W | H'E615 348C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR4 | R/W | H'E615 3490 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR5 | R/W | H'E615 3494 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR6 | R/W | H'E615 3498 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR7 | R/W | H'E615 349C | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR8 | R/W | H'E615 34A0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR9 | R/W | H'E615 34A4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR10 | R/W | H'E615 34A8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR11 | R/W | H'E615 34AC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR12 | R/W | H'E615 34B0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR13 | R/W | H'E615 34B4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR14 | R/W | H'E615 34B8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR15 | R/W | H'E615 34BC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR16 | R/W | H'E615 34C0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR17 | R/W | H'E615 34C4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR18 | R/W | H'E615 34C8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR19 | R/W | H'E615 34CC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR20 | R/W | H'E615 34D0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR21 | R/W | H'E615 34D4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR22 | R/W | H'E615 34D8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR23 | R/W | H'E615 34DC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR24 | R/W | H'E615 34E0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR25 | R/W | H'E615 34E4 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR26 | R/W | H'E615 34E8 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR27 | R/W | H'E615 34EC | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR28 | R/W | H'E615 34F0 | H'FFFF FFFF | 32 | (*2) |
| D0WACR_SRSTCLR29 | R/W | H'E615 34F4 | H'FFFF FFFF | 32 | (*2) |
| D1WACR_SRCR0 | R/W | H'E615 3500 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR1 | R/W | H'E615 3504 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR2 | R/W | H'E615 3508 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR3 | R/W | H'E615 350C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR4 | R/W | H'E615 3510 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR5 | R/W | H'E615 3514 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR6 | R/W | H'E615 3518 | H'0000 0000 | 32 | (*2) |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| D1WACR_SRCR7 | R/W | H'E615 351C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR8 | R/W | H'E615 3520 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR9 | R/W | H'E615 3524 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR10 | R/W | H'E615 3528 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR11 | R/W | H'E615 352C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR12 | R/W | H'E615 3530 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR13 | R/W | H'E615 3534 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR14 | R/W | H'E615 3538 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR15 | R/W | H'E615 353C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR16 | R/W | H'E615 3540 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR17 | R/W | H'E615 3544 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR18 | R/W | H'E615 3548 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR19 | R/W | H'E615 354C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR20 | R/W | H'E615 3550 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR21 | R/W | H'E615 3554 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR22 | R/W | H'E615 3558 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR23 | R/W | H'E615 355C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR24 | R/W | H'E615 3560 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR25 | R/W | H'E615 3564 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR26 | R/W | H'E615 3568 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR27 | R/W | H'E615 356C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR28 | R/W | H'E615 3570 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRCR29 | R/W | H'E615 3574 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR0 | R/W | H'E615 3580 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR1 | R/W | H'E615 3584 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR2 | R/W | H'E615 3588 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR3 | R/W | H'E615 358C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR4 | R/W | H'E615 3590 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR5 | R/W | H'E615 3594 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR6 | R/W | H'E615 3598 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR7 | R/W | H'E615 359C | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR8 | R/W | H'E615 35A0 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR9 | R/W | H'E615 35A4 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR10 | R/W | H'E615 35A8 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR11 | R/W | H'E615 35AC | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR12 | R/W | H'E615 35B0 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR13 | R/W | H'E615 35B4 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR14 | R/W | H'E615 35B8 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR15 | R/W | H'E615 35BC | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR16 | R/W | H'E615 35C0 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR17 | R/W | H'E615 35C4 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR18 | R/W | H'E615 35C8 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR19 | R/W | H'E615 35CC | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR20 | R/W | H'E615 35D0 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR21 | R/W | H'E615 35D4 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR22 | R/W | H'E615 35D8 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR23 | R/W | H'E615 35DC | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR24 | R/W | H'E615 35E0 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR25 | R/W | H'E615 35E4 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR26 | R/W | H'E615 35E8 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR27 | R/W | H'E615 35EC | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR28 | R/W | H'E615 35F0 | H'0000 0000 | 32 | (*2) |
| D1WACR_SRSTCLR29 | R/W | H'E615 35F4 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR0 | R/W | H'E615 3600 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR1 | R/W | H'E615 3604 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR2 | R/W | H'E615 3608 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR3 | R/W | H'E615 360C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR4 | R/W | H'E615 3610 | H'0000 0000 | 32 | (*2) |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| D2WACR_SRCR5 | R/W | H'E615 3614 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR6 | R/W | H'E615 3618 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR7 | R/W | H'E615 361C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR8 | R/W | H'E615 3620 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR9 | R/W | H'E615 3624 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR10 | R/W | H'E615 3628 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR11 | R/W | H'E615 362C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR12 | R/W | H'E615 3630 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR13 | R/W | H'E615 3634 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR14 | R/W | H'E615 3638 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR15 | R/W | H'E615 363C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR16 | R/W | H'E615 3640 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR17 | R/W | H'E615 3644 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR18 | R/W | H'E615 3648 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR19 | R/W | H'E615 364C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR20 | R/W | H'E615 3650 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR21 | R/W | H'E615 3654 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR22 | R/W | H'E615 3658 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR23 | R/W | H'E615 365C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR24 | R/W | H'E615 3660 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR25 | R/W | H'E615 3664 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR26 | R/W | H'E615 3668 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR27 | R/W | H'E615 366C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR28 | R/W | H'E615 3670 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRCR29 | R/W | H'E615 3674 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR0 | R/W | H'E615 3680 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR1 | R/W | H'E615 3684 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR2 | R/W | H'E615 3688 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR3 | R/W | H'E615 368C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR4 | R/W | H'E615 3690 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR5 | R/W | H'E615 3694 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR6 | R/W | H'E615 3698 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR7 | R/W | H'E615 369C | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR8 | R/W | H'E615 36A0 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR9 | R/W | H'E615 36A4 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR10 | R/W | H'E615 36A8 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR11 | R/W | H'E615 36AC | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR12 | R/W | H'E615 36B0 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR13 | R/W | H'E615 36B4 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR14 | R/W | H'E615 36B8 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR15 | R/W | H'E615 36BC | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR16 | R/W | H'E615 36C0 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR17 | R/W | H'E615 36C4 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR18 | R/W | H'E615 36C8 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR19 | R/W | H'E615 36CC | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR20 | R/W | H'E615 36D0 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR21 | R/W | H'E615 36D4 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR22 | R/W | H'E615 36D8 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR23 | R/W | H'E615 36DC | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR24 | R/W | H'E615 36E0 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR25 | R/W | H'E615 36E4 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR26 | R/W | H'E615 36E8 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR27 | R/W | H'E615 36EC | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR28 | R/W | H'E615 36F0 | H'0000 0000 | 32 | (*2) |
| D2WACR_SRSTCLR29 | R/W | H'E615 36F4 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR0 | R/W | H'E615 3700 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR1 | R/W | H'E615 3704 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR2 | R/W | H'E615 3708 | H'0000 0000 | 32 | (*2) |

| Register Name (Abbreviation) | R/W | Address | Initial value | Access Size | Corresponding DxWACRA (x = 0, 1, 2, 3) |
|---|---|---|---|---|---|
| D3WACR_SRCR3 | R/W | H'E615 370C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR4 | R/W | H'E615 3710 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR5 | R/W | H'E615 3714 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR6 | R/W | H'E615 3718 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR7 | R/W | H'E615 371C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR8 | R/W | H'E615 3720 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR9 | R/W | H'E615 3724 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR10 | R/W | H'E615 3728 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR11 | R/W | H'E615 372C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR12 | R/W | H'E615 3730 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR13 | R/W | H'E615 3734 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR14 | R/W | H'E615 3738 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR15 | R/W | H'E615 373C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR16 | R/W | H'E615 3740 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR17 | R/W | H'E615 3744 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR18 | R/W | H'E615 3748 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR19 | R/W | H'E615 374C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR20 | R/W | H'E615 3750 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR21 | R/W | H'E615 3754 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR22 | R/W | H'E615 3758 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR23 | R/W | H'E615 375C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR24 | R/W | H'E615 3760 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR25 | R/W | H'E615 3764 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR26 | R/W | H'E615 3768 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR27 | R/W | H'E615 376C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR28 | R/W | H'E615 3770 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRCR29 | R/W | H'E615 3774 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR0 | R/W | H'E615 3780 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR1 | R/W | H'E615 3784 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR2 | R/W | H'E615 3788 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR3 | R/W | H'E615 378C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR4 | R/W | H'E615 3790 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR5 | R/W | H'E615 3794 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR6 | R/W | H'E615 3798 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR7 | R/W | H'E615 379C | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR8 | R/W | H'E615 37A0 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR9 | R/W | H'E615 37A4 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR10 | R/W | H'E615 37A8 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR11 | R/W | H'E615 37AC | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR12 | R/W | H'E615 37B0 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR13 | R/W | H'E615 37B4 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR14 | R/W | H'E615 37B8 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR15 | R/W | H'E615 37BC | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR16 | R/W | H'E615 37C0 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR17 | R/W | H'E615 37C4 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR18 | R/W | H'E615 37C8 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR19 | R/W | H'E615 37CC | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR20 | R/W | H'E615 37D0 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR21 | R/W | H'E615 37D4 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR22 | R/W | H'E615 37D8 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR23 | R/W | H'E615 37DC | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR24 | R/W | H'E615 37E0 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR25 | R/W | H'E615 37E4 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR26 | R/W | H'E615 37E8 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR27 | R/W | H'E615 37EC | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR28 | R/W | H'E615 37F0 | H'0000 0000 | 32 | (*2) |
| D3WACR_SRSTCLR29 | R/W | H'E615 37F4 | H'0000 0000 | 32 | (*2) |

Note 1. There are no protect registers (DxWACRA (x = 0, 1, 2, 3)) for Read Only register.
Note 2 DxWACRA (x = 0, 1, 2, 3) registers can be accessed only by Domain 0.

## 9.2 Register Description

Legend for Register Description

| | |
|---|---|
| Initial value | : Register value after a reset |
| – | : Undefined value |
| R/W | : Readable/writable. The written value can be read. |
| R | : Read-only. |
| W | : Write-only. The read value is undefined. |

### 9.2.1 Software Reset Register (SRCRn (n = 0 to 29)

Each SRCRn is a 32-bit readable/writable register. Each bit is assigned to a module in the chip. Writing 1 to the bit resets the corresponding module. Writing 1 to a given bit of the software reset clearing register (SRSTCLRn (n = 0 to 29)) clears the corresponding bit of SRCRn and releases the reset. SRCRn and SRSTCLRn does not need read-modify-write sequence. Write 1 to only target module bit of SRCRn or SRSTCLRn. It takes maximum 1 cycle by RCLK to complete software reset after setting 1 to target bit of SRCRn. When release software reset, it is necessary to wait more than 1 cycle by RCLK before setting 1 to target bit of SRSTCLRn. If applying a software reset to a module connected to the AXI, wait the completion of any data transfer.

Register protection function is covered by a function shown in 9.1.1 (5).

#### 9.2.1.1 Software Reset Register 0 (SRCR0)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.2 Software Reset Register 1 (SRCR1)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.3 Software Reset Register 2 (SRCR2)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.4 Software Reset Register 3 (SRCR3)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.5 Software Reset Register 4 (SRCR4)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.6 Software Reset Register 5 (SRCR5)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.7 Software Reset Register 6 (SRCR6)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.8 Software Reset Register 7 (SRCR7)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.9 Software Reset Register 8 (SRCR8)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.10 Software Reset Register 9 (SRCR9)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.11 Software Reset Register 10 (SRCR10)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.12 Software Reset Register 11 (SRCR11)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.13 Software Reset Register 12 (SRCR12)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.14 Software Reset Register 13 (SRCR13)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.15 Software Reset Register 14 (SRCR14)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.16 Software Reset Register 15 (SRCR15)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.17 Software Reset Register 16 (SRCR16)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.18 Software Reset Register 17 (SRCR17)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.19 Software Reset Register 18 (SRCR18)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.20 Software Reset Register 19 (SRCR19)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.21 Software Reset Register 20 (SRCR20)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.22 Software Reset Register 21 (SRCR21)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.23 Software Reset Register 22 (SRCR22)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.24 Software Reset Register 23 (SRCR23)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.25 Software Reset Register 24 (SRCR24)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.26 Software Reset Register 25 (SRCR25)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.27 Software Reset Register 26 (SRCR26)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.28 Software Reset Register 27 (SRCR27)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.29 Software Reset Register 28 (SRCR28)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

#### 9.2.1.30 Software Reset Register 29 (SRCR29)

See the following excel file, in which the bit assignment and the bit description are summarized.

R-CarV4M_UM_009_SRCRn_FSRCHKRAn_FSRCHKRBn_r0p81.xlsx

### 9.2.2 Software Reset Clearing Register (SRSTCLRn (n = 0 to 29))

Each SRSTCLRn is a 32-bit writable register. Each bit is assigned to a module in this chip. Writing 1 to the bit clears the corresponding bit of SRCRn and releases the reset. Write 0 to the reserved bits.

Bit assignment of SRSTCLRn is same with one of SRCRn.

Register protection function is covered by a function shown in Section 9.1.1 (5).

### 9.2.3 Module Stop Control Register (MSTPCRn (n = 0 to 29))

MSTPCRn is 32-bit readable/writable registers which control supply of the clock signal to the modules assigned to the corresponding bits. Module stop signals can be controlled by setting the MSTPCRn (n = 0 to 29).

0: Enables supply of the clock signal to the corresponding module.
1: Stops supply of the clock signal to the corresponding module.

Register protection function is covered by a function shown in Section 9.1.1 (5).

### 9.2.3.1 Module Stop Control Register 0 (MSTPCR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | imppsc | impdmac0 | imp1 | imp0 | spmc | impcnn | — | — | — | — | — | isp0 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MSTP009 | — | MSTP007 | — | MSTP005 | — | — | — | — | MSTP000 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R | R/W | R | R/W | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 28 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 27 | imppsc | 0 | R/W | IMP PSC |
| 26 | impdmac0 | 0 | R/W | IMP DMAC0 |
| 25 | imp1 | 0 | R/W | IMP core1 |
| 24 | imp0 | 0 | R/W | IMP core0 |
| 23 | spmc | 0 | R/W | IMP CNN (Scratchpad memory (SPMC0, SPMC1, SPMC2)) |
| 22 | impcnn | 0 | R/W | IMP CNN |
| 21 to 17 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 16 | isp0 | 1 | R/W | ISP (Channel 1) |
| 15 to 10 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 9 | umfl0 | 1 | R/W | Vision IP (Optical Flow) |
| 8 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 7 | smps0 | 1 | R/W | Vision IP (SFM (Structure Flow Motion), Priority Sampling) |
| 6 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 5 | smpo0 | 1 | R/W | Vision IP (SFM (Structure Flow Motion), Post Processing) |
| 4 to 1 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 0 | rgx | 0 | R/W | 3D Graphics Engine |

### 9.2.3.2 Module Stop Control Register 1 (MSTPCR1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | avs | adg | — | spmi | — | impslv | — | impdta |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R | R/W | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | impd mac1 | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 24 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 23 | avs | 1 | R/W | Adaptive Voltage Scaling (AVS) |
| 22 | adg | 1 | R/W | ADG |
| 21 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 20 | spmi | 0 | R/W | IMP Scratchpad memory (SPMI0, SPMI1) |
| 19 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 18 | impslv | 0 | R/W | IMP AXI slave |
| 17 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 16 | impdta | 0 | R/W | IMP DTA |
| 15 to 6 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 5 | impdmac1 | 0 | R/W | IMP DMAC1 |
| 4 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.3 Module Stop Control Register 2 (MSTPCR2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | avb2:rgmii2 | avb1:rgmii1 | avb0:rgmii0 | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 14 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 13 | avb2:rgmii2 | 1 | R/W | EtherAVB (AVB2), RGMII2 |
| 12 | avb1:rgmii1 | 1 | R/W | EtherAVB (AVB1), RGMII1 |
| 11 | avb0:rgmii0 | 1 | R/W | EtherAVB (AVB0), RGMII0 |
| 10 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.4 Module Stop Control Register 3 (MSTPCR3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | csitop0 | — | — | canfd | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 | csitop0 | 1 | R/W | CSI40 (CSI-2-RX0) |
| 30 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 29 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 28 | canfd | 1 | R/W | CAN-FD |
| 27 to 0 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.5 Module Stop Control Register 4 (MSTPCR4)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | dsitxlink0 | doc2ch | — | — | dis0 | — | ddr1 ddr0 | — | — | — | — | — | — | — | — | csitop1 |
| Initial | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R/W | R/W | R | R | R/W | R | R/W | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 16 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 15 | dsitxlink0 | 1 | R/W | MIPI-DSI (DSI-TX0) |
| 14 | doc2ch | 1 | R/W | DOC |
| 13 to 12 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 11 | dis0 | 1 | R/W | DU |
| 10 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 9 | ddr1:ddr0 | 0 | R/W | LPDDR5/LPDDR4X |
| 8 to 1 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 0 | csitop1 | 1 | R/W | CSI40 (CSI-2-RX1) |

### 9.2.3.6 Module Stop Control Register 5 (MSTPCR5)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | intap | ims1 | ims0 | — | — | imr1 | imr0 | — | — | — | i2c3 | i2c2 | i2c1 | i2c0 | hscif3 | hscif2 |
| Initial | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | hscif1 | hscif0 | fray00 | — | — | — | — | fcpvd0 | — | — | — | — | — | — | — | — |
| Initial | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R | R | R | R | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 | intap | 1 | R/W | Reset for INTC-AP (including GIC600) |
| 30 | ims1 | 1 | R/W | IMR-LX5 (without extension) |
| 29 | ims0 | 1 | R/W | IMR-LX5 (without extension) |
| 28 to 27 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 26 | imr1 | 1 | R/W | IMR-LX5 (with Rotator&Scaler Extension) |
| 25 | imr0 | 1 | R/W | IMR-LX5 (with Rotator&Scaler Extension) |
| 24 to 22 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 21 | i2c3 | 0 | R/W | I2C channel 3 |
| 20 | i2c2 | 0 | R/W | I2C channel 2 |
| 19 | i2c1 | 0 | R/W | I2C channel 1 |
| 18 | i2c0 | 0 | R/W | I2C channel 0 |
| 17 | hscif3 | 1 | R/W | HSCIF channel 3 |
| 16 | hscif2 | 1 | R/W | HSCIF channel 2 |
| 15 | hscif1 | 1 | R/W | HSCIF channel 1 |
| 14 | hscif0 | 1 | R/W | HSCIF channel 0 |
| 13 | fray00 | 0 | R/W | FlexRay (reset synchronized with bus clock shown in Figure 103.1) |
| 12 to 9 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 8 | fcpvd0 | 1 | R/W | FCPVD (FCPV for VSPD0) |
| 7 to 0 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.7 Module Stop Control Register 6 (MSTPCR6)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | rtdm1 | rtdm0 | rpc | pwm | — | — | — | pcie0 | msi5 | msi4 | msi3 | msi2 | msi1 | msi0 | mfi | ivcp1e |
| Initial | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | ispcs1 | ispcs0 | irqc | — | — | inttp | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R | R | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 | rtdm1 | 0 | R/W | RT-DMAC channel 1 |
| 30 | rtdm0 | 0 | R/W | RT-DMAC channel 0 |
| 29 | rpc | 0 | R/W | RPC |
| 28 | pwm | 1 | R/W | PWM9, PWM8, PWM7, PWM6, PWM5, PWM4, PWM3, PWM2, PWM1, PWM0 |
| 27 to 25 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 24 | pcie0 | 1 | R/W | PCIEC ch0 Cold-reset |
| 23 | msi5 | 0 | R/W | MSIOF channel 5 (MSIOF5) |
| 22 | msi4 | 0 | R/W | MSIOF channel 4 (MSIOF4) |
| 21 | msi3 | 0 | R/W | MSIOF channel 3 (MSIOF3) |
| 20 | msi2 | 0 | R/W | MSIOF channel 2 (MSIOF2) |
| 19 | msi1 | 0 | R/W | MSIOF channel 1 (MSIOF1) |
| 18 | msi0 | 0 | R/W | MSIOF channel 0 (MSIOF0) |
| 17 | mfi | 0 | R | MFI |
| 16 | ivcp1e | 1 | R/W | iVCP1E |
| 15 to 14 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 13 | ispcs1 | 1 | R/W | ISP (Channel Selector) for ISP channel 1 |
| 12 | ispcs0 | 1 | R/W | ISP (Channel Selector) for ISP channel 0 |
| 11 | irqc | 0 | R/W | INTC-EX (IRQC) |
| 10 to 9 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 8 | inttp | 0 | R/W | INTC-Monitor (INTTP) |
| 7 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.8 Module Stop Control Register 7 (MSTPCR7)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | vin01 | vin00 | — | aes-acc ch7 | aes-acc ch6 | aes-acc ch5 | aes-acc ch4 | aes-acc ch3 | aes-acc ch2 | aes-acc ch1 | aes-acc ch0 | aes-acc wrap | — | tpu0 | tmu4 | tmu3 |
| Initial | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | tmu2 | mu1 | tmu0 | — | — | sydm2 | sydm1 | — | sdhi0 | scif4 | scif3 | scif2 | scif1 | scif0 | — | — |
| Initial | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 | vin01 | 1 | R/W | VIN1 |
| 30 | vin00 | 1 | R/W | VIN0 |
| 29 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 28 | aes-acc ch7 | 1 | R/W | AES-ACC Ch7 |
| 27 | aes-acc ch6 | 1 | R/W | AES-ACC Ch6 |
| 26 | aes-acc ch5 | 1 | R/W | AES-ACC Ch5 |
| 25 | aes-acc ch4 | 1 | R/W | AES-ACC Ch4 |
| 24 | aes-acc ch3 | 1 | R/W | AES-ACC Ch3 |
| 23 | aes-acc ch2 | 1 | R/W | AES-ACC Ch2 |
| 22 | aes-acc ch1 | 1 | R/W | AES-ACC Ch1 |
| 21 | aes-acc ch0 | 1 | R/W | AES-ACC Ch0 |
| 20 | aes-acc wrap | 1 | R/W | AES-ACC Wrapper |
| 19 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 18 | tpu0 | 1 | R/W | TPU |
| 17 | tmu4 | 1 | R/W | TMU channel 4 |
| 16 | tmu3 | 1 | R/W | TMU channel 3 |
| 15 | tmu2 | 1 | R/W | TMU channel 2 |
| 14 | tmu1 | 1 | R/W | TMU channel 1 |
| 13 | tmu0 | 1 | R/W | TMU channel 0 |
| 12 to 11 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 10 | sydm2 | 0 | R/W | SYS-DMAC2 |
| 9 | sydm1 | 0 | R/W | SYS-DMAC1 |
| 8 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 7 | secrom | 0 | R/W | Secure ROM |
| 6 | sdhi0 | 1 | R/W | SDHI |
| 5 | scif4 | 0 | R/W | SCIF-4 |
| 4 | scif3 | 0 | R/W | SCIF-3 |
| 3 | scif1 | 0 | R/W | SCIF-1 |
| 2 | scif0 | 0 | R/W | SCIF-0 |
| 1 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.9 Module Stop Control Register 8 (MSTPCR8)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | vspd0 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | vin17 | vin16 | vin15 | vin14 | vin13 | vin12 | vin11 | vin10 | vin07 | vin06 | vin05 | vin04 | vin03 | vin02 |
| Initial | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note: "—" : Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits.

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 30 | vspd0 | 1 | R/W | VSPD0 |
| 29 to 14 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 13 | vin17 | 1 | R/W | VIN17 |
| 12 | vin16 | 1 | R/W | VIN16 |
| 11 | vin15 | 1 | R/W | VIN15 |
| 10 | vin14 | 1 | R/W | VIN14 |
| 9 | vin13 | 1 | R/W | VIN13 |
| 8 | vin12 | 1 | R/W | VIN12 |
| 7 | vin11 | 1 | R/W | VIN11 |
| 6 | vin10 | 1 | R/W | VIN10 |
| 5 | vin07 | 1 | R/W | VIN7 |
| 4 | vin06 | 1 | R/W | VIN6 |
| 3 | vin05 | 1 | R/W | VIN5 |
| 2 | vin04 | 1 | R/W | VIN4 |
| 1 | vin03 | 1 | R/W | VIN3 |
| 0 | vin02 | 1 | R/W | VIN2 |

### 9.2.3.10 Module Stop Control Register 9 (MSTPCR9)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | ucmt | tsc2:tsc1 | — | pfc2 | pfc1 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pfc0 | — | cmt3 | cmt2 | cmt1 | cmt0 | — | — | wdt1:wdt0 | wcrc3 | wcrc2 | wcrc1 | wcrc0 | — | — | — |
| Initial | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| R/W: | R/W | R | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 20 | ucmt | 0 | R/W | SUCMT |
| 19 | tsc2:tsc1 | 0 | R/W | TSC4, TSC3, TSC2, TSC1 |
| 18 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 17 | pfc2 | 0 | R/W | PFC2 Bus Domain0 (Table 7.2 in 7. Pin Function (PFC/GPIO)) |
| 16 | pfc1 | 0 | R/W | PFC1 Bus Domain0 (Table 7.2 in 7. Pin Function (PFC/GPIO)) |
| 15 | pfc0 | 0 | R/W | PFC0 Bus Domain0 (Table 7.2 in 7. Pin Function (PFC/GPIO)) |
| 14 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 13 | cmt3 | 1 | R/W | CMT3 |
| 12 | cmt2 | 1 | R/W | CMT2 |
| 11 | cmt1 | 1 | R/W | CMT1 |
| 10 | cmt0 | 1 | R/W | CMT0 |
| 9 to 8 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 7 | wdt1:wdt0 | 0 | R/W | WDT |
| 6 | wcrc3 | 1 | R/W | WCRC3 |
| 5 | wcrc2 | 1 | R/W | WCRC2 |
| 4 | wcrc1 | 1 | R/W | WCRC1 |
| 3 | wcrc0 | 1 | R/W | WCRC0 |
| 2 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.11 Module Stop Control Register 10 (MSTPCR10)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | vspx0 | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 29 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 28 | vspx0 | 1 | R/W | VSPX0 |
| 27 to 0 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.12 Module Stop Control Register 11 (MSTPCR11)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | fcpvx0 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 1 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 0 | fcpvx0 | 1 | R/W | FCPVX0 |

### 9.2.3.13 Module Stop Control Register 12 (MSTPCR12)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | kcrc4 | fso | — | crc3 | crc2 | crc1 | crc0 | — | — | — | — | — | — | — | — | — |
| Initial | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 | kcrc4 | 1 | R/W | KCRC4 |
| 30 | fso | 0 | R/W | RFSO |
| 29 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 28 | crc3 | 1 | R/W | CRC3 |
| 27 | crc2 | 1 | R/W | CRC2 |
| 26 | crc1 | 1 | R/W | CRC1 |
| 25 | crc0 | 1 | R/W | CRC0 |
| 24 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.14 Module Stop Control Register 13 (MSTPCR13)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | mti | — | kcrc7 | kcrc6 | kcrc5 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W |

Note: "—" : Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits.

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 6 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 5 | - | 1 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 4 | mti | 1 | R/W | MTI |
| 3 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 2 | kcrc7 | 1 | R/W | KCRC7 |
| 1 | kcrc6 | 1 | R/W | KCRC6 |
| 0 | kcrc5 | 1 | R/W | KCRC5 |

### 9.2.3.15 Module Stop Control Register 14 (MSTPCR14)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.16 Module Stop Control Register 15 (MSTPCR15)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note: "—" : Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits.

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.17 Module Stop Control Register 16 (MSTPCR16)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.18 Module Stop Control Register 17 (MSTPCR17)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.19 Module Stop Control Register 18 (MSTPCR18)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.20 Module Stop Control Register 19 (MSTPCR19)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.21 Module Stop Control Register 20 (MSTPCR20)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.22 Module Stop Control Register 21 (MSTPCR21)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.23 Module Stop Control Register 22 (MSTPCR22)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.24 Module Stop Control Register 23 (MSTPCR23)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.25 Module Stop Control Register 24 (MSTPCR24)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.26 Module Stop Control Register 25 (MSTPCR25)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.27 Module Stop Control Register 26 (MSTPCR26)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.28 Module Stop Control Register 27 (MSTPCR27)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | csbrg_ir_a3 | — | — | — | — | tsn | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R | R | R | R | R/W | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | cve3 | cve2 | cve1 | cve0 | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 29 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 28 | csbrg_ir_a3 | 0 | R/W | Debug and Trace |
| 27 to 24 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 23 | tsn | 1 | R/W | EtherTSN |
| 22 to 10 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 9 | cve3 | 0 | R/W | CVe3 |
| 8 | cve2 | 0 | R/W | CVe2 |
| 7 | cve1 | 0 | R/W | CVe1 |
| 6 | cve0 | 0 | R/W | CVe0 |
| 5 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| | | | | |

#### 9.2.3.29 Module Stop Control Register 28 (MSTPCR28)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | vdsp1_csb | — | — | — | — | — | — | — | — | vdsp0_csb | — | — | — | fcprc | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R/W | R | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | vdsp1_bus | vdsp0_bus | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 30 | vdsp1_csb | 0 | R/W | VDSP1 (See "U39 VDSP" Block diagram) |
| 29 to 22 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 21 | vdsp0_csb | 0 | R/W | VDSP0 (See "U39 VDSP" Block diagram) |
| 20 to 18 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 17 | fcprc | 1 | R/W | Compression bus in Main Memory domain |
| 16 to 3 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 2 | vdsp1_bus | 0 | R/W | VDSP1 bus block (See "U39 VDSP" Block diagram) |
| 1 | vdsp0_bus | 0 | R/W | VDSP0 bus block (See "U39 VDSP" Block diagram) |
| 0 | - | 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.3.30 Module Stop Control Register 29 (MSTPCR29)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ssi | ssiu | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Target Module for Module Stop |
|---|---|---|---|---|
| 31 to 28 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |
| 27 | ssi | 1 | R/W | SSI |
| 26 | ssiu | 1 | R/W | SSIU |
| 25 to 0 | - | All 0 | R | Reserved. These bits are always read as an initial value. System operation cannot be guaranteed when the different value from the initial value is written to these bits. |

### 9.2.4 Module Stop Status Register (MSTPSRn (n = 0 to 29))

MSTPSRn is a 32-bit readable register that indicates whether the on-chip modules are in the module standby state. Setting a bit in this register to 1 stops supply of the clock signal to the corresponding module and the setting a bit to 0 enables clock supply to the corresponding module. The bit assignment and the initial value are same as that of MSTPCRn (n = 0 to 29).

0: Supply of the clock signal to the corresponding module is enabled.
1: Supply of the clock signal to the corresponding module is stopped.

### 9.2.5 Request/Acknowledge Handshake Select Register (RAHSRn (n = 0 to 29)

RAHSRn is a 32-bit readable/writable register. The bit assignment of RAHSRn is same as that of SRCRn (the initial value of RAHSRn is different from that of SRCRn). The usage method is described in Section 9.3 (4).

Register protection function is covered by a function shown in Section 9.1.1 (5).

### 9.2.6 Domain x Write Access Control Register for yyy (DxWACR_yyy (x = 0 to 3))

DxWACR_yyy is a 32-bit readable/writable register. The register specification is described in Section 9.4.2. The register is applied for the registers below.

- MSTPCRn (n = 0 to 29)
- SRCRn (n = 0 to 29)
- SRSTCLRn (n = 0 to 29)
- RAHSRn (n = 0 to 29))

## 9.3 Operation

### (1) Module Standby

The following is procedure for Module Stop (turn off clock).

1. Set to 1 at the target module bit of MSTPCRn by read-modify-write.
2. Check the bit that the target module of MSTPSRn is 1.

The following is procedure for clear Module Stop (turn on clock).

1. Set to 0 at the target module bit of MSTPCRn by read-modify-write.
2. Check the bit that the target module of MSTPSRn is 0.

### (2) Software Reset

There are three types ((A), (B), (C)) of Software Reset flows as the following:

A) See Figure 9.3.1 (a), 9.3.1 (b), which is applied to issue the Software reset for Master modules in Power Shutoff domain (IMP domain, VIO (Video IO) domain, VC (Video Codec) domain, VIP (Vision IP Processor), GSX (Graphics) domain). See 18. AXI-Bus about which modules corresponds with the Master modules.

B) See Figure 9.3.2, which is applied to issue the Software Reset for Master modules in Always Power On domain (RT (Real Time) domain, Peripheral domain, HSC (High Speed Communication) domain, Main Memory domain, Slave Access Bus domain). See 18. AXI-Bus about which modules corresponds with the Master modules.

C) See Figure 9.3.3, which is applied to issue the Software Reset for Slave modules. These flows are effective to avoid the system hang-up. See 18. AXI-Bus about which modules corresponds with the Slave modules.

### (3) Reset Status Monitor

See Section 9.5.2 (3).

### (4) Force Reset

The following is procedure for Force Reset. When the function is used, the Software Reset is asserted without any handshakes between CPG and target modules. It should be noted that the software reset is forced to be asserted without guaranteeing the system operation in the chip.

1. Writing 0 to the corresponding target module bit of RAHSRn.
2. Writing 1 to the corresponding target module bit of SRCRn.
3. Software reset is asserted to the target module without request/acknowledge handshake between CPG and the module.
4. The corresponding bit of SRCRn can be cleared by writing 1 to the corresponding bit of SRSTCLRn.

### (5) Bus Access Protection

See Section 9.4.2.

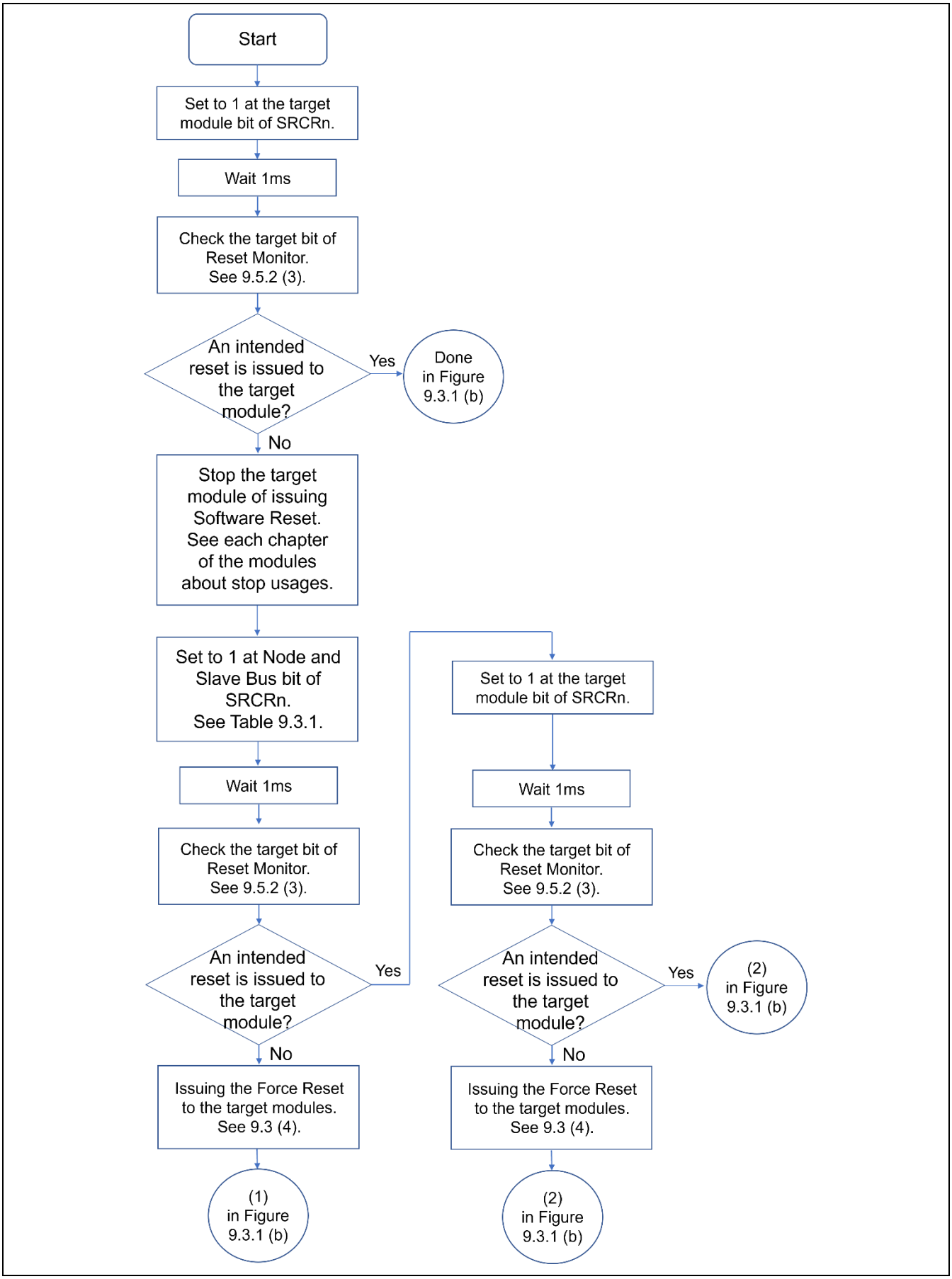



**Figure 9.3.1 (a) Software Reset flow (A)**

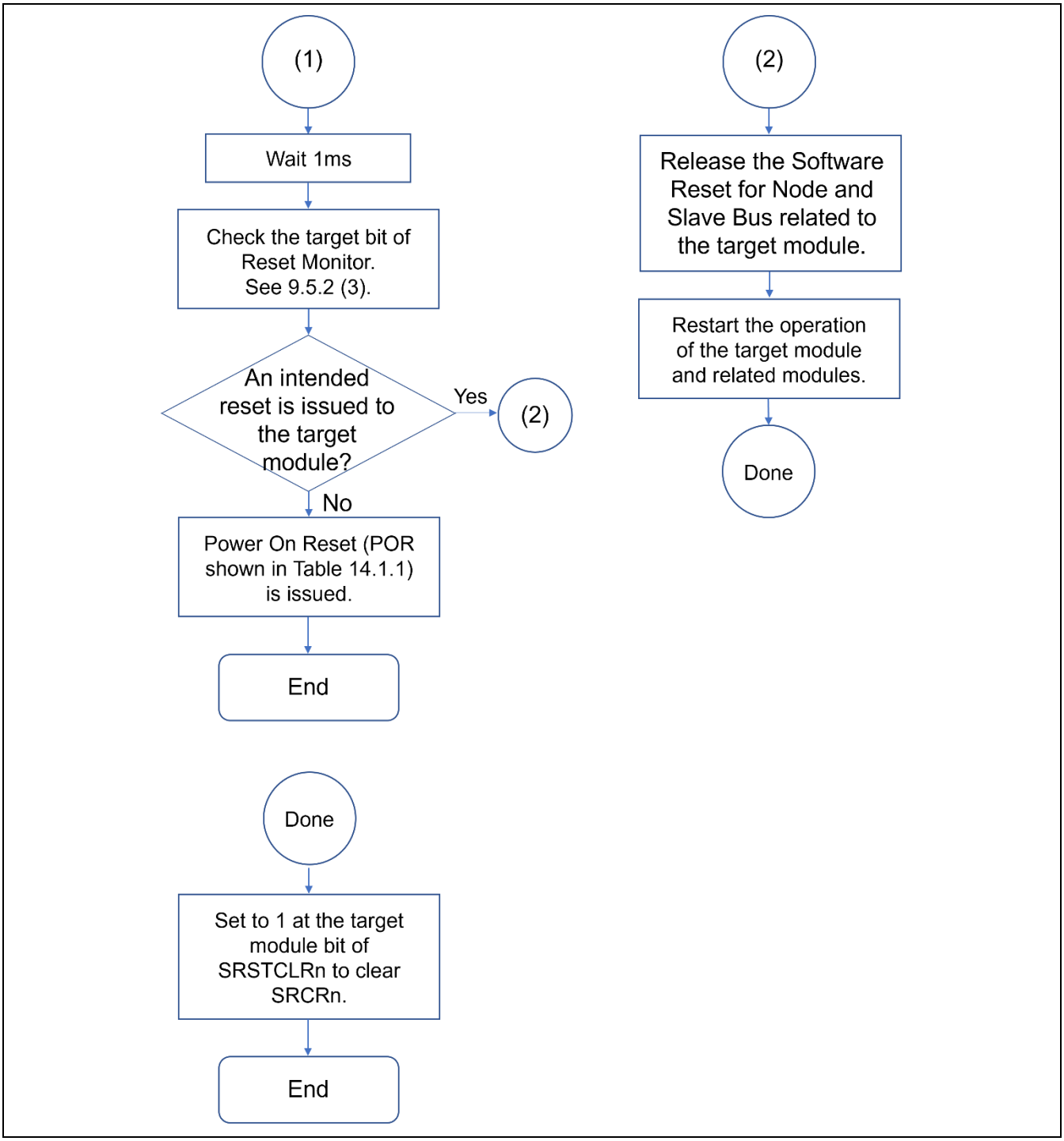



**Figure 9.3.1 (b) Software Reset flow (A) (Continuation from Figure 9.3.1 (a))**

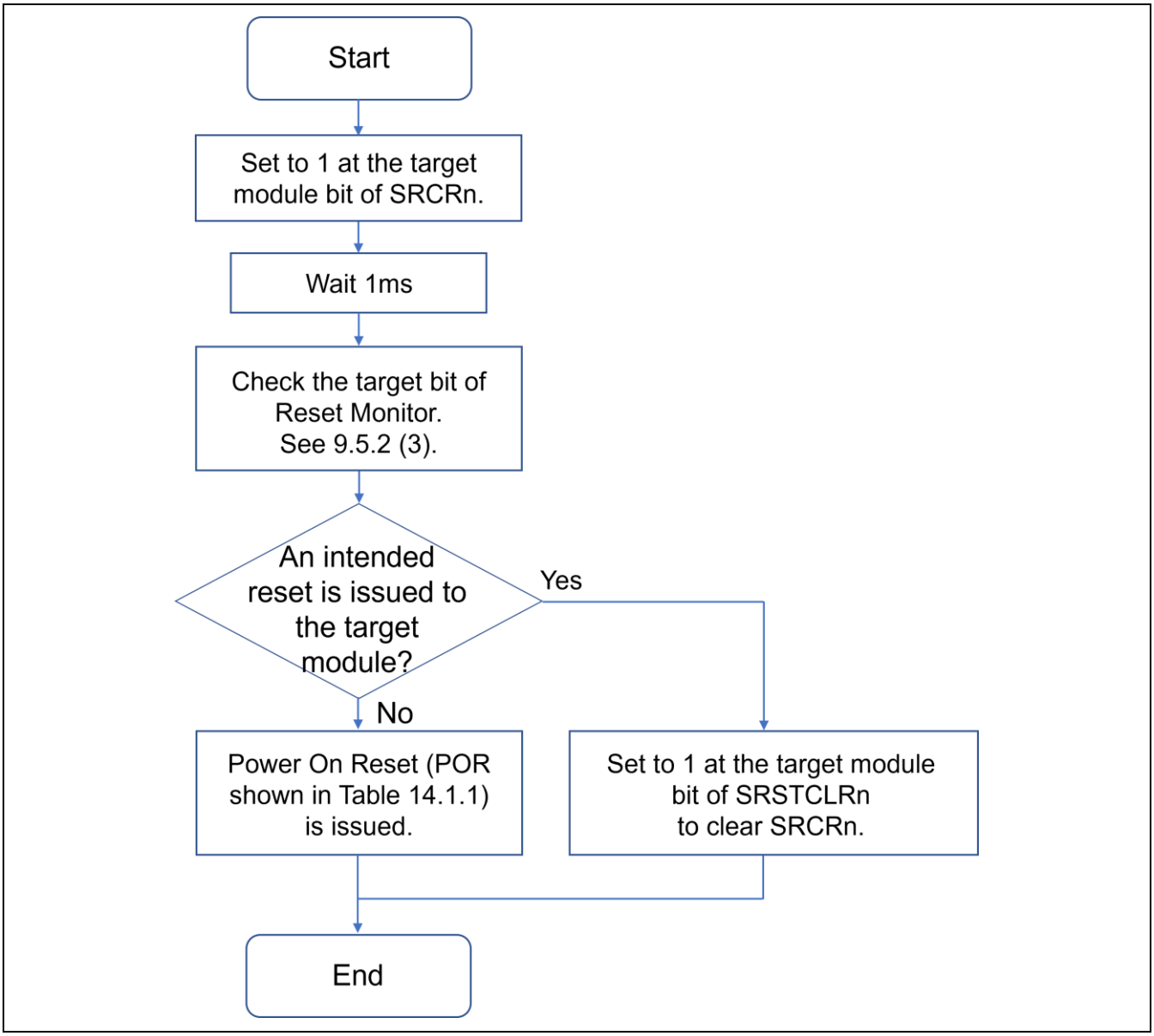



**Figure 9.3.2 Software Reset flow (B)**

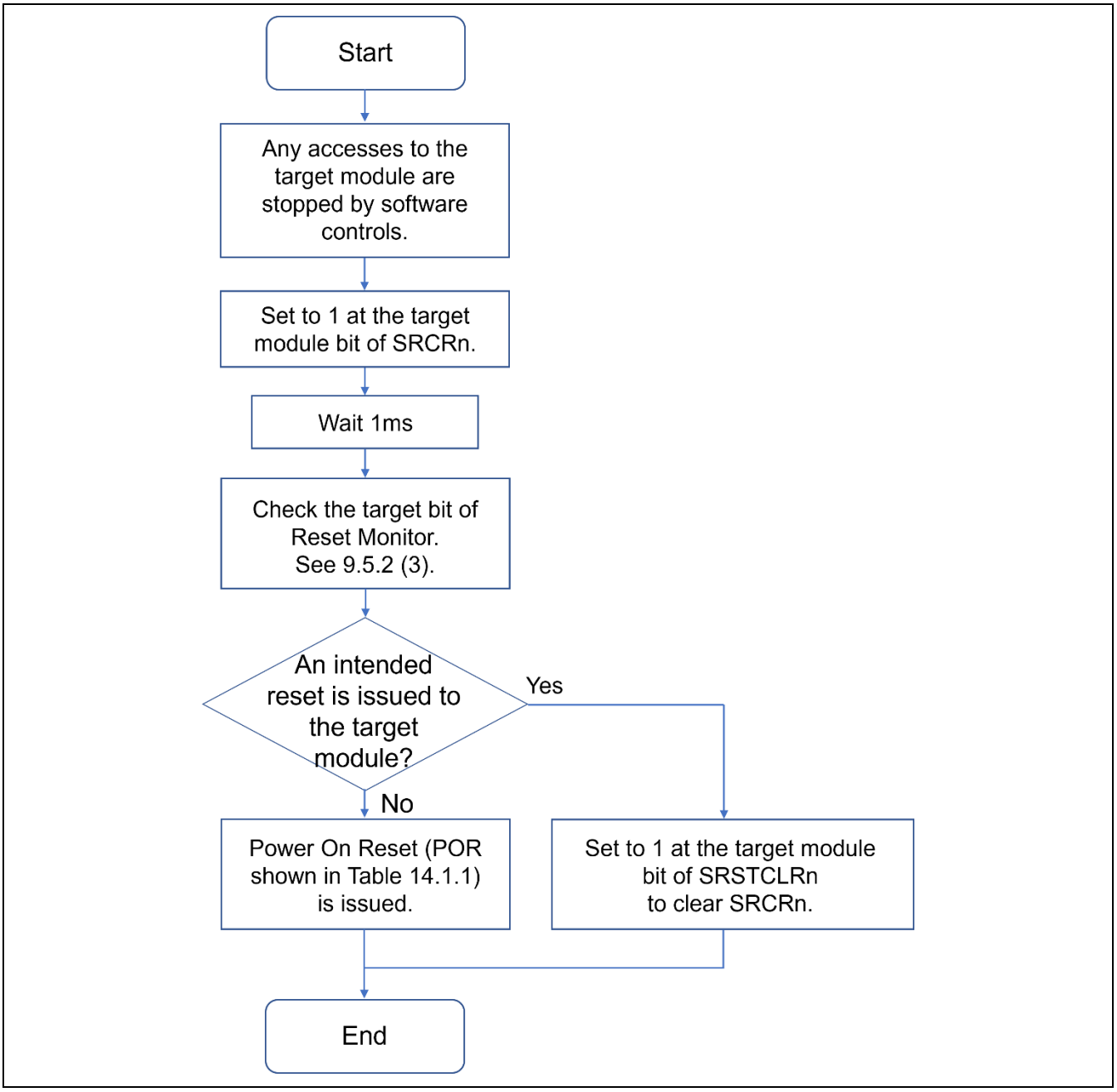



**Figure 9.3.3 Software Reset flow (C)**

**Table 9.3.1 Target Node and Slave Bus for issuing and releasing Software Reset in Software Reset flow (A)**

| Register Name | Bit No. | Bit Name | Target Node and Slave Bus |
|---|---|---|---|
| SRCR1 | Bit 13 | SRT13 | AXI Bus module in IMP domain. |
| SRCR1 | Bit 12 | SRT12 | Bus register of AXI Bus module in IMP domain. |
| SRCR2 | Bit 31 | SRT31 | AXI Slave Bus module in Main Memory domain. |
| SRCR2 | Bit 30 | SRT30 | Bus register of AXI Bus module in Peripheral domain. |
| SRCR2 | Bit 29 | SRT29 | AXI Bus module in Peripheral domain. |
| SRCR2 | Bit 28 | SRT28 | Bus register of AXI Slave Bus module in Slave Access bus domain. |
| SRCR2 | Bit 27 | SRT27 | AXI Slave Bus module in Slave Access bus domain. |
| SRCR2 | Bit 25 | SRT25 | AXI Bus module for Cortex-R52 in RT (Real Time) domain. |
| SRCR2 | Bit 24 | SRT24 | Bus register of AXI Bus module for Cortex-R52 in RT (Real Time) domain. |
| SRCR2 | Bit 23 | SRT23 | Bus register of AXI Bus module for RT-DMAC in RT (Real Time) domain. |
| SRCR2 | Bit 22 | SRT22 | AXI Bus module for RT-DMAC in RT (Real Time) domain. |
| SRCR2 | Bit 21 | SRT21 | Bus register of AXI Bus module in RT (Real Time) domain. |
| SRCR2 | Bit 20 | SRT20 | AXI Bus module in RT (Real Time) domain. |
| SRCR2 | Bit 19 | SRT19 | AXI Bus module for GSX (Graphics) domain. |
| SRCR2 | Bit 18 | SRT18 | Bus register of AXI Bus module for GSX (Graphics) domain. |
| SRCR2 | Bit 8 | SRT8 | APB Bus module for MSIOF in Peripheral domain. |
| SRCR2 | Bit 7 | SRT7 | Bus register of APB Bus module for MSIOF in Peripheral domain. |
| SRCR2 | Bit 6 | SRT6 | Bus register of APB Bus module operated with SASYNCPERD1/2/4φ in Peripheral domain. |
| SRCR2 | Bit 5 | SRT5 | APB Bus module operated with SASYNCPERD1/2/4φ in Peripheral domain. |
| SRCR2 | Bit 4 | SRT4 | APB Bus module in RT (Real Time) domain. |
| SRCR2 | Bit 3 | SRT3 | Bus register of APB Bus module in RT (Real Time) domain. |
| SRCR3 | Bit 0 | SRT0 | Bus register of AXI Slave Bus module in Main Memory domain. |
| SRCR3 | Bit 1 | SRT1 | AXI Slave Bus module (North part) in Slave Access bus domain. |
| SRCR3 | Bit 2 | SRT2 | Bus register of AXI Slave Bus module (North part) in Slave Access bus domain. |
| SRCR3 | Bit 3 | SRT3 | Bus register of AXI Bus module in Peripheral domain. |
| SRCR3 | Bit 4 | SRT4 | AXI Bus module in Peripheral domain. |
| SRCR3 | Bit 7 | SRT7 | Bus module (from CSD STM to PCI). |
| SRCR3 | Bit 8 | SRT8 | Bus register of Bus module (from CSD STM to PCI). |
| SRCR3 | Bit 11 | SRT11 | Bus register of AXI Bus module in VC (Video Codec) domain. |
| SRCR3 | Bit 12 | SRT12 | AXI Bus module in VC (Video Codec) domain. |
| SRCR3 | Bit 16 | SRT16 | Bus register of AXI Bus module in VIO (Video IO) domain. |
| SRCR3 | Bit 17 | SRT17 | AXI Bus module in VIO (Video IO) domain. |
| SRCR3 | Bit 18 | SRT18 | Bus register of AXI Bus module in VIO (Video IO) domain. |
| SRCR3 | Bit 22 | SRT22 | Bus register of AXI Bus module in VIP (Vision IP Processor) domain. |
| SRCR3 | Bit 23 | SRT23 | AXI Bus module in VIP (Vision IP Processor) domain. |
| SRCR11 | Bit 8 | SRT8 | AXI Bus module in VIO (Video IO) domain. |
| SRCR15 | Bit 1 | SRT1 | Bus register of AXI Bus module in HSC (High Speed Communication) domain. |
| SRCR15 | Bit 2 | SRT2 | AXI Bus module in HSC (High Speed Communication) domain. |
| SRCR15 | Bit 22 | SRT22 | APB Bus module in VIO (Video IO) domain. |
| SRCR15 | Bit 23 | SRT23 | Bus register of APB Bus module in VIO (Video IO) domain. |

## 9.4 Usage Notes

### 9.4.1 Write Protect Function for Module Standby and Software Reset registers

Write Protect Function specification is same as that of CPG registers. The detail specification is described in Section 8.4.1.

Example)
When writing a value of "H'0000 00FF" to MSTPCR0 register, the following is procedure
- Write "H'FFFF FF00" (inverted value of "H'0000 00FF") to H'E615 0000.
- After that, write "H'0000 00FF to MSTPCR0 ("H'E615 2D00").

### 9.4.2 Bus Access Protect for Module Standby and Software Reset registers

#### (1) Overview

Overview is same as that of MSTPRST registers. The detail specification is described in Section 8.4.2 (1).

#### (2) Domain x Write Access Control Register (DxWACR_YYY) (x = 0, 1, 2, 3)

Each bit of MSTPCRn, SRCRn, SRSTCLRn, RAHSRn (n = 0 to 29) is protected by DxWACR_MSTPCRn, DxWACR_SRCRn, DxWACR_SRSTCLRn, DxWACR_RAHSRn (x = 0, 1, 2, 3), respectively. DxWACR_YYY can be accessed only by Domain 0.

Figure 9.4.1 shows one example of usage of DxWACR_YYY registers. In this case, only Domain 2 can write a value to MSTPCR0.bit1.

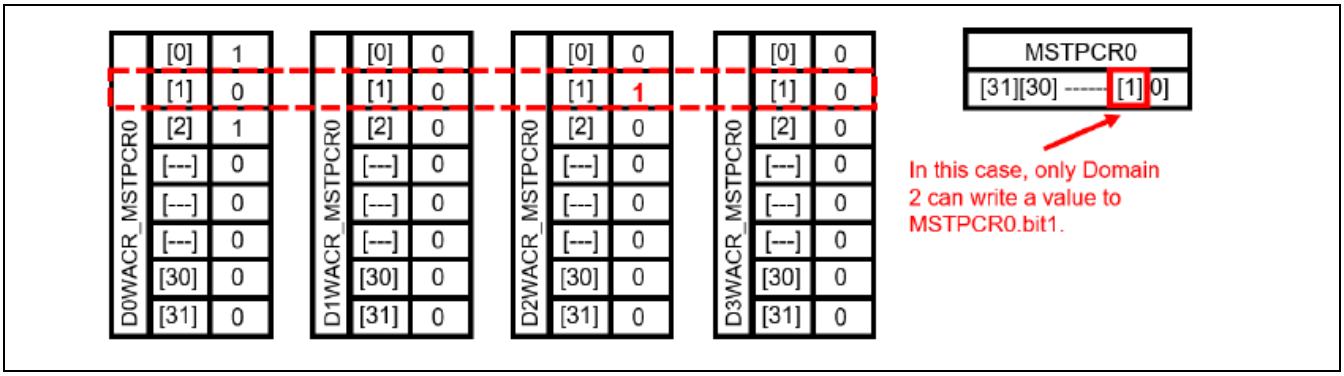



**Figure 9.4.1 Usage of D0WACR_YYY, D1WACR_YYY, D2WACR_YYY, D3WACR_YYY**

#### (3) Check method of illegal write accesses

If there are illegal write accesses for the MSTPRST registers, an error notification is reported to ECM (ECMERRxxxR00 Bit29). In addition, the error notification is cleared by using CPGWPTCSR (Write Access Protect Control/Status Register). In detail, see 8.2.61, in which the register usage is described.

### 9.4.3 Force Reset

When target modules and the connecting bus are under deadlock due to any failures, the modules cannot be initialized by the normal Software Reset (the corresponding bit of RAHSRn is 0). In case of it, the Force Reset function is effective. If

there are not any failures for the modules, the Software Reset (shown in 9.3 (2)) must be used to guarantee the system operation in a chip.

## 9.5 Safety Mechanisms

### 9.5.1 Overview

- CPG (Module Standby, Software Reset) module supports Reset Status Monitor function. Reset Status Monitor is a safety mechanism which confirms that an intended reset is issued and which defects that an unintended reset occurs.

- The access rights for changing independent register bits in CPG (Module Standby, Software Reset) shall be configurable for each Domain 0, 1, 2, 3. The detail specification is described in 9.5.3.

### 9.5.2 Reset Status Monitor

#### (1) Overview

CPG (Module Standby, Software Reset function) can issue reset signals to each module, and it can issue reset signals to each CPU (Cortex-A76 and Cortex-R52). The software on the chip can issue software reset signals to each module and each CPU.
Reset status monitor checks that an intended reset is issued and detects that an unintented reset occurs by using the software registers of FSRCHKSETRn, FSRCHKCLRRn, FSRCHKRAn, FSRCHKRBn (n = 0 to 27).

#### (2) Register Description

As shown in Section 9.1.1 (3), the register corresponds with the following one. The detail specification is described in Section 8.2.7, 8.2.8, 8.2.9, 8.2.10.

- Functional Safety Reset Check Register A (FSRCHKRAn (n = 0 to 29))
- Functional Safety Reset Check Register B (FSRCHKRBn (n = 0 to 29))
- Functional Safety Reset Check Set Register (FSRCHKSETRn (n = 0 to 29))
- Functional Safety Reset Check Clear Register (FSRCHKCLRRn (n = 0 to 29))

#### (3) Operation

In order to check that an intended reset is issued and defect that an unintended reset occurs, the registers are used as shown in the block diagram (Figure 9.1.1). The following is a procedure for using the Reset Status Monitor.

1. Each bit of FSRCHKRAn and FSRCHKRBn can be cleared by writing 1 to the corresponding bit of FSRCHKCLRRn before using the Reset Status Monitor function.

2. When writing 1 to the designated bit of FSRCHKSETRn (n = 0 to 29), the corresponding bits of both FSRCHKRAn and FSRCHKRBn are set to 1.

3. The registers of FSRCHKRAn and FSRCHKRBn can be used to check whether an intended reset is issued or not.

4. Each bit can be cleared by writing 1 to the corresponding bit of FSRCHKCLRRn.

### 9.5.3 Safety Access Protection

#### (1) Overview

CPG (Module Standby, Software Reset) provides a safety access protection mechanism to guarantee “Freedom From Interference (FFI)” when accessing the registers belonging to the modules. Several masters (Cortex-A76, Cortex-R52) take control of different portions and modules of the chip. The access protection mechanism shall enable sharing and splitting the functionality among cores.

The access protection mechanism for the control registers belonging to CPG (Module Standby, Software Reset) module shall be aligned with the Region ID concept. In detail, see Section 9.4.2. The module has critical system control functions (Module Standby, Software Reset). Therefore, the strong access restrictions are necessary for the module.

#### (2) Register Description

As shown in Section 9.4.2 (2), the register corresponds with D0WACRAn_YYY, D1WACRAn_YYY, D2WACRAn_YYY, D3WACRAn_YYY (n = 0 to 63) in Section 9.2.6.

#### (3) Operation

See Section 9.4.2 (1) and 9.4.2 (2).

# 10. Advanced Power Management Unit

## 10.1 Overview

The Advanced Power Management Unit (APMU) is a module to control power supply and clock supply to the AP-System Core and Realtime Core. It also provides the reset vector base address for AP-System Core and Realtime Core.

### 10.1.1 Features

This module includes the following functions and features:

- Controls power supply to each CPU core and L3 cache area in AP-System core.
- Supports following power modes of Arm® Cortex®-A76 CPU cores defined by Arm.
  - On
  - Emulated off
  - Off
- The following power modes of Cortex-A76 CPU cores are not supported.
  - Debug recovery
  - Core dynamic retention
- Supports following power modes of Cortex-A76 CPU cluster defined by Arm.
  - On
  - Functional retention
  - Memory retention
  - Off
  - L3 cache partial powerdown
- The following power mode of Cortex-A76 CPU cluster is not supported.
  - Debug recovery
- Not support boot mode of AP-System Core.
- Controls power supply to each core of Cortex®-R52.
- Supports following power modes of Cortex-R52 defined by Arm.
  - RUN
  - SHUTDOWN
  - Debug over powerdown
- This module has boot address registers for Cortex-A76 and Cortex-R52.
- This module is implemented as DCLS.

Cortex-A76 CPU core power mode defined by Arm is described in the Cortex-A76 Technical Reference Manual (TRM) (Reference [1]). Cortex-A76 CPU cluster power mode defined by Arm is described in the DynamIQ Shared Unit (DSU) TRM (Reference [2]). Cortex-R52 power mode defined by Arm is described in the Cortex-R52 TRM (Reference [4]). The P-channel interface is used to communicate between APMU and Cortex-A76 CPU cores, clusters or Cortex-R52 when APMU controls the power mode for Cortex-A76 CPU cores, clusters or Cortex-R52. The P-channel interface specification is defined by Arm and it is described in the Reference [3].

### 10.1.2 Block Diagram

Figure 10.1 shows block diagram of APMU.

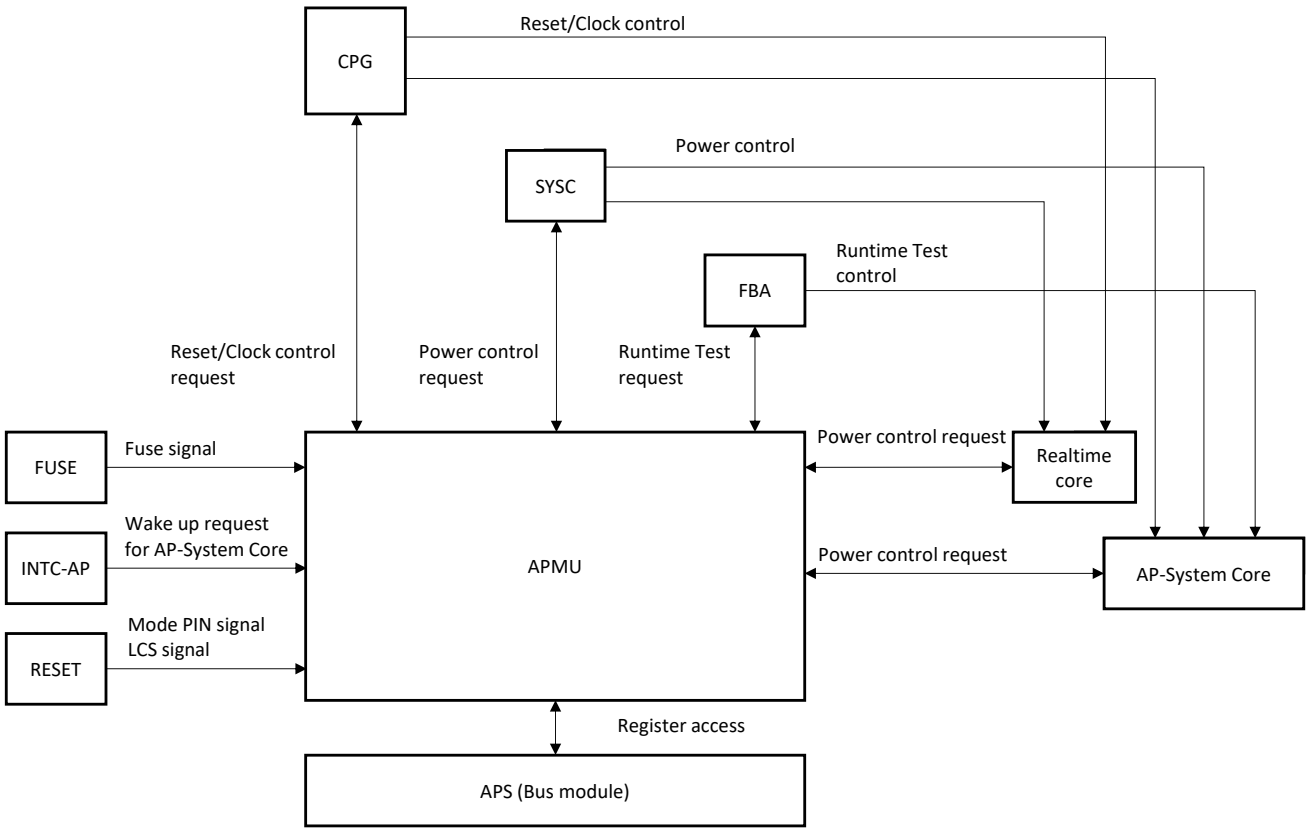



Figure 10.1 Block Diagram of APMU

### 10.1.3 External Pins

There are no external pins related to this function.

### 10.1.4 Register Configuration

Refer to the EXCEL file (R-CarV4M_UM_010_APMU_r0p81.xlsx).

Parameters Table

| Parameters | Range | Description |
|---|---|---|
| n | 0-3 | APMU register domain number |
| r | 0-2 | Realtime Core number |
| i | 0 | Subsystem number for AP-system core |
| x | 0-1 | Cluster number for AP-system core. Labeled Cluster0-Cluster1.<br>Register names expanded according to ”Register Groups Offset Table”.<br>Corresponding to “Module ID” of “CORTEXA76CLUSTER” register groups offset. |
| z | 0-3 | Core number for AP-system core. Labeled Core0-Core3<br>Register names expanded according to ”Register Groups Offset Table”.<br>Corresponding to “Module ID” of “CORTEXA76CORE” register groups offset. |

The following shows an example of Register Groups Offset Table.

| Register groups offset name | Module ID (=ID*1) | Base Address (=BaseAdd*1) | Register name (=RegName*1) | Symbol name (=Symbol *1) |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Sample | #XX | H'YY | Rel0 | _REL0 |

*1. These are the names listed in the following Register Configuration Table.

e.g. If following example are described in the Register Configuration Table and Module ID is #XX, that means as below.

Sample[ID,BaseAdd] = H'YY

Sample[ID,RegName] = Rel0

Sample[ID,Symbol] = _REL0

### 10.1.5 Connected Module

Connected modules with the APMU are described in Table 10.1.

Table 10.1 Connected Module

| Module name | Connected module name | Function of connected module |
|---|---|---|
| APMU | AP-System Core | APMU controls Cortex-A76 power status. |
| | Realtime Core | APMU controls Cortex-R52 reset status. |
| | CPG | APMU and CPG control Cortex-A76 reset status and Cortex-R52 reset status. |
| | SYSC | APMU and SYSC control Cortex-A76 power status and Cortex-R52 power status. |
| | INTC-AP | APMU gets request to wake up Cortex-A76 CPU core from interrupt controller. |
| | FUSE | FUSE defines the configurations of Cortex-A76 and Cortex-R52. APMU refers to their information. |
| | FBA | APMU and FBA control Runtime Test operation of Cortex-A76 CPU cores and clusters. |

### 10.1.6 Reference

[1] Arm® Cortex®-A76 Core Technical Reference Manual, Revision: r4p1
[2] Arm® DynamIQ™ Shared Unit Technical Reference Manual, Revision: r4p1
[3] AMBA Low Power Interface Specification: ARM Q-Channel and P-Channel Interfaces
[4] Arm® Cortex-R52 Processor Technical Reference Manual, Revision r1p4

## 10.2 Register Description

Refer to the EXCEL file (R-CarV4M_UM_010_APMU_r0p81.xlsx).

[Explanation of abbreviation of register]

Initial value: Value of the register after power-on reset

—: Undefined value

R/W: The bit or field is readable and writable.

R/WC0: The bit or field is readable and writable. Writing 0 to the bit initializes the bit. Writing 1 to the bit is ignored.

R/WC1: The bit or field is readable and writable. Writing 1 to the bit initializes the bit. Writing 0 to the bit is ignored.

R: The bit or field is readable only. When writing to the register, write 0 to it.

WO: The bit or field is writable. Note that values read from write-only bits are not guaranteed.

## 10.3 Operation

### 10.3.1 Power Mode

APMU controls power domains for Cortex-A76 CPU cores, clusters and cluster common parts. Each logic has their own power domain. Table 10.2 shows each power domain name and its description. Those power domains have hierarchical structure. Figure 10.2 shows power domain structure. Any power domain must be powered on whenever its upper power domain is powered on.

Table 10.2 Power Domain of AP-System Core

| Power Domain | Description | Power Domain | Description |
|---|---|---|---|
| A1E0D0C0 | Cortex-A76 CPU core 0 | A2E0D0 | Cortex-A76 CPU cluster 0 |
| A1E0D0C1 | Cortex-A76 CPU core 1 | | |
| A1E0D0C2 | Cortex-A76 CPU core 2 | | |
| A1E0D0C3 | Cortex-A76 CPU core 3 | | |

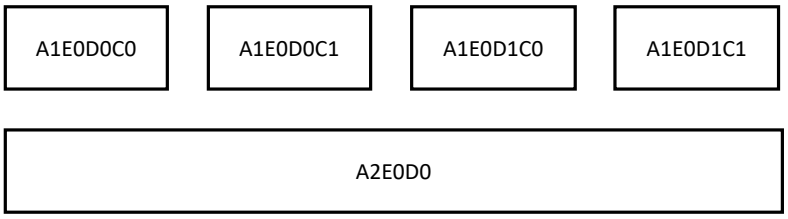



Figure 10.2 Power domain of AP-System Core

#### 10.3.1.1 Cortex-A76 CPU core power mode

Each Cortex-A76 CPU core has following power modes.

- ON mode

  This power mode is the same as On mode defined in Cortex-A76 TRM (Reference [1]). Cortex-A76 CPU core is powered on. When Cortex-A76 CPU core is in this power mode, Cortex-A76 CPU cluster including this core must be in ON mode.

- OFF mode

  This power mode is the same as Off mode defined in Cortex-A76 TRM (Reference [1]). Cortex-A76 CPU core is powered off.

- OFF_EMU mode

  This power mode is the same as Off (emulated) mode defined in Cortex-A76 TRM (Reference [1]). A warm reset of Cortex-A76 CPU core is asserted. So, Cortex-A76 CPU core debug domain is not initialized. And, Cortex-A76 CPU core is not powered off. When Cortex-A76 CPU core is in this power mode, Cortex-A76 CPU cluster including this core must be in ON mode.

- Pseudo OFF mode

  Cortex-A76 CPU core is Off mode defined in Cortex-A76 TRM (Reference [1]). However, Cortex-A76 CPU core is still powered on. This power mode is used for Runtime Test.

#### 10.3.1.2 Cortex-A76 CPU cluster power mode

Each Cortex-A76 CPU cluster has following power modes.

- ON mode

  This power mode is same as On mode defined in DSU TRM (Reference [2]). Cortex-A76 CPU cluster is powered on.

- OFF mode

  This power mode is same as Off mode defined in DSU TRM (Reference [2]). Cortex-A76 CPU cluster is powered off. When Cortex-A76 CPU cluster is in this power mode, all Cortex-A76 CPU cores which belong to this cluster must be in OFF mode.

- FUNC_RET mode

  This power mode is same as Functional retention mode defined in DSU TRM (Reference [2]). Cortex-A76 CPU cluster is powered on. All reset signals are not asserted. All RAM modules are in resume standby mode.

- MEM_RET mode

  This power mode is the same as Memory retention mode defined in DSU TRM (Reference [2]), but Cortex-A76 CPU cluster is powered on. All reset signals are asserted. All RAM modules are in resume standby mode. When Cortex-A76 CPU cluster is in this power mode, all Cortex-A76 CPU cores which belong to this cluster must be in OFF mode.

- L3 cache partial powerdown mode

  This power mode is same as L3 cache partial powerdown mode defined in DSU TRM (Reference [2]), but RAM modules are not powered off. RAM modules go to resume standby mode.

- Pseudo OFF mode

  Cortex-A76 CPU cluster is Off mode defined in DSU TRM (Reference [2]). However, Cortex-A76 CPU cluster is still powered on. When Cortex-A76 CPU cluster is in this power mode, Cortex-A76 CPU cores which belong to this cluster must be in OFF mode or Pseudo OFF mode. This power mode is used for Runtime Test.

#### 10.3.1.3 Cluster common part Power mode

Each Cluster common part has following power modes.

- ON mode

  Cluster common part is powered on.

- OFF mode

  Cluster common part is powered off. When Cluster common part is in this mode, all Cortex-A76 CPU clusters which belong to this Cluster common part must be in OFF mode.

### 10.3.2 Power control

#### 10.3.2.1 Wake up sequence of AP-System Core

Figure 10.3 shows power on sequence of Cortex-A76 CPU core. After Cortex-A76 CPU core is booted, WUP_REQ bit of PWRCTRLCz is cleared to 0. When the power status needs to be confirmed, the status can be confirmed by WUP_REQ bit of PWRCTRLCz.

Figure 10.4 shows power on sequence of Cortex-A76 CPU core by wakeup request from INTC-AP. After Cortex-A76 CPU core is booted, H'0000 000D is set to FSMSTSRCz. When the power status needs to be confirmed, the status can be confirmed by FSMSTSRCz. If unexpected values are read, it means the power mode transition sequence is not completed. Until the expected value is read, polling operation is necessary.

The additional procedure written in the Usage Notes shall be applied. Refer to 10.4.3.

While C4 power is off, AP-System Core cannot wake up. Please turn on C4 power before wakeup AP-System Core. For more detail, please refer to 11. System Controller.

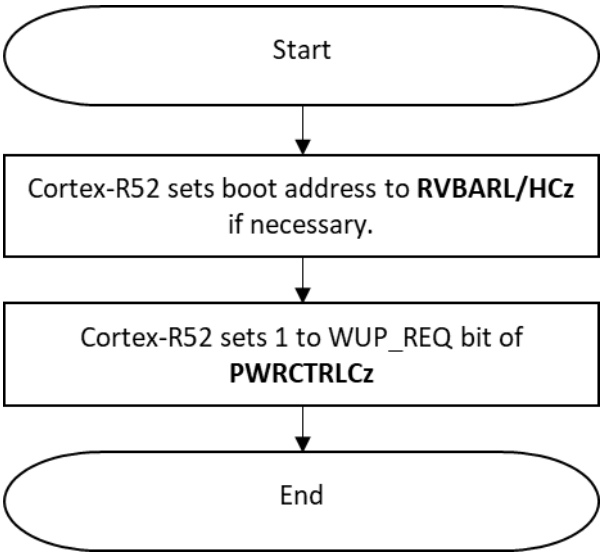



Figure 10.3 Power on sequence of Cortex-A76 CPU core

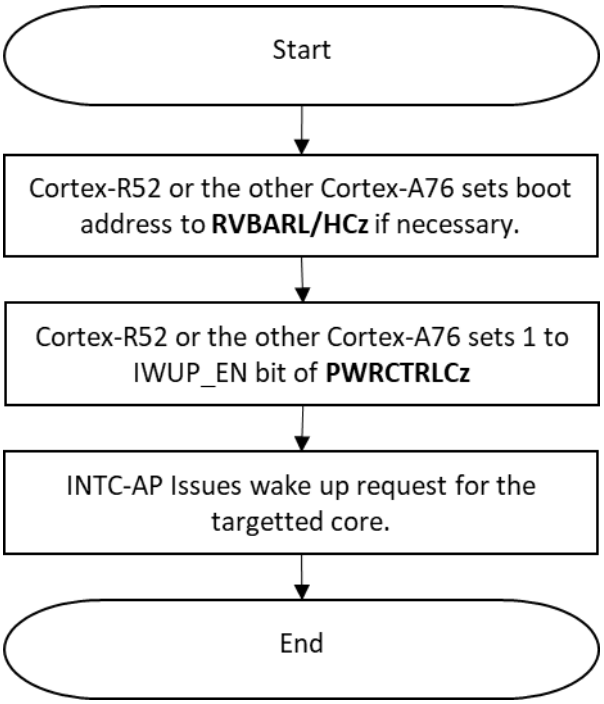



Figure 10.4 Power on sequence of Cortex-A76 CPU core by wakeup request from INTC-AP

#### 10.3.2.2 Power off sequence of AP-System Core

Figure 10.5 shows Cortex-A76 CPU core power off sequence. After Cortex-A76 CPU core is powered off, H'0000 0000 is set to FSMSTSRCz. When the power status needs to be confirmed, the status can be confirmed by FSMSTSRCz. Refer to Reference [1] for the registers written in Bold Italics and details about Power off of Cortex-A76 CPU core.

When Cortex-A76 CPU Core is in debug mode, setting 0 to DBGEN_PPDN bit of SAFECTRLCz is necessary to power off Cortex-A76 CPU core before starting Figure 10.5 sequence. When Cortex-A76 CPU core is in debug mode and DBGEN_PPDN bit of SAFECTRLCz is 1, even if Figure 10.5 sequence is done, Cortex-A76 CPU core is not powered off. Cortex-A76 CPU core goes to OFF_EMU mode. After that, H'0000 001C is set to FSMSTSRCz. When the power status needs to be confirmed, the status can be confirmed by FSMSTSRCz. In Cortex-A76 TRM (Reference [1]), there is a power mode transition from OFF_EMU mode to OFF mode. However, APMU does not support that po wer mode transition.

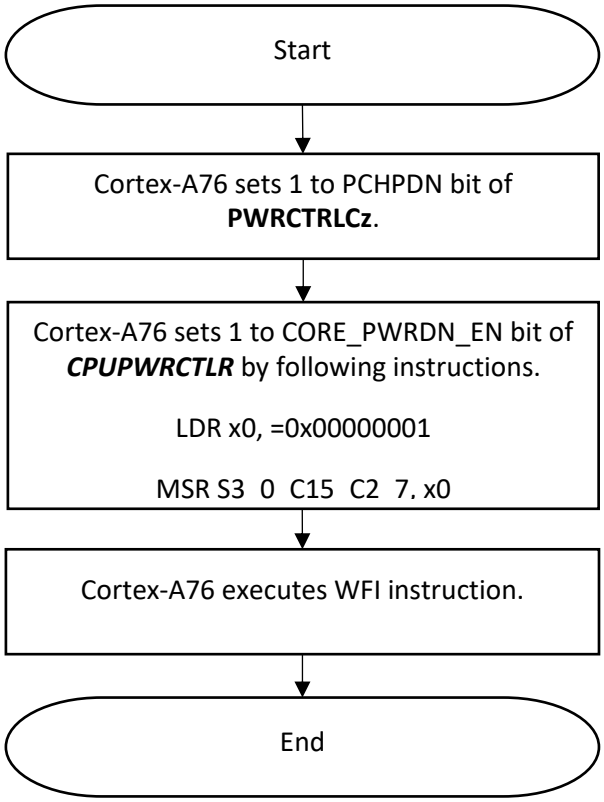



**Figure 10.5 Power off sequence of Cortex-A76 CPU core**

Figure 10.6 shows power off sequence of Cortex-A76 CPU core and cluster. After Cortex-A76 CPU core is powered off, H'0000 0000 is set to FSMSTSRCz. After Cortex-A76 CPU cluster is powered off, H'0000 0000 is set to FSMSTSRCLx. When the power status needs to be confirmed, the status can be confirmed by FSMSTSRCz and FSMSTSRCLx. In addition, if all cores and clusters which belongs to one Cluster common part power domain are powered off by this operation, H'0000 0009 is set to A3FSMSTSRi. Refer to Reference [1] and Reference [2] for the registers written in Bold Italics and details about Power off of Cortex-A76 CPU core and cluster.

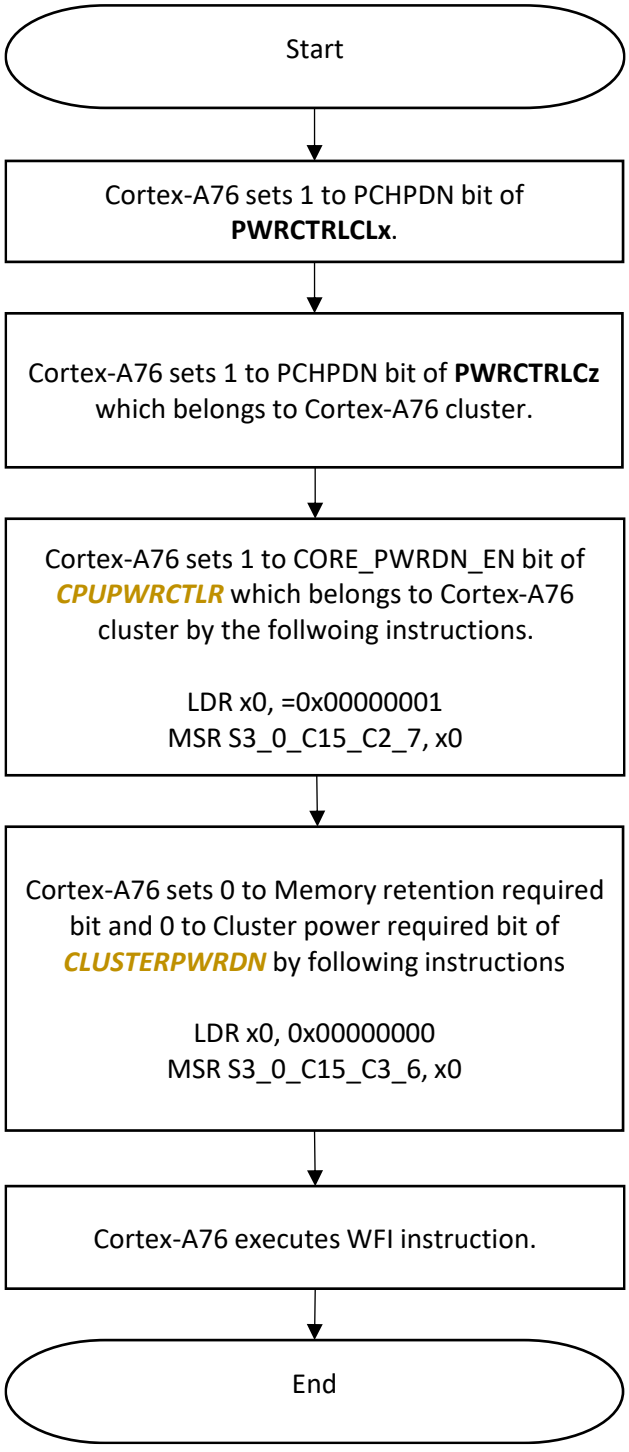



**Figure 10.6 Power off sequence of Cortex-A76 CPU core and cluster**

For the above each case, if unexpected values are read from FSMSTSRCz, FSMSTSRCLx or A3FSMSTSRi, it means the power mode transition sequence is not completed. Until the expected value is read, polling operation is necessary.

The additional procedure written in the Usage Notes shall be applied. Refer to 10.4.3.

#### 10.3.2.3 Power mode transition sequence to go to FUNC_RET/MEM_RET mode

Figure 10.7 shows power mode transition sequence to go to FUNC_RET mode. About enabling Generic Counter, refer to Section 5. After power mode transition is completed, H'0000 0021 is set to FSMSTSRCLx. When the power status needs to be confirmed, the status can be confirmed by FSMSTSRCLx. Refer to Reference [2] for the registers written in Bold Italics and details about FUNC_RET.

Figure 10.8 shows power mode transition sequence to go to MEM_RET mode. After power mode transition is completed, H'0000 0031 is set to FSMSTSRCLx. When the power status needs to be confirmed, the status can be confirmed by FSMSTSRCLx. Refer to Reference [2] for the registers written in Bold Italics and details about MEM_RET.

For the above each case, if unexpected values are read from FSMSTSRCLx, it means the power mode transition sequence is not completed. Until the expected value is read, polling operation is necessary.

The IWUP_EN bit of PWRCTRLCz needs to be set to one when Cortex-A76 CPU core is powered up again by wakeup request from INTC-AP. The PCHWUPEN bit of PWRCTRLCLx needs to be set to one when Cortex-A76 CPU core is powered up again by wakeup request from Cortex-A76 CPU cluster. The PCHWUPEN bit of PWRCTRLCz needs to be set to one when Cortex-A76 CPU core is powered up again by wakeup request from Cortex-A76 CPU core.

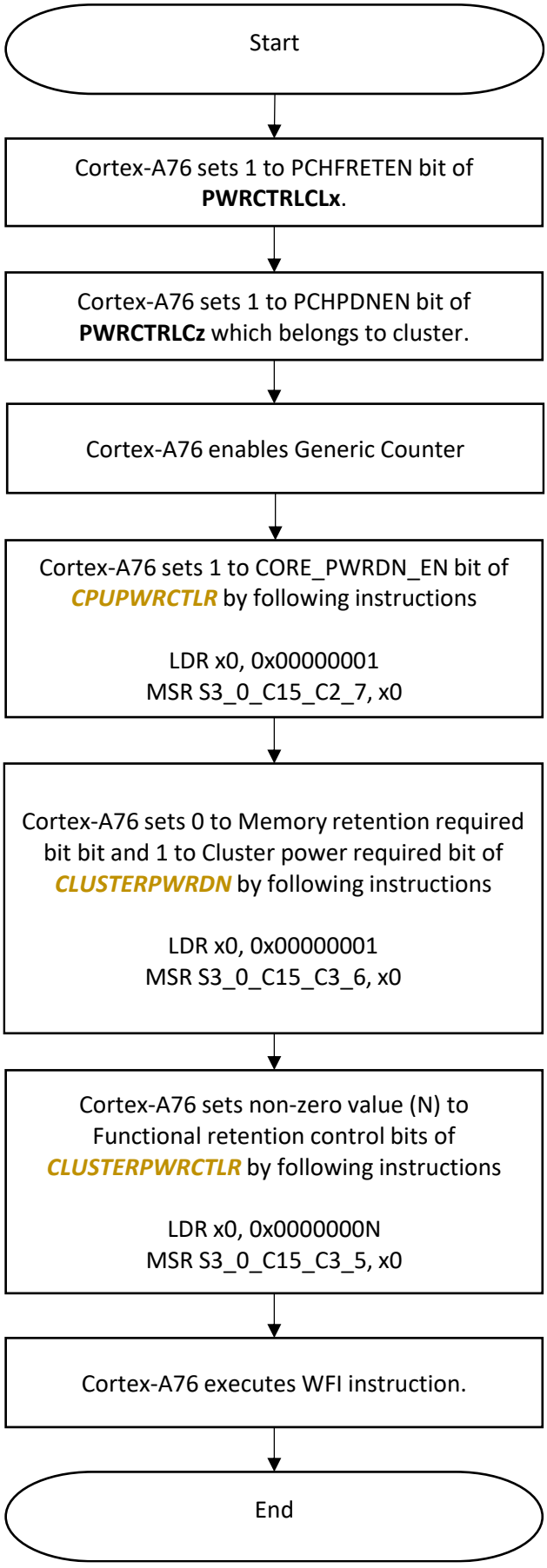



**Figure 10.7 Power mode transition sequence to FUNC_RET mode**

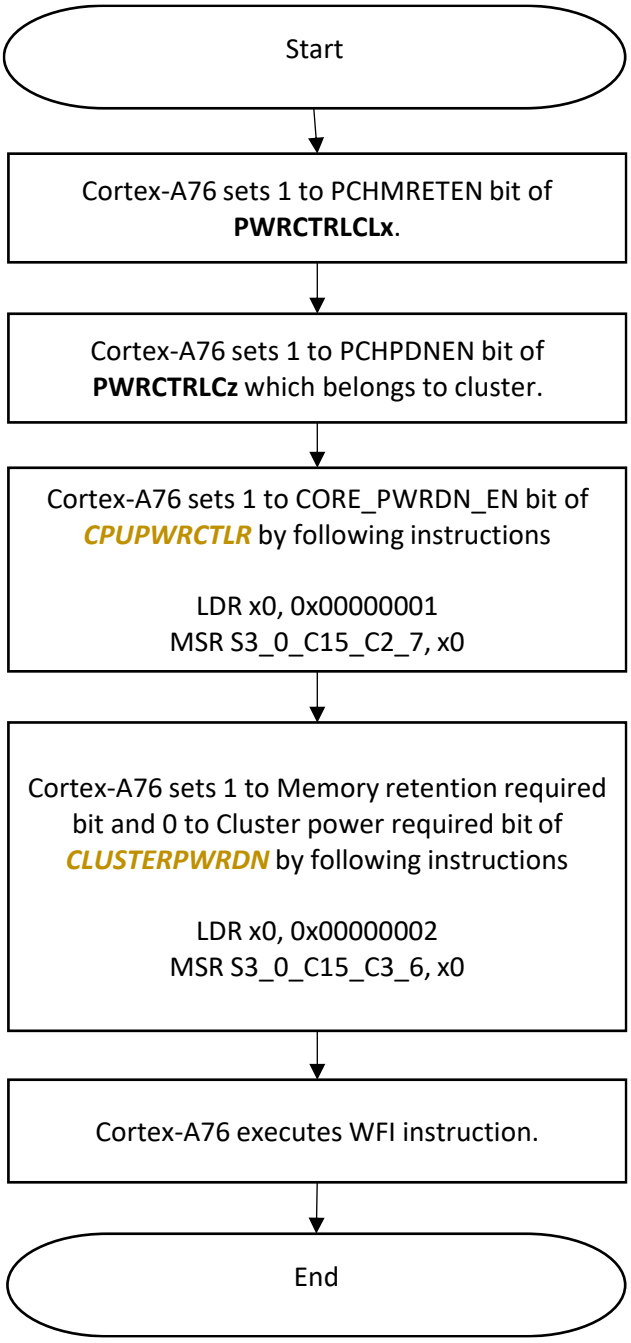



**Figure 10.8 Power mode transition sequence to MEM_RET mode**

#### 10.3.2.4 L3 cache partial power off sequence

L3 cache partial power off mode is supported. The operating cache capacity can be selected from: all, 3/4, 1/2, 1/4, and none. However, each cache capacity size can be selected only when current cache capacity size is specific size. For example, cache capacity none can be selected only when current cache size is 1/4. Figure 10.9 shows L3 cache partial power control sequence. Refer to Reference [2] for the registers written in Bold Italics and details about L3 cache partial power off.

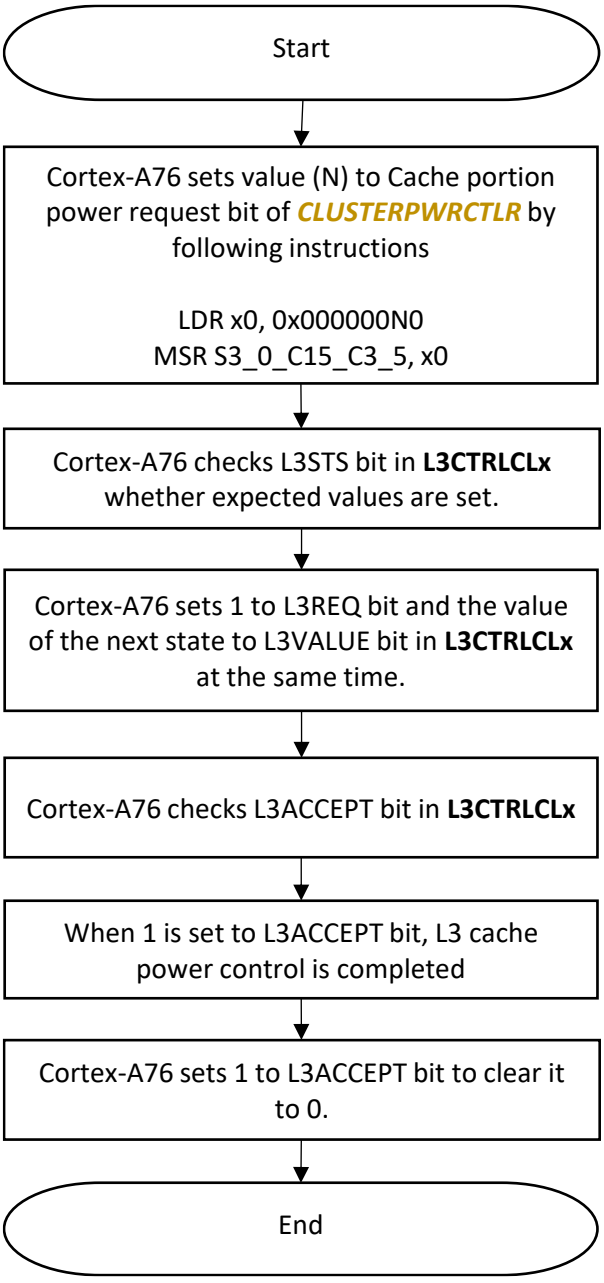



Figure 10.9 L3 cache partial power control sequence

#### 10.3.2.5 Power operation for Cortex-R52

Cortex-R52 supports the following five power modes.

- Core Standby mode (A3CR* domain power is off)
- Normal Operation mode

- Sleep mode (clock stop)
- Core powerdown mode (A3CR* domain power is off)
- Debug over Core powerdown mode

Core Standby mode is a power mode before Cortex-R52 CPU core is booted. Cortex-R52 CPU core 0 is in Core Standby mode only in ICUMX boot. Other Cortex-R52 CPU cores are in Core Standby mode before they are woken up. Each Cortex-R52 CPU core is under reset and there is no operation in Core Standby mode. The Cortex-R52 CPU core leaves Core Standby mode when Cortex-R52 CPU core is woken up by ICUMX, AP-System Core or other Cortex-R52 CPU cores. The Cortex-R52 CPU core is woken up when 0 is set to CR52RST bit in CR52RSTCTRLx. After leaving Core Standby mode, it is impossible to go back to this mode. The boot address can be set by using CR52BARPr or CR52BARr before each core is woken up as needed.

Cortex-R52 CPU cores enters Sleep mode when Cortex-R52 CPU core executes WFI instruction. Clock to Cortex-R52 CPU core is stopped to save the power consumption until Cortex-R52 CPU core receives a wake-up request.

In Core powerdown mode, CPU core power domain (A3RC* power domain) is powered-off. Core powerdown mode has less power than Sleep mode. Debug over Core powerdown mode is for debug Core powerdown mode.

**(1) Core powerdown mode flow**

Figure 10.10 shows core powerdown flow. If core denies the powerdown, PDENY interrupt is occur and core does not transition powerdown. Refer to Reference [4] for details about core powerdown.

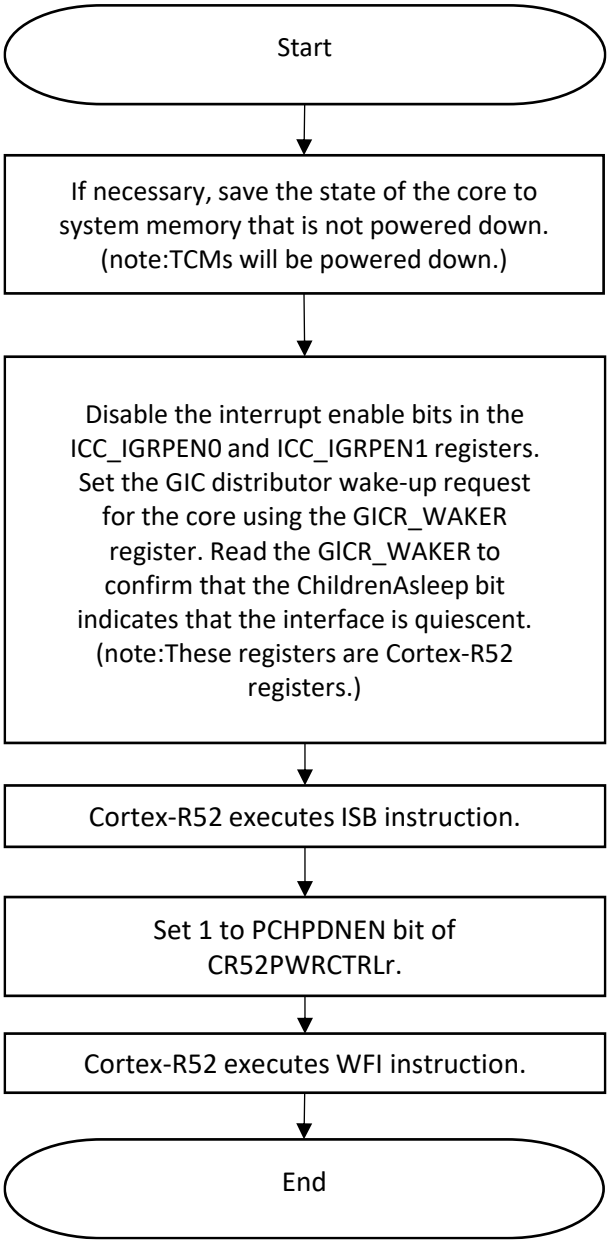



**Figure 10.10 Power mode transition sequence to Core powerdown mode**

### (2) Core powerup mode flow

If PCHWUPEN bit of CR52PWRCTRLr is 1 and wake-up request for the core using the GICR_WAKER comes, Cortex-R52 core will be powered up.

Or, if WUP_REQ bit of CR52PWRCTRLr is set 1, Cortex-R52 core will be powered up.

### (3) Debug over Core powerdown mode flow

When Cortex-R52 is debug mode, even if you execute power down sequence, the core power does not turn off. Some debug information in core power domain is kept.

### 10.3.3 Boot address setting

V4M contains three Cortex-R52 CPU cores, core 0, 1, and 2 (See 6.1.1). Boot address of Cortex-R52 CPU core0 is defined by Mode PIN setting and it is described in CR52BAR0 and CR52BARP0. Table 10.3 shows Boot address and initial value of CR52BAR0 and CR52BARP0 bit field. In addition, for the Secure Boot, the special initial values are set to CR52BAR0 and CR52BARP0, regardless of the Mode PIN setting. It is also described in Table 10.3. About the detail of CR52BAR0 and CR52BARP0, please refer to section 10.2.

Table 10.3 Initial values of BAREN, BTMD[1:0] and Boot address

| **Mode pin setting** | | | | **Initial value** | | |
|---|---|---|---|---|---|---|
| **MD4** | **MD3** | **MD2** | **MD1** | **BAREN** | **BTMD[1:0]** | **Boot address** |
| 0 | 0 | 0 | 0 | 0 | 00 | H’00 0000 0000 |
| 0 | 0 | 0 | 1 | 1 | 01 | H’00 0800 0000 |
| 0 | 0 | 1 | 0 | 1 | 10 | H’00 EB10 0000 |
| 0 | 0 | 1 | 1 | 1 | 10 | H’00 EB10 0000 |
| 0 | 1 | 0 | 0 | 1 | 10 | H’00 EB10 0000 |
| 0 | 1 | 0 | 1 | 1 | 10 | H’00 EB10 0000 |
| 0 | 1 | 1 | 0 | 1 | 10 | H’00 EB10 0000 |
| 0 | 1 | 1 | 1 | 1 | 10 | H’00 EB10 0000 |
| 1 | 0 | 0 | 0 | 1 | 10 | H’00 EB10 0000 |
| 1 | 0 | 0 | 1 | 1 | 10 | H’00 EB10 0000 |
| 1 | 0 | 1 | 0 | 1 | 01 | H’00 0800 0000 |
| 1 | 0 | 1 | 1 | 1 | 01 | H’00 0800 0000 |
| 1 | 1 | 0 | 0 | 1 | 10 | H’00 EB10 0000 |
| 1 | 1 | 0 | 1 | 1 | 10 | H’00 EB10 0000 |
| 1 | 1 | 1 | 0 | 1 | 10 | H’00 EB10 0000 |
| 1 | 1 | 1 | 1 | 1 | 10 | H’00 EB10 0000 |
| **Secure Boot** | | | | | | |
| - | - | - | - | 1 | 10 | H’00 EB10 0000 |

Boot address of Cortex-A76 CPU core is defined by Mode PIN setting and it is described in RVBARLC0, RVBARHC0, RVBARPLC0 and RVBARPHC0. Table 10.4 shows Boot address of Cortex-A76 CPU core. In addition, for the Secure Boot, the special initial values are set to RVBARLC0 and RVBARLPLC0, regardless of the Mode PIN setting. It is also described in Table 10.4. About the detail of RVBARLC0 and RVBARPLC0, please refer to section 10.2.

Table 10.4 Boot Address of Cortex-A76 CPU core for Each Mode Pin Setting

| **Mode pin setting** | | | | |
|---|---|---|---|---|
| **MD4** | **MD3** | **MD2** | **MD1** | **Boot address** |
| 0 | 0 | 0 | 0 | H’00 0000 0000 |
| 0 | 0 | 0 | 1 | H’00 0800 0000 |
| 0 | 0 | 1 | 0 | H’00 EB10 0000 |
| 0 | 0 | 1 | 1 | H’00 EB10 0000 |
| 0 | 1 | 0 | 0 | H’00 EB10 0000 |
| 0 | 1 | 0 | 1 | H’00 EB10 0000 |
| 0 | 1 | 1 | 0 | H’00 EB10 0000 |
| 0 | 1 | 1 | 1 | H’00 EB10 0000 |
| 1 | 0 | 0 | 0 | H’00 EB10 0000 |
| 1 | 0 | 0 | 1 | H’00 EB10 0000 |
| 1 | 0 | 1 | 0 | H’00 0800 0000 |
| 1 | 0 | 1 | 1 | H’00 0800 0000 |
| 1 | 1 | 0 | 0 | H’00 EB10 0000 |
| 1 | 1 | 0 | 1 | H’00 EB10 0000 |
| 1 | 1 | 1 | 0 | H’00 EB10 0000 |
| 1 | 1 | 1 | 1 | H’00 EB10 0000 |
| **Secure Boot** | | | | |
| - | - | - | - | H’00 EB10 0000 |

### 10.3.4 Interrupt function

APMUhas interrupt request signals. It is asserted when a denied state transition has occurred, which is controlled by PDENYINTRCLx and PDENYINTRCz. A denied state transition is detailed in the Reference [3].

To issue interrupt request from APMU, it needs to set 1 to any bits of PDENYINTRCLx or PDENYINTRCz. When a denied state transition has occurred, the corresponding bit of PDENYSTSRCLx or PDENYSTSRCz by APMU is set to 1. At that time, if the corresponding bit of PDENYINTRCLx or PDENYINTRCz is 1, the corresponding bit of INTSTSR is set to 1 and interrupt request signal by APMU is issued.

After interrupt request is issued from APMU, which FSM issuing interrupt request is identified by reading INTSTSR. After interrupt operations are finished, each bit of PDENYSTSRCLx or PDENYSTSRCz must be cleared to 0. By this operation, a corresponding bit of INTSTSR is cleared by APMU. And interrupt request signal is also negated.

### 10.3.5 Reset operation

#### 10.3.5.1 Forced reset procedure

Forced reset asserting to Cortex-A76 CPU core and cluster is done by FRSTR register. If forced reset operation is done, the data inside Cortex-A76 CPU core and cluster is disappeared. So, this function is used for recovery when any fault is found in Cortex-A76 CPU core or cluster. In addition to, FSMLOCKR for each core and cluster is reset. And any value is set to each FSMSTSR for each core and cluster.

When forced reset of Cortex-A76 CPU core is asserted, it is necessary to assert reset signal of Cortex-A76 CPU cluster including target cores at the same time. In other words, forced reset can be done only for each cluster. Forced reset for only core cannot be done. For example, when forced reset asserting for Cortex-A76 CPU core 0 is done, setting 1 to FRSTRC0, FRSTRC1, FRSTRC2, FRSTRC3 and FRSTRCL0 bit in FRSTR is necessary.

#### 10.3.5.2 Reset operation for Cortex-R52

When Cortex-R52 set 1 to RR bit of Hypervisor Reset Management Register, and Cortex-R52 executes WFI instruction, the warm reset of Cortex-R52 is asserted. After that, the warm reset is negated automatically, and Cortex-R52 starts the boot sequence. About the detail of Hypervisor Reset Management Register and warm reset, please refer to Arm Cortex-R52 Processor Technical Reference Manual.

Please refer to “10.3.2.5 Power operation for Cortex-R52” for reset and power operation.

### 10.3.6 Access protection

#### 10.3.6.1 Bus access protection specification (common specification)

APB Bus is available to control Domain access by each 4KB (0x1000) register address area. Figure 10.11 shows an example. The basic idea and detail of this function is written in Section 18.

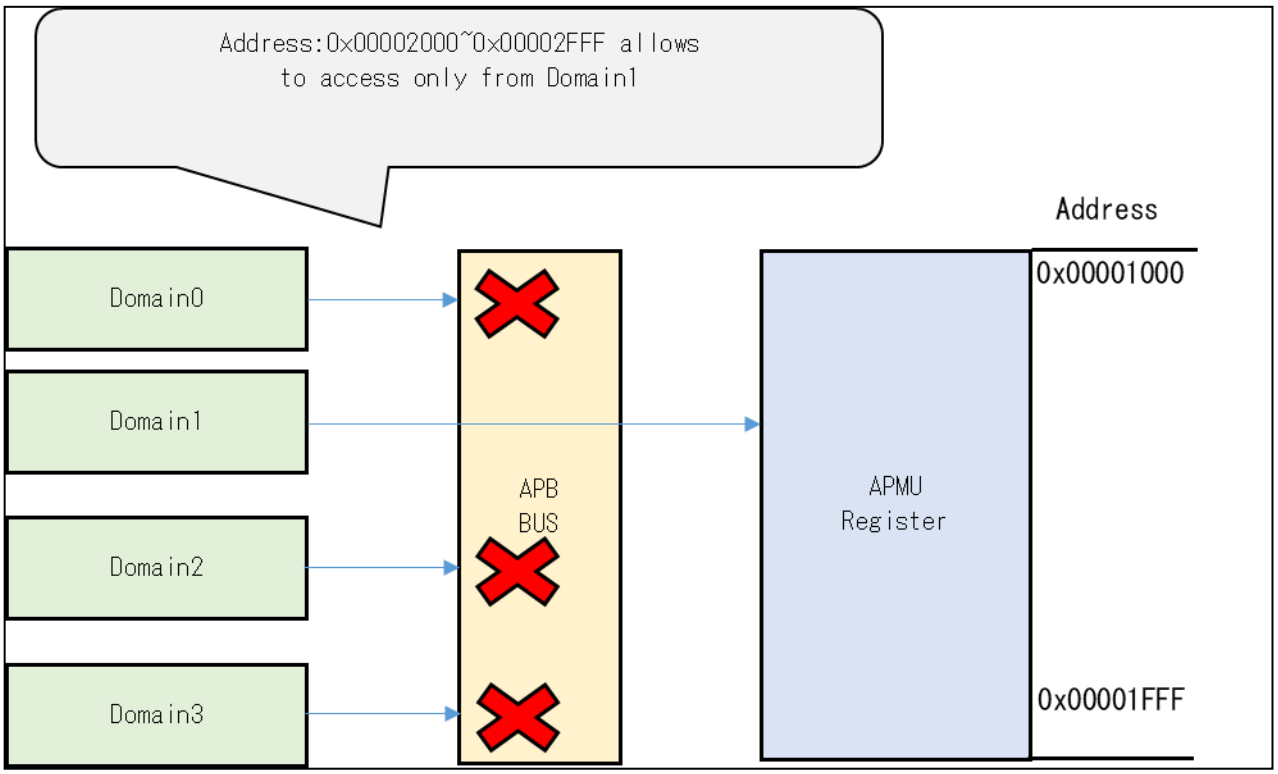



Figure 10.11 Bus access protection

#### 10.3.6.2 APMU access protection specification

**(1) Address and register entity (common specification)**

Each domain has an address area exclusively, but Register entity is only one.

- Domain

  All registers in APMU are accessed through the Domain. There are 4 Domains for APMU.

- Access Group

  Each register in APMU belongs to specific access group. User can assign access groups to each Domain by configuration registers. User can assign 2 or more access groups to 1 Domain. User can assign 1 access group to 2 or more Domains.

Multiple registers belonging to the different group can have different access privileges. However, max is 4 types of access privileges for APMU registers because there are only 4 Domains for APMU. Figure 10.12 shows an example.

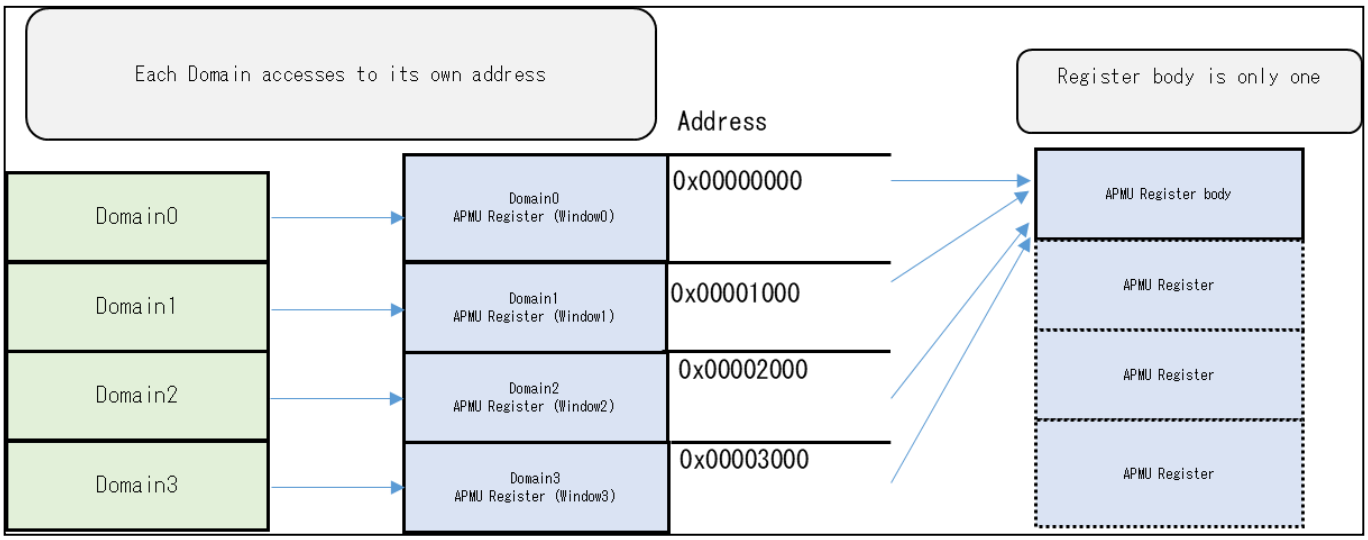



**Figure 10.12 APMU register structure**

### (2) Domain check logic (common specification)

Domain check logic checks from which Domain access and restrict the other Domain to access registers.

APMU has registers, “Domain n Access Enable Register (n = 0-3)”, and it enables Domain control. The detail is written in the following, please refer to 10.3.6.2(3).

[Initial value]

Initial value of Domain 0 access enable register is all 1 (Access enable). Initial value of Domain n access enable register (n = 1-3) is all 0 (Access disable). Domain 0 only can Read/Write enable. Domain 1-3 can Read only (Write disable).

Figure 10.13 shows Domain check logic’s structure.

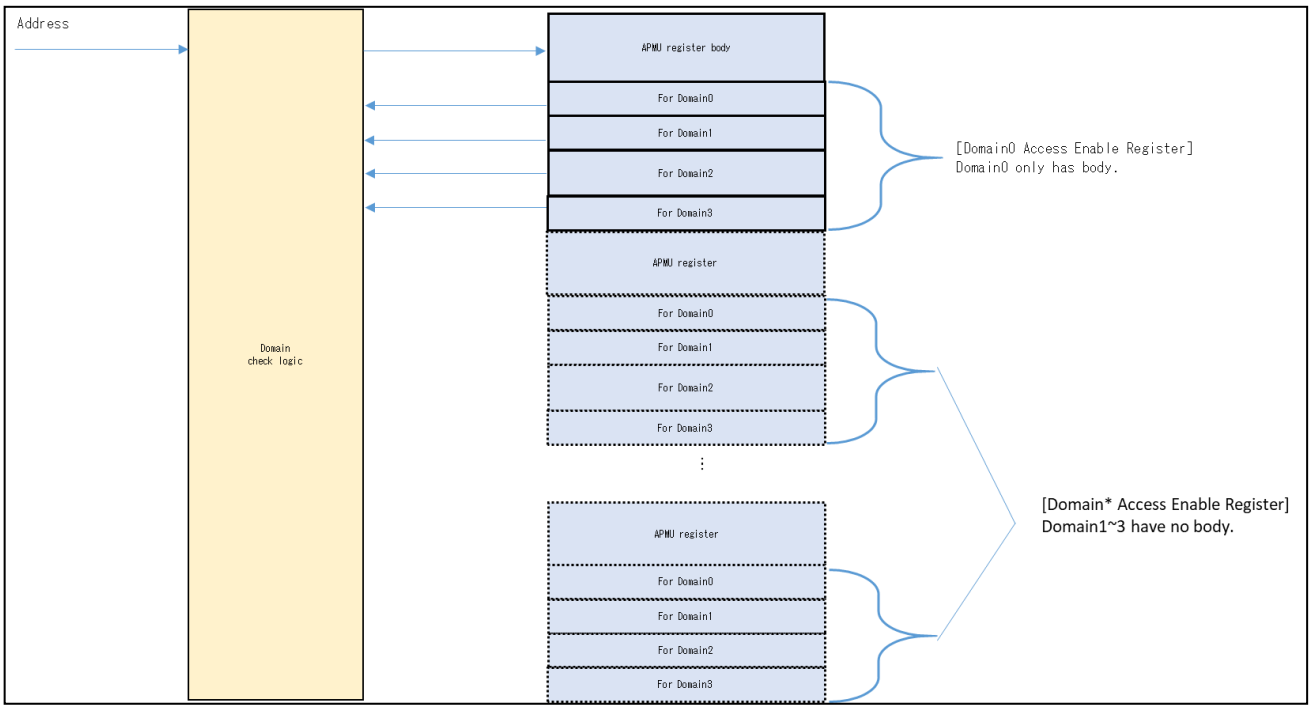



**Figure 10.13 Domain check logic**

### (3) Domain n Access Enable Register (n = 0-3) (APMU specific)

APMU has Registers “Domain n Access Enable Register (n = 0-3)”. They control access privilege of each Domain. Detail of Domain n Access Enable Register and applicable register is described in section 10.2.

Applicable registers of Domain protection are:

- admin registers

  APMU’s general control registers;

- Cortex-R52 registers

  Cortex-R52 reset control registers and CR52BAR;

- Cortex-A76 CPU cluster x registers (x = 0)

  Cortex-A76 CPU cluster x power and reset control registers (x = 0) and RVBAR;

Each register is categorized in the following four groups. The summary of all registers and their groups is described in Table 10.5, Table 10.6, Table 10.7 and Table 10.8.

- Group0

  They are expected to be used for security operation. For example, Reset Vector Base Address Register for secure mode.

- Group1

  They are expected to be used for safety operations. For example, Runtime Test and DCLS configuration of Cortex-A76 CPU clusters.

- Group2

  They are expected to be used for normal power and reset control. For example, power ON/OFF operation.

- Group3

  Reserved. Not used

For example, in the case of the following, only Domain 0 can access Group0 of Cortex-A76 CPU cluster 0's registers. The other Domains cannot access Group0 of Cortex-A76 CPU cluster 0's registers.

D0ACCENR.CL0GRP0 = 1 (Access Enable)

D1ACCENR.CL0GRP0 = 0 (Access Disable)

D2ACCENR.CL0GRP0 = 0 (Access Disable)

D3ACCENR.CL0GRP0 = 0 (Access Disable)

The Domain x Access Enable Register can be written by only Domain 0, regardless of the setting value. Domain 1, 2 and 3 cannot be written. And reading registers is possible by Domain 0, 1, 2 and 3.

Table 10.5 Group for Admin registers

| Register Name | Abbreviation | Group |
|---|---|---|
| Write Protection Control Register | WPCR | - |
| Write Protection Register | WPR | - |
| Domain 0 Access Enable Register | D0ACCENR | - |
| Domain 1 Access Enable Register | D1ACCENR | - |
| Domain 2 Access Enable Register | D2ACCENR | - |
| Domain 3 Access Enable Register | D3ACCENR | - |
| Protect Control/Status Register | PTCSR | Group1 |
| Protect Error Address Register | PTERADR | Group1 |
| DCLS Error Injection Register | DCLSEIJTR | Group1 |
| A3 Power Control Register 0 | A3PWRCTRL0 | Group2 |
| A3 FSM status Register 0 | A3FSMSTSR0 | Group2 |
| A3 FSM Lock Register 0 | A3FSMLOCKR0 | Group2 |
| Interrupt Status Register | INTSTSR | Group2 |
| Error Status Register | ERRSTSR | Group2 |
| Forced Reset Request Register | FRSTR | Group1 |
| Forced Reset Control Register | FRSTCTRL | Group1 |
| PADDR Check Status Register | PADDCHKSTSR | Group1 |
| PWDATA Check Status Register | PWDATACHKSTSR | Group1 |
| APRTMGINT Mask Register | APRTMGINTMASK | Group1 |

Table 10.6 Group for Cortex-R52 registers

| Register Name | Abbreviation | Group |
|---|---|---|
| Cortex-R52 Control Register for Core r | CR52CRr | Group0 |
| Cortex-R52 Reset Control Register for Core r | CR52RSTCTRLr | Group2 |
| Cortex-R52 Power Control Register r | CR52PWRCTRLr | Group1 |
| FSM Lock Register for Cortex-R52 for Core r | FSMLOCKRCR52r | Group2 |
| PDENY Status Register for for Cortex-R52 r | PDENYSTSRCLCR52r | Group1 |
| PDENY Interrupt Register for for Cortex-R52 r | PDENYINTRCLCR52r | Group1 |
| FSM Status Register for Cortex-R52 for Core r | FSMSTSRCR52r | Group2 |
| Group2 General Purpose Register for Cortex-R52 for Core r | G2GPRCR52r | Group2 |
| Cortex-R52 Comparator Enable Register for Core r | CR52CMPENr | Group1 |
| GCNT Error Enable Register for Cortex-R52 for Core r | GCNTERRENCR52r | Group1 |
| Cortex-R52 Reset Vector Base Address Register for Core r | CR52RVBARr | Group2 |
| Cortex-R52 Boot Address Register for Core r | CR52BARr | Group2 |
| Cortex-R52 Reset Vector Base Address Register Protected for Core r | CR52RVBARPr | Group0 |
| Cortex-R52 Boot Address Register Protected for Core r | CR52BARPr | Group0 |

r = 0 - 2

Table 10.7 Group for Cortex-A76 CPU cluster x registers

| Register Name | Abbreviation | Group |
|---|---|---|
| Power Control Register for Cluster x | PWRCTRLCLx | Group2 |
| L3 cache Control Register for Cluster x | L3CTRLCLx | Group2 |
| FSM Lock Register for Cluster x | FSMLOCKRCLx | Group1 |
| PDENY Status Register for Cluster x | PDENYSTSRCLx | Group2 |
| PDENY Interrupt Register for Cluster x | PDENYINTRCLx | Group2 |
| FSM Status Register for Cluster x | FSMSTSRCLx | Group2 |
| Group2 General Purpose Register for Cluster x | G2GPRCLx | Group2 |
| Safety Control Register for Cluster x | SAFECTRLCLx | Group1 |
| GCNT Error Enable Register for Cluster x | GCNTERRENCLx | Group1 |

x = 0

Table 10.8 Group for Cortex-A76 CPU core z registers

| Register Name | Abbreviation | Group |
|---|---|---|
| Power Control Register for Core z | PWRCTRLCz | Group2 |
| FSM Lock Register for Core z | FSMLOCKRCz | Group1 |
| PDENY Status Register for Core z | PDENYSTSRCz | Group2 |
| PDENY Interrupt Register for Core z | PDENYINTRCz | Group2 |
| FSM Status Register for Core z | FSMSTSRCz | Group2 |
| Group2 General Purpose Register for Core z | G2GPRCz | Group2 |
| Safety Control Register for Core z | SAFECTRLCz | Group1 |
| Reset Vector Base Address Register Low for Core z | RVBARLCz | Group1 |
| Reset Vector Base Address Register High for Core z | RVBARHCz | Group1 |
| Reset Vector Base Address Register Protected Low for Core z | RVBARPLCz | Group0 |
| Reset Vector Base Address Register Protected High for Core z | RVBARPHCz | Group0 |

z = 0 - 3

**(4) Write protect logic (APMU specific)**

All registers in APMU except for WPCR and WPR can be protected from unexpected write operation by setting one to WPE bit of WPCR register. When one is set to WPE bit of WPCR register, writing inverted values to WPR is necessary before write access.

The sample procedure is shown in following. In this case, user wants to write H'AAAA 5555 to the target register.

1. Write H'5555 AAAA (= inverted value of H'AAAA 5555) to WPR
2. Write H'AAAA 5555 to the target register

   If the first step is skipped, H'AAAA 5555 is not written to the target register.

## 10.4 Usage Notes

### 10.4.1 Reset operation when AP-System core is in DEBUG mode

When AP-System Core is in DEBUG mode and forced reset function needs to be used, setting zero to FRSTCTRL[30] is necessary.

The above operations must be done during Boot sequence.

### 10.4.2 Enable Access protect function

When access protect function is enabled, setting one to RTGRP3 and CL0GRP3 of each D[n]ACCENR (n=0-3) is necessary.

Setting Group3 affects nothing because there are no registers belonging to Group3.

### 10.4.3 Additional procedure for power on/off control of AP-System Core

When you power-on/off the AP-System Core, you must apply an additional procedure described in this section: either Procedure-1 or Procedure-2.

Refer to 10.3.2.1 for the operation of power-on sequence and 10.3.2.2 for the operation of power-off sequence.

**(1) Procedure-1**

Setting 0 to FRSTCTRL[29] is necessary. This operation must be done before the first Cortex-A76 CPU core is booted.

**(2) Procedure-2**

Do not power-off Cortex-A76 CPU core alone. Cortex-A76 CPU cluster [x] and all CPU cores which belongs to the cluster [x] are necessary to powered-off at a time in the following manner. Where, [x] is cluster number 0.

Power-off sequence

When you power-off Cortex-A76 CPU cluster [x], the following procedure is necessary in the power-off sequence.

1. Clean L1 caches of all CPU cores in the cluster [x] and L3 cache of the cluster [x]. (*1)
2. Power-off threecores in the Cortex-A76 CPU cluster [x]. (*2)
3. Power-off the last core in the Cortex-A76 CPU cluster [x] and the cluster [x] itself.

Notes

*1: Refer to the following document for the example of cache clean.
Example 11-3, section 11.5, ARM Cortex-A Series Programmer's Guide for ARMv8-A version 1.0
(https://developer.arm.com/documentation/den0024/a)

*2: If any of cores in the cluster [x] will never be used, you do not need to power-on that core from the beginning. If you need to use the unpowered core later, power-off the already booted core and its cluster, then power-on all necessary cores in the cluster [x].

## 10.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 10.5.1 Block diagram for functional safety

APMU is implemented as DCLS. Figure 10.14 shows APMU DCLS structure. All sub-modules in APMU are duplicated. One is used as master core and the other is used as checker core. Each core executes same operation and execution result from each core is always compared. If the compare logic detects difference, it judges APMU has failure and it asserts error notification signal. This error notification signal has error injection function. It is controlled by DCLSEIJTR.

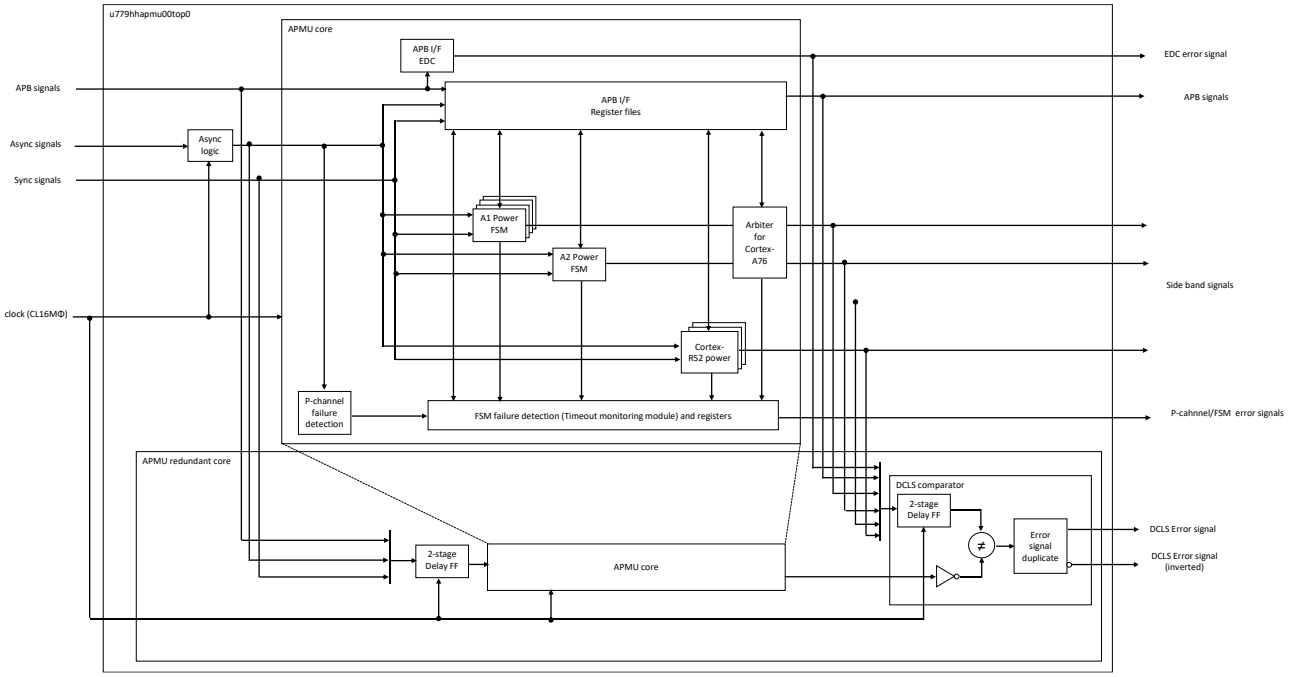



Figure 10.14 APMU DCLS structure

### 10.5.2 Safety Mechanism for Cortex-R52 (DCLS)

#### (1) Overview

The dual core lock step (DCLS) architecture is implemented in Cortex-R52. And its comparator is controlled by CR52CMPENr register.

#### (2) Related Registers

Table 10.9 Register configuration of Safety Mechanism for Cortex-R52 (DCLS)

| Register Name | Abbreviation |
|---|---|
| Cortex-R52 Comparator Enable Register r | CR52CMPEN[r] (r=0-2) |

**(3) Hardware Description**

For more detail about Cortex-R52, please refer to section 6.

### 10.5.3 Safety Mechanism for AP-System Core

**(1) Overview**

AP-System Core does not support DCLS mode.

**(2) Related Registers**

There is no content in this section.

**(3) Hardware Description**

There is no content in this section.

### 10.5.4 Safety Mechanism for P-channel interface

**(1) Overview**

P-channel interface is used to communicate between APMU and Cortex-A76/R52 CPU core and cluster. And Safety Mechanism for this interface is implemented to APMU and Cortex-A76/R52 CPU core and cluster. The block diagram is shown in Figure 10.15.

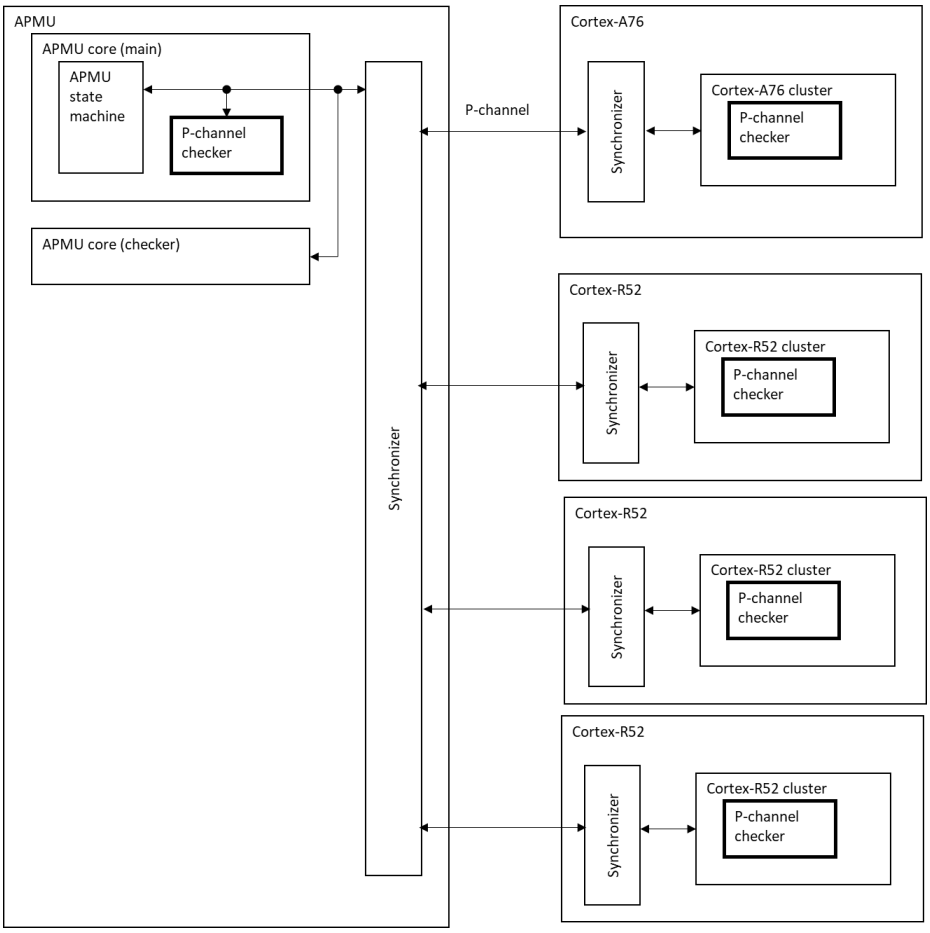



Figure 10.15 Safety Mechanism for P-channel between APMU and Cortex-A76/R52

**(2) Related Registers**

Table 10.10 Related Registers of Safety Mechanism for P-channel Interface

| Register Name | Abbreviation |
|---|---|
| Error Status Register | ERRSTSR |
| A3 FSM Lock Register [i] | A3FSMLOCKR[i] (i=0) |
| FSM Lock Register for Cortex-R52 [r] | FSMLOCKRCR52[r] (r=0-2) |
| FSM Lock Register for Cluster [x] | FSMLOCKRCL[x] (x=0) |
| FSM Lock Register for Core [z] | FSMLOCKRC[z] (z=0-3) |

**(3) Hardware Description**

**a) Safety Mechanism in APMU side**

This Safety Mechanism can detect three types of errors. First is P-channel protocol error, second is P-channel corruption error and third is FSM deadlock error. The error detection flag is stored to FSM Lock Register for each FSM. When error detection flag is set, one is set to each bit of ERRSTSR, and APMU notifies error detecting to ECM.

#### *P-channel protocol error*

P-channel signal from Cortex-A76/R52 CPU core and cluster to APMU is duplicated. And they are compared by this mechanism. This function can be activated after Cortex-A76/R52 CPU core is booted. At that time, before enabling this function, PRTCL_ERR bit must be cleared to 0.

This mechanism is enabled by setting one to PRTCL_EN bit of FSMLOCKRCLx, FSMLOCKRCz or FSMLOCKRCR52r. If this mechanism detects failing, one is set to PRTCL_ERR bit of FSMLOCKRCLx, FSMLOCKRCz or FSMLOCKRCR52r.

#### *P-channel corruption error*

A Toggle order of P-channel signals is defined by Arm. About the detail, refer to the Reference [3]. This mechanism detects failing of this order by monitoring a set of P-channel signal.

This mechanism is enabled by setting one to CRPT_EN bit of FSMLOCKRCLx, FSMLOCKRCz or FSMLOCKRCR52r. If this mechanism detects failing, one is set to CRPT_ERR bit of FSMLOCKRCLx, FSMLOCKRCz or FSMLOCKRCR52r.

#### *FSM deadlock error*

APMU has FSMs for each power domain and Cortex-R52. And each FSM is monitored by a timeout monitoring module. The timeout monitoring module counts transition time from one power mode to other power modes. If the transition time is more than configured time, this mechanism judges FSM is in deadlock.

This mechanism is enabled by setting one to LOCK_EN bit of each FSMLOCKR. Transition time is configured by CNT_VALUE field of FSMLOCKR. A timeout monitoring module counts transition time by 1/16 frequency clock of APMU clock. If this mechanism detects failing, one is set to LOCK_ERR bit of FSMLOCKR.

#### *Error notification signal and handling*

APMU outputs four error notification signals. A corresponding signal is asserted when one is set to any bits of ERRSTSR. The correspondence of error notification signals and bits of ERRSTSR is described in Table 10.11.

If error notification signal is captured by ECM, ECM issues interrupt signal to Cortex-R52. If Cortex-R52 receives interrupt signal to notify APMU failing, Cortex-R52 must read ERRSTSR to identify which FSM detects failure. After that Cortex-R52 reads FSMLOCKR of failing FSM to identify which type of error is detected. After Cortex-R52 finishes error operation, Cortex-R52 clears error flag bit of each FSMLOCKR to zero. By this operation, the corresponding bit of ERRSTSR is cleared by APMU. And error notification signal is also negated by APMU.

Table 10.11 Error notification signal of Safety Mechanism for P-channel interface

| **Error notification signal** | **Bit name of ERRSTSR.** |
|---|---|
| Cortex-R52 CPU core 0 reset control error | CR52_ERR0 |
| Cortex-R52 CPU core 1 reset control error | CR52_ERR1 |
| Cortex-R52 CPU core 2 reset control error | CR52_ERR2 |
| Cortex-A76 CPU cluster 0 power control error | A2C0_ERR, A1C0_ERR, A1C1_ERR, A1C2_ERR, A1C3_ERR |

*Error injection function*

The above three types of errors have error injection function. If one is set to PRTCL_EIJT, CRPT_EIJT or LOCK_EIJT bit by SW, one is set to corresponding error status bits and corresponding error signals are asserted. To negate error notification signal, CPU must clear all error injection bits to zero. After that CPU must clear all error status bits to zero.

**b) Safety Mechanism in Cortex-A76/R52 CPU core and cluster side**

This Safety Mechanism can find two types of errors. First is parity error and second is P-channel signal protocol error. If either one error is detected, Cortex-A76/R52 CPU core notifies error detection to ECM.

*Parity error for P-channel signal*

A parity bit is added to P-channel signal from APMU to Cortex-A76/R52 CPU core and cluster. The parity checking logic is implemented in Cortex-A76/R52 CPU core and cluster. It detects parity error.

*P-channel protocol error*

Toggle ordering of P-channel signals is defined by Arm. About the detail, refer to the Reference [3]. This mechanism detects failing of this ordering.

### 10.5.5 Safety Mechanism for APB bus interface

**(1) Overview**

APMU has EDC checker and generator for APB bus interface.

**(2) Related Registers**

Table 10.12 Related Registers of Safety Mechanism for APB bus Interface

| Register Name | Abbreviation |
|---|---|
| PADDR Check Status Register | PADDCHKSTSR |
| PWDATA Check Status Register | PWDATACHKSTSR |

**(3) Hardware Description**

**For more detail, please refer to Section 15.**

### 10.5.6 Safety Mechanism between Generic Counter and CPUs

**(1) Overview**

Safety mechanism is implemented between Generic Counter and Cortex-A76 CPU core and Cortex-R52 CPU core. It is controlled by GCNTERRENCR52r and GCNTERRENCLx.

**(2) Related Registers**

Table 10.13 Related Registers of Safety Mechanism between Generic Counter and CPUs

| Register Name | Abbreviation |
|---|---|
| GCNT Error Enable Register for Cortex-R52 [r] | GCNTERRENCR52[r] (r=0-2) |
| GCNT Error Enable Register for Cluster [x] | GCNTERRENCL[x] (x=0) |

**(3) Hardware Description**

For more detail about Safety mechanism between Generic Counter and Cortex-R52, please refer to section 6. For more detail about Safety mechanism between Generic Counter and Cortex-A76 CPU core, please refer to section 5.

# 11. System Controller (SYSC)

## 11.1 Overview

### 11.1.1 Features

**Power Control**

SYSC works as a power control for any power domains including.

**AP-system core**

SYSC controls Power-ON and Power-OFF of the Cortex-A76 CPU core at APMU's request. APMU's request can be masked by register settings.

**Realtime core**

SYSC controls Power-ON and Power-OFF of the Realtime core at APMU's request. APMU's request can be masked by register settings.

**GPU**

SYSC controls Power-ON and Power-OFF of the Cortex-A76 CPU core at 3DGE's request. 3DGEs request can be masked by register settings.

**Other**

SYSC controls Power-ON and Power-OFF of other modules by power domain registers.

**Function**

SYSC controls Power-ON and Power-OFF of power domains, and checks the power state status thereof. SYSC has pseudo Power-OFF function.

SYSC and CPG interact since CPG controls clock and reset. When the power state control is completed or violation occurs, the interrupt function for CPUs activates. SYSC controls isolation cells.

**Multi Request Power flow support**

Multiple power control requests able accepted.

This module is sequentially processed by power control requests.

**FuSa**

SYSC has the following functions for FuSa.

- Write access protection
- Isolation Error Detection
- HW Redundancy
- SYSC is POST (Power On Self Test) target

**Policy**

SYSC controls Power Switch and Isolation cells for all power domains.

Basically, we can detect the failure of SYSC function by the following mechanism.

Failure of Power Switch control function.:

If Power Switch is unintentionally Power-OFF, we can detect the failure by using register access, reset feedback etc because the modules in the power domain cannot operate correctly.

Failure of Isolation control function:

If isolation is unintentionally active, we can detect the failure by using register access and reset feedback etc. because the signal is masked by isolation cell.

If isolation cell is failure during Power-OFF, we can detect it by isolation error detection function (SM).

**Functional Safety function**

Create error notification to error correction module when occur error.

**HW redundancy**
Can be error notification when occur violation operation, therefore hardware redundancy of register and FSM in sysc logic. Error correction have 16 violation. And can be mask of error notification.

**SYSC is POST (Power ON Self Test) target**

Power control when period of POST mode. All power ON of power domains before POST mode. Power and isolation control when period of POST mode by SWC.

**Write access protection**
Write access protection function for register. For common registers, write access protection is possible for each register. For power control registers, write access protection is possible for each power domains. If occur write access violation, error notification and store of address.

**Isolation Error detection**
SYSC have error notification of isolation cells.Isolation Function to detect cell failure. It is possible to check in which power domains an error occurred. Error mask function implemented for each error cause.

### 11.1.2 Power Domain Structure

When one of those power domains is Power-OFF state, all power domains which are written left the domain does not change Power-OFF state. For example, when A2E0D0 is Power-OFF state, A1E0D0C0 and A1E0D0C1 power requests are processed after waiting for A2E0D0 Power-ON state.

Table 11.1 Power Domain Structure

| Power Domain | Description | Power Domain | Description | Power Domain | Description |
|---|---|---|---|---|---|
| **A1E0D0C0** | Cortex-A76 CPU core0 *1 | **A2E0D0** | Cortex-A76 CPU cluster *1 | **C4** | |
| **A1E0D0C1** | Cortex-A76 CPU core1 *1 | | | | |
| **A1E0D0C2** | Cortex-A76 CPU core2 *1 | | | | |
| **A1E0D0C3** | Cortex-A76 CPU core3 *1 | | | | |
| | | **A3CR0** | Realtime core0 *1 | – | |
| | | **A3CR1** | Realtime core1 *1 | | |
| | | **A3CR2** | Realtime core2 *1 | | |
| **A23DGB** | 3D Graphics Engine | **A33DGA** | 3D Graphics Engine | **C4** | |
| | | **A3VIP0** | UMF | | |
| | | **A3VIP2** | SMD-PS0, SMD-POST0 | | |
| | | **A3DUL** | PAP System | | |
| | | **A3ISP0** | ISP0 core | | |
| **A1CN0** | CNN | **A2CN0** | High Speed Domain incl. CNN, CNN RAM0(SPMC) and Slim-IMP Dmac0/1 | **C4** | |
| **A1DSP0** | DSP core0 | | | | |
| **A1DSP1** | DSP core1 | | | | |
| | | **A2IMP01** | IMP core 0/1 | | |
| | | **A2PSC** | IMPPSC and IMPDMAC0 | | |
| | | **A2DMA** | IMPDMAC1 | | |
| | | **A2CV0** | CVe cluster 0 | | |
| | | **A2CV1** | Cve cluster 1 | | |
| | | **A2CV2** | Cve cluster 2 | | |
| | | **A2CV3** | Cve cluster 3 | | |
| | | **A3IMR0** | IMR-rs core0 | C4 | |
| | | **A3IMR1** | IMR-rs core1 | | |
| | | **A3IMR2** | IMR-std core0 | | |
| | | **A3IMR3** | IMR-std core1 | | |
| | | **A3VC** | Video Codec | | |
| **A2PCIPHY** | PCI PHY | **A3PCI** | PCI Express | | |

**Table 11.1.2.1 Function located on Always ON power domain**

| Corresponding chapter | Function |
|---|---|
| 7 | Pin Function (PFC/GPIO) |
| 8 | Clock Pulse Generator (CPG) |
| 9 | Module Standby, Software Reset |
| 10 | Advanced Power Management Unit (APMU) |
| 11 | System Controller (SYSC) |
| 13 | Thermal Sensor/Chip Internal Voltage Monitor/Core Voltage Monitor (THS/CIVM/CVM) |
| 14 | Reset (RST) |
| 15 | Interrupt Controller (INTC) |
| 16 | MFIS (Multifunction Interface) |
| 17 | ECM (Error Control Module) |
| 18 | AXI-Bus<br>・ Peripheral domain AXI<br>・ RT (Real Time) domain AXI<br>・ Main Memory domain AXI<br>・ Slave Access Bus<br>・ AXI connected with Main Memory domain AXI, Slave Access Bus |
| 19 | IPMMU which operates with following AXI-Bus<br>・ Peripheral domain AXI<br>・ RT (Real Time) domain AXI<br>・ Main Memory domain AXI<br>・ Slave Access Bus |
| 30 | Direct Memory Access Controller (SYS-DMAC, RT-DMAC) |
| 31 | Boot |
| 33 | External Bus Controller for SDRAM (DBSC5) |
| 34 | System RAM |
| 36 | RT-VRAM |
| 37 | Back-up Buffer |
| 63 | Data Compression Processor (FCPR) |
| 81 | Serial Sound Interface Unit (SSIU) |
| 82 | Serial Sound Interface (SSI) |
| 83 | Audio Clock Generator (ADG) |

| | |
|---|---|
| 102 | CAN-FD |
| 103 | FlexRay |
| 105 | Serial Communication Interface with FIFO (SCIF) |
| 106 | High Speed Serial Communication Interface with FIFO (HSCIF) |
| 107 | I2C Bus Interface (I2C) |
| 109 | Clock-Synchronized Serial Interface with FIFO (MSIOF) |
| 110 | PWM |
| 112 | SPI Multi I/O Bus Controller (RPC-IF) |
| 130 | Secure Engine |
| 132 | AES Accelerator (AES-ACC) |
| 133 | Life Cycle |
| 134 | ICUMX |
| 135 | CRC |
| 136 | Failure Self-DetectionOutoutput(RFSO) |
| 150 | SD Card/MMC Interfaces |
| 154 | Watchdog Timer (RWDT, WWDT, SWDT) |
| 155 | Timer Unit (TPU, CMT0, CMT1/2/3, TMU, SCMT, SUCMT) |
| 169 | HSSTP |
| 170 | Debug and Trace |
| U10.1 | E-FUSE |
| U10.2 | OTP MEM |
| U11 | ROM |
| U61 | DDR PHY |

**Table 11.1.2.2 Function located on C4 power domain**

| Corresponding chapter | Function |
|---|---|
| 18 | AXI-Bus<br>・ CPU domain AXI<br>・ GSX (Graphics) domain AXI<br>・ HSC (High speed Communication) domain AXI<br>・ VC (Video Codec) domain AXI<br>・ VIO (Video IO) domain AXI<br>・ VIP (Vision IP Processor) domain AXI<br>・ IMP domain AXI |
| 19 | IPMMU which operates with following AXI-Bus<br>・ CPU domain AXI<br>・ GSX (Graphics) domain AXI<br>・ HSC (High speed Communication) domain AXI<br>・ VC (Video Codec) domain AXI<br>・ VIO (Video IO) domain AXI<br>・ VIP (Vision IP Processor) domain AXI |

| | |
|---|---|
| | ・ IMP domain AXI |
| 53 | CSI2 |
| 55 | Video Input Module (VIN) |
| 60 | Video Signal Processor (VSP2) |
| 64 | Display Unit (DU) |
| 65 | Display Output Checker (DOC) |
| 67 | DSI/CSI-2 TX |
| 100 | Ethernet (EtherAVB-IF, RGMII) |

**Table 11.1.2.3 Function located on PSO**

| Corresponding chapter | Function |
|---|---|
| 5 | AP-System Core (Cortex-A76 Core0, Core1, Core2, Core3) |
| 6 | Realtime Core (Cortex-R52 Core0, Core1, Core2) |
| 50 | GSX (3D Graphics Processing Unit) |
| 54 | Image Signal Processor (ISP)/Channel Selector (CS) |
| 56 | IMR-LX6 |
| 57 | Video Encoding Processor for Inter-device Video Transfer (iVCP1E) |
| 58 | Image Processing Unit (IMP-X7) |
| 59 | Video Codec Processor |
| 62 | Frame Compression Processor (FCP) |
| 104 | PCIe Controller |
| U20 | Measurement Trace Interface (MTI) |
| U25 | Vision-IP |
| U30 | IMP DMAC |
| U31 | IMP PSC |
| U32 | IMP-X7 CVengine |
| U33 | Convolutional Neural Network (IMP CNN) |
| U34 | IMP-X7_system |
| U36 | Debug Trace Agent (IMP DTA) |
| U37 | IMP Core |
| U39 | VDSP |
| U71 | PAP-SubSystem |

Note
*1: Don’t use SYSC control registers for Cortex-A76 CPU and Realtime power domains. Use APMU to control the power domains.

### 11.1.3 Power Domain Register Number (PDR)

Table 11.2 PDR assignment

| PDR | Power Domain Registers Offset Address | Power Domain [R-Car V4M] |
|---|---|---|
| PDR00 | H'1000 | A1E0D0C0 |
| PDR01 | H'1040 | A1E0D0C1 |
| PDR02 | H'1080 | A1E0D0C2 |
| PDR03 | H'10C0 | A1E0D0C3 |
| PDR16 | H'1400 | A2E0D0 |
| PDR21 | H'1540 | A3CR0 |
| PDR22 | H'1580 | A3CR1 |
| PDR23 | H'15C0 | A3CR2 |
| PDR24 | H'1600 | A33DGA |
| PDR25 | H'1640 | A23DGB |
| PDR31 | H'17C0 | C4 |
| PDR33 | H'1840 | A1DSP0 |
| PDR34 | H'1880 | A2IMP01 |
| PDR35 | H'18C0 | A2PSC |
| PDR36 | H'1900 | A2CV0 |
| PDR37 | H'1940 | A2CV1 |
| PDR38 | H'1980 | A3IMR0 |
| PDR39 | H'19C0 | A3IMR1 |
| PDR40 | H'1A00 | A3VC |
| PDR42 | H'1A80 | A2CN0 |
| PDR44 | H'1B00 | A1CN0 |
| PDR45 | H'1B40 | A1DSP1 |
| PDR47 | H'1BC0 | A2DMA |
| PDR48 | H'1C00 | A2CV2 |
| PDR49 | H'1C40 | A2CV3 |
| PDR50 | H'1C80 | A3IMR2 |
| PDR51 | H'1CC0 | A3IMR3 |
| PDR52 | H'1D00 | A3PCI |
| PDR53 | H'1D40 | A2PCIPHY |
| PDR56 | H'1E00 | A3VIP0 |
| PDR58 | H'1E80 | A3VIP2 |
| PDR60 | H'1F00 | A3ISP0 |
| PDR62 | H'1F80 | A3DUL |

Note: "-" is Reserved.

### 11.1.4 Register Configuration

Table 11.3 lists the SYSC registers.

Base-address of SYSC H'E618 0000

SYSC has Write Access Protection function. The register can only access from permitted protection domains.

There are 4 protection domains (Domain0/1/2/3) by address. Access the same register from each protection domain.

Protection domains select uses 15 and 14 bits. Therefore, for base addresses other than 0, use the following calculation formula.

SYSC register area By address

H'E618 0000 + H’4000 * Protection Domain No.

Do not write to addresses other than those listed below, otherwise normal operation cannot be guaranteed. Values read from other addresses are undefined.

Access size are all 32bits.

[Legend]

Initial value: Register value after a reset

- : Undefined value

R/W: Readable/writable. The written value can be read.

R: Read-only. The write value should always be the initial value.

-/WB: Write-only. The read value is undefined.

R/WC1: Readable/writable. Writing 1 initializes the bit. Writing 0 is ignored.

**Table 11.3 List of Registers**

n: Power Domain Register Number(PDR) , refer to List of PDR and Power domain (Table 11.2).
The register value when the power domain does not exist is unknown.
Example: Offset Address of PDRSR01 is H’1040, PDRSR48 is H’1C00

| Register name | Abbreviation | R/W | Offset Address | Initial Value |
|---|---|---|---|---|
| SYSC Status Register | SYSCSR | R | H'0000 | H'0000 0003 |
| Protect Control/Status Register | SYSCPTCSR | R/W/WC1 | H'0010 | H'0000 0002 |
| Protect Error Address Register | SYSCPTERADR | R/WC1 | H'0014 | H'0000 0000 |
| Extended Function Control Register | SYSCEFCR | R/W | H'0034 | H'0000 0000 |
| Power-ON Status Register0 | SYSCPONSR0 | R | H'0800 | *1 |
| Power-ON Status Register1 | SYSCPONSR1 | R | H'0804 | *1 |
| Power-OFF Status Register0 | SYSCPOFFSR0 | R | H'0808 | *1 |
| Power-OFF Status Register1 | SYSCPOFFSR1 | R | H'080C | *1 |
| Interrupt Status/Clear Register0 | SYSCISCR0 | R/WC1 | H'0810 | H'0000 0000 |
| Interrupt Status/Clear Register1 | SYSCISCR1 | R/WC1 | H'0814 | H'0000 0000 |
| Interrupt Enable Register0 | SYSCIER0 | R/W | H'0820 | H'0000 0000 |

| Register name | Abbreviation | R/W | Offset Address | Initial Value |
|---|---|---|---|---|
| Interrupt Enable Register1 | SYSCIER1 | R/W | H'0824 | H'0000 0000 |
| Interrupt Mask Register0 | SYSCIMR0 | R/W | H'0830 | H'0000 0000 |
| Interrupt Mask Register1 | SYSCIMR1 | R/W | H'0834 | H'0000 0000 |
| Power Domain Status Register n (n=00-63) | PDRSRn | R | H'1000 + n x 64 | *1 |
| Power Domain power-ON Control Register n (n=00-63) | PDRONCRn | W | H'1004 + n x 64 | H'0000 0000 |
| Power Domain power-OFF Control Register n (n=00-63) | PDROFFCRn | W | H'1008 + n x 64 | H'0000 0000 |
| HW Redundant Error Control/Status Register | SYSCRDNCSR | R/W | H'0020 | H'FFFF 0000 |
| HW Redundant Error Injection Register | SYSCRDNIR | R/W | H'0024 | H'0000 0000 |
| APB BUS Status Register | SYSCAPBACR | R/W | H'0030 | H'0000 0000 |
| ISO Error High Status/Clear Register0 | SYSCISOEHSR0 | R/WC1 | H'0860 | H'0000 0000 |
| ISO Error High Status/Clear Register1 | SYSCISOEHSR1 | R/WC1 | H'0864 | H'0000 0000 |
| ISO Error Low Status/Clear Register0 | SYSCISOELSR0 | R/WC1 | H'0868 | H'0000 0000 |
| ISO Error Low Status/Clear Register1 | SYSCISOELSR1 | R/WC1 | H'086C | H'0000 0000 |
| ISO Error High Injection Register0 | SYSCISOEHIR0 | R/W | H'0870 | H'0000 0000 |
| ISO Error High Injection Register1 | SYSCISOEHIR1 | R/W | H'0874 | H'0000 0000 |
| ISO Error Low Injection Register0 | SYSCISOELIR0 | R/W | H'0878 | H'0000 0000 |
| ISO Error Low Injection Register1 | SYSCISOELIR1 | R/W | H'087C | H'0000 0000 |
| ISO Error High Mask Register0 | SYSCISOEHMR0 | R/W | H'0C00 | H'0000 0000 |
| ISO Error High Mask Register1 | SYSCISOEHMR1 | R/W | H'0C04 | H'0000 0000 |
| ISO Error Low Mask Register0 | SYSCISOELMR0 | R/W | H'0C08 | H'0000 0000 |
| ISO Error Low Mask Register1 | SYSCISOELMR1 | R/W | H'0C0C | H'0000 0000 |
| Domain0 Write access control register0 | SYSCD0WACR0 | R/W | H'3000 | H'FFFF FFFF |
| Domain0 Write access control register1 | SYSCD0WACR1 | R/W | H'3004 | H'FFFF FFFF |
| Domain0 Write access control register2 | SYSCD0WACR2 | R/W | H'3008 | H'FFFF FFFF |
| Domain0 Write access control register3 | SYSCD0WACR3 | R/W | H'300C | H'FFFF FFFF |
| Domain0 Write access control register4 | SYSCD0WACR4 | R/W | H'3010 | H'FFFF FFFF |
| Domain0 Write access control register5 | SYSCD0WACR5 | R/W | H'3014 | H'FFFF FFFF |
| Domain1 Write access control register0 | SYSCD1WACR0 | R/W | H'3020 | H'0000 0000 |
| Domain1 Write access control register1 | SYSCD1WACR1 | R/W | H'3024 | H'0000 0000 |
| Domain1 Write access control register2 | SYSCD1WACR2 | R/W | H'3028 | H'0000 0000 |
| Domain1 Write access control register3 | SYSCD1WACR3 | R/W | H'302C | H'0000 0000 |
| Domain1 Write access control register4 | SYSCD1WACR4 | R/W | H'3030 | H'0000 0000 |
| Domain1 Write access control register5 | SYSCD1WACR5 | R/W | H'3034 | H'0000 0000 |
| Domain2 Write access control register0 | SYSCD2WACR0 | R/W | H'3040 | H'0000 0000 |
| Domain2 Write access control register1 | SYSCD2WACR1 | R/W | H'3044 | H'0000 0000 |
| Domain2 Write access control register2 | SYSCD2WACR2 | R/W | H'3048 | H'0000 0000 |
| Domain2 Write access control register3 | SYSCD2WACR3 | R/W | H'304C | H'0000 0000 |
| Domain2 Write access control register4 | SYSCD2WACR4 | R/W | H'3050 | H'0000 0000 |
| Domain2 Write access control register5 | SYSCD2WACR5 | R/W | H'3054 | H'0000 0000 |
| Domain3 Write access control register0 | SYSCD3WACR0 | R/W | H'3060 | H'0000 0000 |
| Domain3 Write access control register1 | SYSCD3WACR1 | R/W | H'3064 | H'0000 0000 |
| Domain3 Write access control register2 | SYSCD3WACR2 | R/W | H'3068 | H'0000 0000 |
| Domain3 Write access control register3 | SYSCD3WACR3 | R/W | H'306C | H'0000 0000 |

| Register name | Abbreviation | R/W | Offset Address | Initial Value |
|---|---|---|---|---|
| Domain3 Write access control register4 | SYSCD3WACR4 | R/W | H'3070 | H'0000 0000 |
| Domain3 Write access control register5 | SYSCD3WACR5 | R/W | H'3074 | H'0000 0000 |

Note
*1:The initial value cannot be confirmed.

## Table 11.7 Register Diagram

| | address | | offset address |
|---|---|---|---|
| Domain0 | E618_0000h | Common Registers | 0000h |
| | | PowerDomainRegisters | 1000h |
| | | Reserved area | 2000h |
| | | Write access control Registers(only Domain0) | 3000h |
| Domain1 | E618_4000h | mirroring from all Domains | |
| Domain2 | E618_8000h | | |
| Domain3 | E618_C000h | | |
| | E618_FFFFh | | |

This Domain is different from Power Domain.
Bus module can protect access by address region (min 4kB unit) each regionID.
ex.) only Cortex-R52 can access to Domain0. Other masters cannot access to Domain0.

Write access protection registers in SYSC can protect write access each regisiter or each Power domain registers.

| Power Domain Registers | Offset Address | Power Domain | |
|---|---|---|---|
| PDR00 | 1000h | A1E0D0C0 | AP-system core0 |
| PDR01 | 1040h | A1E0D0C1 | AP-system core1 |
| PDR02 | 1080h | A1E0D0C2 | AP-system core2 |
| PDR03 | 10C0h | A1E0D0C3 | AP-system core3 |
| PDR04 | 1100h | | |
| PDR05 | 1140h | | |
| PDR06 | 1180h | | |
| PDR07 | 11C0h | | |
| PDR08 | 1200h | | |
| PDR09 | 1240h | | |
| PDR10 | 1280h | | |
| PDR11 | 12C0h | | |
| PDR12 | 1300h | | |
| PDR13 | 1340h | | |
| PDR14 | 1380h | | |
| PDR15 | 13C0h | | |
| PDR16 | 1400h | A2E0D0 | AP-system cluster0 |
| PDR17 | 1440h | | |
| PDR18 | 1480h | | |
| PDR19 | 14C0h | | |
| PDR20 | 1500h | | |
| PDR21 | 1540h | A3CR0 | |
| PDR22 | 1580h | A3CR1 | |
| PDR23 | 15C0h | A3CR2 | |
| PDR24 | 1600h | A33DGA | 3D Graphics Engine (AXM) |
| PDR25 | 1640h | A23DGB | 3D Graphics Engine (AXM) |
| PDR26 | 1680h | | |
| PDR27 | 16C0h | | |
| PDR28 | 1700h | | |
| PDR29 | 1740h | | |
| PDR30 | 1780h | | |
| PDR31 | 17C0h | C4 | |
| PDR32 | 1800h | | |
| PDR33 | 1840h | A1DSP0 | DSP core0 |
| PDR34 | 1880h | A2IMP01 | IMP core 0/1 |
| PDR35 | 18C0h | A2PSC | IMPPSC and IMPDMAC0 |
| PDR36 | 1900h | A2CV0 | CVe cluster 0 |
| PDR37 | 1940h | A2CV1 | CVe cluster 1 |
| PDR38 | 1980h | A3IMR0 | |
| PDR39 | 19C0h | A3IMR1 | |
| PDR40 | 1A00h | A3VC | |
| PDR41 | 1A40h | | |
| PDR42 | 1A80h | A2CN0 | High Speed Domain incl. CNN, CNN RAM0(SPMC) and Slim-IMP Dmac0/1 |
| PDR43 | 1AC0h | | |
| PDR44 | 1B00h | A1CN0 | |
| PDR45 | 1B40h | A1DSP1 | DSP core1 |
| PDR46 | 1B80h | | |
| PDR47 | 1BC0h | A2DMA | IMPDMAC1 |
| PDR48 | 1C00h | A2CV2 | CVe cluster 2 |
| PDR49 | 1C40h | A2CV3 | CVe cluster 3 |
| PDR50 | 1C80h | A3IMR2 | |
| PDR51 | 1CC0h | A3IMR3 | |
| PDR52 | 1D00h | A3PCI | PCI Express |
| PDR53 | 1D40h | A2PCIPHY | PCI PHY |
| PDR54 | 1D80h | | |
| PDR55 | 1DC0h | | |
| PDR56 | 1E00h | A3VIP0 | UMF |
| PDR57 | 1E40h | | |
| PDR58 | 1E80h | A3VIP2 | SMD-PS0, SMD-POST0 |
| PDR59 | 1EC0h | | |
| PDR60 | 1F00h | A3ISP0 | ISP0 core |
| PDR61 | 1F40h | | |
| PDR62 | 1F80h | A3DUL | PAP-Subsystem |
| PDR63 | 1FC0h | | |

| Register | Offset Address |
|---|---|
| PDRSRn | 0 |
| PDRONCRn | 4 |
| PDROFFCRn | 8 |
| Reserved | c |
| Reserved | 10 |
| Reserved | 14 |
| Reserved | 18 |
| Reserved | 1c |
| Reserved | 20 |
| Reserved | 24 |
| Reserved | 28 |
| Reserved | 2c |
| Reserved | 30 |
| Reserved | 34 |
| Reserved | 38 |
| Reserved | 3c |

## 11.2 Register Description

(include Safety Mechanism related register description)

Explanation of abbreviation of register

Initial value: Value of the register after power-on reset

-：Undefined value

R/W：The bit or field is readable and writable.

R/WC0：The bit or field is readable and writable. Writing 0 to the bit initializes the bit. Writing 1 to the bit is ignored.

R/WC1：The bit or field is readable and writable. Writing 1 to the bit initializes the bit. Writing 0 to the bit is ignored.

R：The bit or field is readable only. When writing to the register, write 0 to it.

-/WB：the bit or field is writable. Note that values read from write-only bits are not guaranteed.

### 11.2.1 SYSC Status Register (SYSCSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | BUSY[1:0] | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | *1 | *1 | Reserved *2 |
| 1 to 0 | BUSY[1:0] | *1 | *1 | Indicates whether the SYSC is processing Power-ON or Power-OFF sequence.<br>00: Processing Power-ON or OFF sequence<br>11: Not processing Power-ON or OFF sequence |

Note
*1: Refer to bit assignment table above.
*2: In Reserved, read value is always initial value and write value must be initial value.

### Common Registers

SYSC has Common Registers to control all power domains.

### 11.2.2 Power-ON Status Register0 (SYSCPONSR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Power-ON status for PDR00-31 power domains.<br>0: non-Power-ON state<br>1: Power-ON state<br>The read value of reserved PDR bit is 0. |

Note
*1: Refer to bit assignment table above.

### 11.2.3 Power-ON Status Register1 (SYSCPONSR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Power-ON status for PDR32-63 power domains.<br>0: non-Power-ON state<br>1: Power-ON state<br>The read value of reserved PDR bit is 0. |

Note
*1: Refer to bit assignment table above.

### 11.2.4 Power-OFF Status Register0 (SYSCPOFFSR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Power-OFF Status for PDR31-00 power domains.<br>0: non-Power-OFF state<br>1: Power-OFF state<br>The read value of reserved PDR bit is 0. |

Note
*1: Refer to bit assignment table above.

### 11.2.5 Power-OFF Status Register1 (SYSCPOFFSR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Power-OFF Status for PDR63-32 power domains.<br>0: non-Power-OFF state<br>1: Power-OFF state<br>The read value of reserved PDR bit is 0. |

Note
*1: Refer to bit assignment table above.

### 11.2.6 Interrupt Status/Clear Register0 (complete) (SYSCISCR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Power-ON or OFF process complete interrupt status for PDR00-31 power domains.<br>If SYSCIER0 PDR[n] bit is 1 and Power-ON or OFF process is completed, PDR[n] bit is set to 1.<br>If write 1 to PDR[n], PDR[n] bit is cleared to 0.<br>If PDR[n] is reserved power domain, read value is 0 and write value must be initial value. |

Note
*1: Refer to bit assignment table above.

### 11.2.7 Interrupt Status/Clear Register1 (complete) (SYSCISCR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Power-ON or OFF process complete interrupt status for PDR32-63 power domains.<br>If SYSCIER1 PDR[n] bit is 1 and Power-ON or OFF process is completed, PDR[n] bit is set to 1.<br>If write 1 to PDR[n], PDR[n] bit is cleared to 0.<br>If PDR[n] is reserved power domain, read value is 0 and write value must be initial value. |

Note
*1: Refer to bit assignment table above.

### 11.2.8 Interrupt Enable Register0 (complete) (SYSCIER0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Power-ON or OFF process complete interrupt Enable for PDR00-31 power domains.<br>0: Disable complete interrupt.<br>1: Enable complete interrupt.<br>If PDR[n] is reserved power domain, write value must be initial value. |

Note
*1: Refer to bit assignment table above.

### 11.2.9 Interrupt Enable Register1 (complete) (SYSCIER1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Power-ON or OFF process complete interrupt Enable for PDR32-63 power domains.<br>0: Disable complete interrupt.<br>1: Enable complete interrupt.<br>If PDR[n] is reserved power domain, write value must be initial value. |

Note
*1: Refer to bit assignment table above.

### 11.2.10 Interrupt Mask Register0 (complete) (SYSCIMR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Mask complete interrupt request to INTC for PDR00-31 power domains.<br>0: Not mask complete interrupt request<br>1: Masks complete interrupt request<br>If PDR[n] is reserved power domain, write value must be initial value. |

Note
*1: Refer to bit assignment table above.

### 11.2.11 Interrupt Mask Register1 (complete) (SYSCIMR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Mask complete interrupt request to INTC for PDR32-63 power domains.<br>0: Not mask complete interrupt request<br>1: Masks complete interrupt request<br>If PDR[n] is reserved power domain, write value must be initial value. |

Note
*1: Refer to bit assignment table above.

### 11.2.12 Extended Function Control Register (SYSCEFCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | ADE | ADM | AD- | - | - | - | - | - | - | - | - |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/WC1 | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | - | *1 | *1 | Reserved *2 |
| 10 | ADE | *1 | *1 | All request Done interrupt Enable<br>0: Disable<br>1: Enable |
| 9 | ADM | *1 | *1 | All request Done interrupt Mask<br>0: not mask All request Done interrupt.<br>1: Mask All request Done interrupt. |
| 8 | AD | *1 | *1 | All request Done interrupt status/clear<br>0: Non-detected All request Done interrupt<br>1: Detected All request Done interrupt<br>(when SYSCRSR0, 1, 2, 3 are all 0, this bit is set to 1)<br>When write 1, this bit is cleared to 0.<br>If ADE bit is 0, this bit is not set. |
| 7 to 0 | - | *1 | *1 | Reserved *2 |

Note
*1: Refer to bit assignment table above.

*2: In Reserved, read value is always initial value.

### Power Domain Registers

SYSC has Power Domain Registers for each Power Domain.

### 11.2.13 Power Domain Status Register n (PDRSRn) (n=00-63)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | ON_STATE | - | - | - | OFF_STAT | - | - | - | ON | - | - | - | OFF |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | - | *1 | *1 | Reserved *2 |
| 12 | ON_STATE | *1 | *1 | Indicates status of Power-ON sequence.<br>0:Not processing Power-ON sequence<br>1:Processing Power ON sequence |
| 11 to 9 | - | *1 | *1 | Reserved *2 |
| 8 | OFF_STATE | *1 | *1 | Indicates status of Power-OFF sequence.<br>0: Not processing Power-OFF sequence<br>1: Processing Power OFF sequence |
| 7 to 5 | - | *1 | *1 | Reserved *2 |
| 4 | ON | *1 | *1 | Indicates Power-ON state.<br>0:Not Power-ON state<br>1:Power-ON state |
| 3 to 1 | - | *1 | *1 | Reserved *2 |
| 0 | OFF | *1 | *1 | Indicates Power-OFF state.<br>0:Not Power-OFF state<br>1:Power-OFF state |

Note
*1: Refer to bit assignment table above.
*2: In Reserved, read value is always initial value.

### 11.2.14 Power Domain power-ON Control Register n (PDRONCRn) (n=00-63)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | PWR ON |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | - | *1 | *1 | Reserved *2 |
| 0 | PWRON | *1 | *1 | Power-ON request.<br>If write 1, activate Power-ON request.<br>If the target power domain is already Power-ON, the request is ignored. |

Note
*1: Refer to bit assignment table above.
*2: In Reserved, write value must be 0.

### 11.2.15 Power Domain power-OFF Control Register n (PDROFFCRn) (n=00-63)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | PWR OFF |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | - | *1 | *1 | Reserved *2 |
| 0 | PWROFF | *1 | *1 | Power-OFF request.<br>If write 1, activate Power-OFF request.<br>If the target power domain is already Power-OFF, the request is ignored. |

Note
*1: Refer to bit assignment table above.
*2: In Reserved, write value must be 0.

### 11.2.16 Power Domain power-ON/OFF Error Status Register n (PDRESRn) (n=00-63)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | ERR |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | - | *1 | *1 | Reserved *2 |
| 0 | ERR | *1 | *1 | Power Domain power-ON/OFF Error Status<br>0: Either a preceding power OFF or power ON request was accepted.<br>1: Either a preceding power OFF or power ON request was not accepted. |

Note
*1: Refer to bit assignment table above.
*2: In Reserved, read value is already 0.

**Write access protection Registers**

### 11.2.17 Protect Control/Status Register (SYSCPTCSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | EIEI | EIE | ERR |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | *1 | *1 | Reserved *2 |
| 2 | EIEI | *1 | *1 | Write access protection Error Interrupt request Enable for INTC.<br>0: Disable error interrupt request for INTC<br>1: Enable error interrupt request for INTC |
| 1 | EIE | *1 | *1 | Write access protection Error Interrupt request Enable for ECM (FuSa).<br>0: Disable error interrupt request for ECM<br>1: Enable error interrupt request for ECM |
| 0 | ERR | *1 | *1 | Indicates Write access protection Error status.<br>0: Not detected Write access protection error.<br>1: Detected Write access protection error.<br>If write 1, this bit is cleared to 0.<br>If write 0, it is ignored. |

Note
*1: Refer to bit assignment table above.
*2: In Reserved, read value is always initial value and write value must be initial value.

### 11.2.18 Protect Error Address Register (SYSCPTERADR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | CLR |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/WC1 |

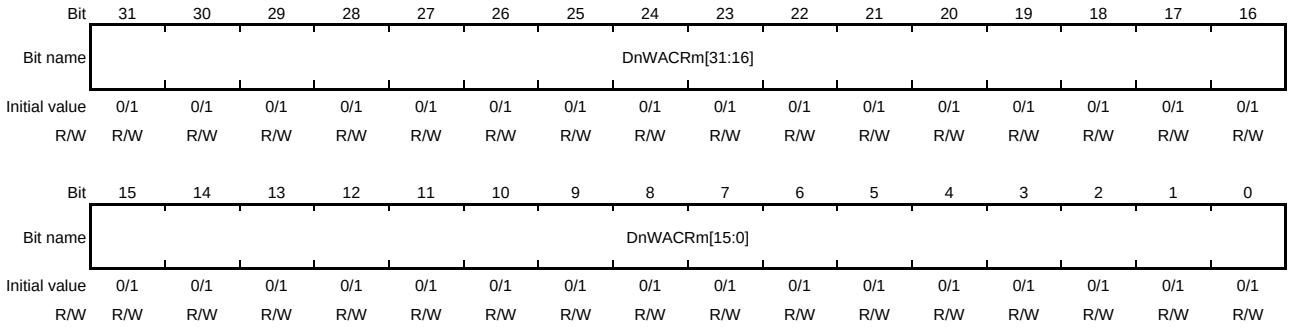
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | ADR[15:0] | | | | | | | | | | | | | | | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | *1 | *1 | Reserved *2 |
| 16 | CLR | *1 | *1 | Write access protection error address clear<br>Before it clear, must clear to SYSCPTCSR.ERR .<br>If write 1, ADR is cleared to 0 and the next error address can be hold.<br>Read value is always 0. |
| 15 to 0 | ADR[15:0] | *1 | *1 | First Write access protection error address.<br>Indicates first Write access protection error address if Write access protection error occurs. It is cleared by CLR bit. |

Note
*1: Refer to bit assignment table above.
*2: In Reserved bits, read value is always initial value and write value must be initial value.

### 11.2.19 Domain n Write Access Control Register (SYSCDnWACRm) (n=0/1/2/3) (m=0/1/2/3)

This register can be written by only Domain0 address. (H'E618_0000− H'E618_3FFC)




Initial value
n=0: H'FFFF_FFFF
n=1,2,3: H'0000_0000

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DnWACRm[31:0] | *1 | *1 | Domain[n] Write access control for Common Registers.<br>For protected address assignment, refer to Table 11.9.<br>0 : Prohibit write access.<br>1 : Permit write access.<br>n: Domain No n=0/1/2/3<br>m: Register No m=0/1/2/3 |

Note
*1: Refer to bit assignment table above.

Table 11.9 Write access protected address assignment for Common Registers

| SYSCDnWACR0 | | SYSCDnWACR1 | | SYSCDnWACR2 | | SYSCDnWACR3 | |
|---|---|---|---|---|---|---|---|
| Bit | Offset address | Bit | Offset address | Bit | Offset address | Bit | Offset address |
| 31 | 07Ch | 31 | 47Ch | 31 | 87Ch | 31 | C7Ch |
| 30 | 078h | 30 | 478h | 30 | 878h | 30 | C78h |
| 29 | 074h | 29 | 474h | 29 | 874h | 29 | C74h |
| 28 | 070h | 28 | 470h | 28 | 870h | 28 | C70h |
| 27 | 06Ch | 27 | 46Ch | 27 | 86Ch | 27 | C6Ch |
| 26 | 068h | 26 | 468h | 26 | 868h | 26 | C68h |
| 25 | 064h | 25 | 464h | 25 | 864h | 25 | C64h |
| 24 | 060h | 24 | 460h | 24 | 860h | 24 | C60h |
| 23 | 05Ch | 23 | 45Ch | 23 | 85Ch | 23 | C5Ch |
| 22 | 058h | 22 | 458h | 22 | 858h | 22 | C58h |
| 21 | 054h | 21 | 454h | 21 | 854h | 21 | C54h |
| 20 | 050h | 20 | 450h | 20 | 850h | 20 | C50h |
| 19 | 04Ch | 19 | 44Ch | 19 | 84Ch | 19 | C4Ch |
| 18 | 048h | 18 | 448h | 18 | 848h | 18 | C48h |
| 17 | 044h | 17 | 444h | 17 | 844h | 17 | C44h |
| 16 | 040h | 16 | 440h | 16 | 840h | 16 | C40h |
| 15 | 03Ch | 15 | 43Ch | 15 | 83Ch | 15 | C3Ch |
| 14 | 038h | 14 | 438h | 14 | 838h | 14 | C38h |
| 13 | 034h | 13 | 434h | 13 | 834h | 13 | C34h |
| 12 | 030h | 12 | 430h | 12 | 830h | 12 | C30h |
| 11 | 02Ch | 11 | 42Ch | 11 | 82Ch | 11 | C2Ch |
| 10 | 028h | 10 | 428h | 10 | 828h | 10 | C28h |
| 9 | 024h | 9 | 424h | 9 | 824h | 9 | C24h |
| 8 | 020h | 8 | 420h | 8 | 820h | 8 | C20h |
| 7 | 01Ch | 7 | 41Ch | 7 | 81Ch | 7 | C1Ch |
| 6 | 018h | 6 | 418h | 6 | 818h | 6 | C18h |
| 5 | 014h | 5 | 414h | 5 | 814h | 5 | C14h |
| 4 | 010h | 4 | 410h | 4 | 810h | 4 | C10h |
| 3 | 00Ch | 3 | 40Ch | 3 | 80Ch | 3 | C0Ch |
| 2 | 008h | 2 | 408h | 2 | 808h | 2 | C08h |
| 1 | 004h | 1 | 404h | 1 | 804h | 1 | C04h |
| 0 | 000h | 0 | 400h | 0 | 800h | 0 | C00h |

Example)
SYSCD0WACR0 bit1 = 0 : H'E618_0004 write access is protected.
SYSCD3WACR3 bit16 = 0 : H'E618_CC40 write access is protected.

### 11.2.20 Domain n Write Access Control Register (SYSCDnWACRm) (n=0/1/2/3) (m=4/5)

This register can be written by only Domain0 address. (H'E618_0000− H'E618_3FFC)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | DnWACRm[31:16] | | | | | | | | | | | | | | | |
| Initial value | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | DnWACRm[15:0] | | | | | | | | | | | | | | | |
| Initial value | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Initial value
n=0: H'FFFF_FFFF
n=1,2,3: H'0000_0000

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DnWACRm[31:0] | *1 | *1 | Domain[n] Write access control for Power domain register.<br>For protected address assignment, refer to Table 11.10.<br>0 : Prohibit write access.<br>1 : Permit write access.<br>n: Domain No n=0/1/2/3<br>m: Register No m= 4/5 |

Note
*1: Refer to bit assignment table above.

Table 11.10 Write access protected address assignment for Power Domain Registers

| SYSCDnWACR4 | | |
|---|---|---|
| Bit | Power Domain Register | Offset address |
| 31 | PDR31 | H'17C0-H'17FC |
| 30 | PDR30 | H'1780-H'17BC |
| 29 | PDR29 | H'1740-H'177C |
| 28 | PDR28 | H'1700-H'173C |
| 27 | PDR27 | H'16C0-H'16FC |
| 26 | PDR26 | H'1680-H'16BC |
| 25 | PDR25 | H'1640-H'167C |
| 24 | PDR24 | H'1600-H'163C |
| 23 | PDR23 | H'15C0-H'15FC |
| 22 | PDR22 | H'1580-H'15BC |
| 21 | PDR21 | H'1540-H'157C |
| 20 | PDR20 | H'1500-H'153C |
| 19 | PDR19 | H'14C0-H'14FC |
| 18 | PDR18 | H'1480-H'14BC |
| 17 | PDR17 | H'1440-H'147C |
| 16 | PDR16 | H'1400-H'143C |
| 15 | PDR15 | H'13C0-H'13FC |
| 14 | PDR14 | H'1380-H'13BC |
| 13 | PDR13 | H'1340-H'137C |
| 12 | PDR12 | H'1300-H'133C |
| 11 | PDR11 | H'12C0-H'12FC |
| 10 | PDR10 | H'1280-H'12BC |
| 9 | PDR09 | H'1240-H'127C |
| 8 | PDR08 | H'1200-H'123C |
| 7 | PDR07 | H'11C0-H'11FC |
| 6 | PDR06 | H'1180-H'11BC |
| 5 | PDR05 | H'1140-H'117C |
| 4 | PDR04 | H'1100-H'113C |
| 3 | PDR03 | H'10C0-H'10FC |
| 2 | PDR02 | H'1080-H'10BC |
| 1 | PDR01 | H'1040-H'107C |
| 0 | PDR00 | H'1000-H'103C |

| SYSCn0WACR5 | | |
|---|---|---|
| Bit | Power Domain Register | Offset address |
| 31 | PDR63 | H'1FC0-H'1FFC |
| 30 | PDR62 | H'1F80-H'1FBC |
| 29 | PDR61 | H'1F40-H'1F7C |
| 28 | PDR60 | H'1F00-H'1F3C |
| 27 | PDR59 | H'1EC0-H'1EFC |
| 26 | PDR58 | H'1E80-H'1EBC |
| 25 | PDR57 | H'1E40-H'1E7C |
| 24 | PDR56 | H'1E00-H'1E3C |
| 23 | PDR55 | H'1DC0-H'1DFC |
| 22 | PDR54 | H'1D80-H'1DBC |
| 21 | PDR53 | H'1D40-H'1D7C |
| 20 | PDR52 | H'1D00-H'1D3C |
| 19 | PDR51 | H'1CC0-H'1CFC |
| 18 | PDR50 | H'1C80-H'1CBC |
| 17 | PDR49 | H'1C40-H'1C7C |
| 16 | PDR48 | H'1C00-H'1C3C |
| 15 | PDR47 | H'1BC0-H'1BFC |
| 14 | PDR46 | H'1B80-H'1BBC |
| 13 | PDR45 | H'1B40-H'1B7C |
| 12 | PDR44 | H'1B00-H'1B3C |
| 11 | PDR43 | H'1AC0-H'1AFC |
| 10 | PDR42 | H'1A80-H'1ABC |
| 9 | PDR41 | H'1A40-H'1A7C |
| 8 | PDR40 | H'1A00-H'1A3C |
| 7 | PDR39 | H'19C0-H'19FC |
| 6 | PDR38 | H'1980-H'19BC |
| 5 | PDR37 | H'1940-H'197C |
| 4 | PDR36 | H'1900-H'193C |
| 3 | PDR35 | H'18C0-H'18FC |
| 2 | PDR34 | H'1880-H'18BC |
| 1 | PDR33 | H'1840-H'187C |
| 0 | PDR32 | H'1800-H'183C |

Example)
SYSCD0WACR4 bit1 = 0 : H'E618_1040-H'E618_107C write access is protected.
SYSCD3WACR5 bit16 = 0 : H'E618_DC00-H'E618_DC40 write access is protected.

## Common Registers (FuSa)

### 11.2.21 HW Redundant Error Control/Status Register (SYSCRDNCSR) (FuSa)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | EIE[15] | EIE[14] | EIE[13] | EIE[12] | EIE[11] | EIE[10] | EIE[9] | EIE[8] | EIE[7] | EIE[6] | EIE[5] | EIE[4] | EIE[3] | EIE[2] | EIE[1] | EIE[0] |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | ERR[15] | ERR[14] | ERR[13] | ERR[12] | ERR[11] | ERR[10] | ERR[9] | ERR[8] | ERR[7] | ERR[6] | ERR[5] | ERR[4] | ERR[3] | ERR[2] | ERR[1] | ERR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | EIE[15:0] | *1 | *1 | HW Redundant Error Interrupt Enable<br>This bit indicates whether the SYSC allows to generate error interrupt to ECM or not when HW Redundant Error is detected.<br>0: Disable Error Interrupt Disable<br>1: Enable Error Interrupt (send error to ECM module). |
| 15 to 0 | ERR[15:0] | *1 | *1 | HW Redundant Error Status<br>Indicates that HW Redundant Error is detected.<br>0: Not detected HW Redundant Error<br>1: Detected HW Redundant Error<br>If write 1 to ERR[n] (n=0-15), ERR[n] bit is cleared to 0.<br>If write 0, it is ignored. |

Note
*1: Refer to bit assignment table above.

### 11.2.22 HW Redundant Error Injection Register (SYSCRDNIR) (FuSa)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | ERIN[15] | ERIN[14] | ERIN[13] | ERIN[12] | ERIN[11] | ERIN[10] | ERIN[9] | ERIN[8] | ERIN[7] | ERIN[6] | ERIN[5] | ERIN[4] | ERIN[3] | ERIN[2] | ERIN[1] | ERIN[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | *1 | *1 | These bits are always read as 0. The write value should always be 0. |
| 15 to 0 | ERIN[15:0] | *1 | *1 | Injects HW Redundant Error<br>By setting those bits, errors of HW redundant are asserted regardless of violated HW Redundant or not<br>0: Not-inject HW Redundant Error.<br>1: Inject HW Redundant Error.<br>SYSCRDNCSR.ERR[n] (n=0-15) bit is set to 1. |

Note
*1: Refer to bit assignment table above.

### 11.2.23 APB BUS Access Check Register (SYSCAPBACR) (FuSa)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | VAL[31] | VAL[30] | VAL[29] | VAL[28] | VAL[27] | VAL[26] | VAL[25] | VAL[24] | VAL[23] | VAL[22] | VAL[21] | VAL[20] | VAL[19] | VAL[18] | VAL[17] | VAL[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | VAL[15] | VAL[14] | VAL[13] | VAL[12] | VAL[11] | VAL[10] | VAL[9] | VAL[8] | VAL[7] | VAL[6] | VAL[5] | VAL[4] | VAL[3] | VAL[2] | VAL[1] | VAL[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | VAL[31:0] | *1 | *1 | APB BUS Access Check Register for FuSa<br>32 bit Write/Read register |

Note
*1: Refer to bit assignment table above.

### 11.2.24 Isolation Error High Status/Clear Register0 (SYSCISOEHSR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Isolation Error High Status/Clear<br>1:Isolation Error form High level<br>0:Not isolation Error form High level |

Note
*1: Refer to bit assignment table above.

### 11.2.25 Isolation Error High Status/Clear Register1 (SYSCISOEHSR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Isolation Error High Status/Clear<br>1:Isolation Error form High level<br>0:Not isolation Error form High level |

Note
*1: Refer to bit assignment table above.

### 11.2.26 Isolation Error Low Status/Clear Register0 (SYSCISOELSR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Isolation Error Low Status/Clear<br>1:Isolation Error form Low level<br>0:Not isolation Error form Low level |

Note
*1: Refer to bit assignment table above.

### 11.2.27 Isolation Error Low Status/Clear Register1 (SYSCISOELSR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Isolation Error Low Status/Clear<br>1:Isolation Error form Low level<br>0:Not isolation Error form Low level |

Note
*1: Refer to bit assignment table above.

### 11.2.28 Isolation Error High Injection Register0 (SYSCISOEHIR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Isolation Error High Injection for PDR00-31 *2<br>0:Not Error Injection<br>1:Error Injection |

Note
*1: Refer to bit assignment table above.
*2 target bit name PDR[31],PDR[25-24],PDR[16],PDR[3-0]
about PDR, see Table 11.2 PDR assignment

### 11.2.29 Isolation Error High Injection Register1 (SYSCISOEHIR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Isolation Error High Injection for PDR32-63 *2<br>0:Not Error Injection<br>1:Error Injection |

Note
*1: Refer to bit assignment table above.
*2 target bit name PDR[62],PDR[60],PDR[58],PDR[56],PDR[53-47],PDR[45-44],PDR[42],PDR[40-33]
about PDR, see Table 11.2 PDR assignment

### 11.2.30 Isolation Error Low Injection Register0 (SYSCISOELIR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Isolation Error Low Injection for PDR00-31 *2<br>0:Not Error Injection<br>1:Error Injection |

Note
*1: Refer to bit assignment table above.
*2 target bit name PDR[31],PDR[25-24],PDR[16],PDR[3-0]
about PDR, see Table 11.2 PDR assignment

### 11.2.31 Isolation Error Low Injection Register1 (SYSCISOELIR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Isolation Error Low Injection for PDR32-63 *2<br>0:Not Error Injection<br>1:Error Injection |

Note
*1: Refer to bit assignment table above.
*2 target bit name PDR[62],PDR[60],PDR[58],PDR[56],PDR[53-47],PDR[45-44],PDR[42],PDR[40-33]
about PDR, see Table 11.2 PDR assignment

### 11.2.32 Isolation Error High Mask Register0 (SYSCISOEHMR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Isolation Error High Mask Status/Clear<br>1:Isolation Error Mask form High level<br>0:Not isolation Error Mask form High level |

Note
*1: Refer to bit assignment table above.

### 11.2.33 Isolation Error High Mask Register1 (SYSCISOEHMR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Isolation Error High Mask Status/Clear<br>1:Isolation Error Mask form High level<br>0:Not isolation Error Mask form High level |

Note
*1: Refer to bit assignment table above.

### 11.2.34 Isolation Error Low Mask Register0 (SYSCISOELMR0)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[31] | PDR[30] | PDR[29] | PDR[28] | PDR[27] | PDR[26] | PDR[25] | PDR[24] | PDR[23] | PDR[22] | PDR[21] | PDR[20] | PDR[19] | PDR[18] | PDR[17] | PDR[16] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[15] | PDR[14] | PDR[13] | PDR[12] | PDR[11] | PDR[10] | PDR[9] | PDR[8] | PDR[7] | PDR[6] | PDR[5] | PDR[4] | PDR[3] | PDR[2] | PDR[1] | PDR[0] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[31:0] | *1 | *1 | Isolation Error Low Mask Status/Clear<br>1:Isolation Error Mask form Low level<br>0:Not isolation Error Mask form Low level |

Note
*1: Refer to bit assignment table above.

### 11.2.35 Isolation Error Low Mask Register1 (SYSCISOELMR1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[63] | PDR[62] | PDR[61] | PDR[60] | PDR[59] | PDR[58] | PDR[57] | PDR[56] | PDR[55] | PDR[54] | PDR[53] | PDR[52] | PDR[51] | PDR[50] | PDR[49] | PDR[48] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit name | PDR[47] | PDR[46] | PDR[45] | PDR[44] | PDR[43] | PDR[42] | PDR[41] | PDR[40] | PDR[39] | PDR[38] | PDR[37] | PDR[36] | PDR[35] | PDR[34] | PDR[33] | PDR[32] |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PDR[63:32] | *1 | *1 | Isolation Error Low Mask Status/Clear<br>1:Isolation Error Mask form Low level<br>0:Not isolation Error Mask form Low level |

Note
*1: Refer to bit assignment table above.

## 11.3 Operation

### 11.3.1 Power Domain Structure

refer to Table 11.1 Power Domain Structure

### 11.3.2 Initial State of Power Domains

Since all power domains are power OFF start.

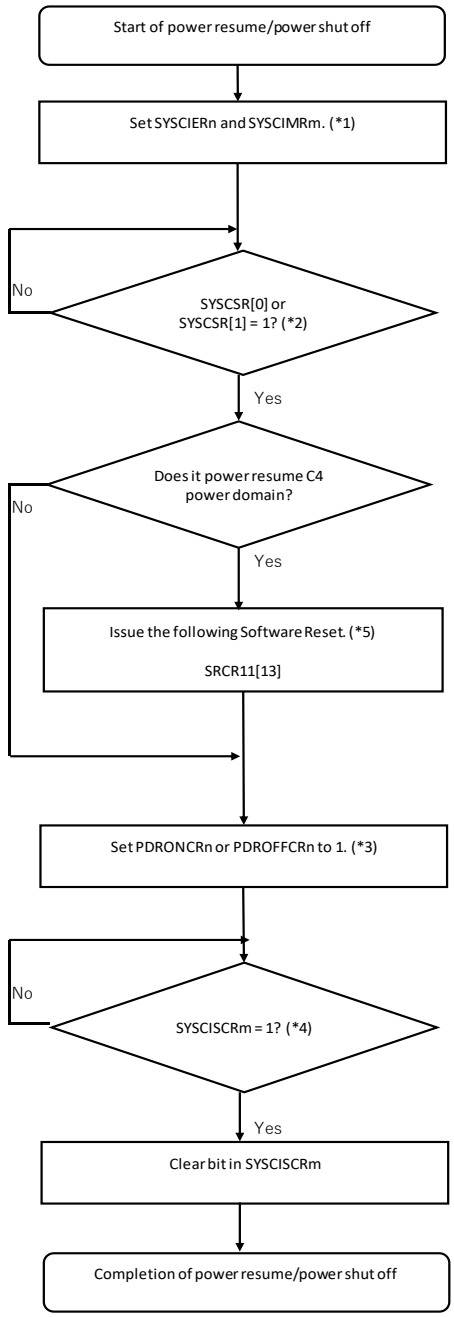
### 11.3.3 Power Control of Arm CPUs

The Cortex-A76 CPU cores and Realtime cores (Cortex-R52) can be powered off by executing the WFI instruction. Similarly, power of Arm CPUs can be resumed either by controlling the APMU (Advanced Power Management Unit) registers or by an IRQ or FIQ interrupt.

For detail of power control by APMU registers, see section 10, Advanced Power Management Unit for Cortex-A76 CPU Core (APMU).

### 11.3.4 Power Control of Non Arm CPU Modules

Power control on modules other than the Arm CPUs can be performed by SYSC registers. Following figure describes power control reference flowchart.




*1 Set interrupt enable register.
For domain 1->32: SYSCIER0, SYSCIMR0
For domain 33->63: SYSCIER1, SYSCIMR1

*2 Check following register, to confirm power resume or shutoff is ready to start.
For power resume : SYSCSR[1]
For power shut off : SYSCSR[0]

*3 For power resume: PDRONCRn
For power shut off: PDROFFCRn
(Refer to section 9.1.6, Power domain structure for supported power domain.)

*4 Wait for the completion of power sequence, by interrupt or register polling
For domain 1->32: SYSCISCR0
For domain 33->63: SYSCISCR1

*5 Assert and release the reset before C4 power domain resume.
(Refer to section 9.3(2)(B) for Software Reset operation.
Refer to sections 9.2.1 and 9.2.2 for reset registers.)

**Figure Power Resume/Shutoff Flowchart for non Arm CPU domain by SYSC Registers**

## 11.4 Usage Notes

It takes several hundreds of microseconds to Power-ON and Power-OFF power domain.

Because actual time required for Power-ON and Power-OFF depends on the status of on-board power line, Power-ON and Power-OFF time is not guaranteed by electrical specification.

To perform BUS control, the power supply is required to be ON.

refer to the following for the table.

chapter 18 AXI-Bus

Before Power-OFF one (or more) of the below power domains, set all modules to the Module Standby state for which belong to the power domain.

Refer to Figure 11.5a.

A3VIP2,0/A3ISP0

Refer to Figure 11.5b.

A3VC/C4/ A2IMP01

Just before Power-OFF for below power domains, clear to Module Standby of modules in these power domains.

Refer to Figure 11.5c.

A2DMA/A2PSC/A2CV3-0/A2CN0/A1DSP1-0

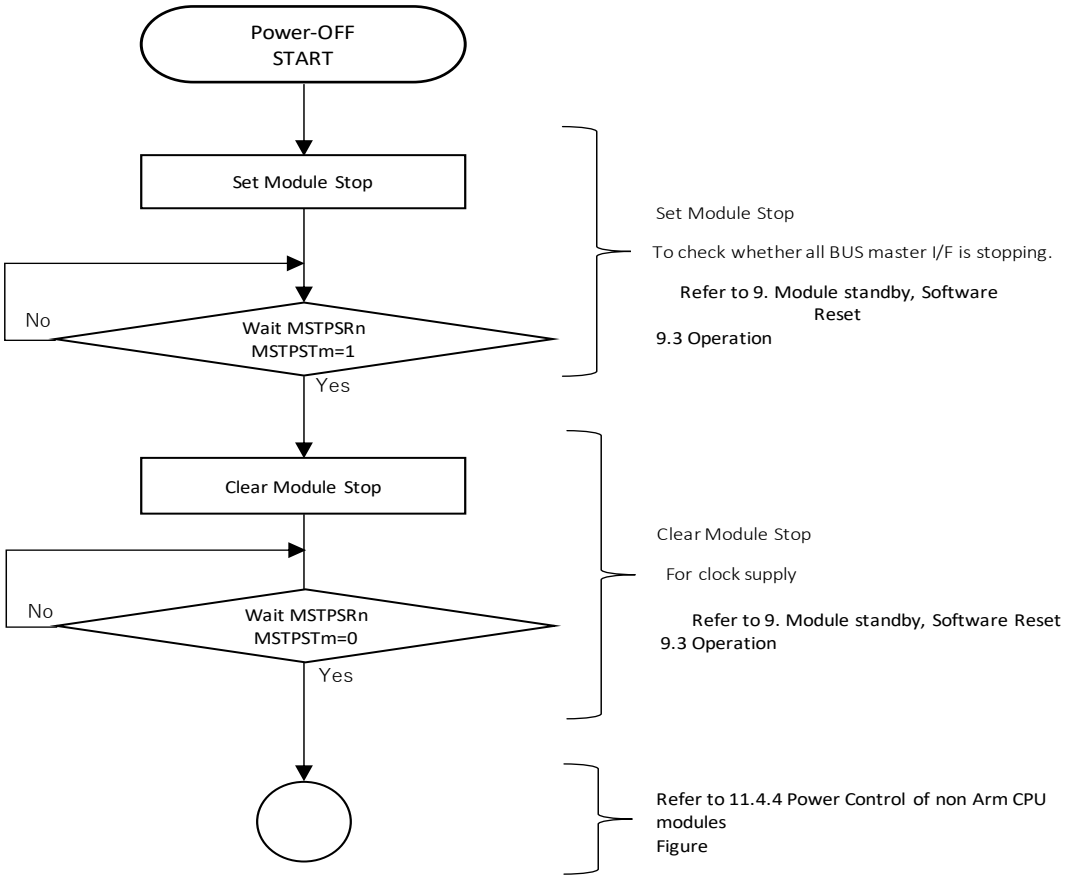



**Figure 11.5a Set and clear module standby**

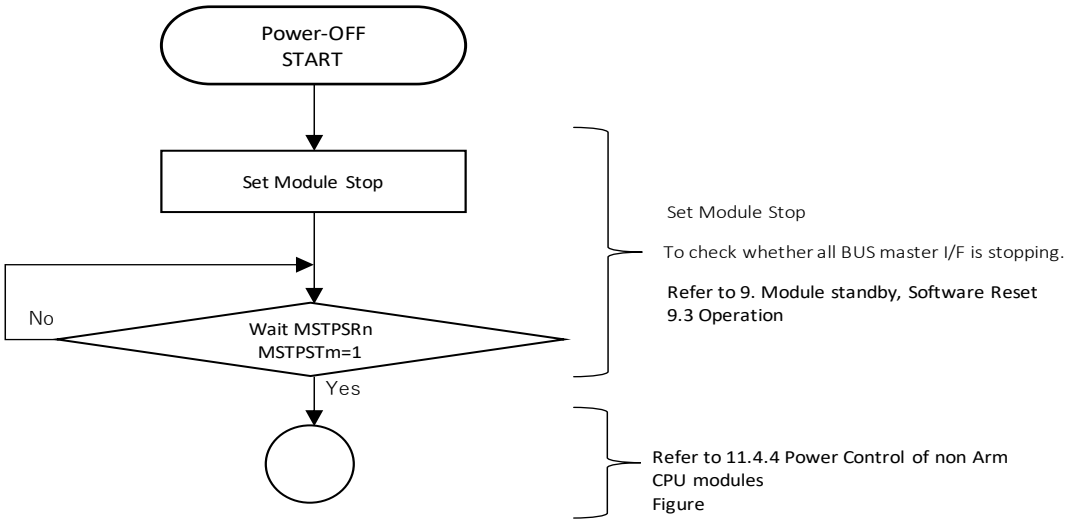



**Figure 11.5b Set Module Standby**

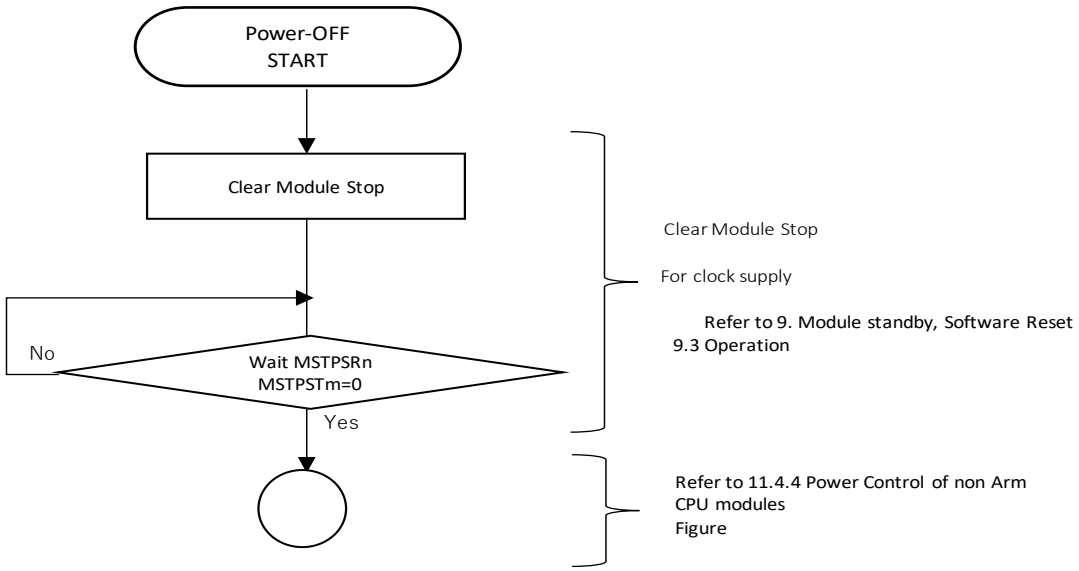



**Figure 11.5c Clear module standby**

## 11.5 Safety Mechanisms

### 11.5.1 Safety Mechanism 1

**a) Overview**

SYSC has the following functions for FuSa.

- Write access protection
- Isolation Error Detection
- HW Redundancy
- SYSC is POST (Power On Self Test) target

**Policy**

SYSC controls Power Switch and Isolation cells for all power domains.

Basically, we can detect the failure of SYSC function by the following mechanism..

Failure of Power Switch control function.:

If Power Switch is unintentionally Power-OFF, we can detect the failure by using register access, reset feedback etc because the modules in the power domain cannot operate correctly.

Failure of Isolation control function:

If isolation is unintentionally active, we can detect the failure by using register access and reset feedback etc. because the signal is masked by isolation cell.

If isolation cell is failure during Power-OFF, we can detect it by isolation error detection function (SM).

**Functional Safety function**

Create error notification to error correction module when occur error.

**HW redundancy**
Can be error notification when occur violation operation, therefore hardware redundancy of register and FSM in sysc logic. Error correction have 16 violation. And can be mask of error notification.

**SYSC is POST (Power ON Self Test) target**

Power control when period of POST mode. All power ON of power domains before POST mode. Power and isolation control when period of POST mode by SWC.

**Write access protection**
Write access protection function for register. For common registers, write access protection is possible for each register. For power control registers, write access protection is possible for each power domains. If occur write access violation, error notification and store of address.

**Isolation Error detection**

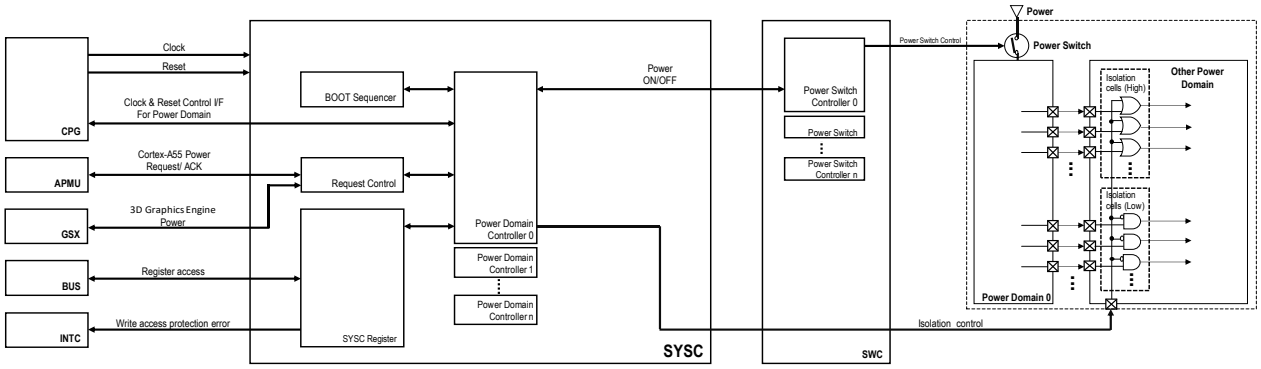
SYSC have error notification of isolation cells. Isolation Function to detect cell failure. It is possible to check in which power domains an error occurred. Error mask function implemented for each error cause.
If target power domain was pseudo power-OFF power domain, this function not work.

**b) Related Registers**

| Register name | Abbreviation |
|---|---|
| HW Redundant Error Control/Status Register | SYSCRDNCSR |
| HW Redundant Error Injection Register | SYSCRDNIR |
| APB BUS Status Register | SYSCAPBACR |
| ISO Error High Status/Clear Register0 | SYSCISOEHSR0 |
| ISO Error High Status/Clear Register1 | SYSCISOEHSR1 |
| ISO Error Low Status/Clear Register0 | SYSCISOELSR0 |
| ISO Error Low Status/Clear Register1 | SYSCISOELSR1 |
| ISO Error High Injection Register0 | SYSCISOEHIR0 |
| ISO Error High Injection Register1 | SYSCISOEHIR1 |
| ISO Error Low Injection Register0 | SYSCISOELIR0 |
| ISO Error Low Injection Register1 | SYSCISOELIR1 |
| ISO Error High Mask Register0 | SYSCISOEHMR0 |
| ISO Error High Mask Register1 | SYSCISOEHMR1 |
| ISO Error Low Mask Register0 | SYSCISOELMR0 |
| ISO Error Low Mask Register1 | SYSCISOELMR1 |
| Domain0 Write access control register0 | SYSCD0WACR0 |
| Domain0 Write access control register1 | SYSCD0WACR1 |
| Domain0 Write access control register2 | SYSCD0WACR2 |
| Domain0 Write access control register3 | SYSCD0WACR3 |
| Domain0 Write access control register4 | SYSCD0WACR4 |
| Domain0 Write access control register5 | SYSCD0WACR5 |
| Domain1 Write access control register0 | SYSCD1WACR0 |
| Domain1 Write access control register1 | SYSCD1WACR1 |
| Domain1 Write access control register2 | SYSCD1WACR2 |
| Domain1 Write access control register3 | SYSCD1WACR3 |
| Domain1 Write access control register4 | SYSCD1WACR4 |
| Domain1 Write access control register5 | SYSCD1WACR5 |
| Domain2 Write access control register0 | SYSCD2WACR0 |
| Domain2 Write access control register1 | SYSCD2WACR1 |
| Domain2 Write access control register2 | SYSCD2WACR2 |
| Domain2 Write access control register3 | SYSCD2WACR3 |
| Domain2 Write access control register4 | SYSCD2WACR4 |
| Domain2 Write access control register5 | SYSCD2WACR5 |
| Domain3 Write access control register0 | SYSCD3WACR0 |
| Domain3 Write access control register1 | SYSCD3WACR1 |
| Domain3 Write access control register2 | SYSCD3WACR2 |
| Domain3 Write access control register3 | SYSCD3WACR3 |
| Domain3 Write access control register4 | SYSCD3WACR4 |
| Domain3 Write access control register5 | SYSCD3WACR5 |

### c) Hardware Description

Connected to the internal peripheral bus (the APB bus), SYSC operates in coordination with CPG, RESET module, and APMU; it contains a module that controls the objects of power shutoff.



# 12. Adaptive Voltage Scaling (AVS) [for specific customers only]

## 12.1 Overview

Adaptive voltage scaling (AVS) mitigates power variation by adjusting VDD_xx voltage.

### 12.1.1 Features

- AVS allows lower voltage operation without operating frequency degradation if there is a room to speed. Up to 3 adjustable voltage setting called as AVS code is offered. AVS can be employed only VDD_xx voltage. (refer to 12.3.1)
- Register-AVS
  Adjust VDD voltage by register values corresponding to AVS code after boot (refer to 12.3.2)

  AVS code is not available via AVS pins which are described in 004_R-CarV4M_Pin_Multiplex_D_r0p51.xlsx. VOLCOND register shows AVS code.

### 12.1.2 Block Diagram

A block diagram of the Adaptive Voltage Scaling (AVS) module is shown in Figure 12.1.

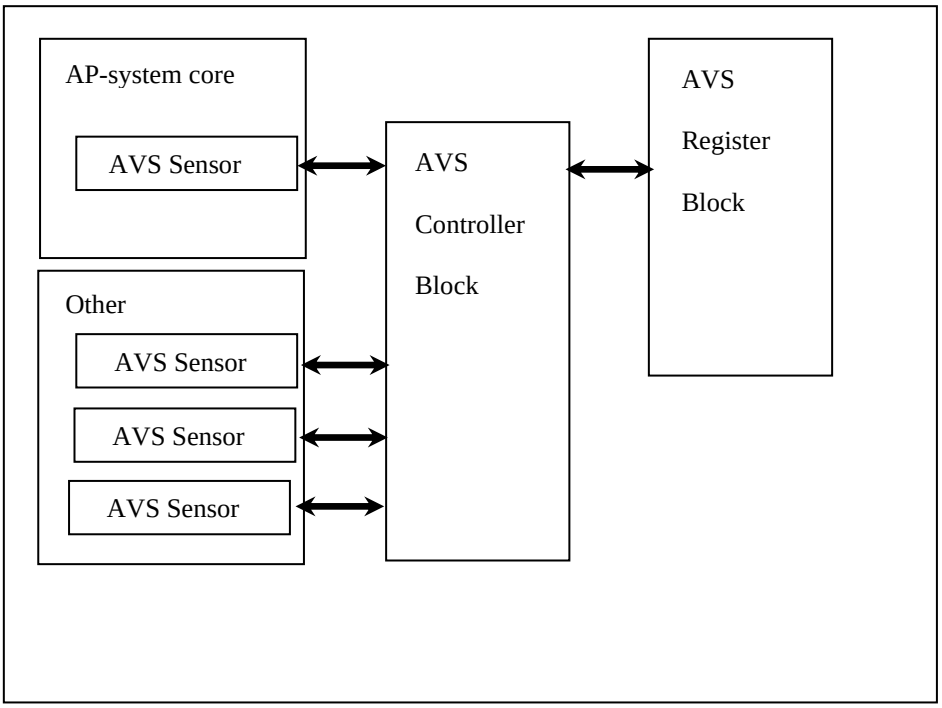



**Figure 12.1 Block Diagram of Adaptive Voltage Scaling (AVS)**

**Table 12.1 Block Diagram Description Table**

| Name | Description |
|---|---|
| AVS Sensor* | Sensor for checking module process variation. |
| AVS Controller | Module that controls information on each sensor. |
| AVS Register | A group of registers that instructs the sensor and stores acquired information. |

### 12.1.3 External Pins

#### (1) Connection diagram

As shown in Figure 12.2, the PMIC output voltage terminal is connected to the VDD_xx terminal, it is possible to it is possible to change the PMIC voltage via I2C after activating the PMIC at a specified voltage.

AVS code is not available via AVS pins which are described in 004_R-CarV4M_Pin_Multiplex_D_r0p51.xlsx. VOLCOND register shows AVS code.

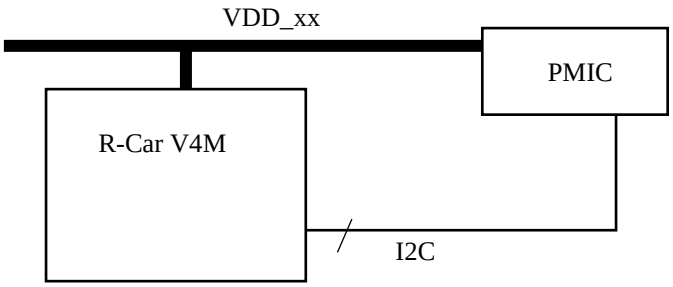



**Figure 12.2 Connection Diagram of Adaptive Voltage Scaling (AVS)**

For how to change the VDD_xx voltage after startup.

| Name | Description |
|---|---|
| PMIC | Power management IC which has the output voltage switching function. |

### 12.1.4 Register Configuration

Table 12.2 is a list of registers.

For the access address of this register, add the Offset Address of Table 12.2 Register Configuration to "BASE ADDRESS" below.

BASE ADDRESS = 0xE60A 0000

[Legend for register description]

| | |
|---|---|
| Initial value: | Register value after a reset |
| —: | Undefined value |
| R/W: | Readable/writable. The written value can be read. |
| R: | Read-only. Writing is prohibit. |
| R/WC0: | Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored |
| R/WC1: | Readable/writable. Writing 1 initializes the bit. Writing 0 is ignored. |
| W: | Write-only. Reading this bit is prohibited. When the bit is reserved, the write value should always be 0. |
| __/W: | Write-only. The read value is undefined. |

**Table 12.2 List of Registers of the Adaptive Voltage Scaling (AVS)**

| Register Name | Mnemonic | R/W | Offset Address | Initial value | Size | R-Car V4M |
|---|---|---|---|---|---|---|
| Adaptive Dynamic Voltage ADJust Parameter (non-AVS) | ADVADJP | R | 0x080 | 0x0000 0000 | 32 | √ |
| Adaptive Dynamic Voltage ADJust Parameter (AVS) | ADVADJP | R | 0x080 | 0x0000 0XXX | 32 | √ |

Note: Please do not access other than the above address.

(include Safety Mechanism related registers)

### 12.1.5 Connected Module

Table 12.3 is a list of the modules that are connected to this module.

**Table 12.3 Connected Modules**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| AVS | AP-system core | System CPU core |
| | CNN | Convolution neural network engine |
| | FUSE | For ADVADJP |

## 12.2 Register Description

[Legend for register description]

| | |
|---|---|
| Initial value: | Register value after a reset |
| —: | Undefined value |
| R/W: | Readable/writable. The written value can be read. |
| R: | Read-only. Writing is prohibit. |
| R/WC0: | Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored |
| R/WC1: | Readable/writable. Writing 1 initializes the bit. Writing 0 is ignored. |
| W: | Write-only. Reading this bit is prohibited. When the bit is reserved, the write value should always be 0. |
| __/W: | Write-only. The read value is undefined. |

### 12.2.1 Adaptive Dynamic Voltage ADJust Parameter (ADVADJP)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | VOLCOND[8:0] | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved (AVS)<br>These bits are always read as 0. Writing is prohibited. |
| 8 to 0 | VOLCOND[8:0] | * | R | AVS code<br>Initial values are derived from Fuse. |

Note *: Table 12.4 VOLCOND[8:0] shows the initial value of this register.

### 12.2.2 VOLCOND[8:0]

The Table 12.4 is a list of AVS code.
Refer to “Power supply employing AVS”, which describes the voltage value corresponding to each AVS code.

**Table 12.4 VOLCOND[8:0] Table**

| | VOLCOND | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **[8]** | **[7]** | **[6]** | [5] | [4] | [3] | [2] | [1] | [0] |
| Non-AVS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AVS0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| AVS1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| AVS2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

Notes: Other values should be handled as Non-AVS.
x: undefined (internal code)

(include Safety Mechanism related register description)

## 12.3 Operation

### 12.3.1 AVS code

(non-AVS product) The register of ADVADJP are always 0.

AVS code, which cannot be changed, is determined by shipment test and written as Fuse values.

### 12.3.2 Register-AVS

The voltage of VDD_xx can be adjusted manually by using register values of AVS code after boot. That should be completed before AP-system core and CNN is started running.

## 12.4 Block Diagram

AVS does not connect peripheral module.

## 12.5 Interface specification

### 12.5.1 Clock and Reset Implementation

Description item information: Name the used in UM, Signal net name, Max frequency and Condition of reset

Reset

| Pin name | Net name | Feedback Reset Pin name | Feedback Reset net name |
|---|---|---|---|
| async_rst | x_apreset_advfsc_c4_p | async_rst_out | advfsc_apresetst_cl |
| | | | |

Clock

| Name | Type of Clock | Pin name | Net name | Frequency |
|---|---|---|---|---|
| CPϕ | Bus clock | clk | x_cl16mck_top | CL16Mφ |
| | | | | |

Clock limitations(Frequency): CPϕ > RCLK

Table 12.5
.2 Inter-module clock true/false table

| From \ To | ACLK | MCLK | SDCLK* | PCLK |
|---|---|---|---|---|
| ACLK | Sync | Async | - | - |
| MCLK | Async | Sync | Sync | Async |
| SDCLK* | - | Sync | Sync | - |
| PCLK | - | Async | - | Sync |

- : N/A

# 13.Thermal Sensor/Chip Internal Voltage Monitor/Core Voltage Monitor (THS/CIVM/CVM)

## 13.1 Overview

This LSI provides a thermal sensor module that measures the temperature (Tj) inside the LSI. The thermal sensor module also includes a chip internal voltage monitoring module that measures voltage supply inside the LSI.

### 13.1.1 Features

#### 13.1.1.1 THS Features

**(1) Each of the thermal sensors measures temperature Tj**

**(2) R-Car V4M provides two thermal sensor modules (THS1/CIVM1, THS2/CIVM2)**

**(3) This module generates errors when the detected temperature Tj within the LSI rises above or falls below a specified temperature**

**Table 13.1 Placed Point of THS**

| THS | Placed point | Rationale of location |
|---|---|---|
| THS1 | around Cortex-R52 | CPU Core 0 is candidate of hotspot because usually it is used as primary core in cluster |
| THS2 | around Cortex-A76 | CPU Core 0 is candidate of hotspot because usually it is used as primary core in cluster. |

#### 13.1.1.2 CIVM Features

**(1)** Each of the chip internal voltage monitors in the thermal sensor module measures voltage supply inside the LSI

**(2) R-Car V4M monitors VDD voltage supply**

**Table 13.2 Monitoring Point of CIVM**

| CIVM | Monitoring Point | Monitor Voltage | Power shut-off control |
|---|---|---|---|
| CIVM1 | VDD around Coretex-R52 | VDD | None (always on) |
| CIVM2 | VDD around Coretex-A76 | VDD | None (always on) |

#### 13.1.1.3 CVM Features

**(1) This module detects the lower or upper value of VDD's voltage and notifies the state thereof by using the VTHREF0 and VTHSENSE0 pins or by reading the CVM_VMOUT_BK register**

**(2) R-Car V4M provides two CVM modules (TSC1/CVM1, TSC2/CVM2)**

TSC : Thermal Sensor Controller

**Table 13.3 Monitoring Point of CVM**

| CVM | Monitoring Point | Monitor Voltage | Power shut-off control |
|---|---|---|---|
| CVM1 | VDD around Coretex-R52 | VDD | None (always on) |
| CVM2 | VDD around Coretex-A76 | VDD | None (always on) |

### 13.1.2 Block Diagram

Figure 13.1 shows a block diagram of the thermal sensor modules. Each thermal sensor module consists of a thermal sensor (THS), a chip internal voltage monitor (CIVM), a core voltage monitor (CVM) and a thermal sensor controller (TSC). THS measures Tj temperature, and CIVM measures a voltage. CVM detects the lower or upper value of VDD's voltage. TSC controls THS, CIVM and CVM.

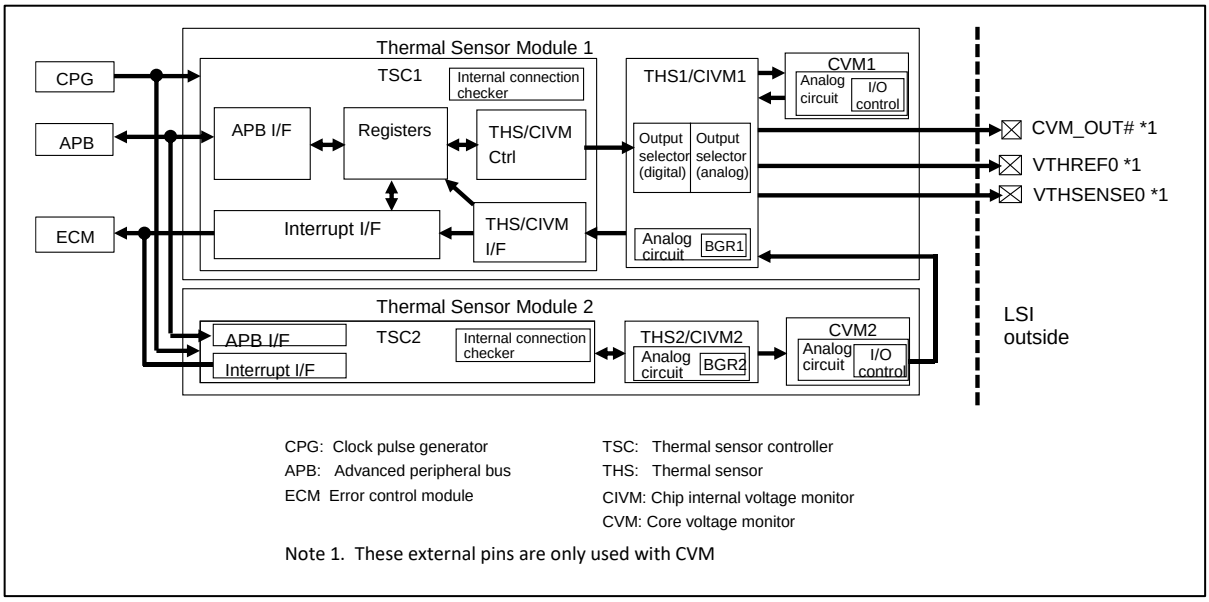



**Figure 13.1 Block Diagram of Thermal Sensor Modules**

### 13.1.3 External Pins

Table 13.4 is a list of external pins. These external pins are only used with CVM. For the processing of these pins, refer to Usage Note. The external pins detail are as follows：

-Digital Pin 1 pin, Core Supply Voltage Failure Detection.

-Analog Pin 2 pin, Overvoltage, Undervoltage, Illegal, Notification

-No state retention

**Table 13.4 Pin Configuration**

| Pin Name | I/O | Function |
|---|---|---|
| VTHSENSE0 | Analog I/O | VDD undervoltage indicator |
| VTHREF0 | Analog I/O | VDD overvoltage indicator |
| CVM_OUT# | Digital output | VDD anomaly(undervoltage or overvoltage) indicator |

For noise reduction, connect the capacitor on the VTHREF0/VTHSENSE0 pins on the board as shown in Figure 13.2. Connect C1 and C2 to VSS (GND) at one point. However, if there are no noise concerns or if noise reduction is considered in other ways, these capacitances are not mandatory. If the routing to the connection destination of VTHREF0 and VTHSENSE0 runs in parallel wiring (like differential wiring) C1 and C2 are not always necessary.

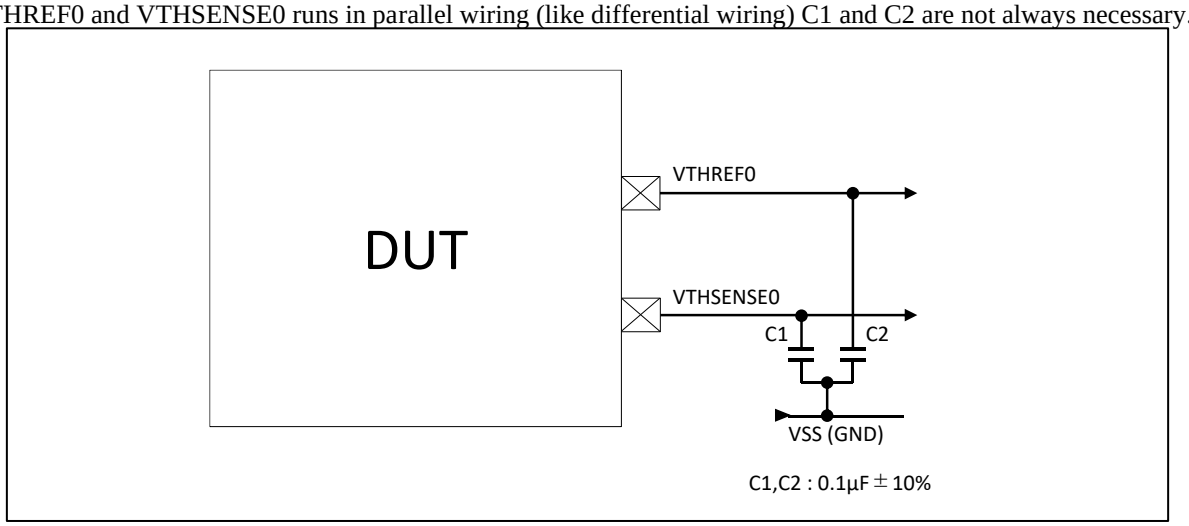



**Figure 13.2 Connect capacitors to Analog Pin**

### 13.1.4 Register configuration

Please refer to R-CarV4M_UM_013_THS_CIVM_CVM_r0p70.xlsx.

### 13.1.5 Connected Module

Table 13.5 is a list of the modules that are connected to this module.

**Table 13.5 Connected Modules**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| THS/CIVM | APB | Controls register accesses |
| | CPG | Outputs clocks |
| | ECM | Error Control module |
| | Module standby | Stops clocks |
| | Software reset | Execute software reset |

## 13.2 Register description

Please refer to R-CarV4M_UM_013_THS_CIVM_CVM_r0p70.xlsx.

## 13.3 Operation

### 13.3.1 Thermal Sensor/Chip Internal Voltage Monitor Operation Mode

Figure 13.3 shows the state transition diagram of Finite State Machine. To change the operating mode, set the register according to Table 13.6 How to Set Each Mode Entry. Switching the operating mode in a procedure not described in this table is prohibited.

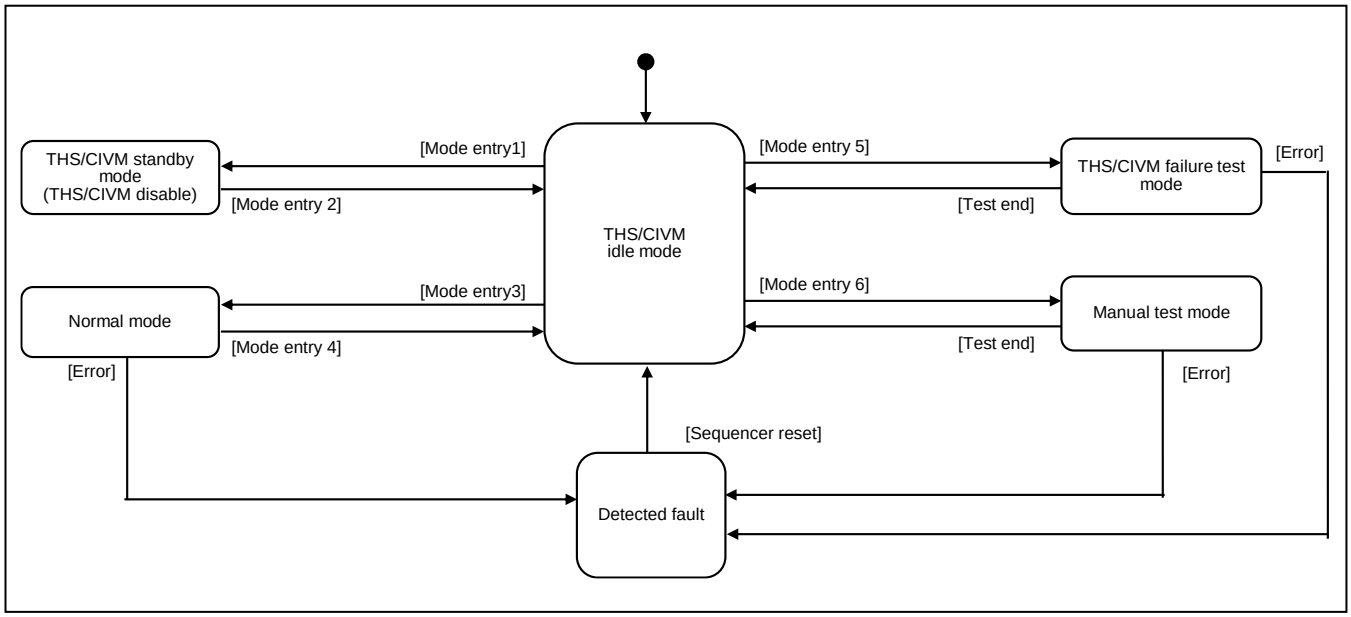



**Figure 13.3 THS/CIVM control FSM State Transition**

**Table 13.6 How to Set Each Mode Entry**

| Register | Mode Entry 1 | Mode Entry 2 | Mode Entry 3 | Mode Entry 4 | Mode Entry 5 | Mode Entry 6 |
|---|---|---|---|---|---|---|
| TH_EN_B | 0 -> 1 | 1 -> 0 | 0 | 0 | 0 | 0 |
| THSST | 0 | 0 | 0 -> 1 | 1 -> 0 | 0 | 0 |
| CIVMTST | 0 | 0 | 0 | 0 | 0 -> 1 | 0 |
| MANTST | 0 | 0 | 0 | 0 | 0 | 0 -> 1 |
| CTRSEL | 0 | 0 | 0 | 0 | 0 | 0 |

### 13.3.2 THS Idle Mode

The initial state after the reset is THS idle mode. This mode must be maintained for at least 100 [us] until the BGR stabilizes after power-on.

### 13.3.3 Normal Mode

Normal mode of thermal sensor module outputs the digital codes of temperature and voltage which are measured by thermal sensor module at a constant period. The measured digital code can be monitored with the TEMP_CODE and VOLT_CODE registers.

### 13.3.3.1 THS/CIVM operation

The THS operation uses an A/D converter to obtain the sensing voltage from the thermal sensor. The CIVM operation uses the same A/D converter to get the chip internal voltage. Repeat THS operation, CIVM operation, or both during Normal mode.THS operation and CIVM operation are enabled by register configuration. If both operations are enabled, the acquisition time is increased because the A/D converter is used alternately.

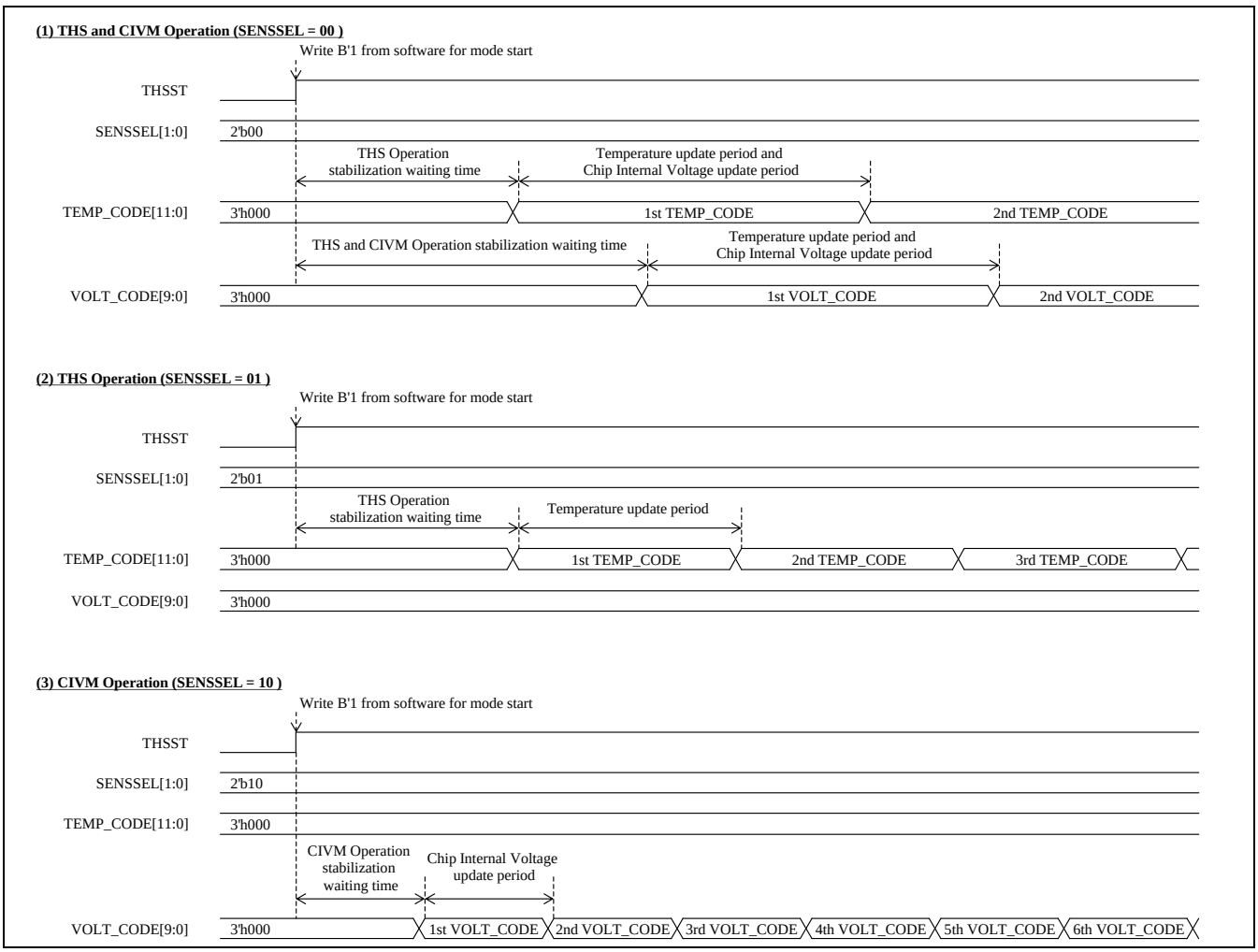



**Figure 13.4 THS/CIVM operation timing chart**

### 13.3.3.2 Temperature change detection

Temperature change can be detected during Normal mode. When the measured value crosses the set temperature threshold, an error is notified to the external module. After starting normal mode, you should check that the first measurement value is within the threshold temperature range.
The block diagrams of temperature error are shown in Figure 13.5. The temperature threshold for thermal sensor module has three (Thermal Sensor ch1, Thermal Sensor ch2 and Thermal Sensor ch3). The temperature change errors for TSC1 and TSC2 can detect to exceed (TEMP1, TEMP2 and TEMP3) or to fall below (TEMPD1, TEMPD2 and TEMPD3), and the error signal of its module is set to 1 by is detected each of TEMP or TEMPD.

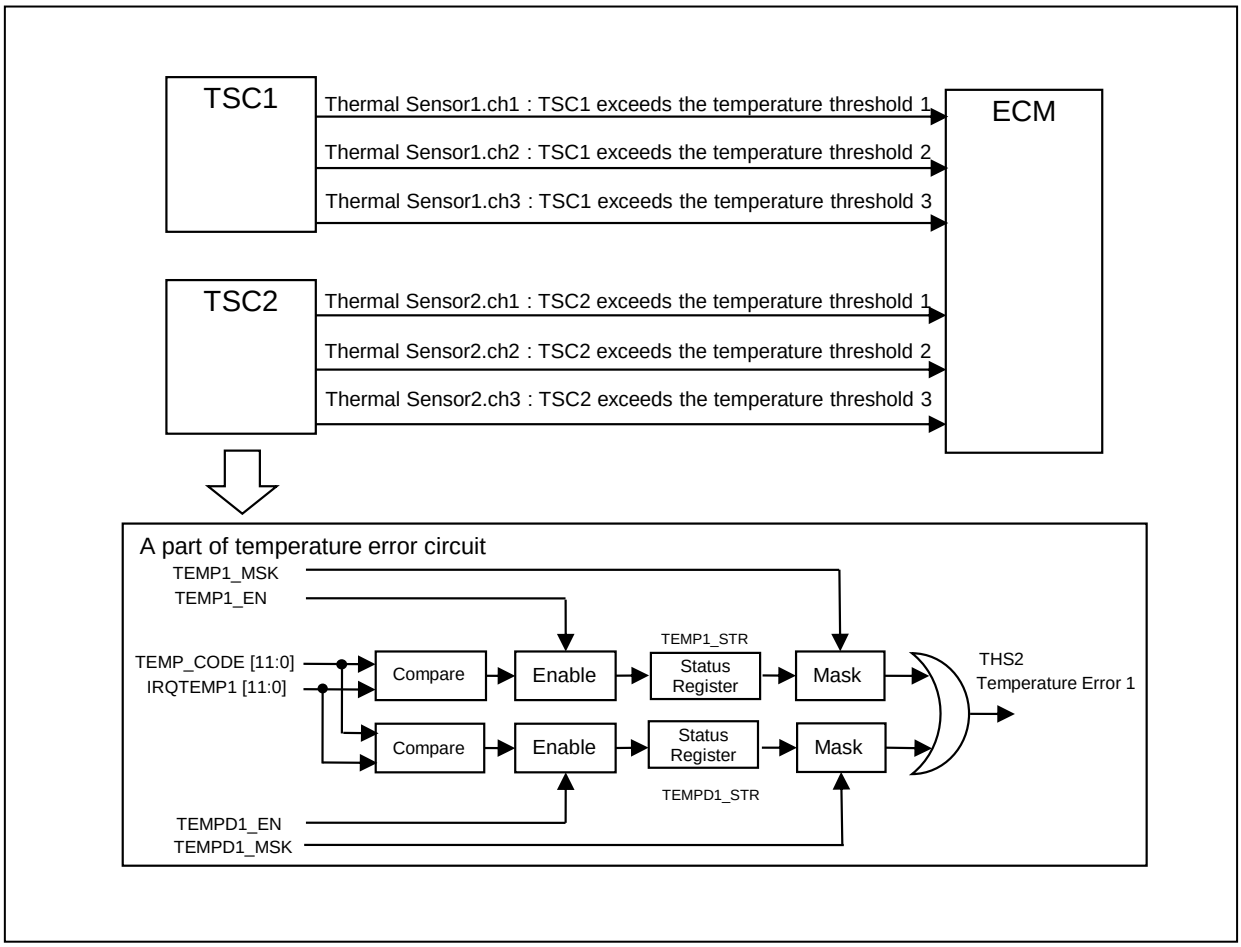



**Figure 13.5 Block Diagram of Temperature Error**

#### 13.3.3.3 Setting of Normal Mode

Figure 13.6 and Figure 13.7 show a flowchart of the normal mode setting.

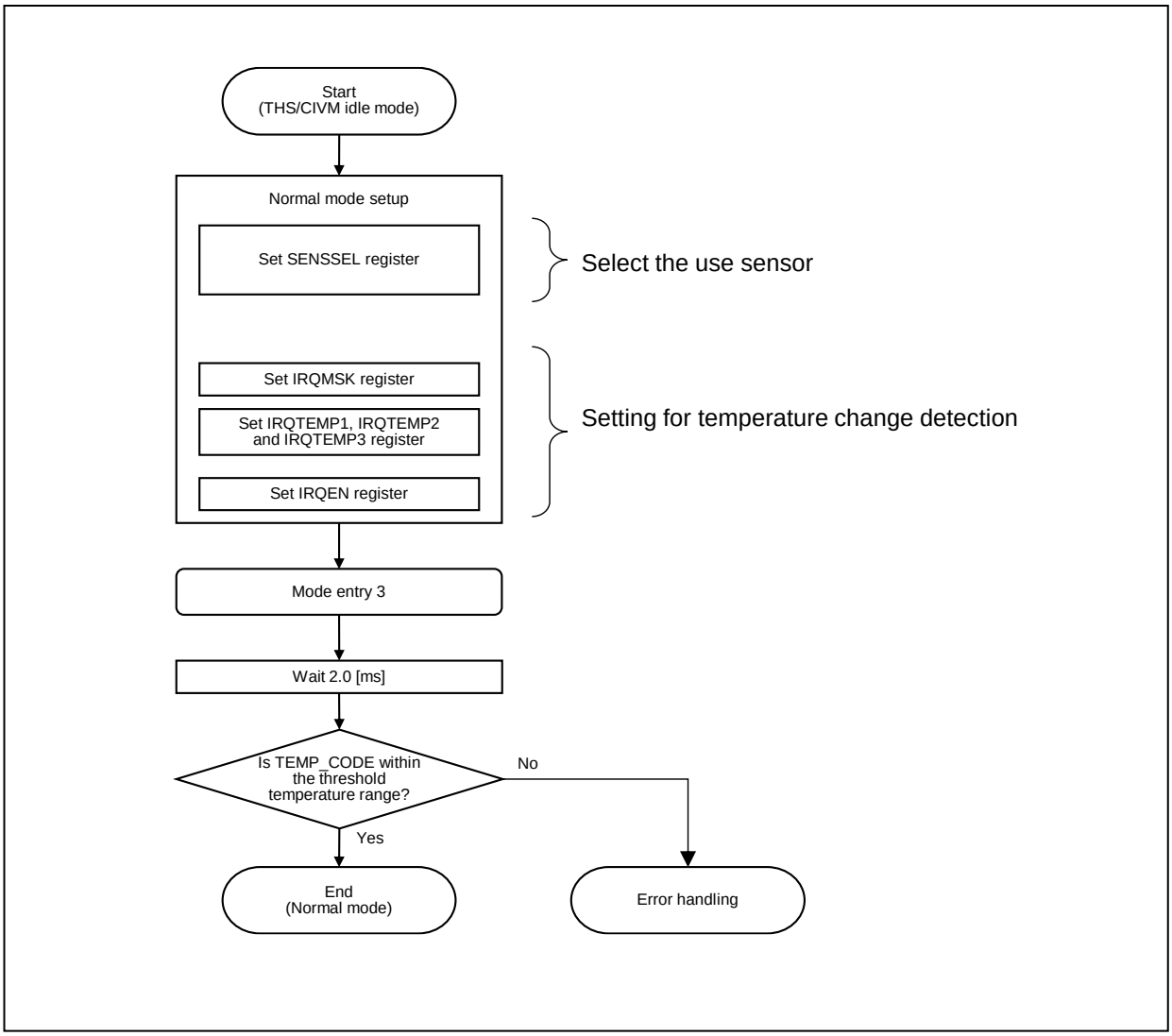



**Figure 13.6 Normal mode start setting**

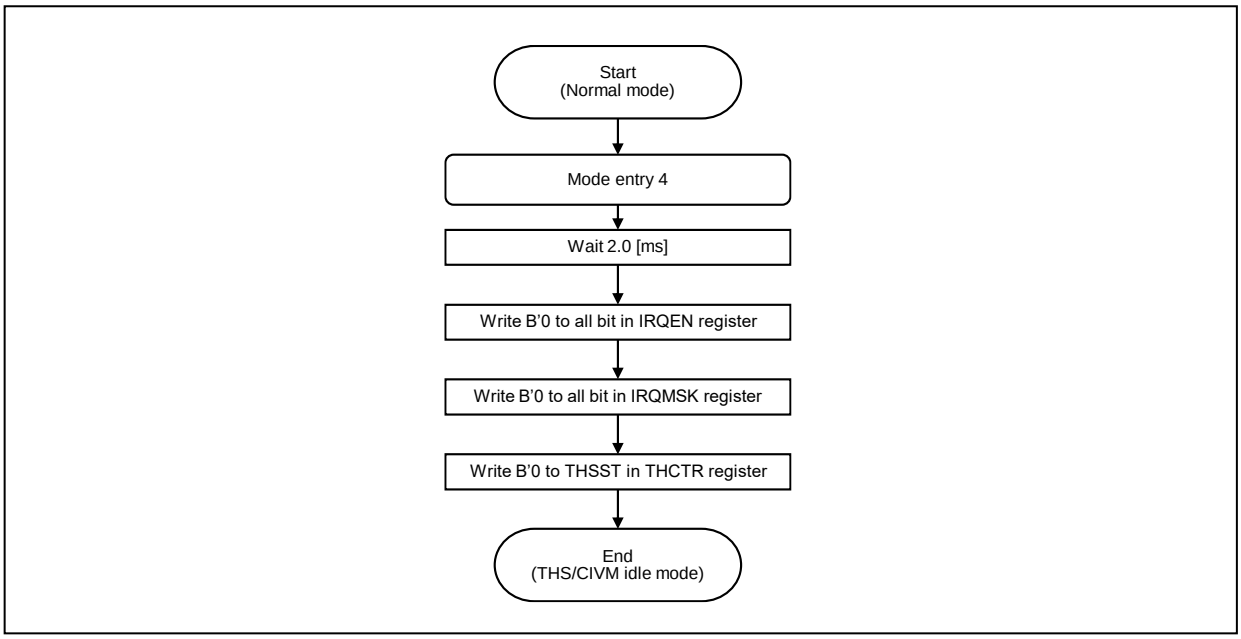



**Figure 13.7 Normal mode stop setting**

#### 13.3.3.4 The conversion formula for Thermal Sensor

The temperature conversion formula is shown below.

When TEMP_CODE <THCODE2[11:0]

$$\text{Temperature}[°\text{C}]\{\text{dec}\} = \frac{167\{\text{dec}\}\times(\text{PTAT3}[11:0]-\text{PTAT2}[11:0])(\text{THCODE3}[11:0]-\text{TEMP_CODE}[11:0])}{(\text{PTAT3}[11:0]-\text{PTAT1}[11:0])(\text{THCODE3}[11:0]-\text{THCODE2}[11:0])} - 41\{dec\}$$

When TEMP_CODE ≥THCODE2[11:0]

$$\text{Temperature}[°\text{C}]\{\text{dec}\} = \frac{167\{\text{dec}\}\times(\text{PTAT1}[11:0]-\text{PTAT2}[11:0])(\text{THCODE1}[11:0]-\text{TEMP_CODE}[11:0])}{(\text{PTAT1}[11:0]-\text{PTAT3}[11:0])(\text{THCODE2}[11:0]-\text{THCODE1}[11:0])} + 126\{dec\}$$

{dec} is decimal value. The other terms are hex value.

The values of the terms PTAT1, PTAT2, PTAT3, THCODE1, THCODE2, and THCODE3 contained in the above conversion formula are stored in registers. The correspondence between terms and registers is shown in the table below.

| Formula term | Register bit name |
|---|---|
| PTAT1 | PTAT_PF_U_SR1 |
| PTAT2 | PTAT_PF_R_SR1 |
| PTAT3 | PTAT_PF_L_SR1 |
| THCODE1 | THCODE_U_SR1 |
| THCODE2 | THCODE_R_SR1 |
| THCODE3 | THCODE_L_SR1 |

#### 13.3.3.5 The conversion formula for Chip Internal Voltage monitor

The chip internal voltage conversion formula is shown below.

$$\text{Voltage[mV]}\{\text{dec}\} = 1.39\{\text{dec}\} \times \text{VOLT_CODE[9:0]} - 131\{\text{dec}\}$$

{dec} is decimal value. The other terms are hex value.

#### 13.3.3.6 The conversion formula for temperature error setting of temperature

IRQTEMP1 to 3[11:0] calculate by using below.

First, calculate Tj_T. It is

$$\text{Tj_T}\{\text{dec}\} = \frac{(\text{PTAT2[11:0]} - \text{PTAT3[11:0]}) \times 167\{\text{dec}\}}{(\text{PTAT1[11:0]} - \text{PTAT3[11:0]})} - 41\{\text{dec}\} \approx 20 \text{ to } 60\ [^{\circ}\text{C}]\{\text{dec}\}$$

Note : 20 to 60 [°C]{dec} is an indication of the range of Tj_T.

Next, calculate IRQTEMP1 to 3[11:0].

If setting temperature, STEMP, is less than or equal to Tj_T,

$$\text{IRQTEMP1 to 3[11:0]} = \text{THCODE3[11:0]} - \frac{(\text{STEMP} + 41\{\text{dec}\})(\text{PTAT3[11:0]} - \text{PTAT1[11:0]})(\text{THCODE3[11:0]} - \text{THCODE2[11:0]})}{167\{\text{dec}\} \times (\text{PTAT3[11:0]} - \text{PTAT2[11:0]})}.$$

If setting temperature, STEMP, is more than Tj_T,

$$\text{IRQTEMP1 to 3[11:0]} = \text{THCODE1[11:0]} - \frac{(\text{STEMP} - 126\{\text{dec}\})(\text{PTAT1[11:0]} - \text{PTAT3[11:0]})(\text{THCODE2[11:0]} - \text{THCODE1[11:0]})}{167\{\text{dec}\} \times (\text{PTAT1[11:0]} - \text{PTAT2[11:0]})}.$$

STEMP : temperature error setting of temperature $[^{\circ}\text{C}]\{\text{dec}\}$

{dec} is decimal value. The other terms are hex value.

The values of the terms PTAT1, PTAT2, PTAT3, THCODE1, THCODE2, and THCODE3 contained in the above conversion formula are stored in registers. See section 13.3.3.4 for the correspondence between terms and registers.

### 13.3.4 THS/CIVM Standby Mode

Figure 13.8 shows a flowchart of the THS/CIVM standby mode setting. This mode disables THS/CIVM.

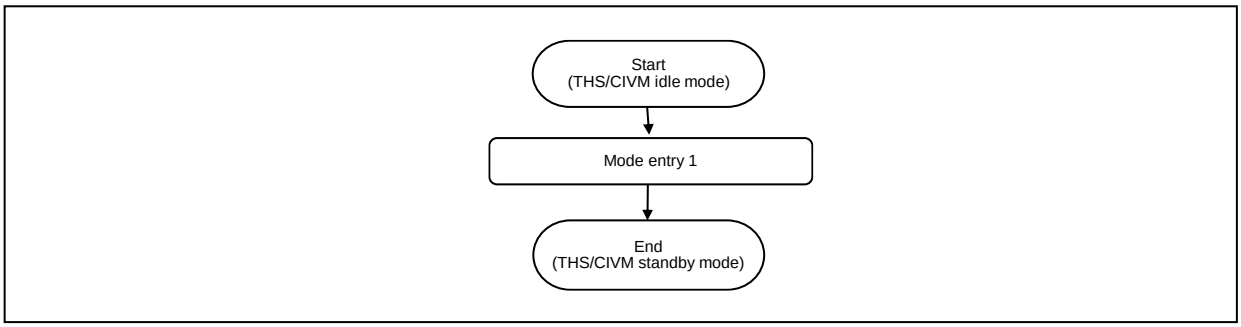



**Figure 13.8 standby mode setting**

Figure 13.9 shows a flowchart of idle setting from THS/CIVM standby mode. In case of transition from THS / CIVM standby mode to idle mode, wait for 100 [us] or more and then move to the next mode (normal mode, etc.).

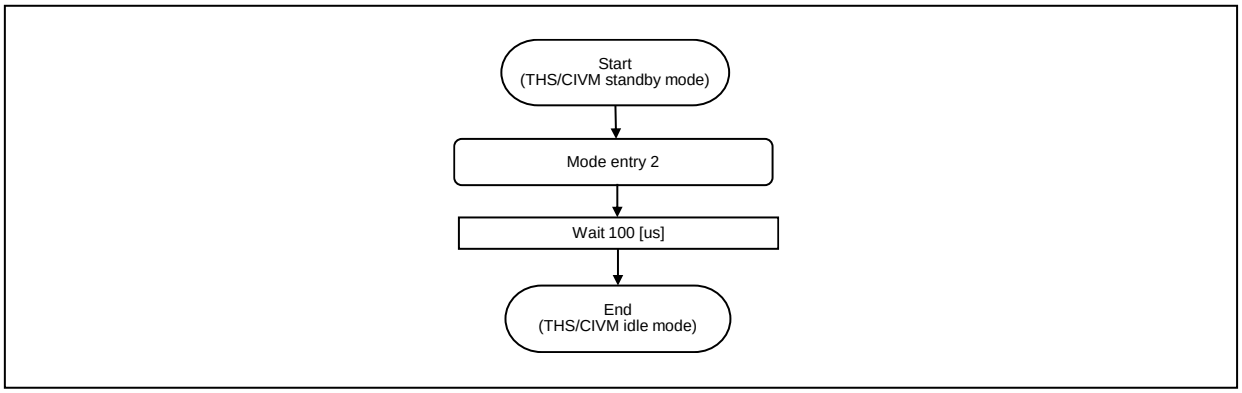



**Figure 13.9 Setting from standby mode to idle mode**

### 13.3.5 Thermal Sensor Module Standby Mode

This module supports Module Standby Mode. Module Standby Mode of TSC1 and TSC2 are controlled together. FSM must be set to THS/CVM idle mode or THS/CVM standby mode before the module can be set to Module Standby Mode. In the restart method, clear Module Standby register, execute a software reset. Refer to “Module Standby, Software Reset” section for the detail of setting method.

### 13.3.6 Software Reset

This module supports software reset. Software reset of TSC1 and TSC2 are controlled together. Refer to “Module Standby, Software Reset” section for the detail of setting method. FSM must be set to THS/CVM idle mode or THS/CVM standby mode before software reset. Several registers are not cleared by software reset.

## 13.4 Usage Notes

(1) When TH_EN_B of THCTR register set from 1 to 0, control register after 100 [us]. During this time, the software will have to wait.

(2) Leave external LSI pins (VTHSENSE0 and VTHREF0) open.

(3) Use digital code (TEMP register and VOLT register) when measuring Tj and chip internal voltage.

(4) When Tj exceeds 125[°C], correct operation and device characteristics of this module are not guaranteed.

(5) It is reducted spike noise that is shorter than the following period in each feature.

THS Features : 1000[us]

CIVM Features : 500[us]

CVM Features (analog output) : 7000[us]

CVM Features (digital output) : Filter is implemented to reduce spike noise. See 200.6.6 section.

## 13.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 13.5.1 THS/CIVM Characteristics Check

#### 13.5.1.1 THS characteristic failure determination

This test determines the failure of the BGR reference(VREF) voltage for each THS and the failure of the A/D conversion value of the proportional to absolute temperature(PTAT) voltage relative to VREF.

Figure 13.10 shows a block diagram for characteristic failure determination. Each THS uses a resistor-divider to generate voltages equivalent to 126[°C] and -41[°C] from the BGR circuit's reference voltage, which can be A/D converted. In addition, the InV[B1] output of THS1 is supplied to THS2, and A/D conversion can be performed based on the VREF of each THS. Using these paths, the voltages equivalent to 126[°C] and -41[°C] generated by THS1 can be A/D converted at each THS, and the result can be used to detect BGR reference voltage failures.

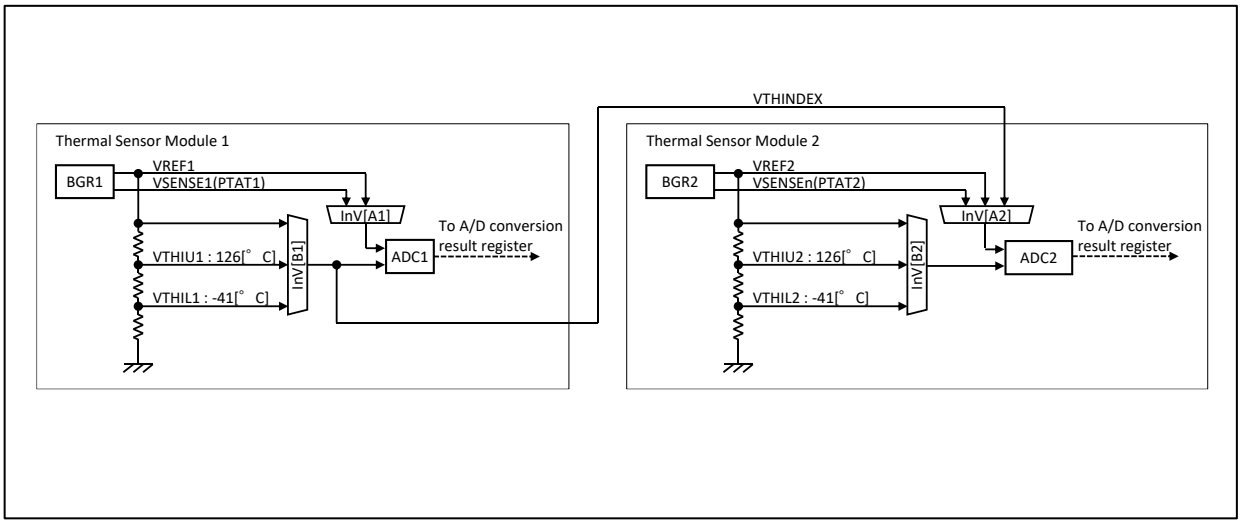



**Figure 13.10 Block diagram for characteristic failure determination**

The flowchart of the A/D conversion control necessary for BGR reference voltage failure determination is shown in Figure 13.11 and Figure 13.12. Figure 13.11 is a flowchart for A/D conversion with reference to the VREF voltage of THS1.

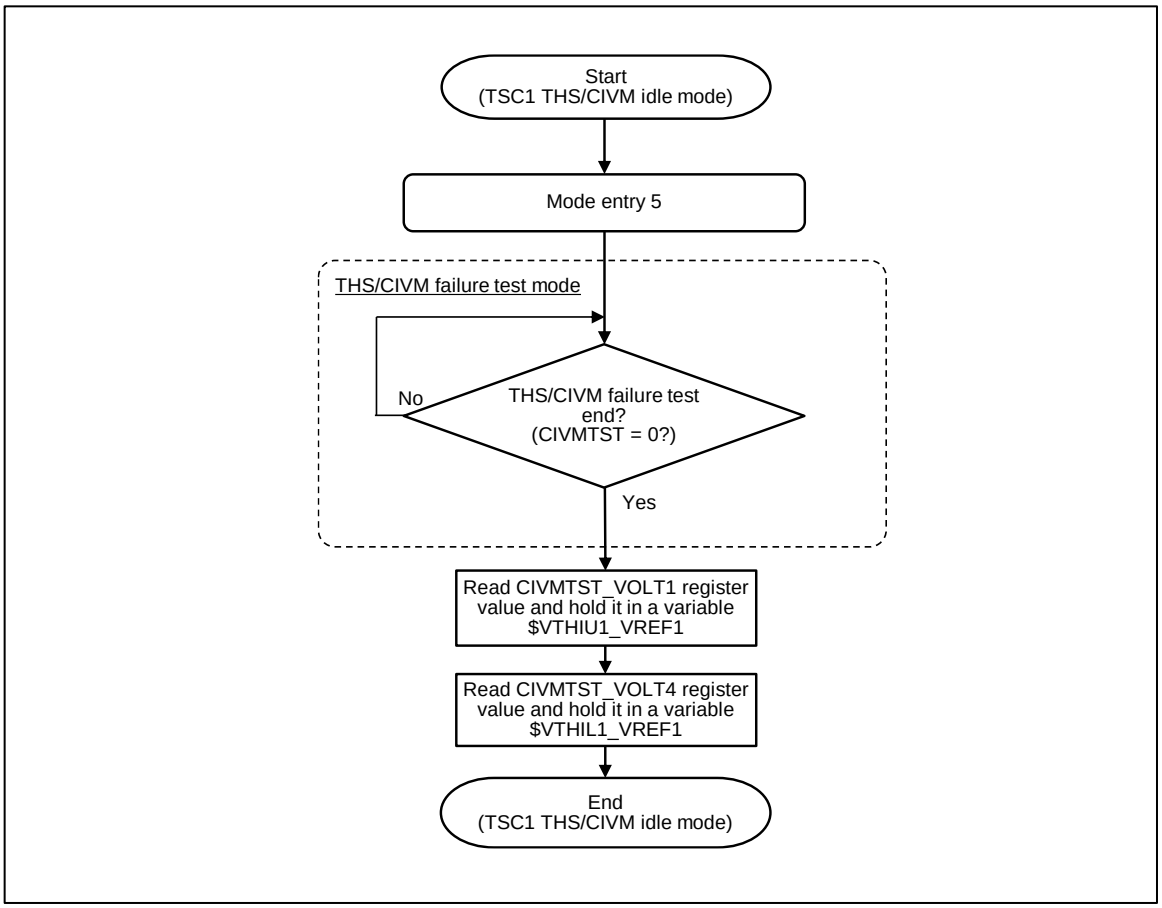



**Figure 13.11 A/D conversion flowchart for VREF voltage test(1/2)**

Figure 13.12 is a flowchart for A/D conversion with reference to the VREF voltage of each THS. All THS must work together for this A/D conversion. In order to fix the output of the InV[B1] of THS1, it is necessary to control by a different register setting than the FSM. During this time, do not allow the FSM of THS1 to transition to anything other than THS/CIVM idle mode. Each THS other than THS1 uses Manual mode to A/D conversion of the supplied test voltage.

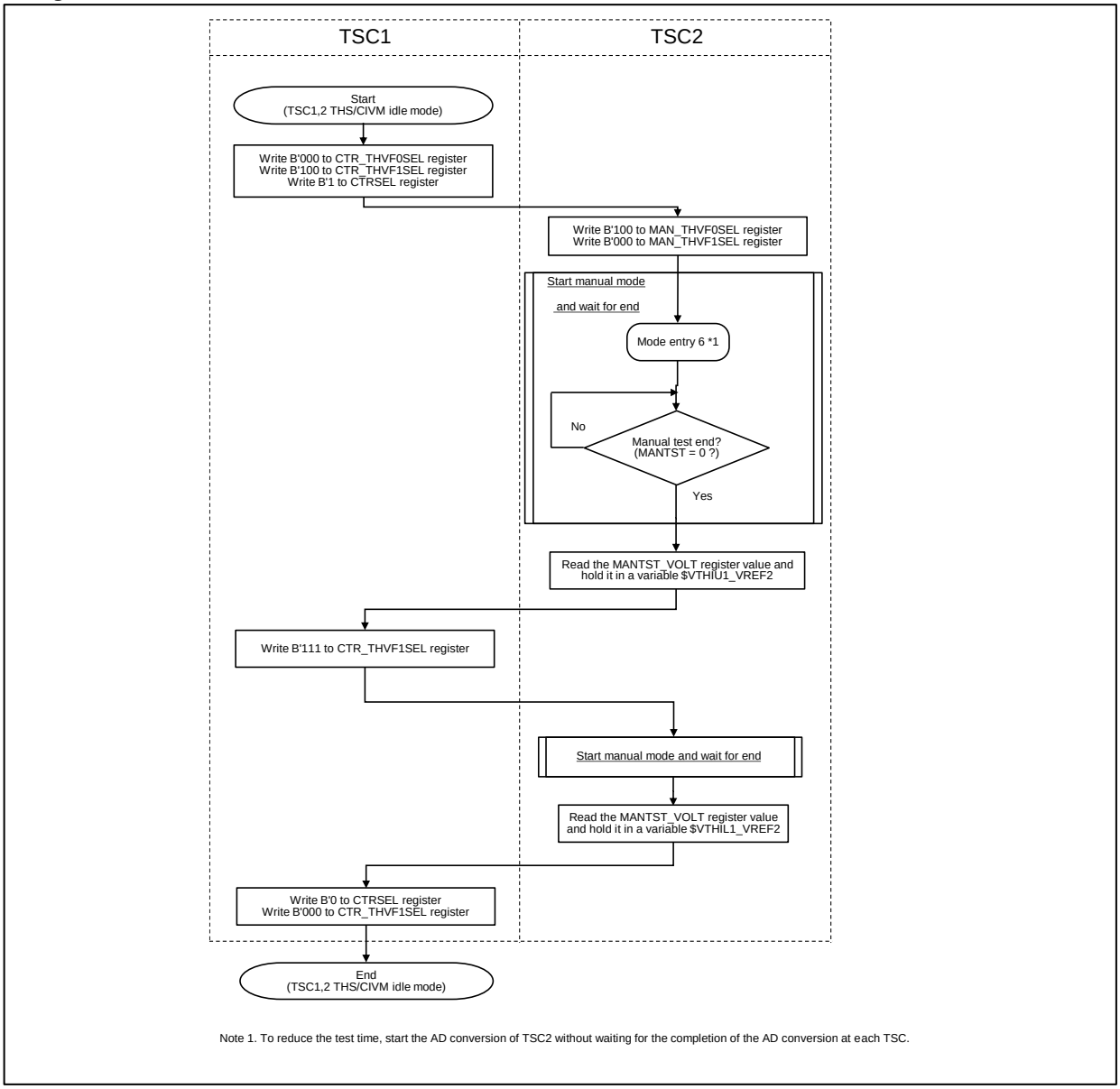



**Figure 13.12 A/D conversion flowchart for VREF voltage test(2/2)**

The determination formula is shown below. Under normal conditions, the following formula holds.

$$2800\{dec\} \leq \$VTHIU1_VREF1 \leq 3500\{dec\}$$

$$2800\{dec\} \leq \$VTHIU1_VREF2 \leq 3500\{dec\}$$

That is each value of $VTHIU1_VREF1, $VTHIU1_VREF2 is all in the range 2800{dec} to 3500{dec}.

$$1700\{dec\} \leq \$VTHIL1_VREF1 \leq 2200\{dec\}$$

$$1700\{dec\} \leq \$VTHIL1_VREF2 \leq 2200\{dec\}$$

That is each value of $VTHIL1_VREF1, $VTHIL1_VREF2 is all in the range 1700{dec} to 2200{dec}.

{dec} is decimal value.

Figure 13.13 shows a flowchart of the PTAT voltage failure determination. To determine the failure of the A/D conversion value of the PTAT voltage, the TEMP_CODE acquired by each THS is used.

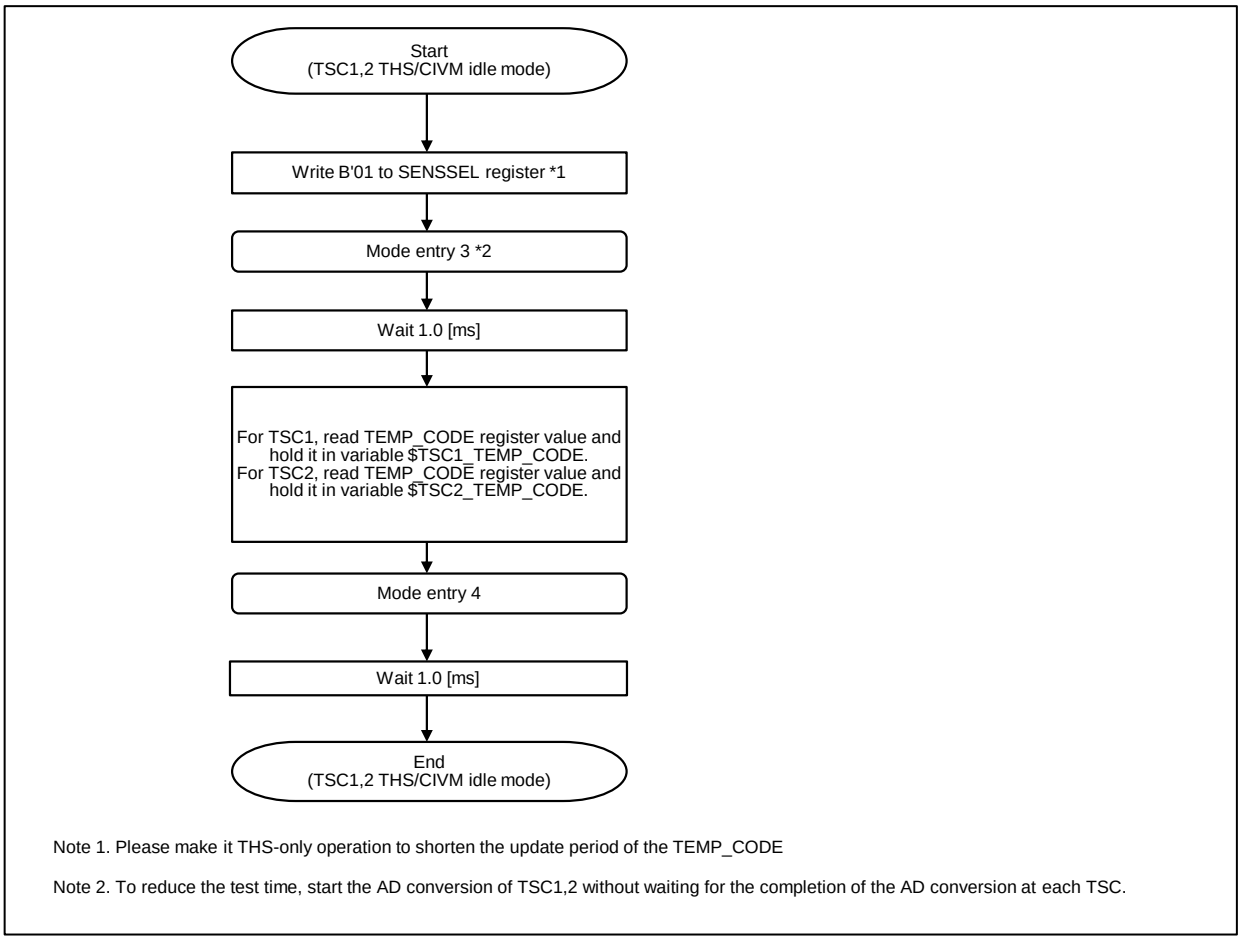



**Figure 13.13 PTAT voltage failure determination flowchart**

The determination formula is shown below.

Calculate the difference between $TSC1_TEMP_CODE and other TEMP_CODE.

$$\mathrm{DIF_TEMP_CODE2} = \$\mathrm{TSC2_TEMP_CODE} - \$\mathrm{TSC1_TEMP_CODE}$$

Under normal conditions, the following formula holds.

$$-325\{\mathrm{dec}\} \leq \mathrm{DIF_TEMP_CODE2} \leq 325\{\mathrm{dec}\}$$

{dec} is decimal value.

### 13.5.2 Fault Detection During Operation

Figure 13.14 shows a block diagram of thermal sensor module error. In addition to the temperature change detection shown in section 13.3.3.2, the thermal sensor controller notifies the external module of an error if fault is detected during operation. The detailed errors are as follows：

- ERR_ACKTOV : Digital code acquisition timeout (9.6[ms] or more)

- ERR_THCNTOV : Digital code counter overflow

- ERR_THCODE : Obtained illegal digital code (all 0 or all 1)

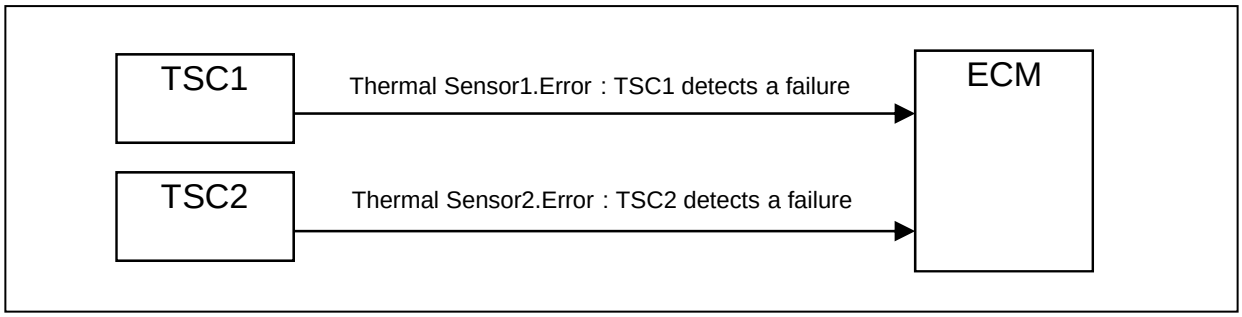



**Figure 13.14 Block diagram of thermal sensor module error**

### 13.5.3 Error Injection

Error injection function is implemented for errors output from TSC to ECM. Figure 13.15 shows a block diagram of the error injection function for temperature change detection. Figure 13.16 shows the error injection flowchart for temperature change detection(13.3.3.2 section).

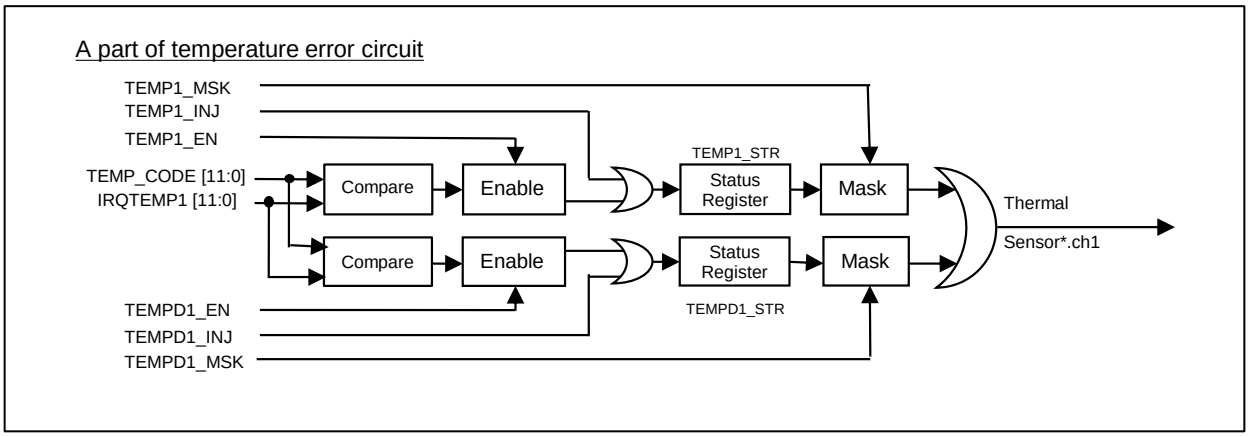



**Figure 13.15 Temperature change detection error injection block diagram**

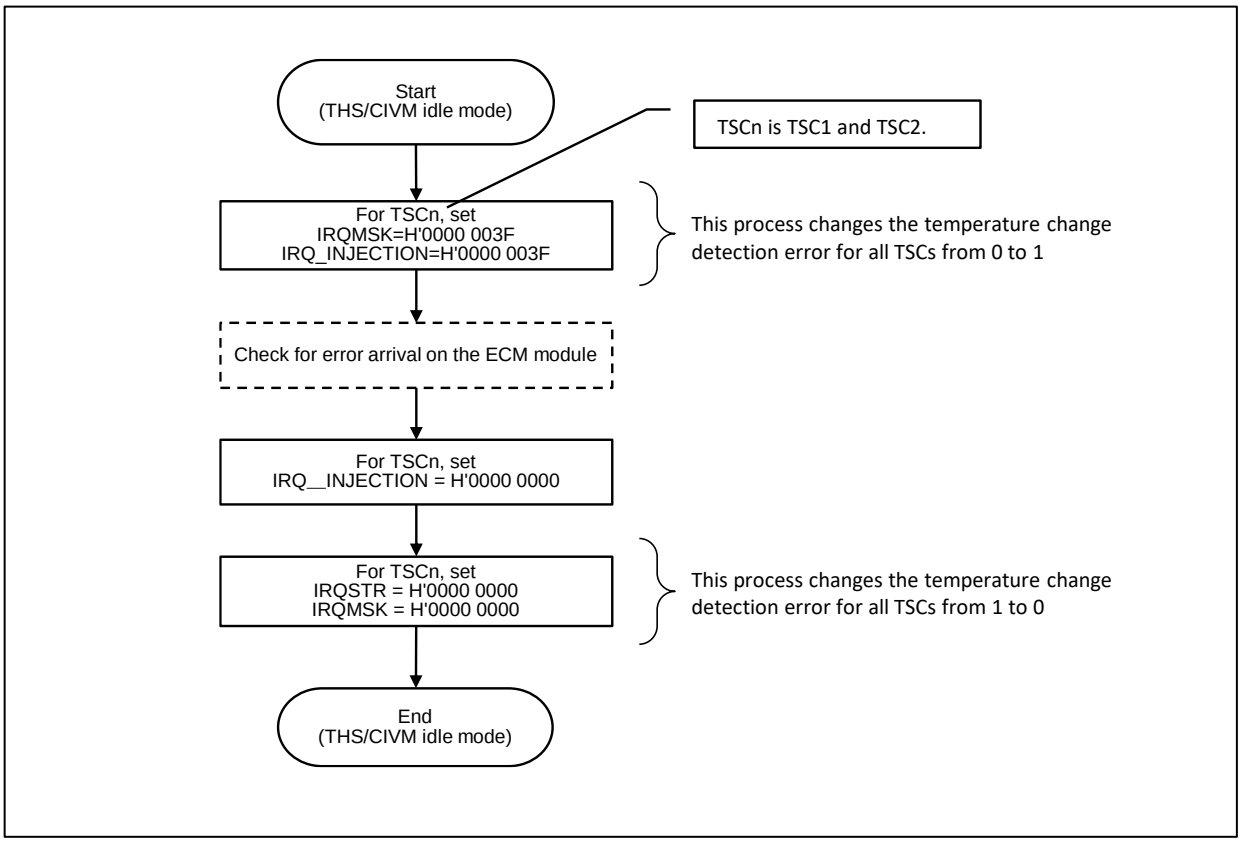



**Figure 13.16 Temperature change detection error injection flowchart**

Figure 13.17 shows a block diagram of the error injection function for failure detection error. Figure 13.18 shows error injection flowchart for failure detection error (13.5.2 section).

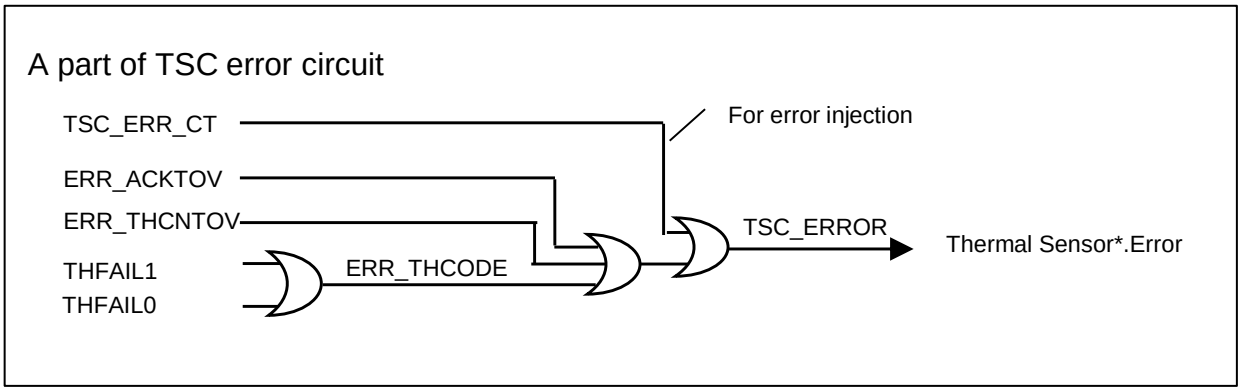



**Figure 13.17 Failure detection error injection block diagram**

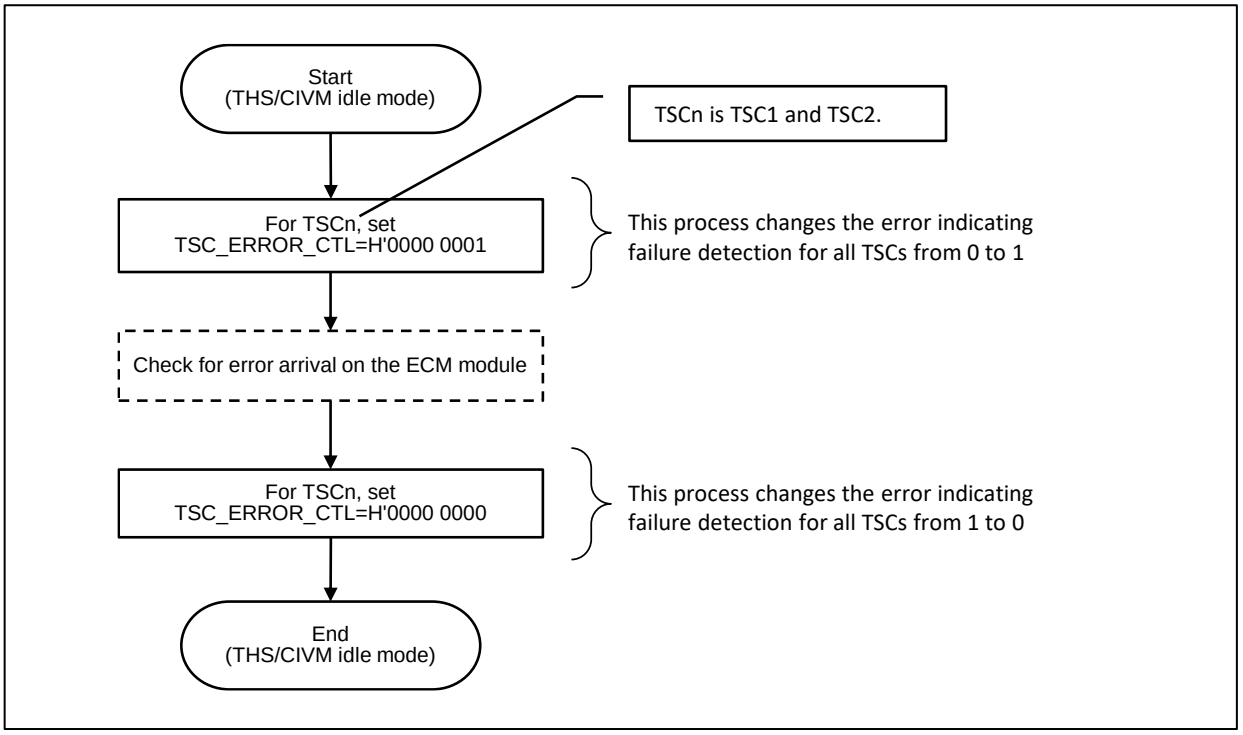



**Figure 13.18 Failure detection error injection flowchart**

### 13.5.4 Core Voltage Monitor

Figure 13.19 and Figure 13.20 depicts the block diagram of CVMs implemented in R-Car V4M. R-Car V4M has two CVMs monitoring distant positions on the same net. Each CVM compares the potential of the monitoring point with its own Bandgap Reference, and it asserts error signals in case it is beyond the preprogrammed threshold (overvoltage) or it is below another preprogrammed threshold (undervoltage). The threshold configuration is copied to and latched in the VDDQ18 domain so that the VDD anomaly does not destroy the configuration. The threshold configuration can be changed from the time Power on Reset is released until the VM_LOCK register is set to 1. Setting VM_LOCK to 1 completely retains the threshold configuration latched to the VDDQ18 domain and cannot be changed until Power on Reset.

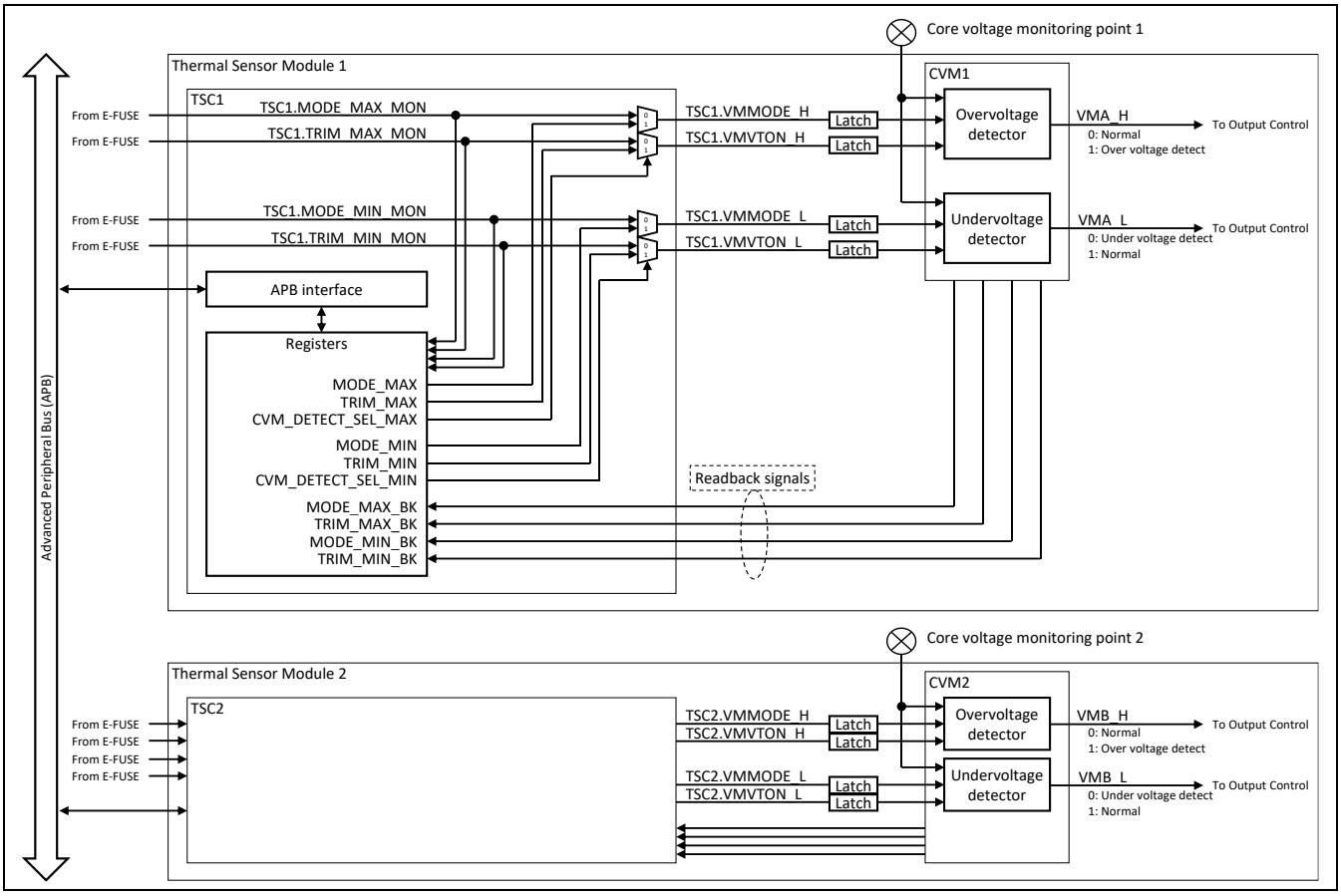



**Figure 13.19 Voltage Detector part of CVM Block Diagram**

The two CVMs shall be consolidated to deliver the error states.

Undervoltage and Overvoltage is respectively sent out to VTHREF0 pin and VTHSENSE0 pin. Core Supply Voltage Failure Detection is sent to CVM_OUT# pin. Table 13.7 shows the truth table of the external terminal output for CVM. The initial state (after Power on Reset release) is VMCHPL bits are all 0 (all CVMs are disable state), and CVM_OUT# output Low. Insert Low Pass Filter between Out Put Buffer and Digital External Pin.

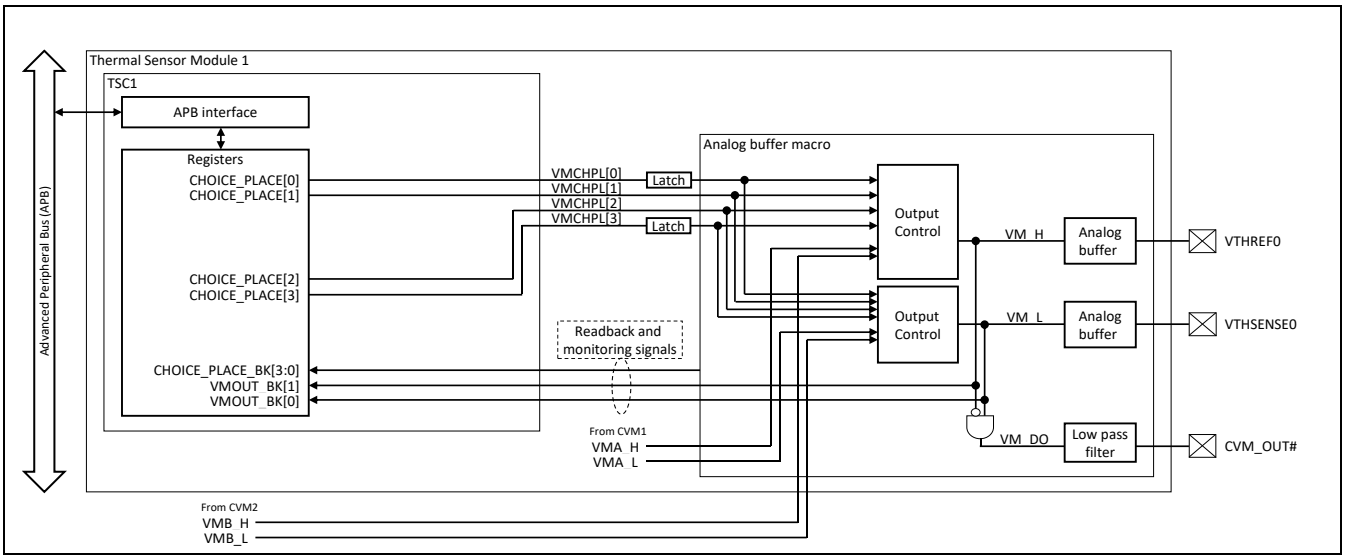



**Figure 13.20 Output Control part of CVM Block Diagram**

**Table 13.7 CVM external pins output truth table**

| Input | | Output | | | State |
|---|---|---|---|---|---|
| VM_L | VM_H | VM_DO (CVM_OUT#[V]) | VTHSENSE0[V] | VTHREF0[V] | |
| 0 | 0 | 0 (≈0) | ≈0 | ≈0.8 | Undervoltage |
| 0 | 1 | 0 (≈0) | ≈0 | ≈1.24 | Illegal |
| 1 | 0 | 1 (≈1.8) | ≈0.8 | ≈0.8 | Normal voltage |
| 1 | 1 | 0 (≈0) | ≈0.8 | ≈0 | Overvoltage |

**Table 13.8 CVM VM_H output control truth table**

| Input | | | | | | Output |
|---|---|---|---|---|---|---|
| VMCHPL | | | | VMA_H | VMB_H | VM_H |
| [0] | [1] | [2] | [3] | | | |
| 0 | 0 | 0 | 0 | * | * | 0 |
| 1 | 0 | 0 | 0 | 0 | * | 0 |
| | | | | 1 | * | 1 |
| 0 | 1 | 0 | 0 | * | 0 | 0 |
| | | | | * | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| | | | | 1 | * | 1 |
| | | | | * | 1 | 1 |
| * | * | 1 | * | * | * | 1 |
| * | * | * | 1 | * | * | 1 |

**Table 13.9 CVM VM_L output control truth table**

| Input | | | | | | Output |
|---|---|---|---|---|---|---|
| VMCHPL | | | | VMA_L | VMB_L | VM_L |
| [0] | [1] | [2] | [3] | | | |
| 0 | 0 | 0 | 0 | * | * | 0 |
| 1 | 0 | 0 | 0 | 0 | * | 0 |
| | | | | 1 | * | 1 |
| 0 | 1 | 0 | 0 | * | 0 | 0 |
| | | | | * | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | * | 0 |
| | | | | * | 0 | 0 |
| | | | | 1 | 1 | 1 |
| * | * | 1 | * | * | * | 0 |
| * | * | * | 1 | * | * | 0 |

#### 13.5.4.1 CVM failure detection test & CVM initial sequence

CVM failure detection test is shown Figure 13.21. Figure 13.22 to Figure 13.26 and Figure 13.28 show the predefined process of Figure 13.21. This test checks CVM function. In this test, the output of the external pins (CVM_OUT#, VTHSENSE0, VTHREF0) changes because the CVM is intentionally set to normal, illegal, undervoltage, and overvoltage states. Do not use external pins until the CVM configuration is complete.

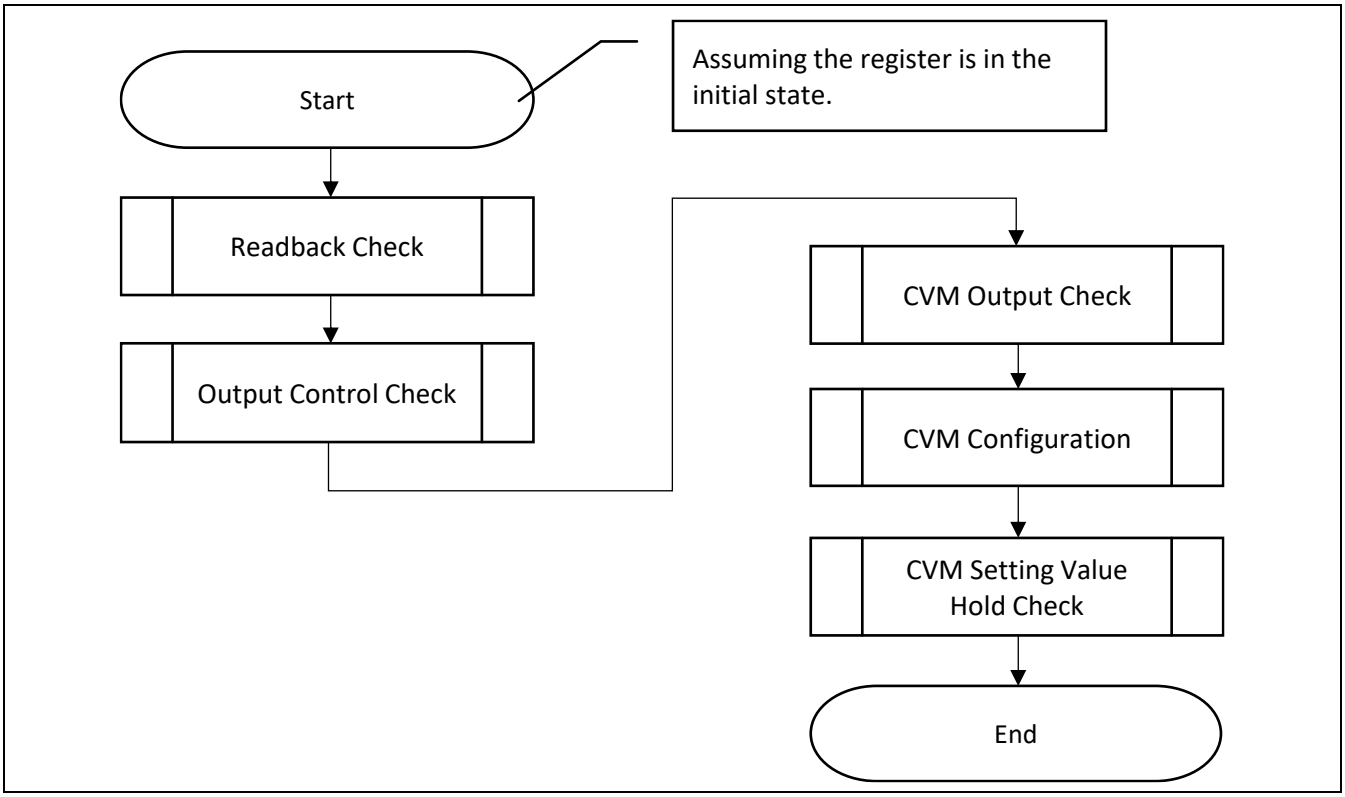



**Figure 13.21 CVM failure detection test**

Readback Check is shown Figure 13.22. This test uses the CVM Detection Voltage Manual Setting Register to comprehensively switch between the overvoltage and undervoltage thresholds to ensure that the settings and Readback values all match.

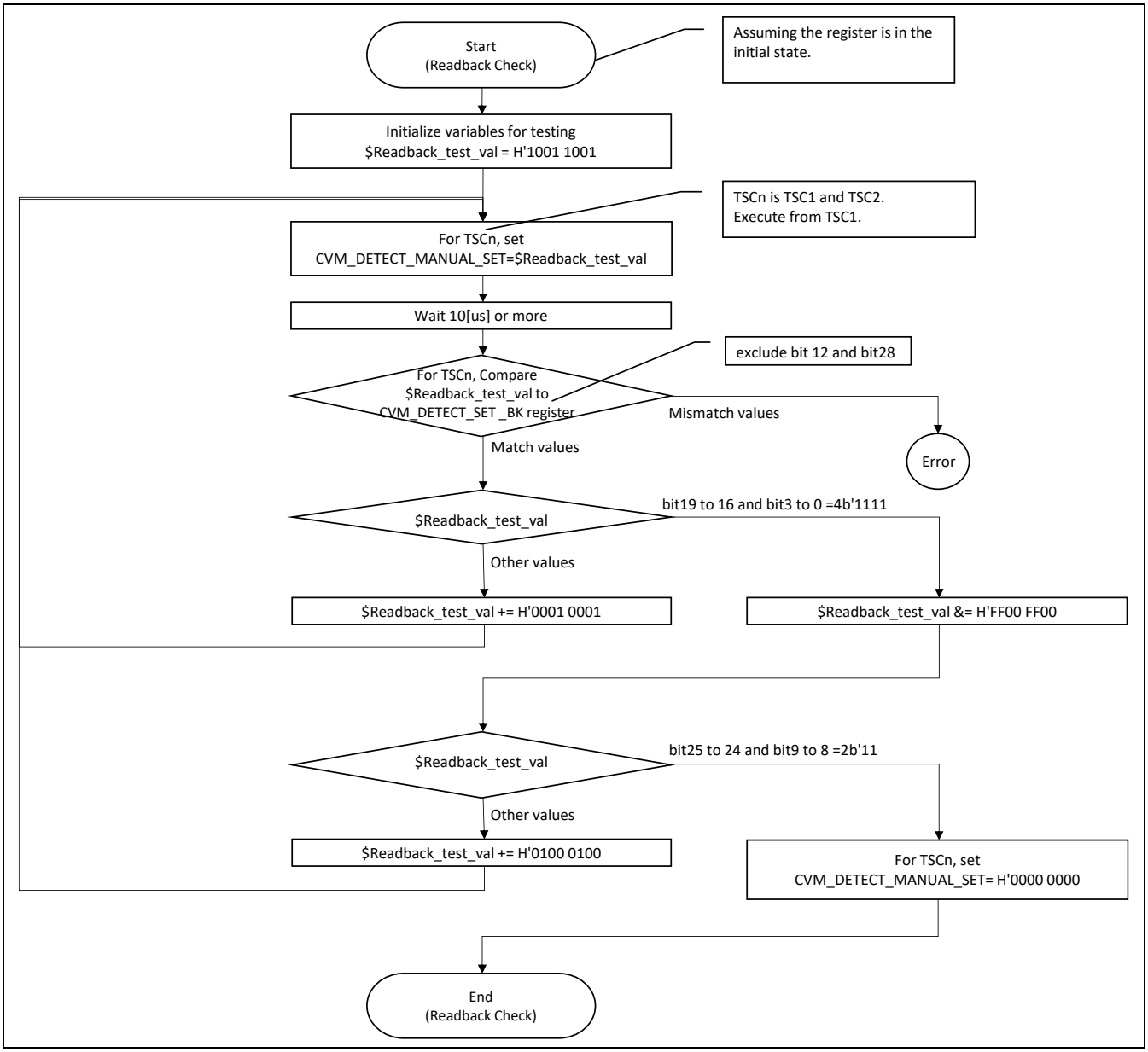



**Figure 13.22 Predefined process “Readback Check”**

The Output Control Check is shown Figure 13.23. This test changes the threshold voltage settings for each CVM to detect Overvoltage or Undervoltage, and verifies that only the CVM results selected by CHOICE_PLACE bits is combined (VM_H and VM_L) output. Table 13.10 shows the combination table used in this test.

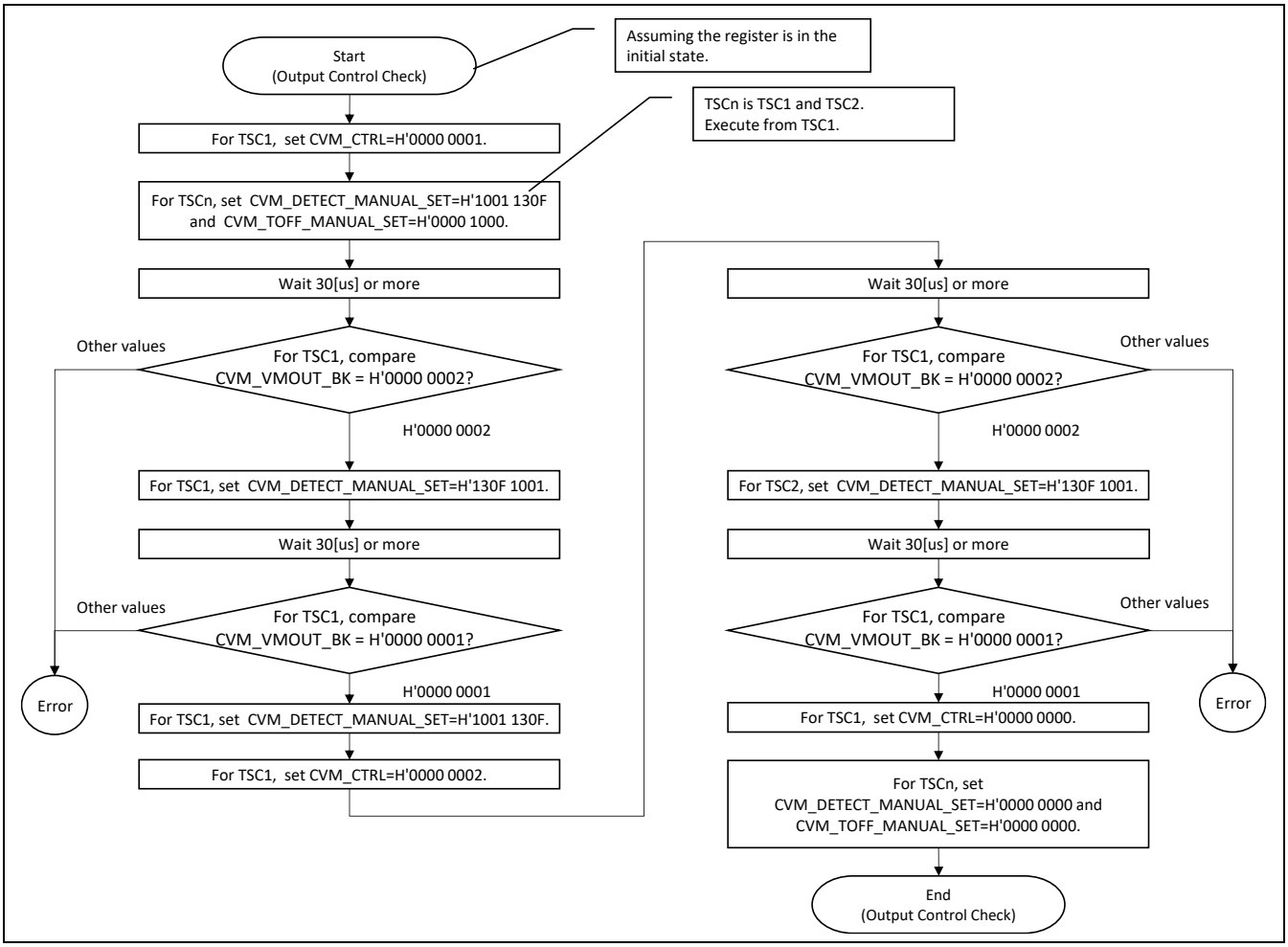



**Figure 13.23 Predefined process "Output Control Check"**

**Table 13.10 Output Control Check combination table**

| Input | | | | | | | | Output | |
|---|---|---|---|---|---|---|---|---|---|
| VMCHPL | | | | VMA_L | VMB_L | VMA_H | VMB_H | VM_L | VM_H |
| [0] | [1] | [2] | [3] | | | | | | |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| | | | | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| | | | | 1 | 1 | 0 | 0 | 1 | 0 |

CVM Output Check is shown Figure 13.24. This test sets CVM to an overvoltage, undervoltage, illegal, and normal state to check the external pin outputs for each state. Table 13.11 shows the combination table used in this test.

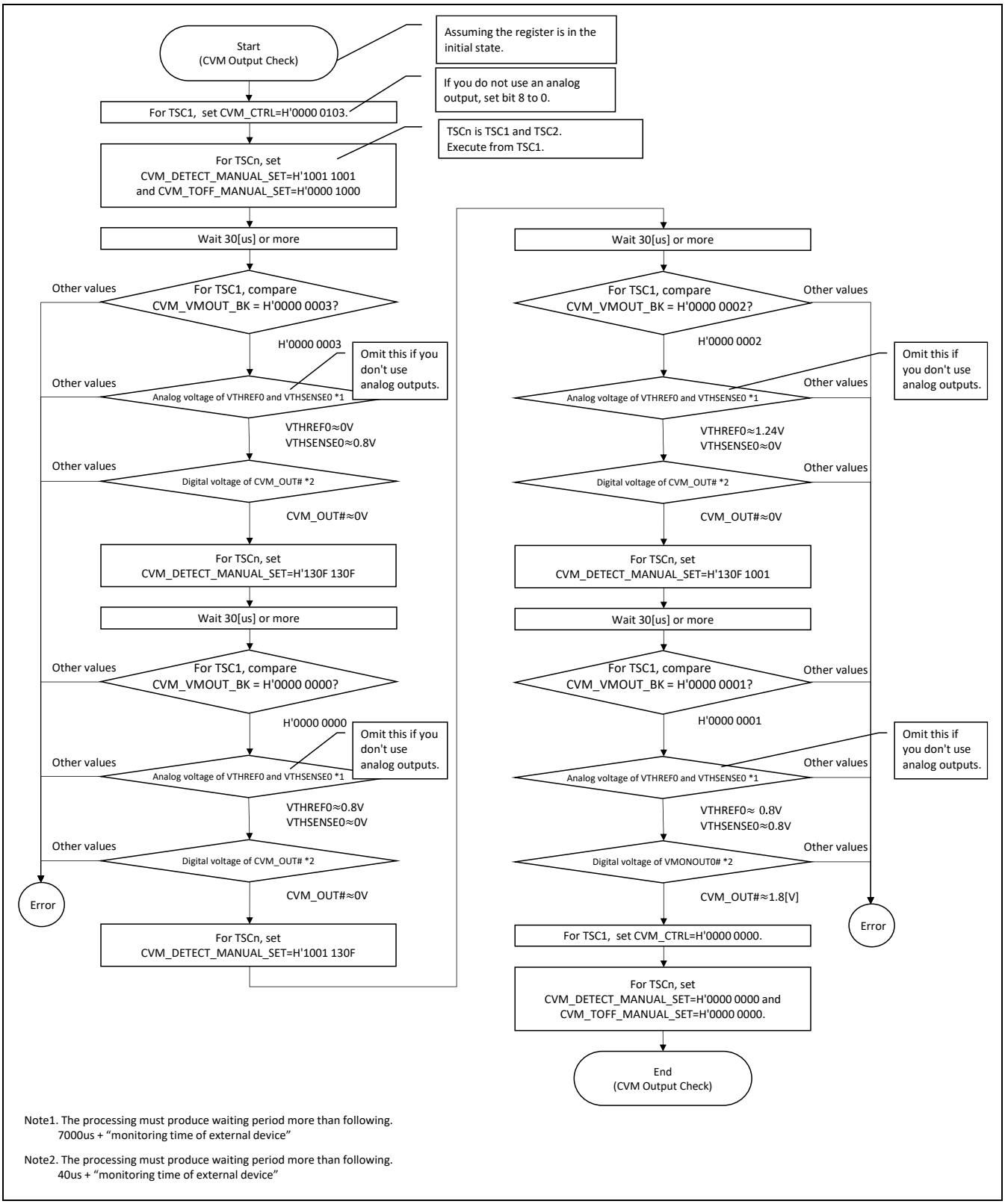



**Figure 13.24 Predefined process "CVM Output Check"**

**Table 13.11 CVM Output Check combination table**

| Input | | | | | | Output | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| VMCHPL | | | | VMA_L, VMB_L | VMA_H, VMB_H | VM_L | VM_H | CVM_OUT# [V] | VTHSENSE0[V] | VTHREF0[V] |
| [0] | [1] | [2] | [3] | | | | | | | |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | ≈0 | ≈0.8 | ≈0 |
| | | | | 0 | 0 | 0 | 0 | ≈0 | ≈0 | ≈0.8 |
| | | | | 0 | 1 | 0 | 1 | ≈0 | ≈0 | ≈1.24 |
| | | | | 1 | 0 | 1 | 0 | ≈1.8 | ≈0.8 | ≈0.8 |

CVM Configuration is shown Figure 13.25. CVM configuration sets overvoltage and undervoltage thresholds using CVM trimming values read from E-fuse. If necessary, set analog output settings and digital output LPF settings. Figure 13.26 shows a block diagram of the optional CVM function selection register. After these settings, make sure that all settings and Readback values match and that the CVM is in a normal state, and set the VM_LOCK to hold the settings.

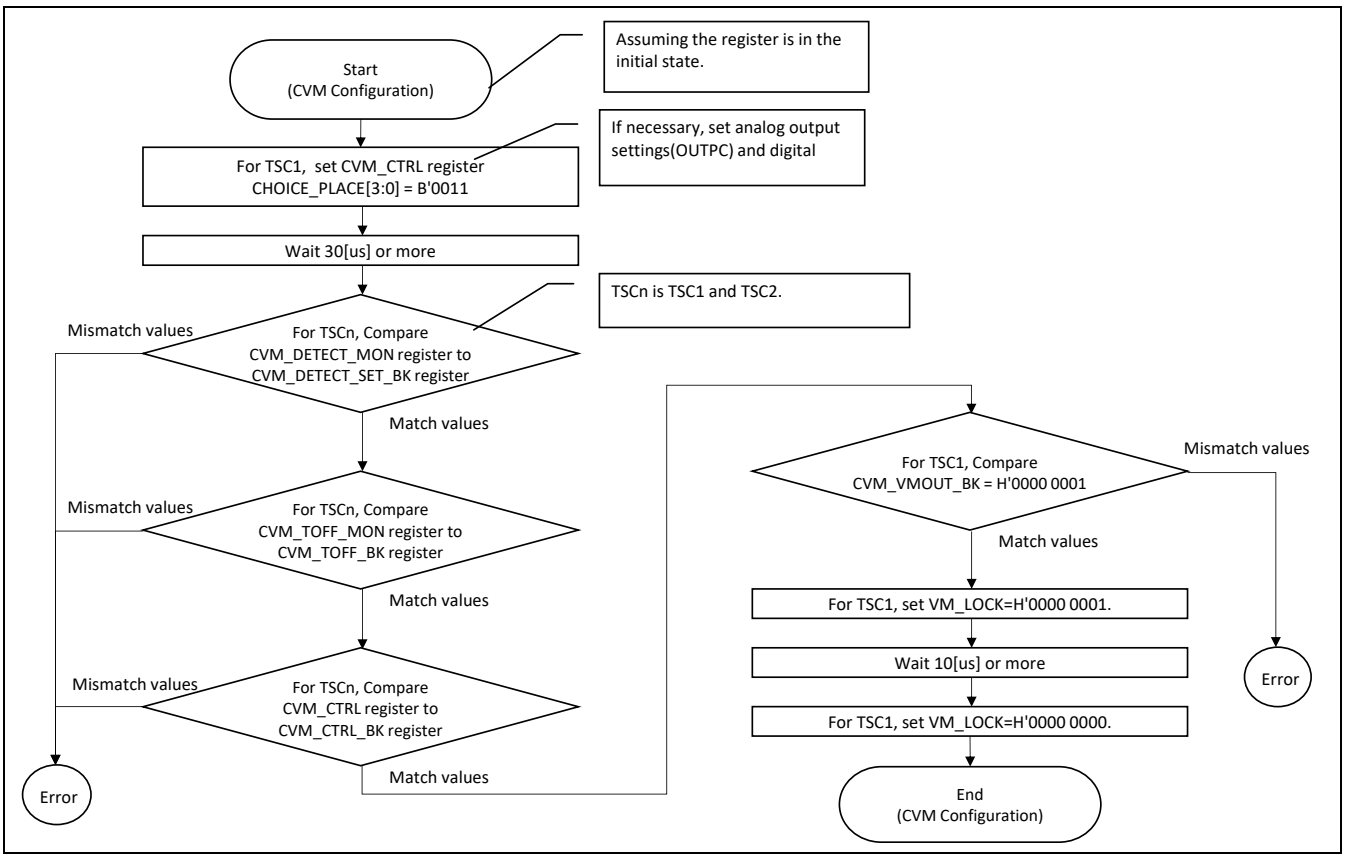



**Figure 13.25 Predefined process "CVM Configuration"**

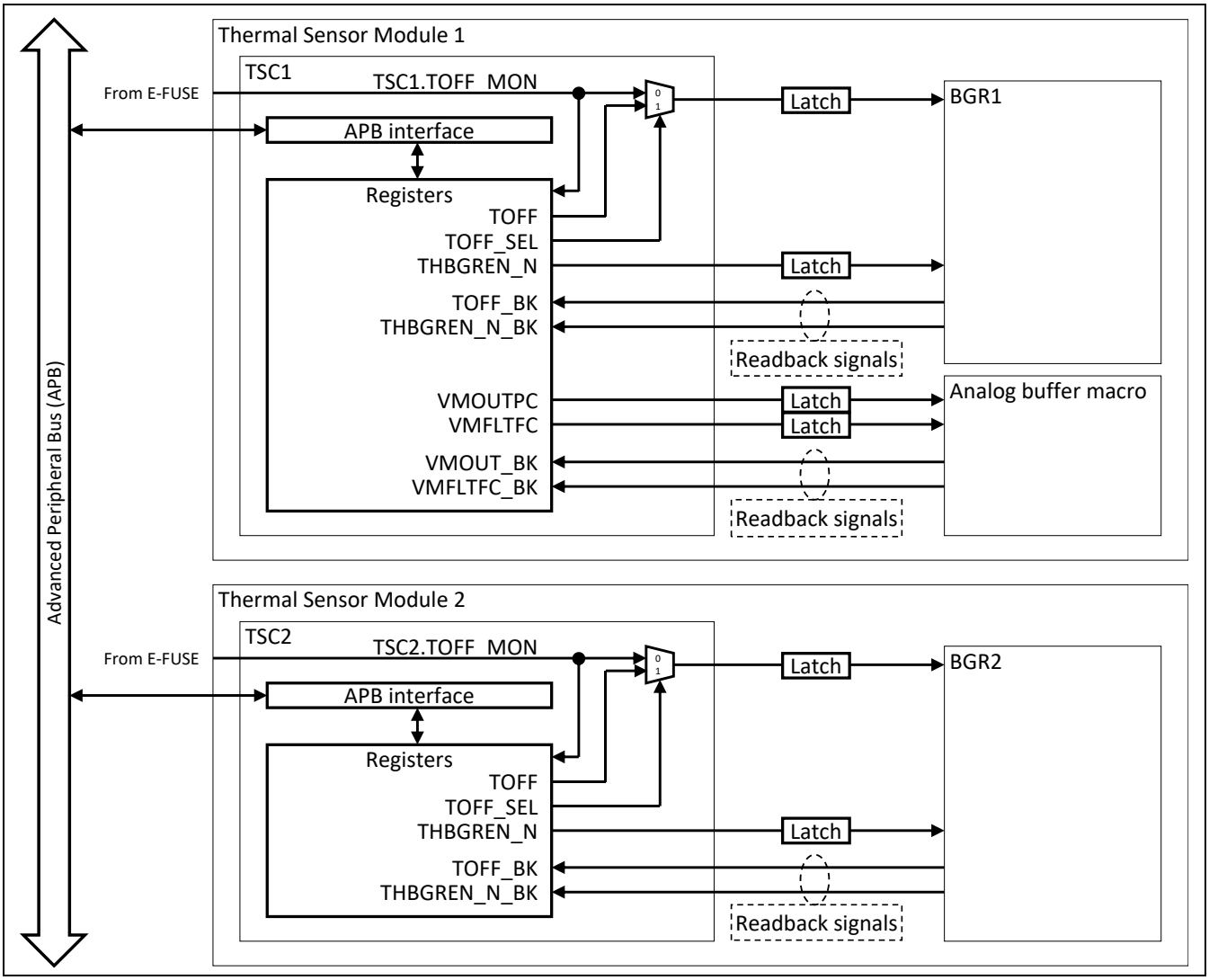



**Figure 13.26 Optional CVM function selection register**

CVM Setting Value Hold Check is shown Figure 13.27. This test verifies that the settings held by the VM_LOCK feature do not change.

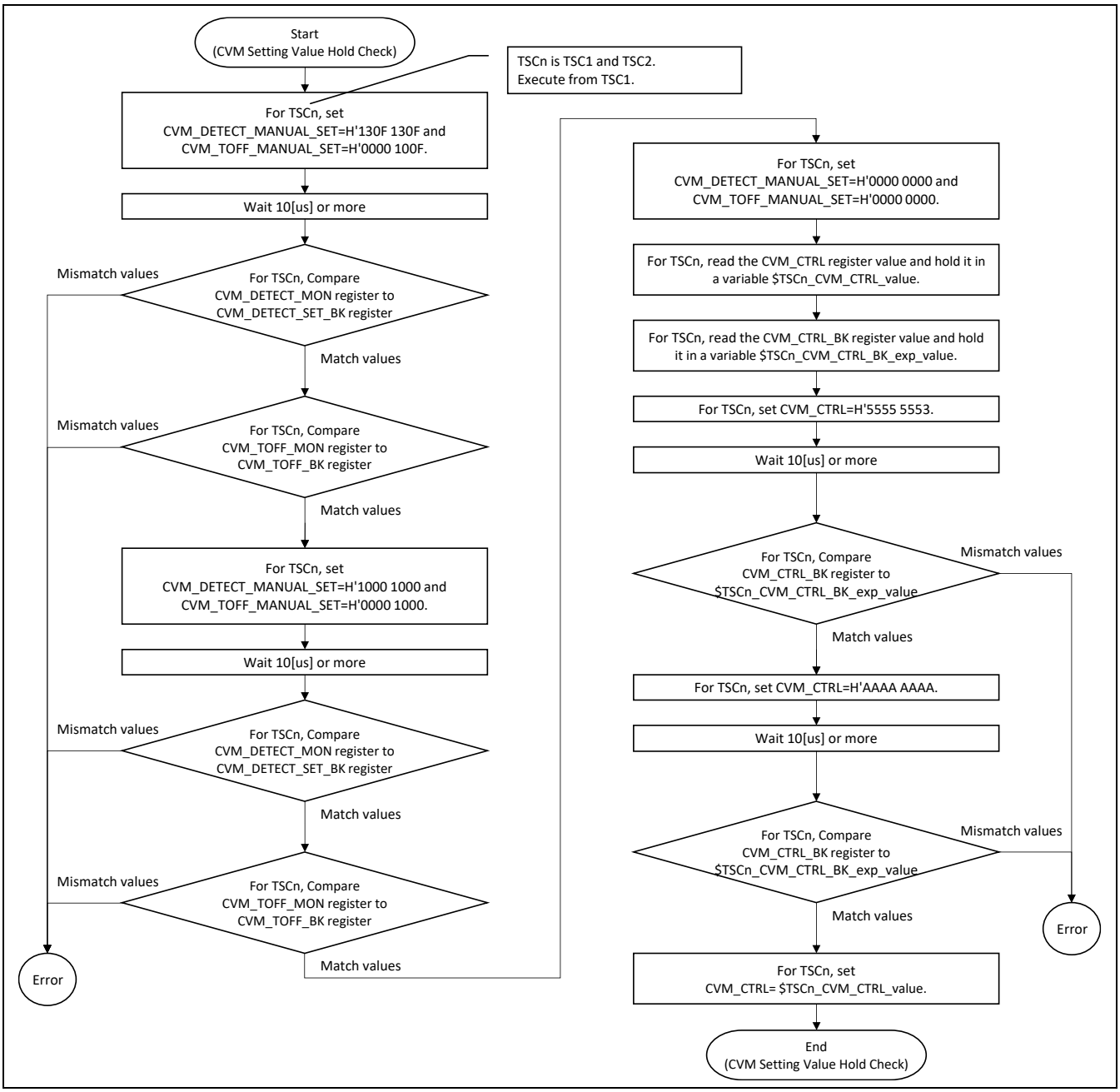



**Figure 13.27 Predefined process "CVM Setting Value Hold Check"**

### 13.5.5 Operation example

Figure 13.28 shows an operation example.

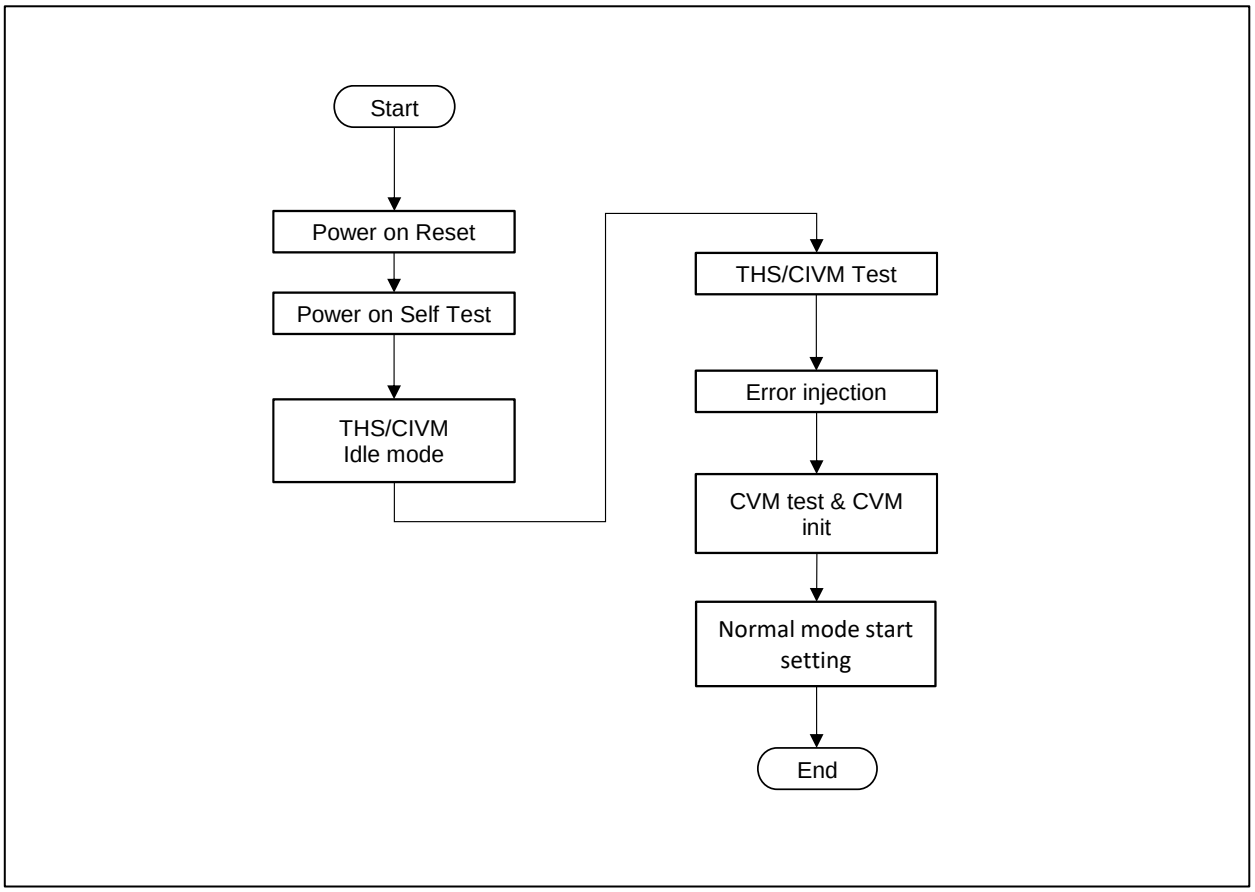



**Figure 13.28 operation example**

# 14. Reset (RST)

## 14.1 Overview

The reset (RST) consists of the bus interface block and registers related to reset control of Power On Reset (POR) and Software Power On Reset (SWPOR). Timing of the bus interface block is based on CL16Mφ.

### 14.1.1 Features

#### (1) Mode Monitor Register

The Mode Pin values can be monitored by Mode Monitor Register 0 and 1 (MODEMR0 and MODEMR1). The detail register specification is described in Sec.14.2.1 and 14.2.2.

#### (2) Reset Control Function

Table 14.1.1 shows the reset categories and reset sources in a chip.

**Table 14.1.1 Reset Categories and Reset Sources**

| Category | Source |
|---|---|
| Power On Reset (POR) | External reset pin (PRESET#) is available for initialization of a whole chip. The sequence for turning on the power is described in Sec.14.3.1. |
| Software Power On Reset (SWPOR) (*1) | A reset can be issued to a whole chip by software register (SRESCR0.SPRES=1). The register specification is described in Sec.14.2.6. |
| | WDT resets can be issued to a whole chip by the RWDT and System WDT modules. The specification of these modules is described in Sec.154.1 and 154.3, respectively. |

Notes:
(1) Some registers of Reset (RST) are not initialized by SWPOR. See Figure 14.1.4 about the reset target modules.

#### (3) Boot Address Registers for ICUMXB

Reset (RST) controls the boot address for ICUMXB as shown in Sec.14.2.11 (ICUMXBAR). The boot address register for Cortex-A76 and Cortex-R52 is controlled by APMU.

#### (4) PRESETOUT# [Functional Safety]

The safety mechanism is supported by PRESETOUT#. It can be judged by monitoring PRESETOUT# whether Reset (RST) module has any failures. When the boot sequence is succeeded without any failures, PRESETOUT# value is changed from LOW to HIGH.

#### (5) Bus Access Protection [Functional Safety]

Register access protection is applied for registers of Reset (RST). The detail specification is described in Sec.14.4.1. In addition, the function is a safety mechanism to guarantee “Freedom From Interference (FFI)”. In detail, see Sec.14.5.3. The register specification is described in Sec.14.2.14.

### 14.1.2 Block Diagram

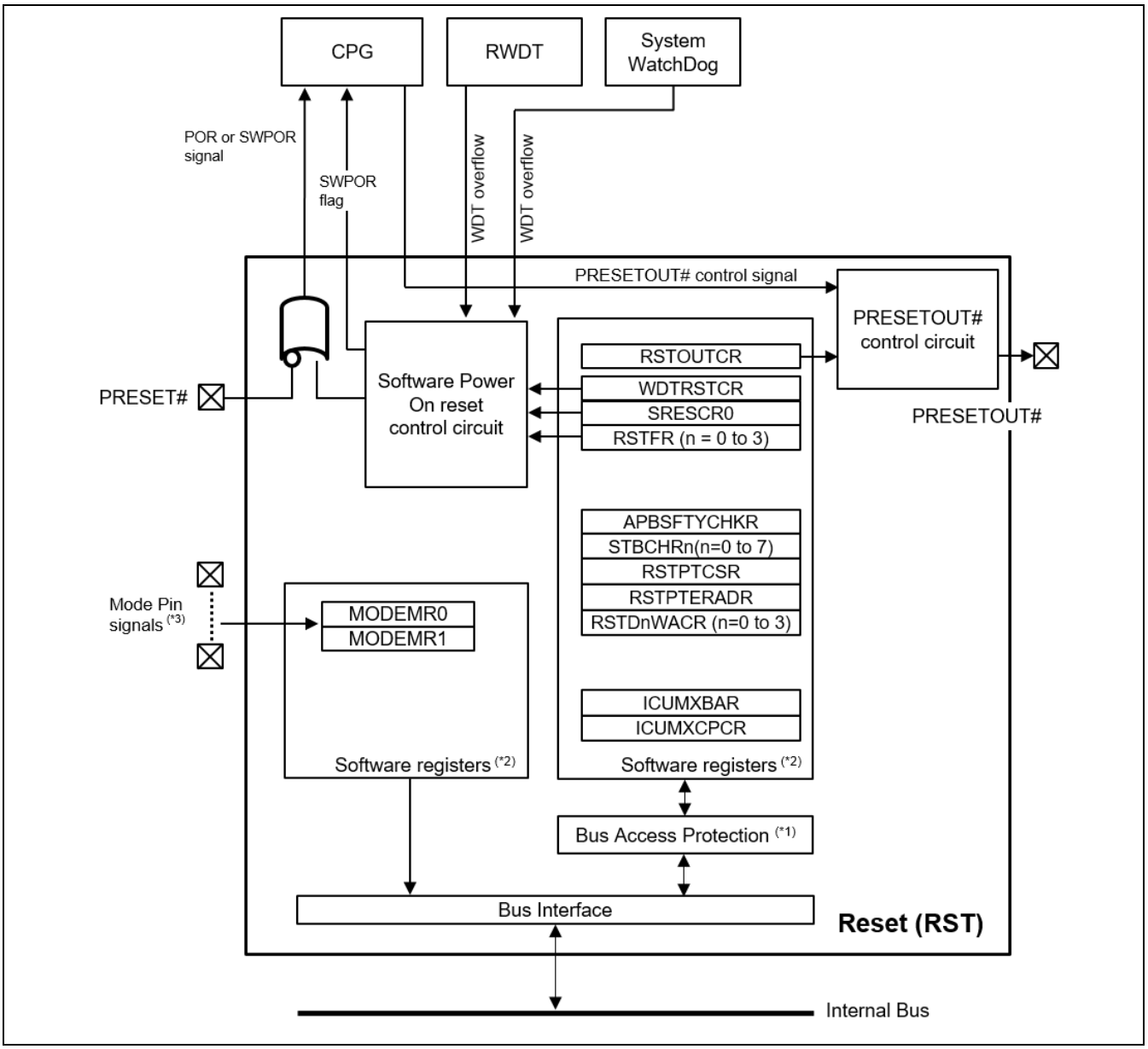



**Figure 14.1.1 Block Diagram of RESET (RST)**

Notes:

(1) Except MODEMR0 and MODEMR1, all registers are protected by Bus Access Protect shown in Sec.14.4.1.

(2) Reset (RST) has software registers. The details specification is described in Sec.14.2.

(3) When PRESET# is negated, the levels on Mode Pin are latched by RESET (RST). The latched signals are provided for target modules. In addition, the signals are sent to MODEMR0 and MODEMR1. The values can be monitored by the registers. See Sec.3.2 about the Mode Pin specification.

### 14.1.3 External Pins

The following table lists the pins connected to the RESET (RST).

**Table 14.1.2 External Pins for RST**

| Name | I/O | Function |
|---|---|---|
| PRESET | Input | Power-on-reset<br>The low-level input on this pin places the LSI in the power-on-reset state. |
| PRESETOUT | Input | Reset output (*6) |
| MD0 | Input | Mode 0. Free Run Mode / Step Up mode Selection (*3) |
| MD1 | Input | Mode 1. Boot Device select (*1) |
| MD2 | Input | Mode 2. Boot Device select (*1) |
| MD3 | Input | Mode 3. Boot Device select (*1) |
| MD4 | Input | Mode 4. Boot Device select (*1) |
| MD5 | Input | Mode 5. Reserved |
| MD6 | Input | Mode 6. Master Boot Processor select(*3) |
| MD7 | Input | Mode 7. Master Boot Processor select(*3) |
| MD8 | Input | Mode 8. Reserved |
| MD9 | Input | Mode 9. EXTAL/XTAL select (*5). |
| MD10 | Input | Mode 10. Debug Mode select(*2) |
| MD11 | Input | Mode 11. Debug Mode select(*2) |
| MD13 | Input | Mode 13. PLL multiplication ratio setting (*5). |
| MD14 | Input | Mode 14. PLL multiplication ratio setting (*5). |
| MD16 | Input | Mode 16. Reserved. |
| MD17 | Input | Mode 17. DDR clock frequency setting. |
| MD18 | Input | Mode 18. SSCG enable / disable for PLL1(*4) |
| MD19 | Input | Mode 19. DDR clock frequency setting. |
| MD20 | Input | Mode 20. Debug Mode select(*2) |
| MD21 | Input | Mode 21. Debug Mode select(*2) |
| MD22 | Input | Mode 22. DDR Mode select |
| MD23 | Input | Mode 23. POST target select (*8) |
| MD24 | Input | Mode 24. Reserved. |
| MD25 | Input | Mode 25. Field BIST enable / disable control |
| MD27 | Input | Mode 27. DDR Mode select. |
| MD29 | Input | Mode 29. RCLK reference clock select (*5). |
| MD31 | Input | Mode 31. SCIF Download Mode select (*1) |
| MD32 | Input | Mode 32. SCIF Download Mode select (*1) |
| MD33 | Input | Mode 33. Reserved |
| MD35 | Input | Mode 35. Reserved |
| MD36 | Input | Mode 36. SSCG enable / disable for PLL1(*4) |
| MD37 | Input | Mode 37. SSCG enable / disable for PLL1(*4) |
| MDT0 | Input | MDT0. Debug Mode select(*2) |

Notes

1) See 31 Boot (Table 31.1 Mode Pin Configuration (1)
2) See 170 Debug and Trace (Table JTAG selection table).
3) See 8. CPG (8.1.5.10 Step-up mode)
4) See 8. CPG (8.1.5.1 PLL1 Initial Multiplication Ratio).
5) See 8. CPG (8.1.5 Clock Operating Modes).
6) See 8. CPG (8.1.5.9 Step-up Mode).
7) See 8. CPG (8.1.5.3 PLL3 Multiplication ratio)
8) See Table 171.2.

### 14.1.4 Register Configuration

Register address space of Reset module is separated by 4 domains. The base address of each domain is as bellow. The detail specification is described in Sec.14.4.1.

| Domain number | Base address |
|---|---|
| 0 | H'E616 0000 |
| 1 | H'E616 4000 |
| 2 | H'E616 8000 |
| 3 | H'E616 C000 |

**Table 14.1.3 RESET (RST) Registers Details**

| Register Name | Mnemonic | R/W | Offset Address | Initial value | Size | Corresponding DnWACR (n = 0, 1, 2, 3) |
|---|---|---|---|---|---|---|
| Mode Monitor Register 0 | MODEMR0 | R | H'0000 | H'---- ---- | 32 | (*2) |
| Mode Monitor Register 1 | MODEMR1 | R | H'0004 | H'-000 000- | 32 | (*2) |
| Watchdog Timer Reset Control Register | WDTRSTCR | R/W | H'0010 | H'0000 8003 | 32 | RSTDnWACR [4] |
| PRESETOUT0# Control Register | RSTOUTCR | R/W | H'0014 | H'0000 0001 | 32 | RSTDnWACR [5] |
| Soft Power On Reset Control Register 0 | SRESCR0 | R/W | H'0018 | H'0000 0000 | 32 | RSTDnWACR [6] |
| Reset Flag Register 0 | RSTFR0 | R/W | H'0020 | H'0000 0000 | 32 | RSTDnWACR [8] |
| Reset Flag Register 1 | RSTFR1 | R/W | H'0024 | H'0000 0000 | 32 | RSTDnWACR [9] |
| Reset Flag Register 2 | RSTFR2 | R/W | H'0028 | H'0000 0000 | 32 | RSTDnWACR [10] |
| Reset Flag Register 3 | RSTFR3 | R/W | H'002C | H'0000 0000 | 32 | RSTDnWACR [11] |
| Standby Flag Register 0 | STBCHR0 | R/W | H'0030 | H'0000 0000 | 32 | RSTDnWACR [12] |
| Standby Flag Register 1 | STBCHR1 | R/W | H'0034 | H'0000 0000 | 32 | RSTDnWACR [13] |
| Standby Flag Register 2 | STBCHR2 | R/W | H'0038 | H'0000 0000 | 32 | RSTDnWACR [14] |
| Standby Flag Register 3 | STBCHR3 | R/W | H'003C | H'0000 0000 | 32 | RSTDnWACR [15] |
| Standby Flag Register 4 | STBCHR4 | R/W | H'0040 | H'0000 0000 | 32 | RSTDnWACR [16] |
| Standby Flag Register 5 | STBCHR5 | R/W | H'0044 | H'0000 0000 | 32 | RSTDnWACR [17] |
| Standby Flag Register 6 | STBCHR6 | R/W | H'0048 | H'0000 0000 | 32 | RSTDnWACR [18] |
| Standby Flag Register 7 | STBCHR7 | R/W | H'004C | H'0000 0000 | 32 | RSTDnWACR [19] |
| APB bus Safety Check Register | APBSFTYCHKR | R/W | H'0050 | H'0000 0000 | 32 | RSTDnWACR [20] |
| ICUMXB Boot Address Register | ICUMXBAR | R/W | H'0054 | H'0000 00-- | 32 | RSTDnWACR [21] |
| ICUMXB CPinit Control Register | ICUMXCPCR | R/W | H'0058 | H'0000 000- | 32 | RSTDnWACR [22] |
| Protect Control/Status Register | RSTPTCSR | R/W | H'0068 | H'0000 0002 | 32 | RSTDnWACR [26] |
| Protect Error Address Register | RSTPTERADR | R | H'006C | H'0000 0000 | 32 | RSTDnWACR [27] |
| Domain 0 Write Access Control Register | RSTD0WACR | R/W | H'3800 | H'FFFF FFFF | 32 | (*1) |
| Domain 1 Write Access Control Register | RSTD1WACR | R/W | H'3A00 | H'0000 0000 | 32 | (*1) |
| Domain 2 Write Access Control Register | RSTD2WACR | R/W | H'3C00 | H'0000 0000 | 32 | (*1) |
| Domain 3 Write Access Control Register | RSTD3WACR | R/W | H'3E00 | H'0000 0000 | 32 | (*1) |

Note 1: The register can be accessed by Domain 0.

Note 2: There are no protect registers for Read Only register.

**Table 14.3 Initialization condition of RST Register**

| Register Name | Mnemonic | Initialization condition |
|---|---|---|
| Mode Monitor Register 0 | MODEMR0 | PRESET# only |
| Mode Monitor Register 1 | MODEMR1 | PRESET# only |
| Watchdog Timer Reset Control Register | WDTRSTCR | PRESET# only |
| PRESETOUT# Control Register | RSTOUTCR | PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| Soft Power On Reset Control Register 0 | SRESCR0 | PRESET# only |
| Reset Flag Register 0 | RSTFR0 | PRESET# only |
| Reset Flag Register 1 | RSTFR1 | PRESET# only |
| Reset Flag Register 2 | RSTFR2 | PRESET# only |
| Reset Flag Register 3 | RSTFR3 | PRESET# only |
| Standby Flag Register 0 | STBCHR0 | PRESET# only |
| Standby Flag Register 1 | STBCHR1 | PRESET# only |
| Standby Flag Register 2 | STBCHR2 | PRESET# only |
| Standby Flag Register 3 | STBCHR3 | PRESET# only |
| Standby Flag Register 4 | STBCHR4 | PRESET# only |
| Standby Flag Register 5 | STBCHR5 | PRESET# only |
| Standby Flag Register 6 | STBCHR6 | PRESET# only |
| Standby Flag Register 7 | STBCHR7 | PRESET# only |
| APB bus Safety Check Register | APBSFTYCHKR | PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| ICUMXB Boot Address Register | ICUMXBAR | Depend on WDTRSTCR.RESBAR2S PRESET# only or PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| ICUMXB CPinit Control Register | ICUMXCPCR | Depend on WDTRSTCR.RESBAR2S PRESET# only or PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| Protect Control/Status Register | RSTPTCSR | PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| Protect Error Address Register | RSTPTERADR | PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| Domain 0 Write Access Control Register | D0WACR | PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| Domain 1 Write Access Control Register | D1WACR | PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| Domain 2 Write Access Control Register | D2WACR | PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |
| Domain 3 Write Access Control Register | D3WACR | PRESET#, Soft Power On Reset(WDT reset or SRESCR.SPRES=1) |

### 14.1.5 Connected modules

**Table 14.1.3 Connected modules**

| Module name | Connected module name | Function of connected module |
|---|---|---|
| RST | CPG | Reset control |
| | RWDT | Mask RWDT reset request |
| | System WatchDog | Mask System WatchDog reset request |
| | ICUMXB | Reset and Boot address control |

## 14.2 Register Description

Legend for Register Description

| | |
|---|---|
| Initial value | : Register value after a reset |
| – | : Undefined value |
| R/W | : Readable/writable. The written value can be read. |
| R | : Read-only. |
| W | : Write-only. The read value is undefined. |
| R/WC1 | : Readable/writable. Writing 1 inializes the bit. Writing 0 is ignored. |

### 14.2.1 Mode Monitor Register 0 (MODEMR0)

MODEMR0 is a 32-bit read-only register, which can be accessed only in longwords.
This register is initialized only by PRESET#.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MD31 | — | MD29 | — | MD27 | — | MD25 | MD24 | MD23 | MD22 | MD21 | MD20 | MD19 | MD18 | MD17 | MD16 |
| Initial value: | — | 0 | — | 0 | — | 0 | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | MD14 | MD13 | — | MD11 | MD10 | MD9 | MD8 | MD7 | MD6 | MD5 | MD4 | MD3 | MD2 | MD1 | MD0 |
| Initial value: | 0 | — | — | 0 | — | — | — | — | — | — | *1 | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note 1 : Initial value of MD5 bit is 1'b0 when LCS = SE.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | MD31 | Depend on mode setting | R | The value of MD31 |
| 30 | — | 0 | R | Reserved. When read, returns 0. |
| 29 | MD29 | Depend on mode setting | R | The value of MD29 |
| 28 | — | 0 | R | Reserved. When read, returns 0. |
| 27 | MD27 | Depend on mode setting | R | The value of MD27 |
| 26 | — | 0 | R | Reserved. When read, returns 0. |
| 25 | MD25 | Depend on mode setting | R | The value of MD25 |
| 24 | MD24 | Depend on mode setting | R | The value of MD24 |
| 23 | MD23 | Depend on mode setting | R | The value of MD23 |
| 22 | MD22 | Depend on mode setting | R | The value of MD22 |
| 21 | MD21 | Depend on mode setting | R | The value of MD21 |
| 20 | MD20 | Depend on mode setting | R | The value of MD20 |
| 19 | MD19 | Depend on mode setting | R | The value of MD19 |
| 18 | MD18 | Depend on mode setting | R | The value of MD18 |
| 17 | MD17 | Depend on mode setting | R | The value of MD17 |
| 16 | MD16 | Depend on mode setting | R | The value of MD16 |
| 15 | — | 0 | R | Reserved. When read, returns 0. |
| 14 | MD14 | Depend on mode setting | R | The value of MD14 |
| 13 | MD13 | Depend on mode setting | R | The value of MD13 |
| 12 | — | 0 | R | Reserved. When read, returns 0. |
| 11 | MD11 | Depend on mode setting | R | The value of MD11 |
| 10 | MD10 | Depend on mode setting | R | The value of MD10 |
| 9 | MD9 | Depend on mode setting | R | The value of MD9 |
| 8 | MD8 | Depend on mode setting | R | The value of MD8 |
| 7 | MD7 | Depend on mode setting | R | The value of MD7 |
| 6 | MD6 | Depend on mode setting | R | The value of MD6 |
| 5 | MD5 | Depend on mode setting *1 | R | The value of MD5 |
| 4 | MD4 | Depend on mode setting | R | The value of MD4 |
| 3 | MD3 | Depend on mode setting | R | The value of MD3 |
| 2 | MD2 | Depend on mode setting | R | The value of MD2 |
| 1 | MD1 | Depend on mode setting | R | The value of MD1 |
| 0 | MD0 | Depend on mode setting | R | The value of MD0 |

### 14.2.2 Mode Monitor Register 1 (MODEMR1)

MODEMR1 is a 32-bit read-only register, which can be accessed only in longwords.
This register is initialized only by PRESET#.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | MDT0 | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | MD37 | MD36 | MD35 | — | MD33 | MD32 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | 0 | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved. When read, returns 0. |
| 28 | MDT0 | Depend on mode setting | R | The value of MDT0 |
| 27 to 6 | — | All 0 | R | Reserved. When read, returns 0. |
| 5 | MD37 | Depend on mode setting | R | The value of MD37 |
| 4 | MD36 | Depend on mode setting | R | The value of MD36 |
| 3 | MD35 | Depend on mode setting | R | The value of MD35 |
| 2 | — | All 0 | R | Reserved. When read, returns 0. |
| 1 | MD33 | Depend on mode setting | R | The value of MD33 |
| 0 | MD32 | Depend on mode setting | R | The value of MD32 |

### 14.2.3 Watchdog Timer Reset Control Register (WDTRSTCR)

WDTRSTCR is a 32-bit readable/writable register, which can be accessed only in longwords. This register specifies whether the watchdog timer overflow should be masked or not as a reset trigger. The upper word of the write value should always be the code value H'A55A. If the code value is read, it is always read as 0.

This register is initialized only by PRESET#.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Code value (H'A55A) | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR2S | — | — | — | — | — | — | — | — | — | — | — | — | — | SWDT_RSTMSK | RWDT_RSTMSK |
| Initial value: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Code value (H'A55A)<br>When read, returns 0. |
| 15 | RESBAR2S | 1 | R/W | Select BAR2 registers reset condition<br>0: ICUMXBAR and ICUMXCPCR are initialized only by PRESET#.<br>1: ICUMXBAR and ICUMXCPCR are initialized by PRESET#, WDT reset, and Soft Power On Reset (SRESCSR.PRES=1). |
| 14 to 2 | — | All 0 | R | Reserved<br>When read, returns 0. The write value should always be 0. |
| 1 | SWDT_RSTMSK | 1 | R/W | System WatchDog Reset Mask<br>This bit is used to mask the detection of System WatchDog overflow.<br>0: Reset request<br>1: Not reset request |
| 0 | RWDT_RSTMSK | 1 | R/W | RWDT Reset Mask<br>This bit is used to mask the detection of RWDT overflow.<br>0: Reset request<br>1: Not reset request |

**Note:**

1. If a conflict between "WDTRSTCR.RWDT_RSTMSK (from 0 to 1)" and "SPOR asserting" occurs, it cannot be guaranteed that the value of RSTFRn.RCRWDT bit is certain. After the WDTRSTCR.RWDT_RSTMSK is changed from 0 to 1, the value of RWTCSRA.WOVF in RWDT (Sec. 154.1) should be checked instead of the value of RSTFRn.RCRWDT.

2. If a conflict between "WDTRSTCR.SWDT_RSTMSK (from 0 to 1)" and "SPOR asserting" occurs, it cannot be guaranteed that the value of RSTFRn.RCSWDT bit is certain. After the WDTRSTCR.SWDT_RSTMSK is changed from 0 to 1, the value of SWTCSRA.WOVF in SWDT (Sec. 154.1) should be checked instead of the value of RSTFRn.RCSWDT.

### 14.2.4 PRESETOUT# Control Register (RSTOUTCR)

RSTOUTCR is a 32-bit readable/writable register, which can be accessed only in longwords. This register controls the output level of external LSI pin RESETOUT# by software.

This register is initialized by PRESET# and Soft Power On Reset (WDT reset or SRESCR0.SPRES=1).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RES OUT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>When read, returns 0. The write value should always be 0. |
| 0 | RESOUT | 1 | R/W | PRESETOUT# control by software<br>This bit is used to specify the output level of PRESETOUT#.<br>0: PRESETOUT# is asserted<br>1: PRESETOUT# is negated |

### 14.2.5 Soft Power On Reset Control Register 0 (SRESCR0)

SRESCR0 is a 32bit readable/writeable register, which can be accessed only in longwords. The upper word of the write value should always be the code value H' 5AA5. If the code value is read, it is always read as 0.

This register is initialized only by PRESET#.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Code value (H'5AA5) | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPRES | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Code value (H'5AA5)<br>When read, returns 0. |
| 15 | SPRES | 0 | R/W | Soft Power On Reset<br>Soft Power On Reset is asserted.<br>This bit is initialized on Power On Reset.<br>0：Ignored<br>1：Soft Power On Reset is asserted.<br>Note: Soft Power On Reset is asserted when 1 is written to this bit.<br>When Power On Reset process is completed, this bit gets cleared<br>(The range of initialization is same with Soft Power On Reset by WDT) |
| 14 to 0 | — | All 0 | R | Reserved<br>When read, returns 0. The write value should always be 0. |

### 14.2.6 Reset Flag Register n (RSTFRn) (n=0 to 3)

RSTFRn is a 32bit readable/writeable register, which can be accessed only in longwords. Write access is performed only to clear the flag bits. This register sets the reset flags when Soft Power On Reset (Power On Reset other than PRESET#) is detected. By reading this register at the time of reboot etc., it is known that which kind of reset was detected.

This register is initialized only by PRESET#.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | RCPRES | RCSWDT | RCRWDT | RPF |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved<br>When read, returns 0. The write value should always be 0. |
| 3 | RCPRES | 0 | R/W | Soft Power On Reset Factor Bit<br>0：SRESCR.SPRES is not Soft Power On Reset factor.<br>1：SRESCR.SPRES is Soft Power On Reset factor. |
| 2 | RCSWDT | 0 | R/W | Soft Power On Reset Factor Bit<br>0：System-WDT is not Soft Power On Reset factor.<br>1：System-WDT is Soft Power On Reset factor. |
| 1 | RCRWDT | 0 | R/W | Soft Power On Reset Factor Bit<br>0：RWDT is not Soft Power On Reset factor.<br>1：RWDT is Soft Power On Reset factor. |
| 0 | RPF | 0 | R/W | Soft Power On Reset detection bit<br>0：Soft Power On Reset is not detected.<br>1：Soft Power On Reset is detected. |

**Note:**
See “Note” in 14.2.4 Watchdog Timer Reset Control Register (WDTRSTCR).

### 14.2.7 Standby Flag Register n (STBCHRn) (n=0 to 7)

STBCHRn are 32-bit readable/writable registers, which can be accessed only in longwords. These registers are defined purely for software purpose. Setting these registers doesn't influence the LSI operation. As these bits are initialized only by PRESET#, the programmer can use this register at their convenience.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STBn[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STBn[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | STBn[31:0] | All 0 | R/W | defined purely for software purpose |

### 14.2.8 APB bus Safety Check Register (APBSFTYCHKR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHK[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHK [15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

This register is functional safety use only. Keep initial value.

### 14.2.9 ICUMXB Boot Address Register (ICUMXBAR)

ICUMXBAR is a 32-bit readable/writable register, which can be accessed only in longwords.
This register is initialized by PRESET# when WDTRSTCR.RESBAR2S=0. This register is initialized by PRESET# and Soft Power On Reset (WDT reset or SRESCR0.SPRES=1) when WDTRSTCR.RESBAR2S=1.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBAR2[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBAR2[15:9] | | | | | | | — | — | — | — | BAREN | — | — | BTMD[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 | 0 | 0 | *2 | *2 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | RBAR2[31:9] | All 0 | R/W | ICUMXB Boot Address2<br>When the ICUMXB accesses to the range of physical address from 0x0000 0000 to 0x0000 01FF, the address bit in [31:9] is replaced by RBAR2[31:9]. |
| 8 to 5 | — | All 0 | R | Reserved<br>When read, returns 0. The write value should always be 0. |
| 4 | BAREN *3 | *1 | R/W | BAREN bit<br>0: RBAR2 is not valid.<br>1: RBAR2 is valid. |
| 3 to 2 | — | All 0 | R | Reserved<br>When read, returns 0. The write value should always be 0. |
| 1 to 0 | BTMD[1:0] | *2 | R/W | Specifies the Boot area of ICUMXB<br>During Hardware Power On Reset the initial value of these bits depends upon MD4, MD3, MD2 and MD1 pin setting.<br>00: RBAR2[31:9] is assigned to Boot address.<br>01: Prohibited.<br>10: Boots from BootROM.<br>11: Prohibited. |

Note *1 When MD4!=0 or MD3!=0 or MD2!=0 or MD1!=0, initial value is 1. When others, initial value is 0.

Note *2 Initial value depends upon MD4, MD3, MD2 and MD1 pin setting.

Note *3 Do not rewrite BAREN=1 simultaneously with RBAR2 and BTMD. After completing the setting of RBAR2 and BTMD with BAREN=0, set only BAREN=1.
For example: Change RBAR2[31:9] from A to B, and BTMD[1:0]=00
Step 1: Write RBAR2[31: 9]=B, BAREN=0, BTMD[1:0]=00
Step 2: Write RBAR2[31: 9]=B, BAREN=1, BTMD[1:0]=00

**Table 14.2.1 Initial values of BAREN, BTMD[1:0], and boot address**

| Mode pin setting | | | | Initial values | | | Note |
|---|---|---|---|---|---|---|---|
| MD4 | MD3 | MD2 | MD1 | BAREN | BTMD[1:0] | Boot address | |
| 0 | 0 | 0 | 0 | 0 | 00 | H'00 0000 0000 | |
| 0 | 0 | 0 | 1 | 1 | 01 | H'00 0800 0000 | This setting is prohibited |
| 0 | 0 | 1 | 0 | 1 | 10 | H'00 EB10 4000 | |
| 0 | 0 | 1 | 1 | 1 | 10 | H'00 EB10 4000 | |
| 0 | 1 | 0 | 0 | 1 | 10 | H'00 EB10 4000 | |
| 0 | 1 | 0 | 1 | 1 | 10 | H'00 EB10 4000 | This setting is prohibited |
| 0 | 1 | 1 | 0 | 1 | 10 | H'00 EB10 4000 | |
| 0 | 1 | 1 | 1 | 1 | 10 | H'00 EB10 4000 | |
| 1 | 0 | 0 | 0 | 1 | 10 | H'00 EB10 4000 | This setting is prohibited |
| 1 | 0 | 0 | 1 | 1 | 10 | H'00 EB10 4000 | This setting is prohibited |
| 1 | 0 | 1 | 0 | 1 | 01 | H'00 0800 0000 | |
| 1 | 0 | 1 | 1 | 1 | 01 | H'00 0800 0000 | |
| 1 | 1 | 0 | 0 | 1 | 10 | H'00 EB10 4000 | This setting is prohibited |
| 1 | 1 | 0 | 1 | 1 | 10 | H'00 EB10 4000 | |
| 1 | 1 | 1 | 0 | 1 | 10 | H'00 EB10 4000 | This setting is prohibited |
| 1 | 1 | 1 | 1 | 1 | 10 | H'00 EB10 4000 | |

### 14.2.10 ICUMXB CPinit Control Register (ICUMXCPCR)

ICUMXCPCR is a 32-bit readable/writable register, which can be accessed only in longwords. This register is initialized by PRESET# when WDTRSTCR.RESBAR2S=0.

This register is initialized by PRESET# and Soft Power On Reset (WDT reset or SRESCR0.SPRES=1) when WDTRSTCR.RESBAR2S=1.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ICUMX CPINIT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

Note *1 : When MD7=1 and MD6=0, initial value is 1, in other case initial value is 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>When read, returns 0. The write value should always be 0. |
| 0 | ICUMXCPINIT | *1 | R/W | 0: Reset to ICUMXB is asserted. ICUMXB is not booted.<br>1: Reset to ICUMXB is de-asserted. ICUMXB is booted. |

Note *1 : When MD7=1 and MD6=0, initial value is 1, in other case initial value is 0.

Operation of ICUMXB boot by Cortex-R52 is below.

Step1: Cortex-R52 boot.
Step2: Setting ICUMXBAR
Step3: Dummy read to ICUMXBAR.
Step4: Changing ICUMXCPINIT value from 0 to 1. To Change the value from 1 to 0 is prohibited.

ICUMXBAR and ICUMXCPINIT register values are not used in Secure LCS.
Secure (SE) LCS is one of Life Cycle states, which is defined by secure engine at booting.

### 14.2.11 Protect Control/Status Register (RSTPTCSR)

RSTPTCSR is a 32-bit readable/writable register, which can be accessed only in longwords.
This register is initialized by PRESET# and Soft Power On Reset (WDT reset or SRESCR0.SPRES=1).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EIE | ERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | EIE | 1 | R/W | Enable error interrupt request of write access protection<br>0: Disable error interrupt request<br>1: Enable error interrupt request<br>When EIE=1 and ERR=1, issue error interrupt request to ECM. |
| 0 | ERR | 0 | R/WC1 | Display the status of error detection on secure access protection<br>0: Not detect error.<br>1: Detect error of write access protection to protected regsiters which are specified RSTDnWPTCSR.<br>Write 1 to this bit to clear this flag bit. |

### 14.2.12 Protect Error Address Register (RSTPTERADR)

RSTPTERADR is a 32-bit read-only register, which can be accessed only in longwords.
This register is initialized by PRESET# and Soft Power On Reset (WDT reset or SRESCR0.SPRES=1).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDR[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 1 | ADDR[15:1] | All 0 | R | Offset address[15:1] of the first illegal write access to the protected registers.<br><br>When write access is occurred to the potected registers in RESET module, the address offset value is set to these bits automatically. The set address is one of the first illegal access. |
| 0 | ADDR[0] | 0 | R/WC1 | Offset address[0] of the first illegal write access to the protected registers.<br><br>When write access is occurred to the potected registers in RESET module, the address offset value is set to these bits automatically. The set address is one of the first illegal access.<br>When write 1 to the bit, ADDR[15:0] Is cleared. |

### 14.2.13 Domain n Write Access Control Register (RSTDnWACR) (n=0 to 3)

RSTDnWACR is a 32-bit readable/writable register, which can be accessed only in longwords. This register is specified write access premission of each regsiters from the domain n.
This register is initialized by PRESET# and Soft Power On Reset (WDT reset or SRESCR.SPRES=1).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | RSTPTERADR | RSTPTCSR | — | — | — | ICUMXCPCR | ICUMXBAR | APBSFTYCHKR | STBCHR7 | STBCHR6 | STBCHR5 | STBCHR4 |
| Initial value: | | | | | | | | | | | | | | | | |
| n=0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| n=1 to3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STBCHR3 | STBCHR2 | STBCHR1 | STBCHR0 | RSTFR3 | RSTFR2 | RSTFR1 | RSTFR0 | | SRESCR0 | RSTOUTCR | WDTRSTCR | — | — | — | — |
| Initial value: | | | | | | | | | | | | | | | | |
| n=0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| n=1 to3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | R/W | R/W | R/W | R | R | R | R |

**[RSTDnWCR (n=0)]**

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 1 | R | Reserved<br>These bits are always read as 1. The write value should always be 1. |
| 27 | RSTPTERADR | 1 | R/W | Write permission to RSTPTERADR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 26 | RSTPTCSR | 1 | R/W | Write permission to RSTPTCSR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 25 to 23 | — | All 1 | R | Reserved<br>These bits are always read as 1. The write value should always be 1. |
| 22 | ICUMXCPCR | 1 | R/W | Write permission to ICUMXCPCR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 21 | ICUMXBAR | 1 | R/W | Write permission to ICUMXBAR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 20 | APBSFTYCHKR | 1 | R/W | Write permission to APBSFTYCHKR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 19 | STBCHR7 | 1 | R/W | Write permission to STBCHR7 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 18 | STBCHR6 | 1 | R/W | Write permission to STBCHR6 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 17 | STBCHR5 | 1 | R/W | Write permission to STBCHR5 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 16 | STBCHR4 | 1 | R/W | Write permission to STBCHR4 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | STBCHR3 | 1 | R/W | Write permission to STBCHR3 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 14 | STBCHR2 | 1 | R/W | Write permission to STBCHR2 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 13 | STBCHR1 | 1 | R/W | Write permission to STBCHR1 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 12 | STBCHR0 | 1 | R/W | Write permission to STBCHR0 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 11 | RSTFR3 | 1 | R/W | Write permission to RSTFR3 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 10 | RSTFR2 | 1 | R/W | Write permission to RSTFR2 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 9 | RSTFR1 | 1 | R/W | Write permission to RSTFR1 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 8 | RSTFR0 | 1 | R/W | Write permission to RSTFR0 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 7 | — | 1 | /W | Reserved<br>These bits are always read as 1. The write value should always be 1. |
| 6 | SRESCR0 | 1 | R/W | Write permission to SRESCR0 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 5 | RSTOUTCR | 1 | R/W | Write permission to RSTOUTCR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 4 | WDTRSTCR | 1 | R/W | Write permission to WDTRSTCR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 3 to 0 | — | All 1 | R | Reserved<br>These bits are always read as 1. The write value should always be 1. |

**[RSTDnWACR (n=1 to 3)]**

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 | RSTPTERADR | 0 | R/W | Write permission to RSTPTERADR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 26 | RSTPTCSR | 0 | R/W | Write permission to RSTPTCSR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 25 to 23 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 22 | ICUMXCPCR | 0 | R/W | Write permission to ICUMXCPCR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 21 | ICUMXBAR | 0 | R/W | Write permission to ICUMXBAR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 20 | APBSFTYCHKR | 0 | R/W | Write permission to APBSFTYCHKR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 19 | STBCHR7 | 0 | R/W | Write permission to STBCHR7 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 18 | STBCHR6 | 0 | R/W | Write permission to STBCHR6 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 17 | STBCHR5 | 0 | R/W | Write permission to STBCHR5 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 16 | STBCHR4 | 0 | R/W | Write permission to STBCHR4 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 15 | STBCHR3 | 0 | R/W | Write permission to STBCHR3 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 14 | STBCHR2 | 0 | R/W | Write permission to STBCHR2 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 13 | STBCHR1 | 0 | R/W | Write permission to STBCHR1 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 12 | STBCHR0 | 0 | R/W | Write permission to STBCHR0 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 11 | RSTFR3 | 0 | R/W | Write permission to RSTFR3 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 10 | RSTFR2 | 0 | R/W | Write permission to RSTFR2 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 9 | RSTFR1 | 0 | R/W | Write permission to RSTFR1 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 8 | RSTFR0 | 0 | R/W | Write permission to RSTFR0 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 7 | SRESCR1 | 0 | R/W | Write permission to SRESCR1 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 6 | SRESCR0 | 0 | R/W | Write permission to SRESCR0 register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 5 | RSTOUTCR | 0 | R/W | Write permission to RSTOUTCR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 4 | WDTRSTCR | 0 | R/W | Write permission to WDTRSTCR register from domain n.<br>0 : Prohibit write access.<br>1 : Permit write access. |
| 3 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

## 14.3 Operation

### 14.3.1 Power-On Reset by PRESET# Pin

- For the power-on oscillation settling time (tOSC), see Section 200, Electrical Characteristics. After the state on the PRESET# pin input is changed from a low level to high level, the internal reset state continues until the flash ROM reset time (longer than 50 µs), the flash ROM holding time (longer than 5 µs) and the reset holding time
- Figure 14.3.1 shows a sequence for turning on the power. When the power is turned on, ensure that a low level is input to the PRESET# pin. A low level input is also needed on the TRST# pin.

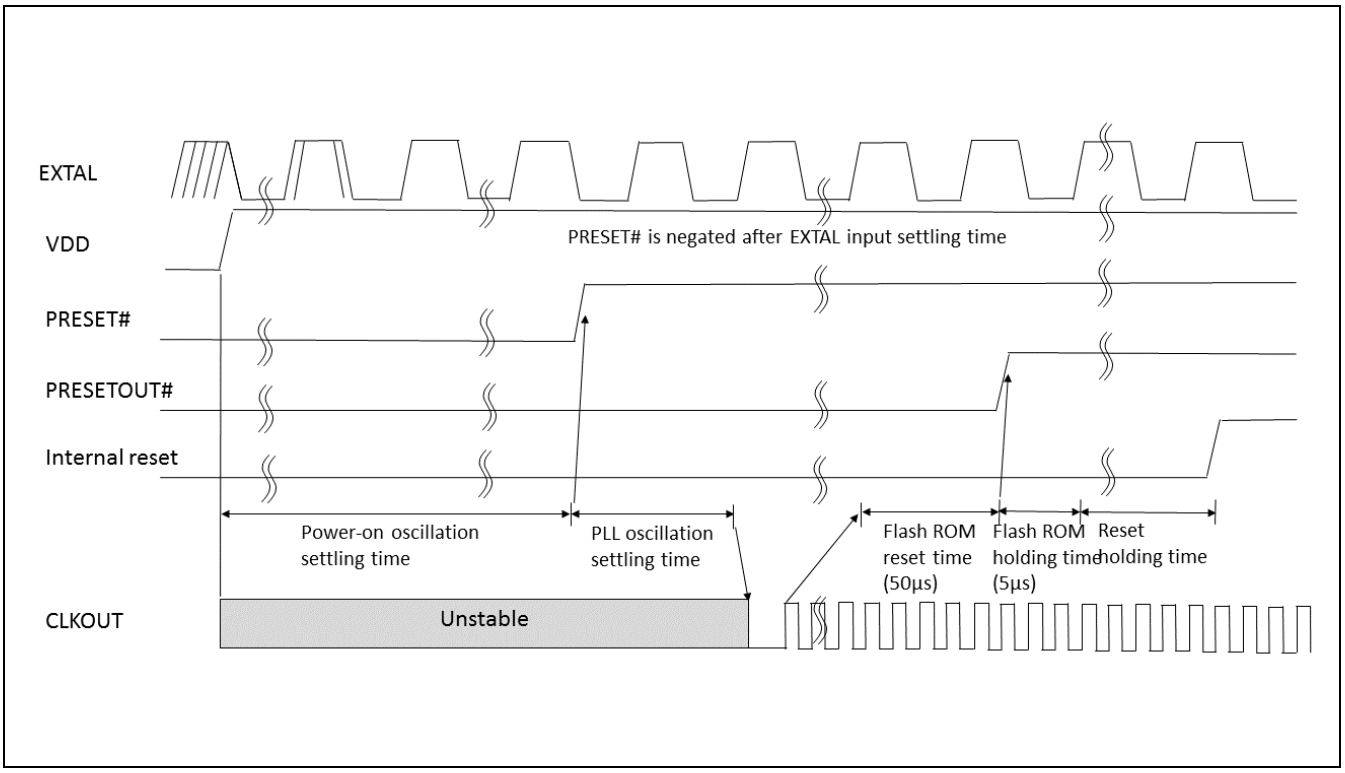



**Figure 14.3.1 Free-Running Mode**

### 14.3.2 WDT reset

WDT resets can be issued by the RWDT and System WDT modules.
WDTRSTCR controls the permission or prohibition of WDT resets.
The reset (RST) module is not initialized by a WDT reset.

## 14.4 Usage Notes

### 14.4.1 Bus Access Protection for Reset (RST) registers

**(1) Overview**

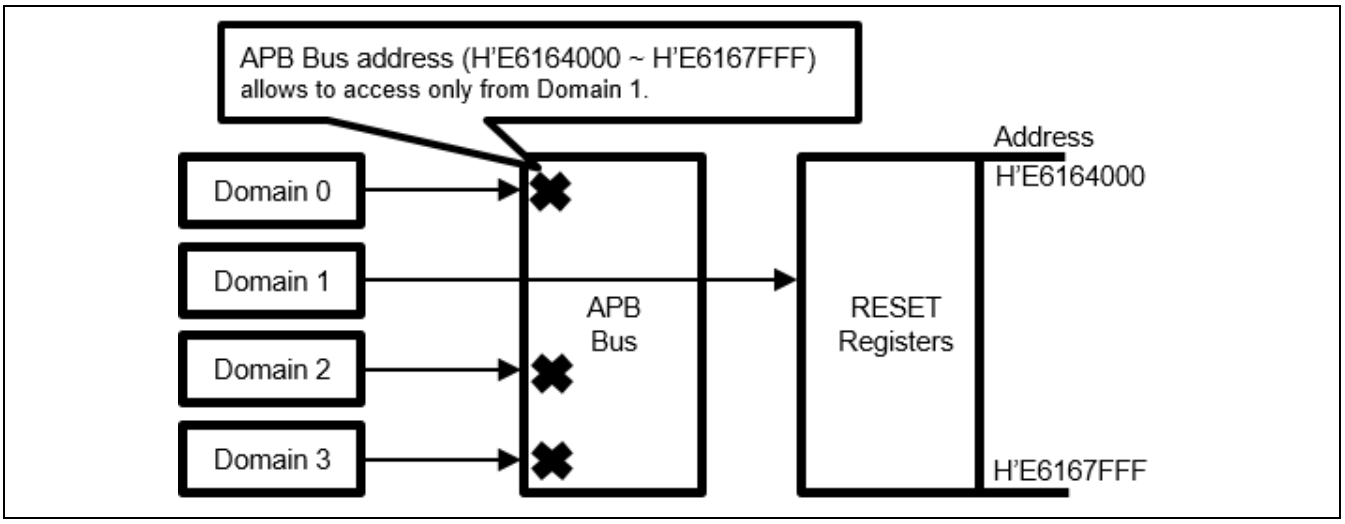



**Figure 14.4.1 Bus Access Protection for Reset (RST) registers**

Reset (RST) register address map is assigned from H'E6160000 to H'E616FFFF. The address map is separated into 4 groups which is accessed by each domain (Domain 0, Domain 1, Domain 2, Domain 3). The Physical address of Reset (RST) register body from 0x0000_0000 to 0x0000_3FFF.

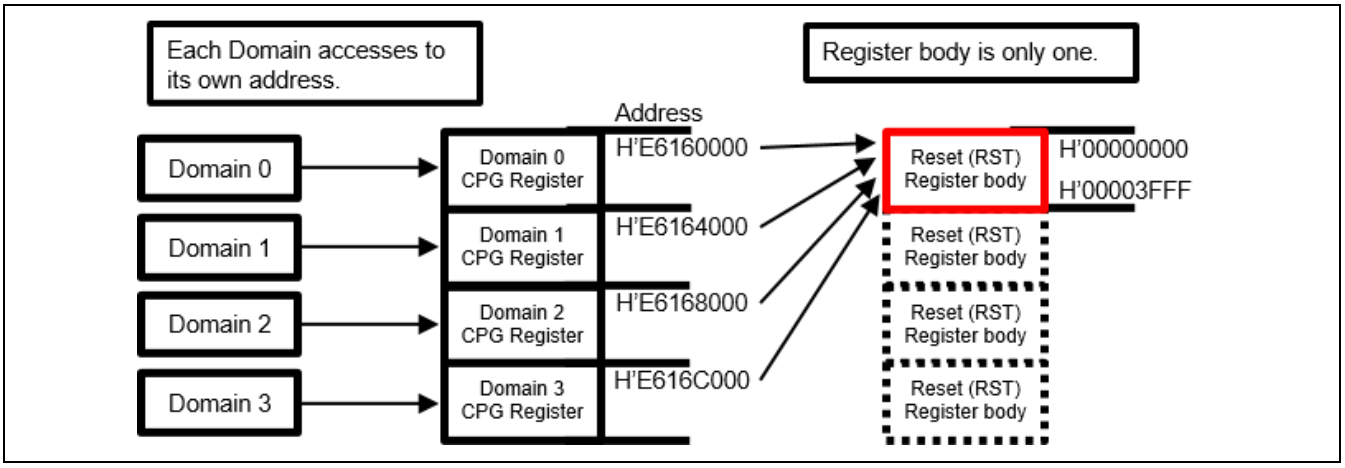



**Figure 14.4.2 Address Map Structures**

### (2) Domain n Write Access Control Register (RSTDnWACR) (n = 0, 1, 2, 3)

Each register in “Reset (RST) Register body” is protected by “Domain n Write Address Control Register (RSTDnWACR (n = 0, 1, 2, 3)). For example of the case below (Figure 14.4.3), D2WACR.bit15 = 1, D{0, 1, 3}WACR.bit15 = 0. Only Domain 2 can write access to “STBCHR3”.

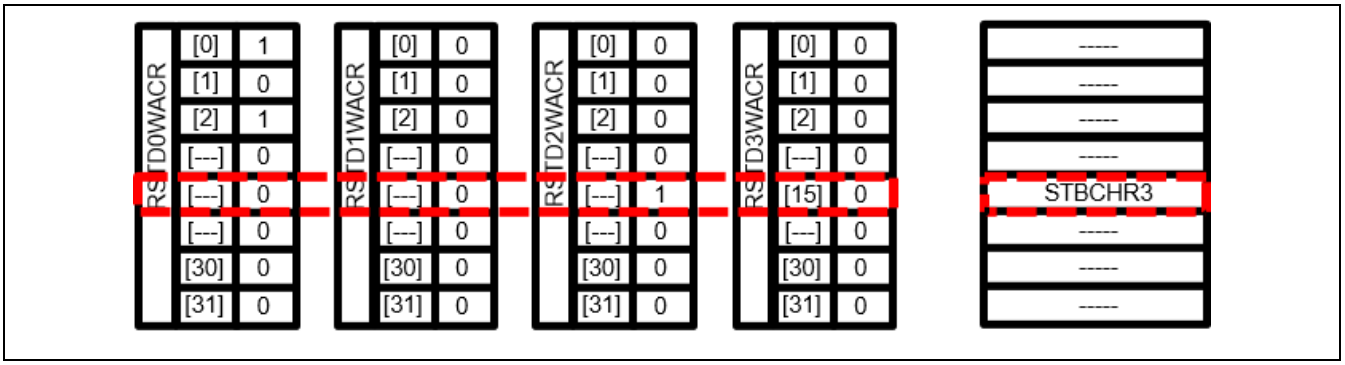



**Figure 14.4.3 Usage of RSTDnWACR (n = 0, 1, 2, 3)**

The register corresponds with RSTD0WACR, RSTD1WACR, RSTD2WACR, RSTD3WACR in Sec.14.2.14.

### (3) Check method of illegal write accesses

If there are illegal write accesses for the Reset (RST) registers, an error notification is reported to ECM (ECMERRxxxR00 Bit31). In addition, the error notification is cleared by using RSTPTCSR (Protect Control/Status Register). In detail, see 14.2.12, in which the register usage is described.

## 14.5 Safety Mechanisms

### 14.5.1 Overview

- The safety mechanism is supported by PRESETOUT0#. It can be judged by monitoring PRESETOUT0# whether Reset (RST) module has any failures.
- The access rights for changing independent registers in Reset (RST) shall be configurable for each Domain 0, 1, 2, 3. See 14.5.3 in detail.

### 14.5.2 PRESETOUT#

The safety mechanism is supported by PRESETOUT#. It can be judged by monitoring PRESETOUT# whether Reset (RST) module has any failures. When the boot sequence is succeeded without any failures, PRESETOUT# value is changed from LOW to HIGH. In addition, it can be judged that the Power on Self Test is succeeded.

### 14.5.3 Bus Access Protect

Reset (RST) provides a safety access protection mechanism to guarantee “Freedom From Interference (FFI)” when accessing the registers belonging to the Reset (RST) module. Several masters (e.g. Cortex-R52, Cortex-A76) take control of different portions and modules of the chip. The access protection mechanism shall enable sharing and splitting the functionality among cores.

The access protection mechanism for the control registers belonging to Reset (RST) module shall be aligned with the Region ID concept. In detail, see Sec.14.4.1. Reset (RST) has critical system control functions such as Reset Control Function. Therefore, the strong access restrictions are necessary for Reset (RST).

# 15. Interrupt Controller (INTC)

## 15.1 Overview

This module handles system and realtime domain interrupt events.

This module handles system domain interrupt events. This module is integrated Arm GIC-600 (r1p6) which is compliant with Arm GICv3.

### 15.1.1 Features

- There is one INTC-Monitor for one interrupt signal of the safety-related module
- INTC-Monitors use a dedicated clock from the Timer clock generator.
- Clock frequency of the Timer Clock generator can be configured by the Counter control register. The Counter control register and its access error status registers can be accessed in secure mode. (Non-secure access is prohibited.)
- When non-secure access to the secure registers are occurred, the access error signal is output.
  - The error's factor is kept in register
- All INTC-Monitors have each control registers and status bits.
- Each control registers and status bits of INTC-Monitors can be accessed in non-secure mode.
- INTC-Monitor can monitor the following two items.
  - Whether the interrupt signal is asserted within the expected time
  - Whether the interrupt signal is de-asserted within the expected time
- When the expected time is exceeded, the error signal is output.
  - The error's factor is kept in status register
  - The error signals are output from status registers respectively
- INTC-AP handles system domain interrupt events.
  - Interrupt controller for AP-System core
  - 32 programmable priority levels
  - Arm-IRQ interrupt generation
  - INTC-AP handles peripheral interrupt
  - GIC600 (r1p6) implemented
- Handles the external NMI/IRQ as a common module
- Edge-triggered by rising edge, falling edge or both edges
- Level-sensitive to the high or low values

- Provides IRQ signal de-bouncing, noise reduction and chattering

### 15.1.2 Block Diagram

Figure 15.1 show block diagram of connection between INTC and related modules. INTC module consists of synchronization, selector and INTC-Monitor parts.

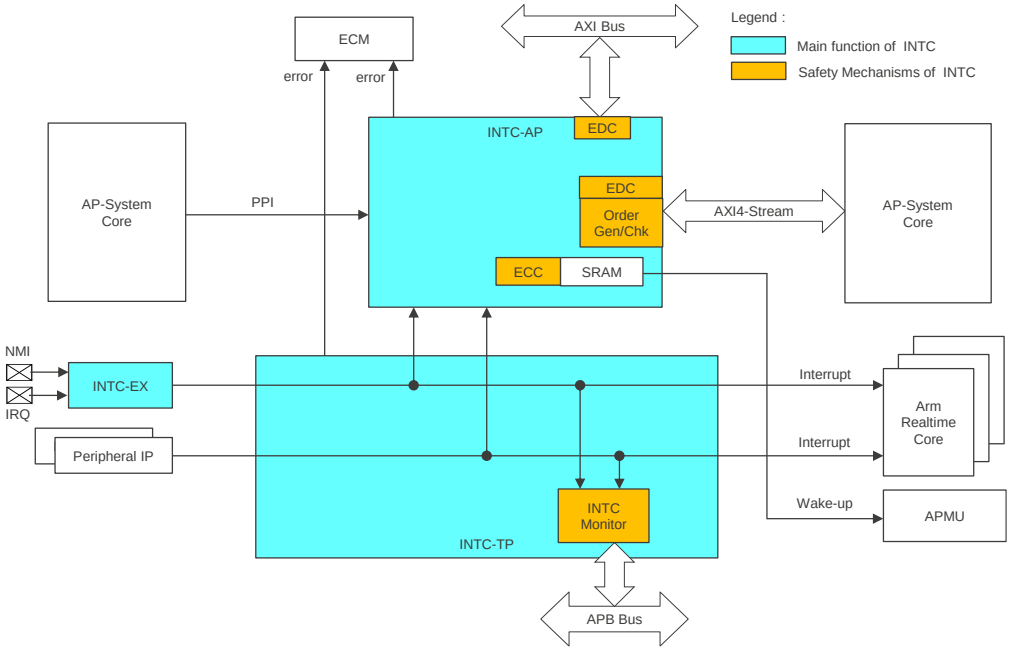



**Figure 15.1 Block diagram of connection between INTC and related modules**

### 15.1.3 External Pins

Table 15.1 shows the INTC-EX pin configuration.

**Table 15.1 Pin Configuration**

| Name | Function | I/O | Description |
|---|---|---|---|
| IRQ[d] (d=0-5) | External interrupt input pins | Input | Input of IRQ[d] interrupt request signals from externaldevice. All IRQ signals are always active. |
| NMI | External non maskable interrupt input pins | Input | Input of NMI interrupt request signal from external device. |

### 15.1.4 Conntected Module

Table 15.2, Table 15.3 and Table 15.4 shows the connected modules to INTC. As for the feature of the connected modules, refer to the chapter of each module.

**Table 15.2 Connected Modules of INTC-AP**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| INTC-AP | AP system core | Application CPU core block |
| | APMU | Send a request to wake up the Application CPU core to APMU. |
| | AXI-BUS | AXI-Bus, APB-Bus |
| | INTC-TP | Collect interrupts from each IPs, INTC-EX |
| | CPG | Clock Supply |
| | RST | Module Reset |
| | ECM | Error Interrupt Controller |

**Table 15.3 Connected Modules of INTC-TP**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| INTC-TP | IPs | Interrupt collection from IPs |
| | Realtime core | Realtime CPU core |
| | INTC-AP | Interrput controller for Application CPU core |
| | INTC-EX | Provide interrupts from external device |
| | AXI-BUS | APB-Bus |
| | CPG | Clock Supply |
| | RST | Module Reset |
| | ECM | Error Interrupt Controller |

**Table 15.4 Connected Modules of INTC-EX**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| INTC-EX | INTC-TP | Collect interrupts from each IPs, INTC-EX |
| | AXI-BUS | APB-Bus |
| | CPG | Clock Supply |
| | RST | Module Reset |

### 15.1.5 Register configuration

Refer to R-CarV4M_UM_015_INTC_r0p81.xlsx.

## 15.2 Register description

Refer to R-CarV4M_UM_015_INTC_r0p81.xlsx.

## 15.3 Operation

### 15.3.1 Interrupt Description

Refer to 015_R-CarV4M_INTC_Interrupts_Mapping_r0p81.xlsx

as interrupt table.

## 15.4 Usage Notes

### 15.4.1 INTC-AP Register Configuration and Function Description

INTC-AP is based on Arm Generic Interrupt Controller. For more information, see Arm Generic Interrupt Controller Architecture Specification and section 5, AP-System Core.

960 SPI interrupts are integrated into Interrupt ID [959:32]. Also 256 LPIs (Locality-specific Peripheral Interrupts) are supported.

The INTC-AP supports 4 CPU cores which are mapped as shown below.

**Table 15.5 Port mapping for INTC-AP**

| | |
|---|---|
| CPU0 interface | AP-System core (Cortex-A76) CPU core 0 |
| CPU1 interface | AP-System core (Cortex-A76) CPU core 1 |
| CPU2 interface | AP-System core (Cortex-A76) CPU core 2 |
| CPU3 interface | AP-System core (Cortex-A76) CPU core 3 |
| CPU4 interface | — |
| CPU5 interface | — |
| CPU6 interface | — |
| CPU7 interface | — |

Table 15.6 shows the base address of GIC600.

**Table 15.6 GIC600 Base address**

| Functions | Base Address |
|---|---|
| GIDC/GICDA : Distributor Registers | H'F100_0000 |
| GICD message-based interrupts (GICA) : Distributor registers (GICA) for message-based SPIs | H'F101_0000 |
| GICT : trace and debug functions Registers | H'F102_0000 |
| GICP : Performance Monitoring Unit functions Registers | H'F103_0000 |
| GITS : ITS control register | H'F104_0000 |
| GITS translate : ITS translation register | H'F105_0000 |
| GICR (LPI) : Redistributor registers for control and physical LPIs | H'F106_0000 + CPUn x H'2_0000 |

| | |
|---|---|
| GICR (SGI) : Redistributor registers for SGIs and PPIs | H'F107_0000 + CPUn x H'2_0000 |

For more information of these registers, see Arm Cortex-A76 Technical Reference Manual and Arm GIC-600 (r1p6) Generic Interrupt Controller Technical Reference Manual.

### 15.4.2 Interrupts Mapping

Refer to 015_R-CarV4M_INTC_Interrupts_Mapping_r0p81.xlsx

as interrupt table.

Note: SGI is Software Generated Interrupt, PPI is Private Peripheral Interrupt and SPI is Shared Peripheral Interrupt. All SPI interrupts should be level-sensitive configuration. If you want more detail information, refer to Arm Cortex-A76 Technical Reference Manual and Arm GIC-600 (r1p6) Generic Interrupt Controller Technical Reference Manual.

### 15.4.3 NMI Mask Lock Feature

NMI can be masked by setting NMI Mask Lock Set Register. If you set the lock code to NMI lock code register and set the same lock code to NMI lock set register, then NMI cannot be masked by software (locked NMI mask). If you want to unlock NMI mask, you need to set DBGEN bit of NMI debug control register and set the unlock code to the NMI debug code register.

### 15.4.4 Procedure of NMI Mask Lock

- Lock feature

1. Write the lock code to NMI lock code register
   ex) Write32 (INTC-EX_BASE + H'0A04), H'00A CCE55
2. Write the lock code to NMI mask lock set register
   ex) Write32 (INTC-EX_BASE + H'0A00), H'00AC CE55
3. Write access to NMI lock code register and NMI mask lock set register
   — Check whether these registers cannot be updated

- Unlock feature

1. Set enable bit to NMI debug control register
   ex) Write32 (INTC-EX_BASE + H'0A08), H'0000 0001
2. Write unlock code to NMI debug code register
   ex) Write32 (INTC-EX_BASE + H'0A0C), H'558C CAFF
3. Write access to NMI lock code register and NMI mask lock register
   — Check whether these registers can be updated

### 15.4.5 Instructions for using INTC-AP

For reducing power consumption, software reset for INTC-AP is asserted and module stop for INTC-AP is enabled at after reset released.
Before wake-up AP System Cores, we must negate software reset for INTC-AP and disable module stop for INTC-AP.

This is the procedure of negating software reset and disabling module stop.
For details on these registers, refer to section 8.2 Register Description, in section 8, "Clock Pulse Generator (CPG)" and section 9.2 Register Description, in section 9, "Module Standby, Software Reset".

1. Negate software reset for INTC-AP.

(1) Write H'7FFF_FFFF to the CPG write protect register (CPGWPR).

(2) Write H'8000_0000 to the Software Reset Clearing Register 5 (SRSTCLR5).

(3) Write H'FFF7_FFFF to the CPG write protect register (CPGWPR).

(4) Write H'0008_0000 to the Software Reset Clearing Register 11 (SRSTCLR11).

2. Disable module stop for INTC-AP.

(1) Write H'7FFF_FFFF to the CPG write protect register (CPGWPR).

(2) Write H'8000_0000 to the Module Stop Control Register 5 (MSTPCR5).

### 15.4.6 Restriction of interrupt control in CPU power off

When a CPU core power off like as changing low power mode, all the interrupts for the CPU core target of power off must be deasserted.
This restriction is also described in Arm GIC-600 (r1p6) Generic Interrupt Controller Technical Reference Manual, chapter 3.6.2 “Processor core power management”.

## 15.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

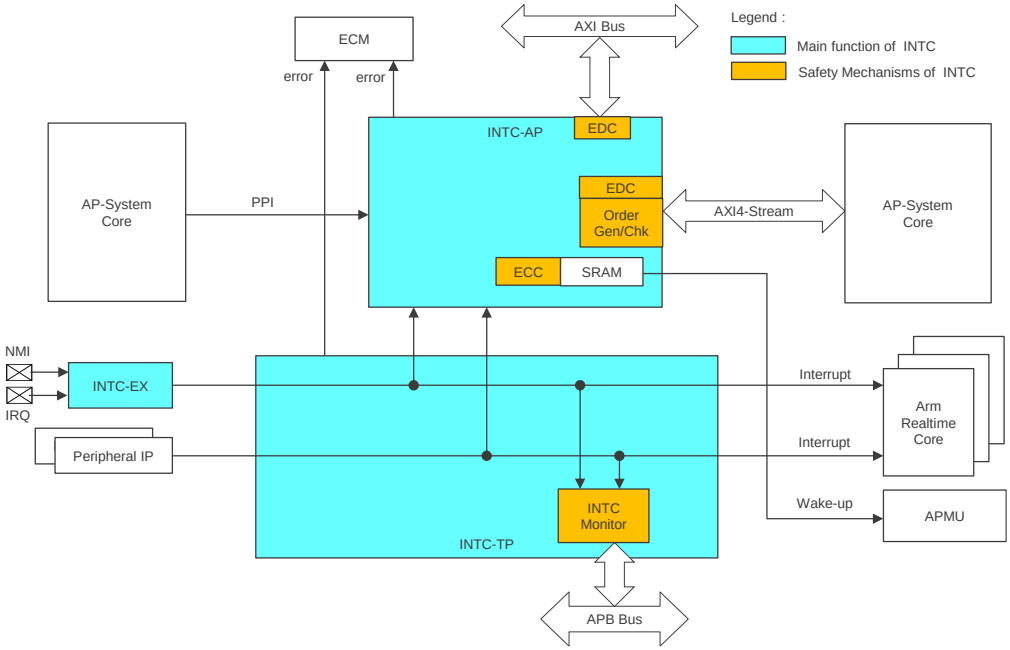



**Figure 15.2 Block diagram of Safety Mechanisms**

### 15.5.1 Block diagram for functional safety

Peripheral's interrupt is implemented by interrupt monitoring

EDC protection for AXI, AXI4-stream and APB BUS channel

### 15.5.2 Safety Mechanism

- Interrupt monitoring
- EDC protection in BUS channel

# 16. Multifunctional Interface (MFIS)

## 16.1 Overview

The multifunctional interface (MFIS) provides an interface between the AP System Core domain and the Realtime Core domain. The MFIS supports the issuing of interrupts for the AP System Core domain and the Realtime Core domain by each core.

### 16.1.1 Features

- By writing to specific bits of the internal registers of the MFIS from the AP System Core domain, interrupts can be sent to the Realtime Core domain. Similarly, interrupts can also be sent from the Realtime Core domain to another core domain.
- Different software-controlled interrupts are provided to each CPU and enabling high-speed data transfer using interrupts.
- The resource shared with each CPU cores can be realized by use of the lock register.
- Access Protection of Security supported
- PRR Product Register inside (individual address)

**Indices**

| Index | Description |
|---|---|
| i | Throughout this section, the individual number of AP System cores are identified by the index “i”.<br>And “i” means the number for sender/receiver communication between AP system Core and Realtime core. |
| k | Throughout this section, the individual number of Realtime cores are identified by the index “k”.<br>And “k” means the number for sender/receiver of communication between AP system Core and Realtime core. |
| m | Throughout this section, “m” means the number for sender of communication between Realtime cores. |
| n | Throughout this section, “n” means the number for receiver of communication between Realtime cores. |
| j | Throughout this section, “j” means the number of MFIS lock register. |

### 16.1.2 Block Diagram

Figure 16.1 shows a block diagram of the MFIS.

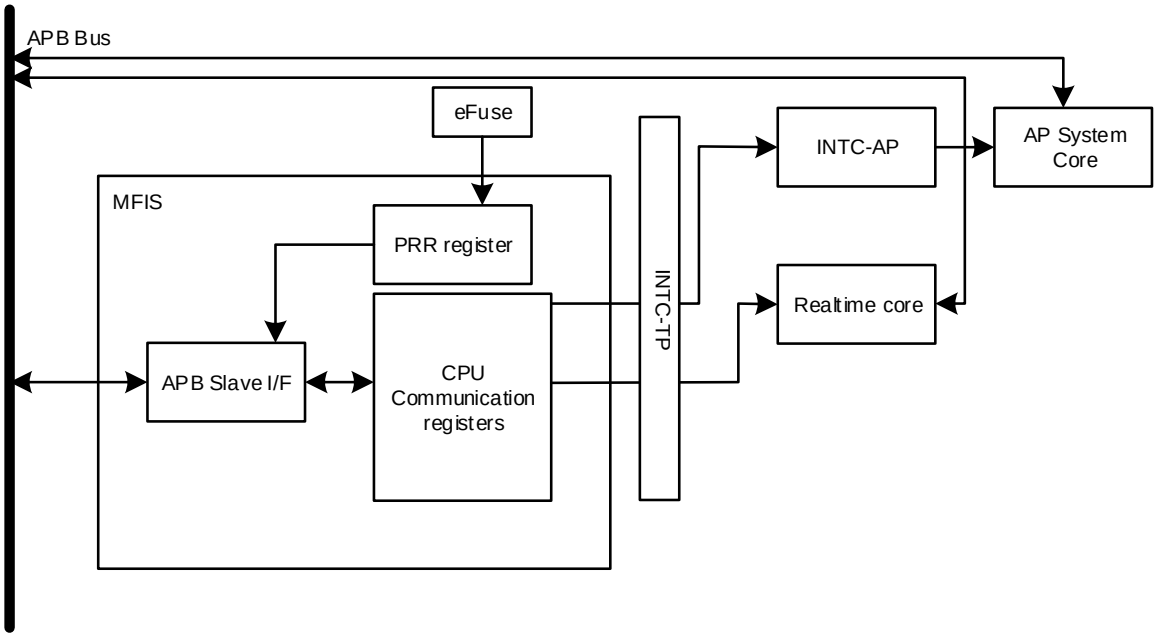



**Figure 16.1 MFIS Block Diagram**

### 16.1.3 External Pins

There are no external pins in this module.

### 16.1.4 Connected Module

Table 16.1 shows the connected modules to MFIS. As for the feature of the connected modules, refer to the chapter of each module.

**Table 16.1 Connected Modules**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| MFIS | AP System Core | AP System Core |
| | Realtime Core | Realtime Core |
| | AXI-BUS | APB-Bus |
| | CPG | Clock Supply |
| | RST | Module Reset |
| | INTC | Interrupt handling |

### 16.1.5 Register configuration

Refer to R-CarV4M_UM_016_MFIS_r0p81.xlsx

## 16.2 Register description

Refer to R-CarV4M_UM_016_MFIS_r0p81.xlsx

## 16.3 Operation

### 16.3.1 Mutex Control

To implement exclusive access to resources shared among the AP System Core, Realtime Core, MFIS lock registers (MFISLCKRj [j=0-63]) can be used to apply mutex control. An example of the procedure for mutex control for the two CPUs by using the lock registers is given below.

1. For example, if the CPU0 is to use resource A, the CPU reads MFISLCKR0 (here we assume that MFISLCKR0 is used for resource A).
2. The CPU0 reads the LCK bit in MFISLCKR0 to confirm that its current value is 0. This causes setting of the LCK bit in MFISLCKR0 to 1.
3. The CPU0 proceeds to use resource A. As long as resource A is in use by the CPU0, the CPU1 reads MFISLCKR0 before attempting to start using resource A. The value 1 read from the LCK bit (indicating that resource A is in use) indicates that resource A is not available to the CPU1.
4. When the CPU0 is finished with resource A, it sets the LCK bit in MFISLCKR0 to 0 (indicating that resource A is not currently in use).

## 16.4 Usage Notes

No description on this section.

## 16.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 16.5.1 Hardware Redundancy for MFIS

#### (1) Overview

Hardware redundancy is applied to the MFIS in order to detect fault of functional logic.

#### (2) Hardware Description

Figure 16.2 shows a block diagram of MFIS protection for functional safety. Not only the body of MFIS logic but also the control registers are checked by lock-step operation. When an error related to the functional safety is detected, this error is reported through the external system.

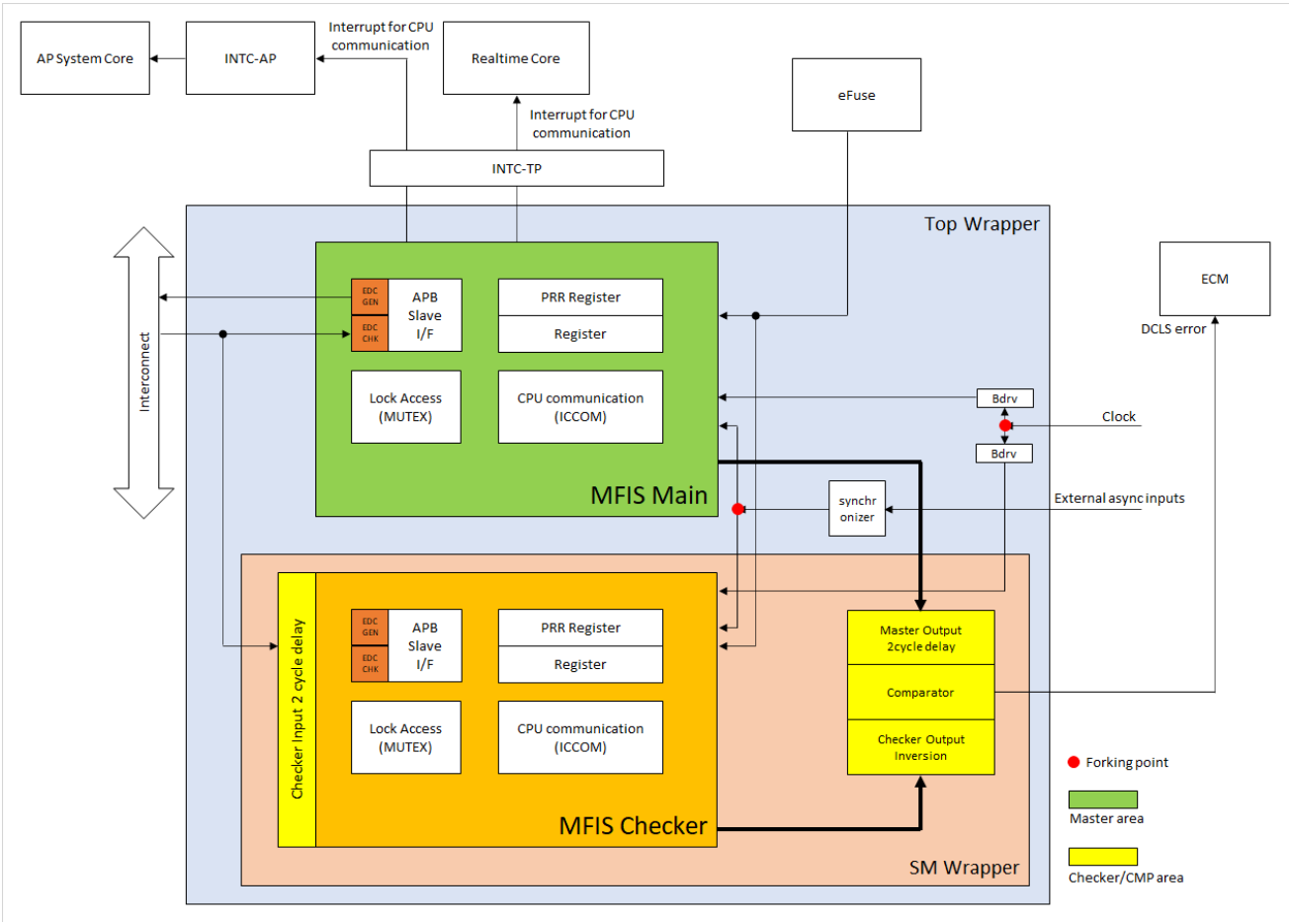



Figure 16.2 Block diagram of MFIS protection for Functional Safety

### 16.5.2 Region ID Protection for MFIS

#### (1) Overview

A Region ID Protection (RGID) provides “Freedom From Interfere (FFI)”.
This safety mechanism checks a consistency of master and slave. A Region ID protection is a function of safety mechanism that prevents unexpected access.
For details, refer to section 18, AXI-Bus.

#### (2) Hardware Description

MFIS is provided Region ID Protection each 4KB block can set its own Region ID.
To apply Region ID Protection, refer to section 18.5.9, Region ID protection, in section 18, AXI-Bus.
Table 16.2 shows register assignment for each Region ID block in MFIS and related register in AXI-Bus.

Before access to registers in MFIS, apply Region ID Protection set as below.

1. Set Region ID Protection setting for each MFIS region by RGIDR_MODID register.
   RGIDR_MODID[15:0] bits can set protection access from the master set Region 15 to 0.
   Ex: To set read permission only a master Region ID = 1 to Address area of “MFIS - Region 1”,
   write 32’h0000 0002 to RGIDR_MODID[RGIDR_MFI01] register.
2. Set Region ID for each master by RGIDM_MODID register.
   RGIDM_MODID[3:0] bits can set Region ID of the master.
   Ex: To set Realtime core [k] Region ID = 1, write 32’h0000 0001 to RGIDM_MODID[RGIDM_CR52SS[k]] register.

**Table 16.2 Register assignment for each Region ID block**

| Category | Target Register | Register Address | Function | Comment in Address Map | Related Register in AXI-Bus |
|---|---|---|---|---|---|
| Common | MFISLCKR0～63<br>MFIEDCSIDADDR<br>MFIEDCSIDWDATA<br>MFISOFTMDR<br>MFISBTSTSR<br>MFISCMPERRSTSR<br>MFISWPCNTR<br>MFISWACNTR<br>MFIERRINJ | H’E626_00C0<br>～H’E626_0944 | Access from Both CPU (Cortex-A76 CPU core i [i=0,1,2,3], Cortex-R52 core k [k = 0,1,2]) | MFIS - Region 0 | Read:<br>RGIDR_MODID[RGIDR_MFI00]<br>Write:<br>RGIDR_MODID[RGIDW_MFI00] |
| Independent | MFISARIICR[k]0<br>MFISARIMBR[k]0 | H’E626_1400～<br>H’E626_1500 | Access from Cortex-A76 CPU core 0 and Cortex-R52 core k [k = 0,1,2] only | MFIS - Region 1 | Read:<br>RGIDR_MODID[RGIDR_MFI01]<br>Write:<br>RGIDR_MODID[RGIDW_MFI01] |
| Independent | MFISARIICR[k]1<br>MFISARIMBR[k]1 | H’E626_2408～<br>H’E626_2508 | Access from Cortex-A76 CPU core 1 and Cortex-R52 core k [k = 0,1,2] only | MFIS - Region 2 | Read:<br>RGIDR_MODID[RGIDR_MFI02]<br>Write:<br>RGIDR_MODID[RGIDW_MFI02] |
| Independent | MFISARIICR[k]2<br>MFISARIMBR[k]2 | H’E626_3410～<br>H’E626_3510 | Access from Cortex-A76 CPU core 2 and Cortex-R52 core k [k = 0,1,2] only | MFIS - Region 3 | Read:<br>RGIDR_MODID[RGIDR_MFI03]<br>Write:<br>RGIDR_MODID[RGIDW_MFI03] |
| Independent | MFISARIICR[k]3<br>MFISARIMBR[k]3 | H’E626_4418～<br>H’E626_4518 | Access from Cortex-A76 CPU core 3 and Cortex-R52 core k [k = 0,1,2] only | MFIS - Region 4 | Read:<br>RGIDR_MODID[RGIDR_MFI04]<br>Write:<br>RGIDR_MODID[RGIDW_MFI04] |
| Independent | MFISAREICR0[i]<br>MFISAREMBR0[i]<br>MFISRREICR0[n]<br>MFISRREMBR0[n]<br>(n = 1,2) | H’E626_9404～<br>H’E626_9474 | Access from Cortex-A76 CPU core i [i=0,1,2,3], Cortex-R52 core n [n = 0,1,2] only | MFIS - Region 9 | Read:<br>RGIDR_MODID[RGIDR_MFI09]<br>Write:<br>RGIDR_MODID[RGIDW_MFI09] |

| Category | Target Register | Register Address | Function | Comment in Address Map | Related Register in AXI-Bus |
|---|---|---|---|---|---|
| Independent | MFISAREICR1[i]<br>MFISAREMBR1[i]<br>MFISRREICR1[n]<br>MFISRREMBR1[n]<br>(n = 0,2) | H'E626_A424<br>～<br>H'E626_A484 | Access from Cortex-A76 CPU core i [i=0,1,2,3], Cortex-R52 core n [n = 0,1,2] only | MFIS - Region 10 | Read:<br>RGIDR_MODID[RGIDR_MFI10]<br>Write:<br>RGIDR_MODID[RGIDW_MFI10] |
| Independent | MFISAREICR2[i]<br>MFISAREMBR2[i]<br>MFISRREICR2[n]<br>MFISRREMBR2[n]<br>(n = 0,1) | H'E626_B464<br>～<br>H'E626_B494 | Access from Cortex-A76 CPU core i [i=0,1,2,3], Cortex-R52 core n [n = 0,1,2] only | MFIS - Region 11 | Read:<br>RGIDR_MODID[RGIDR_MFI11]<br>Write:<br>RGIDR_MODID[RGIDW_MFI11] |

# 17. Error Control Module (ECM)

## 17.1 Overview

The ECM is to collect all hardware errors and notify to the system and decide the reaction of the specified error.

### 17.1.1 Features

Error Control Module (ECM) has the following feature

- Collect all hardware errors and notify to the system
- Decide the reaction of the specified error.
  - Assert interrupt internal or external
  - RAM error count is implemented
- External error control logic is implemented
  - Control error assert/negate timing
  - Initial condition of external error signal is active
- Support generating error interrupt by software
- Dynamic error notification mode supported
- 4x additional error notification signal supported
- APB bus is implemented for register access
- 43x Error Control Registers are implemented in ECM.
- Up to 2048 error can be handled.

### 17.1.2 Block Diagram

Figure 17.1 shows a block diagram of the ECM.

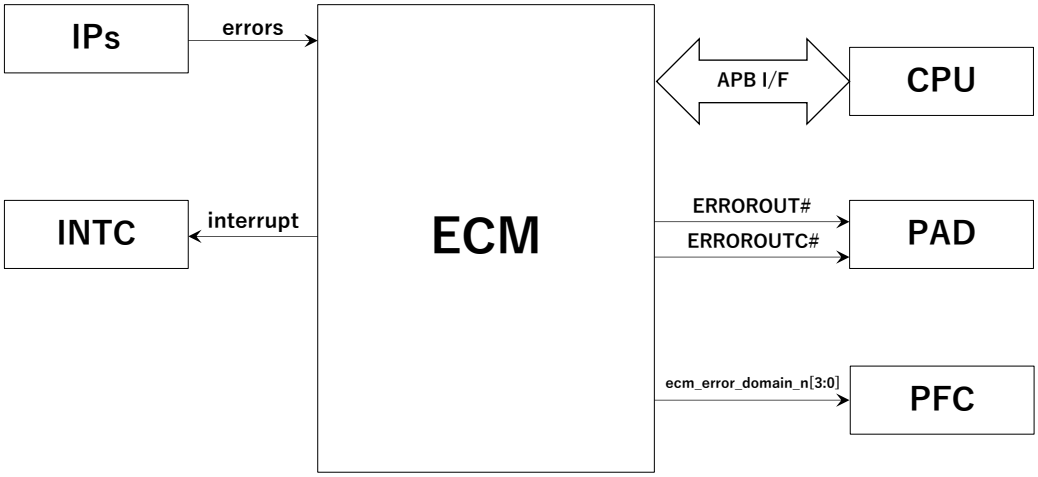



**Figure 17.1 ECM Block Diagram**

### 17.1.3 External Pins

Table 17.1 shows the pin configuration of ECM.

Table 17.1 Pin Configuration

| Name | Function | I/O | Description |
|---|---|---|---|
| ERROROUT# | ERROR OUT pin | output | Error notification for external system. |
| ERROROUTC# | ERROR OUT control | output | Error notification for external system, PFC. |

### 17.1.4 Connected Module

Table 17.2 shows the connected modules to ECM. As for the feature of the connected modules, refer to the chapter of each module.

Table 17.2 Connected Modules

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| ECM | IPs | Error collection from IPs |
| | AXI-BUS | APB-Bus |
| | CPG | Clock Supply |
| | RST | Module Reset |
| | PFC | Pin Function |
| | INTC | Interrupt handling |

### 17.1.5 Register configuration

## 17.2 Register description

Note: Don't access (write) the registers described as “reserved” in register description.

(Refer to R-CarV4M_UM_017_ECM_r0p81.xlsx)

This attachment file shows bit assignment of ECMERRxxxR[n].

(Refer to 017_R-CarV4M_ECM_Map_table_V4M_for_D_r0p81.xlsx)

Note:
In R-Car V4M, there is some errors that is “Initial Value” column is “1” but “ECMERRxxxRn meanings of this error [R-Car V4M]” is Reserved. Therse errors are removed from R-Car V4H but keeping the initial value of ECMERRCTLR[n]=1 in R-Car V4M.
So these errors aren’t asserted in R-Car V4M and it’s no problem to change the ECMERRCTLR[n]=0.

## 17.3 Operation

No description on this section.

## 17.4 Usage Notes

No description on this section.

## 17.5 Safety Mechanisms

### 17.5.1 Error Management System

#### (1) Overview

The ECM module supports error management in order to notify either the system interrupt controller (INTC) or the external systems of hardware errors, such as RAM ECC errors.

#### (2) Hardware Description

#### (a) Error Checking function

The enable bit of ECMERRCTLR is set, the corresponding error can be reported to ECM and the status of the corresponding error is set in ECMERRSTSR. The reporting target which is notified of the error status can be selected from the INTC and the external pin by the corresponding bit of ECMERRTGTR.
Figure 17.2 shows the block diagram of error checking function.

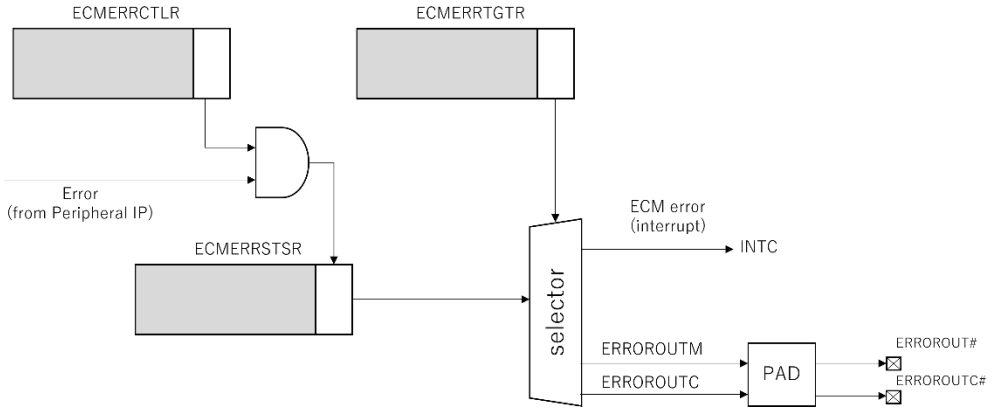



**Figure 17.2 ECM Block Diagram of Error Checking function**

1. Set 1'b0 to a corresponding error bit in ECMERRTGTR[n] register.
   When this bit is set 1'b1, error detection target is set to INTC.

2. Set 1'b1 to a corresponding error bit in ECMERRCTLR[n] register to be enable error detection.
   Not to set this bit before setting for ECMERRTGTR[n] register isn't completed.

#### (b) Pseudo Error Insertion

For the sake of software development of error notification to the external systems, the ECM module is equipped with the pseudo error generation logic. If you want to generate a hardware error and notify either the system interrupt controller (INTC) or the Control Domain of the error by using this function, you have to set both the safety error insertion control register (SAFCTLR) and safety set error insertion enable register (SAFSTERRENR).

Before generating pseudo error notification, the following procedure is required. At first, set "1" to SAFCTLR.DBGEN and select the target error registers by setting SAFCTLR.REGSEL [5:0]. After setting "1" to the target bit of SAFSTERRENR, the corresponding error is generated. If the corresponding bit of SAFCLERRENR is set to "1", then clear the status of this error notification. By checking the value of the corresponding bit of "SAFSTSR", it can be checked whether error notification is generated.

**(c) Write Protection**

In order not to update the functional registers by mistake, the write access protection function need to introduce.

If the target address is the same as the value of the target address field, then update target register and clear the internal access flag. If not, internal access flag is only cleared. Once the write access fail, the target address needs to be set in ECMWACNTR register again. When “WPD” bit in ECMWPCNTR register is set, the write access can be done without any permission.

Figure 17.3 shows the general flow of the write access of the target registers.

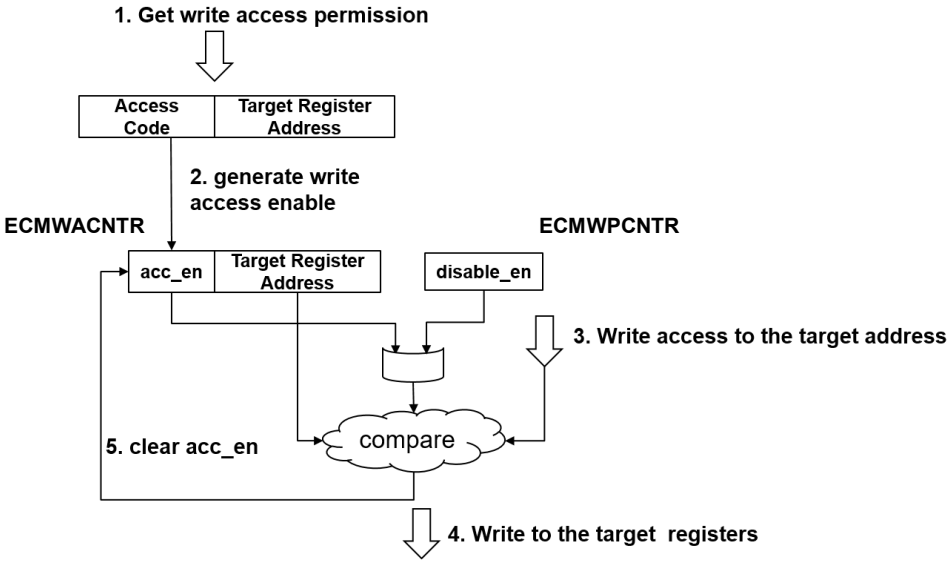



**Figure 17.3 Data flow of write access of the target registers**

**(d) Error Count function**

In case that the internal RAM is protected by ECC, the number of the 1-bit or multi-bit errors can be count-up. When the number of the errors exceed the maximum number of errors, then ECM issues the interrupt related to this error. The maximum number of errors can be controlled by software.

Below is detailed information about the error count function.

- When the bit for ECMERRCTLR[n] corresponding to an ECM error is set to 1, enabling ECM error detection, the error count for ECMERRCNTR[m], which is subject to the ECM error, becomes active.

- When an ECC 1-bit Error is detected in the RAM of the module subject to the ECM error, CECN_1bit is incremented. Clearing the cause of the error at the source will result in an increment upon the next error occurrence. (※1)

- An error notification is sent to ECM when the count value reaches the same as the MECN_1bit setting value.

- When an ECC multi-bit Error is detected, CECN_mbit is incremented. Clearing the cause of the error at the source will result in an increment upon the next error occurrence. (※1)

- An error notification is sent to ECM when the count value reaches the same as the MECN_mbit setting value.

- For ECM errors, errors are separated into ECC 1-bit Error and ECC multi-bit Error, with each error assigned its respective counter.

- For ECM errors where ECC 1-bit Error and ECC multi-bit Error are not separated, it indicates that an error notification is sent when the error count reaches MECN_1bit (1-bit Error) or MECN_mbit (multi-bit Error).

- The behavior regarding the ECM error counter is the same regardless of the ECM error.

About the list of the target modules which is controlled by each ECM Error Count Registers, refer to "V4MECMERRCNTR_UM" sheet in the attachment file, "017_R-CarV4M_ECM_Map_table_V4M_for_D_r0p71.xlsx".

※1: To continuously count the error counter, the operation flow of error occurrence,
count-up → error cause clearing → error occurrence count-up
is required.
Errors occurring before the clearing of the error cause cannot be detected, hence there are limitations such as not observing cases of frequent occurrences in a short time.

[Additional Notes on Error Counter]

- When the combination of MECN_1bit and CECN_1bit as well as MECN_mbit and CECN_mbit reach the same value, the count value is reset to 0x1.

- These counters are cleared to 0 and the counting operation continues when a 1 is written to the corresponding bit in ECMERRSTSR[n] to clear the error.

- After the counter value reaches the 5-bit length full b'11111', the counting operation returns to 0x1, and counting does not continue after reaching full.

**(e) Port Safe State Function**

To keep safe operating state (Port Safe State) in case of failure, and then stop it, Port Safe State operate to receive fault information from ECM and make Pin for each domain.

In details, refer to section 7.3.9, Port Safe State Function, in section 7, Pin Function (PFC/GPIO).

**(f) Control External Error Request**

To ensure that external system controller receive this error request, need to be controlled the assert/de-assert timing of the external error request.

Once the external error signal is asserted, then kick the internal timer of the external request holding. When the enable bit of the ECMEXTRQHLDCNTR register is set, the external error signal cannot be de-asserted until the internal timer is overflowed.

On the other hand, once the external error signal is de-asserted, then kick the internal timer of the external request masking. When the enable bit of the ECMEXTRQMSKCNTR register is set, the external error signal cannot be asserted until the internal timer is overflowed. Figure 17.4 shows the block diagram of the external error request control feature.

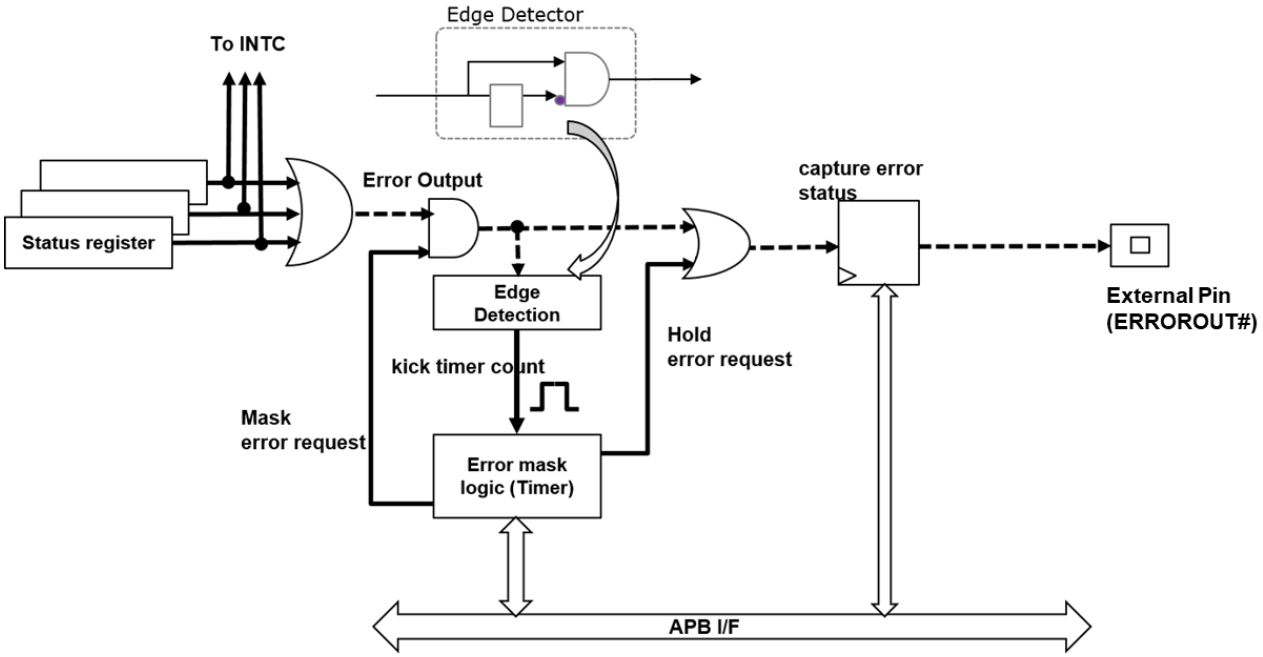



**Figure 17.4 Block Diagram of the external error request control**

# 18. AXI-Bus

## 18.1 Overview

AXI-bus is the On-Chip interconnect bus based on the AMBA® Advanced extensible Interface (AXI) Protocol Specification. AXI-bus has the hierarchical structure and there are the following domains.

- Peripheral domain AXI
- HSC (High speed Communication) domain AXI
- RT (Real Time) domain AXI
- VIO (Video IO) domain AXI
- VC (Video Codec) domain AXI
- VIP (Vision IP Processor) domain AXI
- CPU domain AXI
- GSX (Graphics) domain AXI
- IMP domain AXI
- Main Memory domain AXI
- Slave Access Bus

**Features**

- Bus protocol: AXI3 with QoS Control
- Source clock: S0D2_{domain}φ, S0D3_{domain}φ, SVD1_{domain}φ, VIOBUSφ, VCBUSφ, MSOφ, SASYNCPER2φ, SASYNCRTφ, CL16Mφ
- Bus width: 32bits/64bits/128bits/256bits/512bits
- Quality-of-Service (QoS)
- Provide flexible SDRAM address mapping. Linear mapping allocates a continuous address space to every single memory controller. Split mapping provides an interleaved address space to four memory controller at 4KB granularity.
- Decompression of lossless compressed image
- Virtualization and Security Support
  - Access protection based on Region ID setting for FFI (Freedom From Interference).
  - Access protection based on secure attribute.
  - Provide Bus Domain Protection.
- Low Power function with dynamic module stop function

### 18.1.1 Block diagram

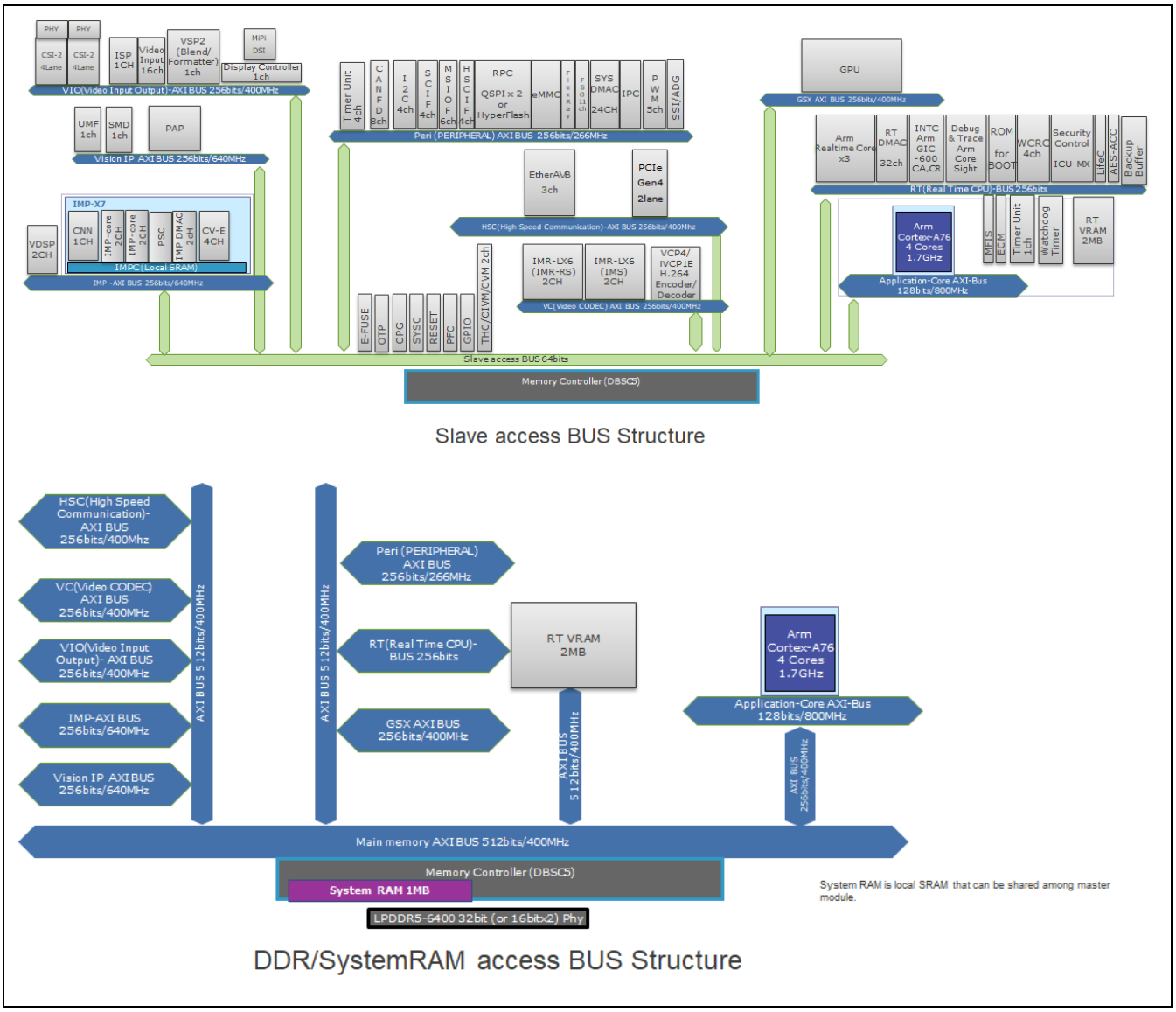



Figure 18.1 Block diagram of AXI-Bus

### 18.1.2 Register configuration

Please refer to R-CarV4M_UM_018_AXI-bus_r0p81.xlsx.

## 18.2 Register description

Please refer to R-CarV4M_UM_018_AXI-bus_r0p81.xlsx.

## 18.3 Operation

### 18.3.1 DRAM Split/Linear Address mapping

Section 18.3.1 is reserved on R-CarV4M.

### 18.3.2 QoS arbitration

AXI-bus has the QoS arbitration to guarantee latency and bandwidth. A concept of the time shared control which is being called time division slot allocation is introduced. The arbitration can be controlled by assigning bandwidth of each IP to time slot.

### 18.3.3 Memory Protection

Main Memory domain AXI-bus has memory protected area function. This function for DRAM as shown in Figure 18.2 is that DRAM area can be divided into 64 areas respectively and each area can be set access privilege to 2 secure groups. This function for System RAM as shown in Figure 18.3 is that System RAM area can be divided into 16 areas respectively and each area can be set access privilege to 2 secure groups.

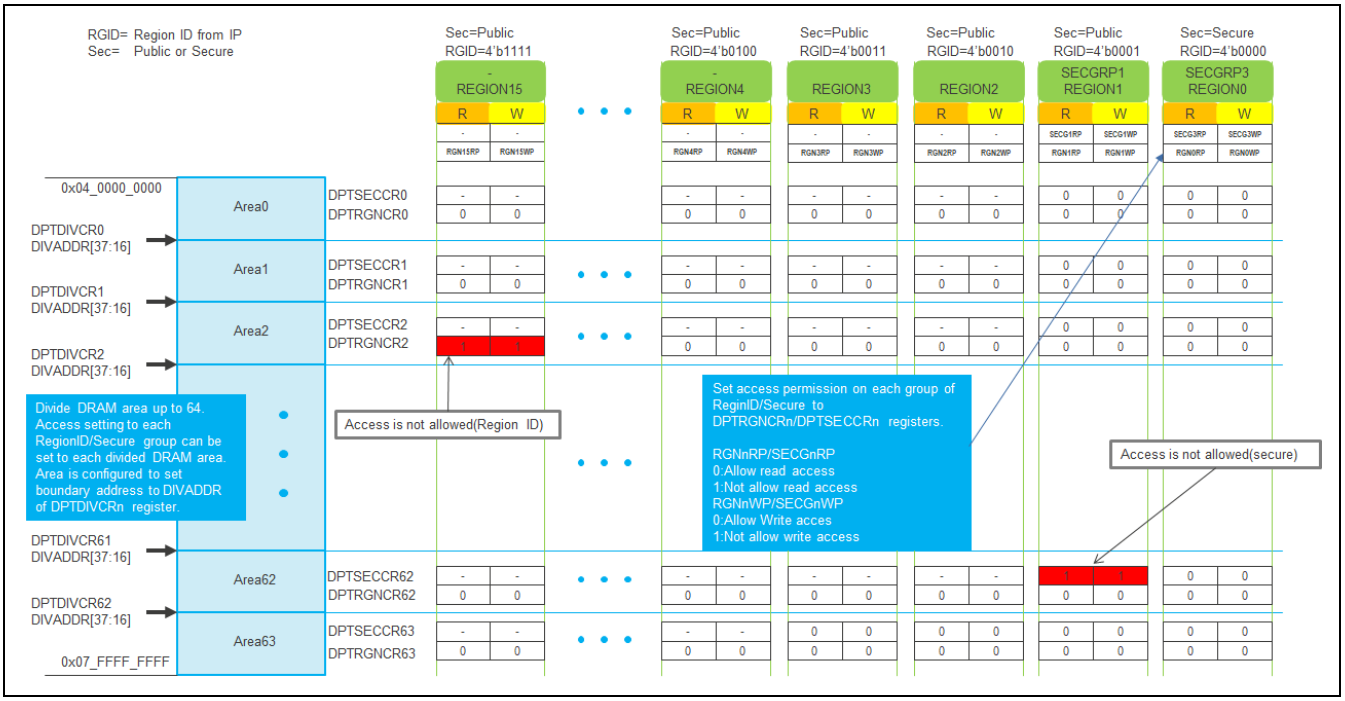



**Figure 18.2 DDR Memory Protected Area Function Concept**

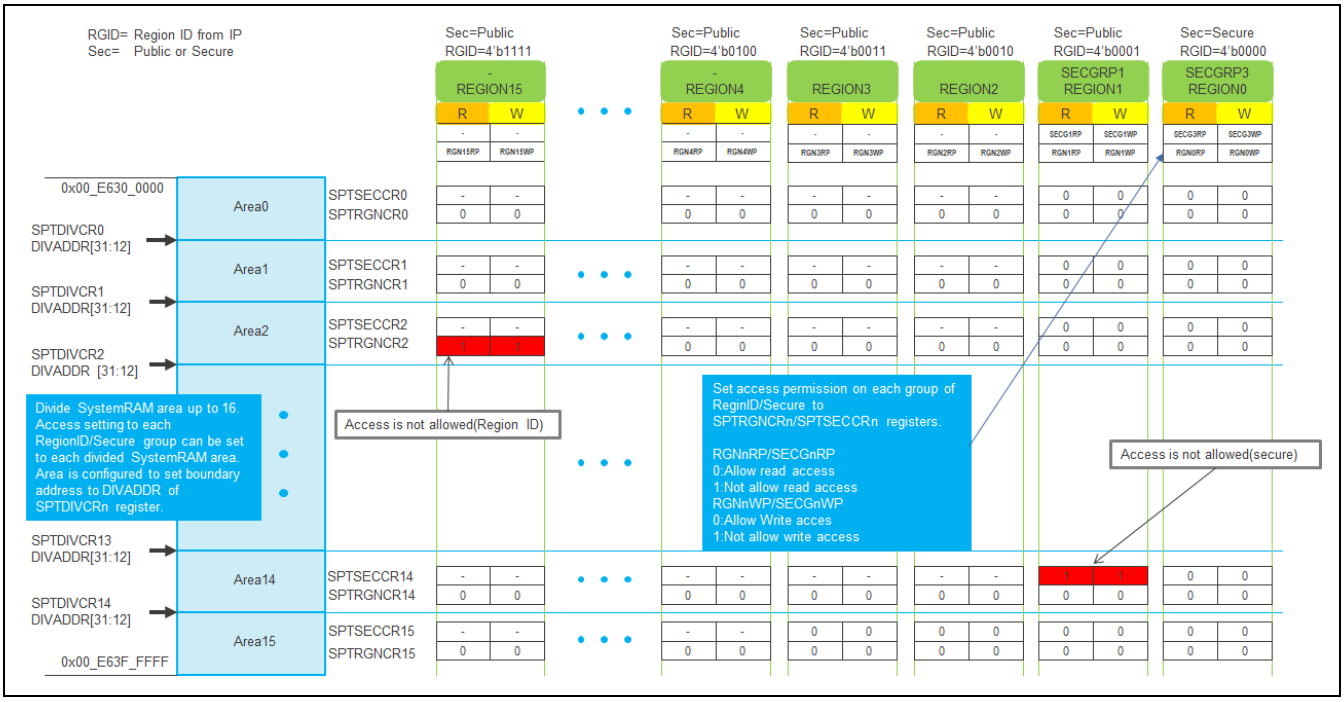



**Figure 18.3 System RAM Memory Protected Area Function Concept**

## 18.4 Usage Notes

### 18.4.1 Access Protection for Secure/Safety Region

A Region ID Protection (RGID) provides “Freedom From Interfere (FFI)”.
By using RGID, a user can specify flexible access protection up to 16 region in a system.
For register setting flow, please refer section 18.5.9(3).

### 18.4.2 Bus Domain Protection

A Bus Domain Protection (BDP) provides register or bit unit access protection corporate with slave module.
Combination with RGID, a user can specify access policy of each bit level. As example, a master A can access to bit 15 of slave A register while a master B cannot access to bit 15 of it.

### 18.4.3 QoS System

AXI-bus has the QoS arbitration to guarantee latency and bandwidth. A concept of the time shared control which is being called time division slot allocation is introduced. The arbitration can be controlled by assigning bandwidth of each IP to time slot.

## 18.5 Safety Mechanism for AXI-bus

There are safety mechanisms inside AXI-Bus system.

- AXI-bus ECM (Error Collection Module)
- Destination ID Checker
- Illegal Response Detection.
- Protocol Check
- Register Slice Order check
- Redundant Integration for protocol conversion
- Bus Protocol EDC
- Default slave and dummy slave error notification
- Region ID protection
- Bus Domain Protection
- EDC protection for QoS SRAM

This section describes each of safety mechanism and register description.

Notice: All AXI-Bus safety mechanism is always enable.

### Overview of Safety Mechanism of AXI-bus

Please refer to 018_R-CarV4M_AXI-bus_overview_structure_of_SM_r0p30.pdf.

### 18.5.1 AXI-bus ECM (Error Collection Module)

#### (1) Overview

The AXI-bus ECM module manages all AXI-bus error on each domain of this SoC.
This module collects all AXI-bus errors and reflects them to status register. One error signal notification is connected to ECM module or INTC module. ICISTP errors and SEC errors are connected to INTC module. Other errors are connected to ECM module. This manages interrupt enable/disable for all AXI-bus errors (routing error/illegal response/timeout counter/protocol check/dummy slave/order check/Burst ID check/Redundant integration/Region ID protection). This also manages error injection function for all AXI-bus safety mechanisms.

#### (2) Error Notification from AXI-bus ECM to ECM

There is one error signal connection from AXI-bus ECM to ECM.

| AXI-Bus ECM Domain | Corresponding ECM Error Status Register |
|---|---|
| TOP0 (Slave Access Bus) | Bit 6 of ECMERRSTSR7 |
| TOP1 (Slave Access Bus) | Bit 23 of ECMERRSTSR39 |

| | |
|---|---|
| TOP2 (Slave Access Bus) | Bit 19 of ECMERRSTSR39 |
| TOP3 (Slave Access Bus) | Bit 24 of ECMERRSTSR39 |
| RT0 | Bit 7 of ECMERRSTSR7 |
| RT1 | Bit 17 of ECMERRSTSR39 |
| RT2 | Bit 16 of ECMERRSTSR39 |
| RT3 | Bit 18 of ECMERRSTSR39 |
| PER00 | Bit 9 of ECMERRSTSR7 |
| PER01 | Bit 20 of ECMERRSTSR39 |
| PER02 | Bit 21 of ECMERRSTSR39 |
| PER03 | Bit 22 of ECMERRSTSR39 |
| HSC | Bit 8 of ECMERRSTSR7 |
| IMP | Bit 11 of ECMERRSTSR7 |
| VIO0 | Bit 3 of ECMERRSTSR7 |
| VIO1 | Bit 27 of ECMERRSTSR39 |
| VIO2 | Bit 28 of ECMERRSTSR39 |
| VC0 | Bit 4 of ECMERRSTSR7 |
| VIP0 | Bit 2 of ECMERRSTSR7 |
| 3DG (GSX domain AXI) | Bit 5 of ECMERRSTSR7 |
| MM | Bit 10 of ECMERRSTSR7 |

There is one error signal connection from AXI-bus ECM to INTC.

About assignment, please refer attached “R-Car V4M series User's Manual [015_R-CarV4M_INTC_Interrupts_Mapping_r0p50.xlsx]“ as interrupt table.

| AXI-Bus ECM Domain | Corresponding INT ID in SPI_Assignment sheet. |
|---|---|
| TOP0 | 0x026F |
| TOP1 | 0x0280 |
| TOP2 | 0x0277 |
| TOP3 | 0x0278 |
| PER00 | 0x026C |
| PER01 | 0x027D |

| | |
|---|---|
| PER02 | 0x027E |
| PER03 | 0x027F |
| HSC | 0x026D |
| RT0 | 0x026E |
| RT1 | 0x0275 |
| RT2 | 0x0274 |
| RT3 | 0x0276 |
| IMP | 0x026A |
| VIO0 | 0x027B |
| VIO1 | 0x0272 |
| VIO2 | 0x027C |
| VC0 | 0x0271 |
| VIP0 | 0x0273 |
| 3DG (GSX domain AXI) | 0x0270 |
| MM | 0x026B |

**(3) Operation**

For bit assignment of AXI-bus ECM, please refer 018_AXI-Bus_ECM_assignment_r0p81.xlsx.

1. Status (ST) register

When error occurs, corresponding bit is asserted. Reserved bit doesn't guarantee read value of status register. When corresponding bit of INTEN register is set 1, error notification is informed from AXI-bus ECM to ECM or INTC. To clear the status, please write 1 to corresponding bit.

2. Interrupt Enable (INTEN) register

When a bit is set 1 and corresponding error occurs, error notification is informed form AXI-bus ECM to ECM or INTC. Reserved bit shall be set 1'b0.

3. DUMMY register

When a bit is set 1, dummy error notification is informed from AXI-bus ECM to ECM or INTC.

### 18.5.2 Destination ID Checker.

**(1) Overview**

This safety mechanism checks consistency between destination ID of transaction and ID of slave module. When data is sent to a slave module from the master, the slave module will check whether the ID of the transaction is addressed to itself or not.

**(2) Hardware Description**

Every AXI master I/F of each bus module, it generates DSTID (Destination ID) that specifies destination of slave I/F of each module. When a transaction reaches slave I/F of each bus module, a routing check function compares DSTID of transaction and that of defined value of each slave. If it is not matched, it is recognized as routing error.

For APB slave I/F, two stage routing check is applied. First stage of routing check is completed before AXI to APB conversion logic. Then DSTID is re-generated at the entrance of APB I/F (different logic of APB decode logic), then it is transferred with transaction. When a transaction is reached to APB slave I/F, a routing check function checks consistency under same manner of AXI slave case.

This also covers a transaction between a master and slave that cannot be covered by Region ID protection.

When an AXI-bus ECM module receives an error notification signal from each Destination ID Checker, it issues an error notification to ECM.

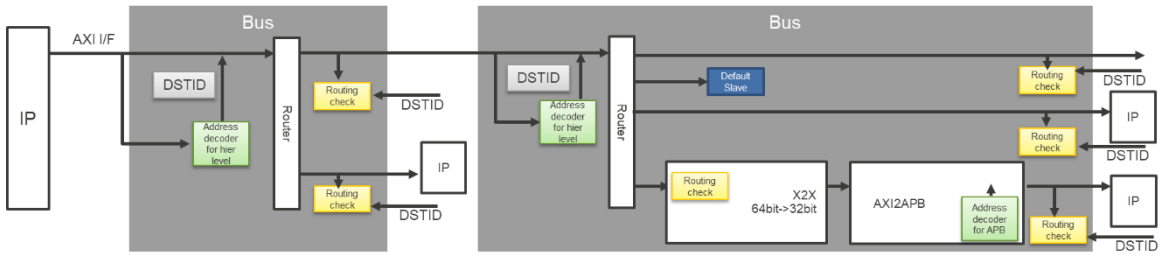



**Figure 18.4** Routing Check function

### 18.5.3 Illegal Response Check

**(1) Overview**

This safety mechanism uses Outstanding Counter and Transaction ID (TID) table of FIXED ERROR function. The Outstanding Counter can be used for illegal response error as explained in section 18.5.1. The safety mechanism uses illegal response error to detect the fault when response exceeds requests. The Transaction ID table can be used for illegal response of in-order transaction.

**(2) Hardware Description**

There are four types of detection, Request Timeout, Over Outstanding Requests, Wrong Response Detection, and Wrong Order Detection for in-order transaction in the FIXED ERROR function.

The Request Timeout is a function that performs to count cycles of latency from transaction requests to transaction acceptance. When it exceeds certain period, it is recognized as wrong operation of bus-controller.

The Over Outstanding Requests is a function that performs counting number of accepted requests. When it exceeds defined number (supported outstanding requests for each master module), it is recognized as wrong operation of bus-controller.

The Wrong Response Detection is a function that performs checking SRCID of response transaction. A SRCID is a unique value of each master module. When a transaction is accepted, AXI-bus puts this SRCID on transaction. When it reaches slave module, AXI-bus record SRCID value and put it on response packet. When a response transaction is back to master module, this function checks consistency of SRCID value of master module and SRCID value on response transaction. When it is mismatch, it is recognized as wrong operation of bus-controller.

The Wrong Order Detection for in-order transaction is a function that performs transaction order check among in-order requests. When a transaction is accepted, AXI-bus puts unique Transaction ID (TID) on transaction and record them in transaction table (FIFO) with AXI ID (Same AXI ID is recognized as in-order transaction.). This function expects TID order in transaction table among in-order transaction (same AXI ID entry). When a response transaction back to master module, this function checks TID value and AXI ID. When an order of TID is incorrect among same AXI-ID transaction, it is recognized as wrong operation of bus-controller.

When the AXI-bus ECM module receives an error notification signal from each FIXED ERROR function, it issues an error notification to ECM.

**FIXED ERROR:**

Figure 18.5 shows the structure of Outstanding Counter and Request Timeout Counter function. This function is consisted of several counters.

- Outstanding counter
- Timer

Outstanding Counters watches how many requests are not completed.
Request num increment logic and response num decrement logic are the logic controller for Outstanding Counter.
When a module issues a request to bus module, then request num increment logic increases the value of Outstanding Counter. When a bus module receives a response, then response num decrement logic decreases the value of Outstanding Counter. When an Outstanding Counter is exceeded outstanding setting at bus controller, it is recognized as wrong operation of bus controller “2) request number violation”.

When an Outstanding Counter is less than 0, this function recognizes it to receive illegal response from slave module as “3) response number violation”. For check targets of FIXED ERROR, see Bit description table of FIXERRST registers.

A timer of request timeout exceeds cycles that is specified by FDT_X*1 setting then this function recognize it as “1) Request timeout error”
*1: X is a target module name listed as shown in Register description (section 18.2).

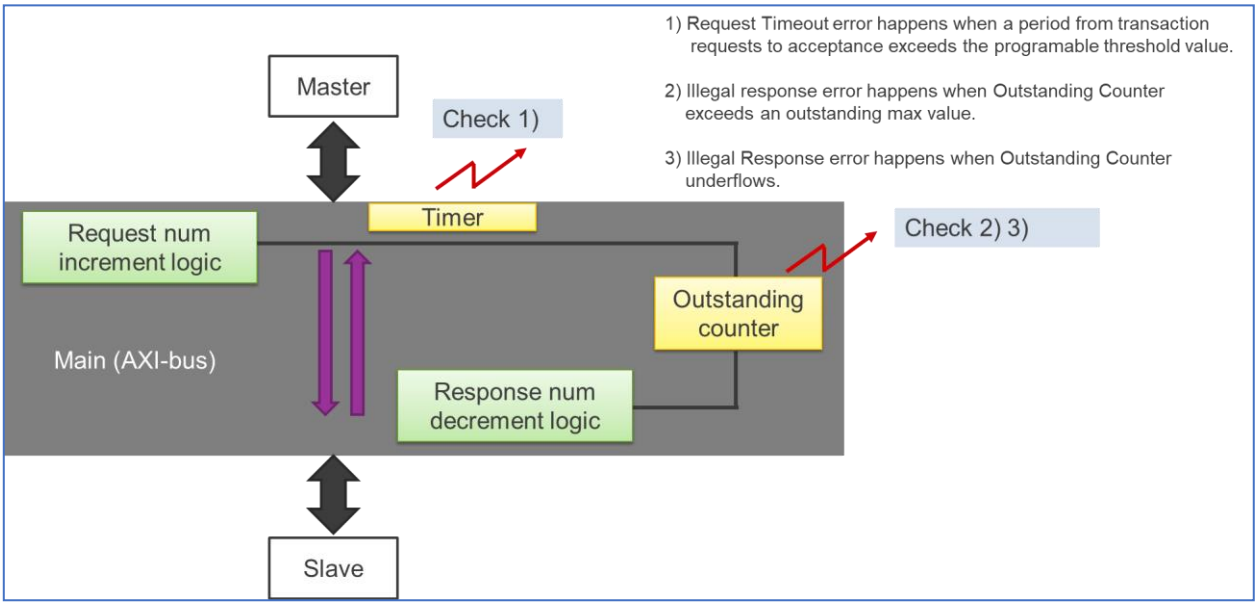



**Figure 18.5** Overview of Outstanding Counter function and Request Timeout function.

**TID check:**

Figure 18.6 shows the structure of TID function.

This function is consisted of Transaction table (FIFO) and timer.

Transaction table records AXI ID, Slave ID (SlvID) and TID. TID is unique transaction ID to recognize order of in-order transaction. An entry of this table is managed by FIFO algorithm. Entry number 0 is always the oldest transaction. A timeout counter of this function only counts cycles from a transaction becomes the oldest. When the oldest transaction is updated, then this counter is initialized. When value of this timeout counter exceeds cycles that is specified by FDT_X*1 setting, then this function recognize it as “transaction timeout error”.
*1: X is a target module name listed as shown in Register description (section 18.2).

When in-order transaction back from slave, this function checks whether in-order transaction (same AXI ID) keeps the order based on TID value. When it is not matched, it is recognized as illegal response of in-order transaction.

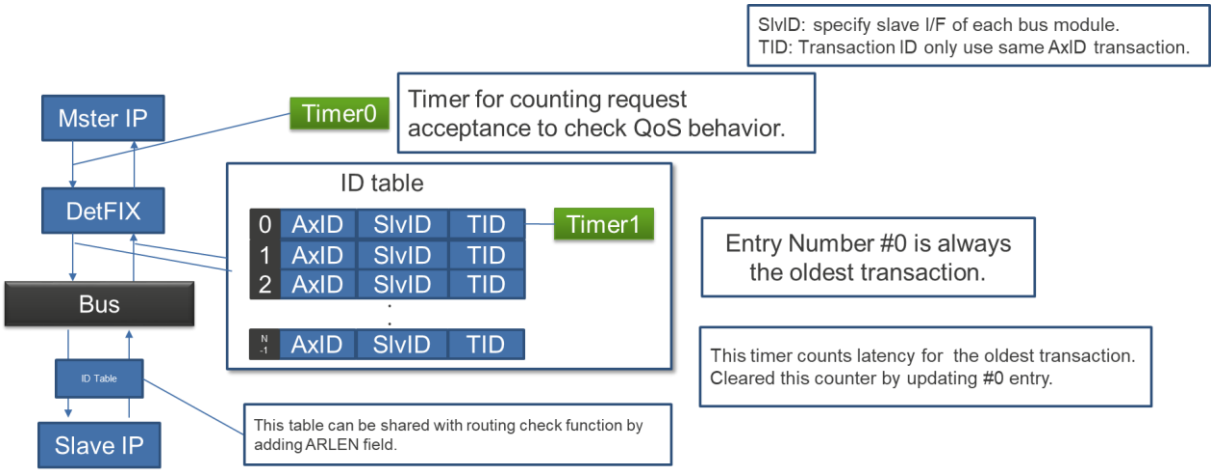



**Figure 18.6** FIXED ERROR function

When the AXI-bus ECM module receives an error notification signal from FIXED ERROR function, it issues an error notification to ECM.

### 18.5.4 Protocol Check

#### (1) Overview

This safety mechanism uses Wchk/Rchk function and LEN/SIZE check function. The Wchk/Rchk uses data consistency check can be used BurstID function. The safety mechanism detects wrong data transfer. The LEN/SIZE check checks consistency among AR/AW channel and actual R/W behavior. The safety mechanism detects a potential of hang-up transaction at the entrance of bus module.

#### (2) Hardware Description

The Wchk/Rchk function check the number of R and W channel order in burst transaction. It puts number of burst transaction (BurstID) on transaction in descending order and check them:
The order of Burst ID (Whether they are in descending order)
The value of RLAST/WLAST (RLAST/WLAST=1 only when the BurstID =0)

When the order of BurstID or the timing of RLAST/WLAST is incorrect, then it is recognized as protocol error by malfunction of bus module.

The LEN/SIZE check function checks consistency of LEN/SIZE value which specifies transaction length and bit width of one cycle transaction and actual data transaction (R or W channel) behavior. When a request (AR/AW channel) is

received this function records LEN/SIZE value of transaction, then it checks actual burst length of data channel and transaction size based on LEN/SIZE value. If it is not matched, this safety mechanism recognized it as protocol error.

When the AXI-bus ECM module receives an error notification signal from Protocol Checker, it issues error notification to ECM.

### 18.5.5 Register Slice Order check

#### (1) Overview

Register Slice Order check for AXI-bus is a safety mechanism that detects incorrect order of transaction, transaction insertion, repetition, and lost due to wrong operation of buffer management of Register Slice.

#### (2) Hardware Description

Register Slice Order check function checks the order inside Register Slice. When a transaction is received both master and slave side, it puts the order number in ascending order. When a transaction is output, then it checks the order number whether it is in ascending order. If this is not matched, it is recognized as wrong operation (Insertion/Lost/Repetition/Resequencing/of Register Slice in AXI-bus.

When the AXI-bus ECM receives error notification signal from each Register Slice Order check function, it issues an error notification to ECM.

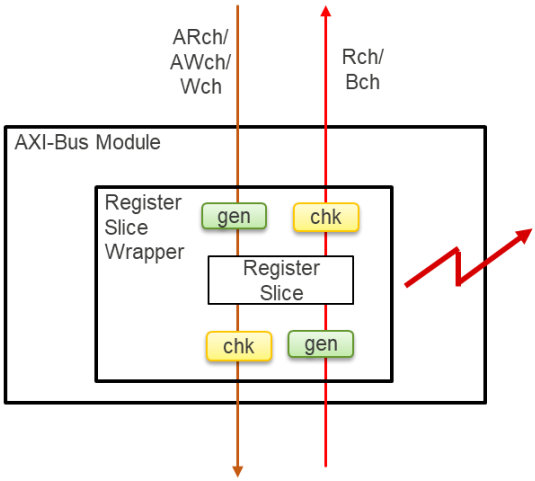



Figure 18.7 Register Slice Order check function for AXI-bus

### 18.5.6 Redundant Integration

#### (1) Overview

Hardware redundancy is applied to the following AXI-bus elements in order to detect fault functional logic.

- X2X (bit width conversion)
- X2P (Protocol conversion logic from AXI to APB)
- uTLB
- Main Memory Interconnect
- Node, Trunk Bus module

Hardware redundancy implementation is different in each AXI domain bus.

| AXI domain bus | Implementation of redundancy |
|---|---|
| Peripheral | Redundancy with 2 delay FFs |
| HSC | Redundancy with 2 delay FFs |
| RT | Redundancy with 2 delay FFs |
| VIO | Redundancy without 2 delay FFs |
| VC | Redundancy without 2 delay FFs |
| VIP | Redundancy without 2 delay FFs |
| CPU | Redundancy with 2 delay FFs |
| GSX | No redundancy |
| IMP | Redundancy without 2 delay FFs |
| Main Memory | Redundancy with 2 delay FFs |
| Slave Access Bus | Redundancy with 2 delay FFs |

**(2) Hardware Description**

Figure 18.8 to Figure 18.9 shows the block diagram of AXI-bus protection for functional safety.

A redundant integration for X2X/X2P/uTLB supports all signals redundancy while main memory interconnects supports transaction signals except for DATA (Read/Write Data). For Read/Write data, EDC code for Read/Write data is redundant. Then by checking consistency of EDC and data at master and checker logic, it can detects wrong operation inside main memory interconnect.

Figure 18.8 shows redundancy with 2 delay FF. Master core has 2 delay FF at output ports and checker core has 2 delay FF at input ports. VIO/VC/VIP/IMP AXI bus domain doesn’t have these 2 delay FF. And, GSX AXI bus domain doesn’t have redundancy protection.

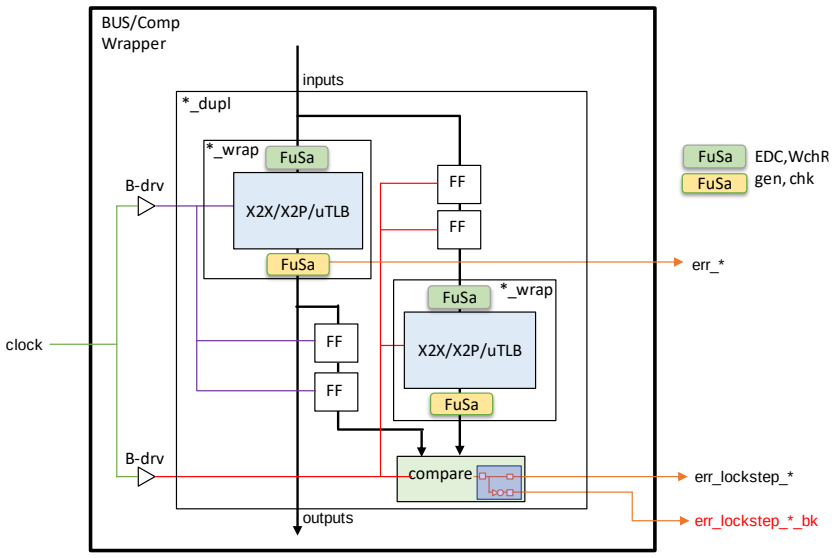



**Figure 18.8** A structure of redundant integration of X2X/X2P/uTLB.

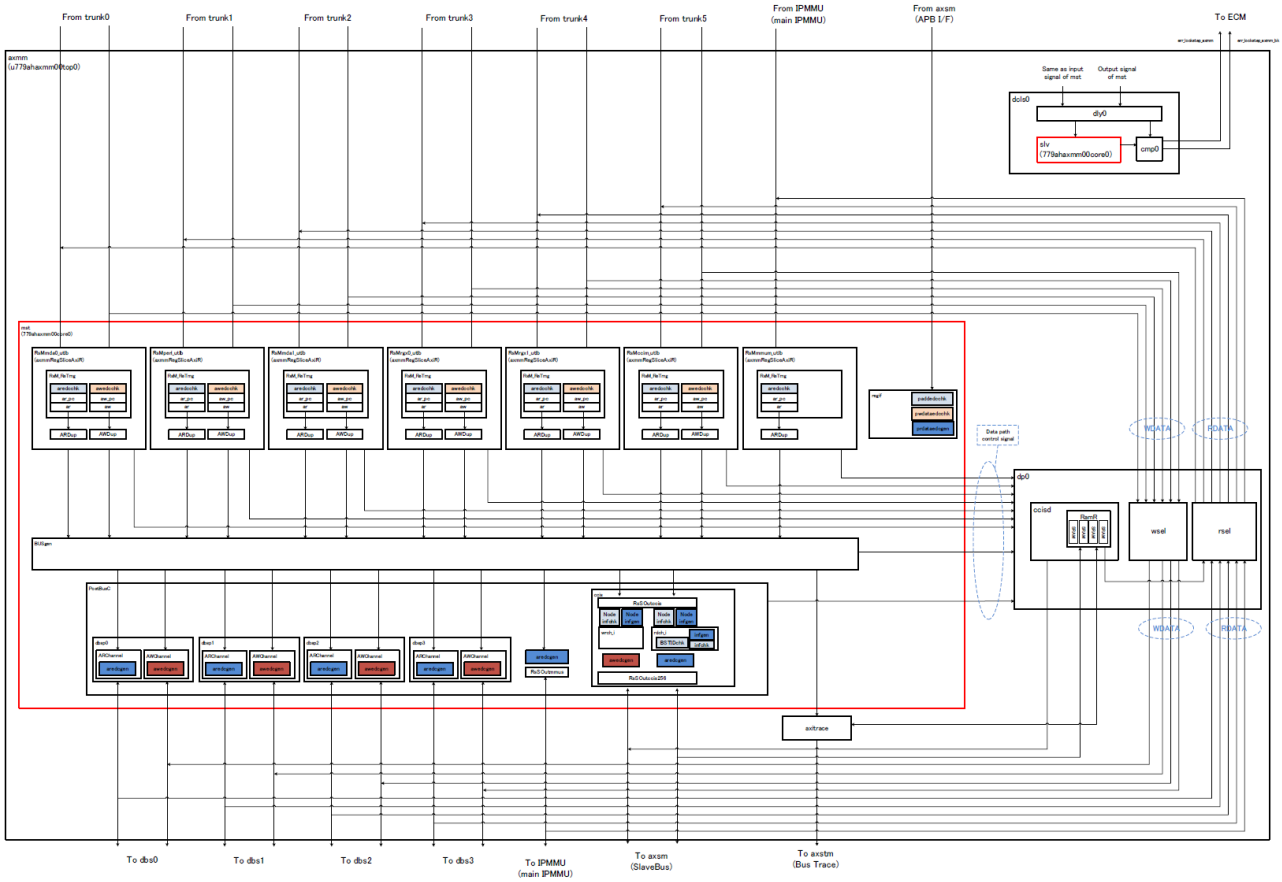



**Figure 18.9** A redundant integration of Main Memory Interconnect.

### 18.5.7 Bus Protocol EDC

#### (1) Overview

EDC protection is applied to all transactions on AXI-bus in order to detect the wrong data transfer caused by data/command (including address) corruption or wrong operation of bus-controller.

#### (2) Hardware Description

An error detection code (EDC) is a function for detecting bus transaction errors by comparing check bits. The check bits are a compressed code generated from target bus transaction signals such as address, transaction size, and read/write data. Figure 18.10 shows an image of an EDC generator and EDC checker. The check bits are generated in the EDC generator on a master module side and transferred with the bus transaction signal to a target slave module. When the slave module receives both the bus transaction signal and check bits, the EDC checker compares the received check bits and the check bits generated in the EDC checker. If they do not match, it can be determined that a transaction error has occurred due to corruption of the signal. Error signal from each EDC checker is input to AXI-bus ECM module. AXI-bus ECM module issues an interrupt to ECM when receive error signal from any EDC checker. T

Figure 18.10, Figure 18.11 shows integration path.

Transaction Data in Figure 18.11 represents a set that includes the data, address, and command line.

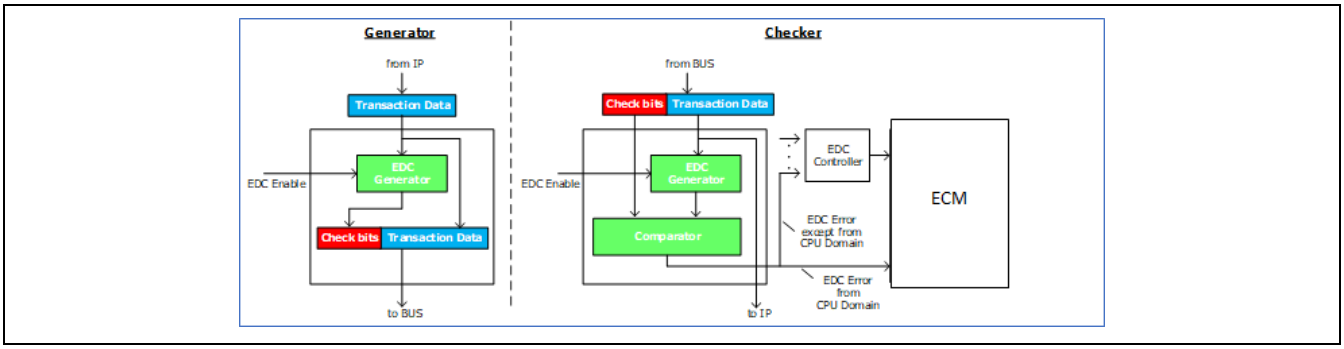



**Figure 18.10** A structure of EDC

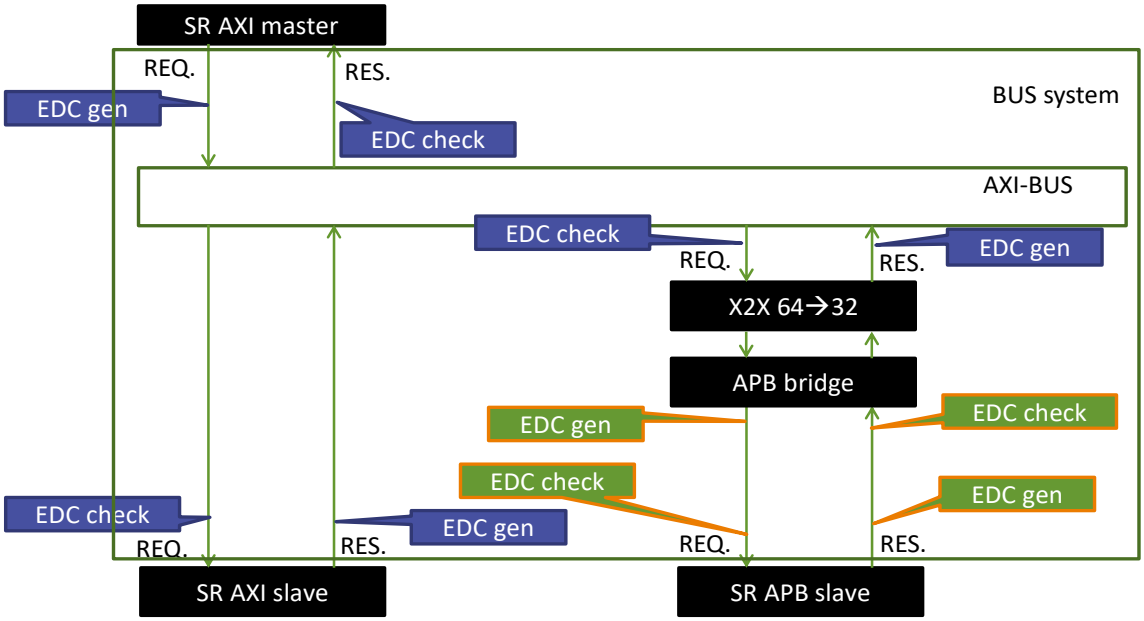



**Figure 18.11** A structure of EDC on AXI-Bus

### 18.5.8 Dummy slave

#### (1) Overview

A dummy slave is a function in order to avoid AXI bus hang-up when a target slave module is in module standby (clock is suspended) by accepting a transaction and inserting error response to master module instead of slave module. This function also notifies an error to AXI-bus ECM when it inserts error response as safety mechanism for transaction lost and delay.

#### (2) Hardware Description

A dummy slave is a function that returns error response to master module (origin of AXI transaction) when it detects a transaction arrival during a module standby for target slave is asserted. Error signal from each dummy slave is input to AXI-bus ECM module. An error status is captured in the ICISTPSTX registers in AXI-bus ECM module. AXI-bus ECM module issues an interrupt to INTC when receive error signal from any dummy slave where ICISTPINTENX are set.

### 18.5.9 Region ID protection

#### (1) Overview

This safety mechanism checks a consistency of master and slave A Region ID protection is a function of safety mechanism that prevents unexpected access.

#### (2) Hardware Description

A Region ID (RGID) protection is a function that blocks illegal access from un-expected master to slave module. Figure 18.12 shows a concept of RGID protection. Figure 18.13 shows a structure of RGID protection. The total number of RGID is 16. A value of RGID value is transferred with AXI transaction as USER extension signals. When it is reached to AXI Slave I/F, this function checks RGID value on a transaction and that on acceptable RGID value defined on AXI Slave I/F. A transaction is issued to slave module when it is matched. An error response is issued to master module when it is not matched. Error signal from each RGID protection is input to AXI-bus ECM module. AXI-bus ECM module issues an interrupt to ECM when receive error signal from any RGID protection.

Each of slave I/F, multi RGID values can be set as acceptable RGID for Read and Write channel independently. This provides “Read only”, “Read and Write”, “Access Prohibited” configuration for each RGID value.

Region ID Master Setting Register is a register that set a RGID value for each Master I/F. Region ID Read (Write) Enable Register is a register that set acceptable RGID value for each Slave I/F.

For SYS-DMAC, RT-DMAC, this function also provides independent RGID value for one AXI Master I/F. It output possible RGID value for internal function of those IPs such as “each DMAC channel”. There are internal registers for RGID value setting inside those IPs and each those IPs compared a value from possible RGID and setting value. When it is matched, setting value can be used. With this function, each DMAC channel can be used with different access protection policy.

For a specific module listed below this function also provides Region ID Read/Write Enable Register dividing one slave address space into several area. A granularity of this divided area, see chapter 2, Area Map.

Target List: CPG/RESET/APMU/PFC/GPIO/PCIe/SYS-DMAC/RT-DMAC/Generic Counter/APSREG/AXCC/DBSC/SYSC
For a SYSRAM/DDR area, this function provides area configuration mechanism that define target area of one Region ID Read/Write Enable setting. The total number of configuration area is 64 for SYSRAM and DDR area. Figure XX shows a function of SYSRAM and DDR area configuration.

If they do not match, it can be determined that an access error has occurred due to RGID value mismatch. Error signal from each RGID protection is input to AXI-bus ECM module. AXI-bus ECM module issues an interrupt to ECM when receive error signal from any RGID protection.

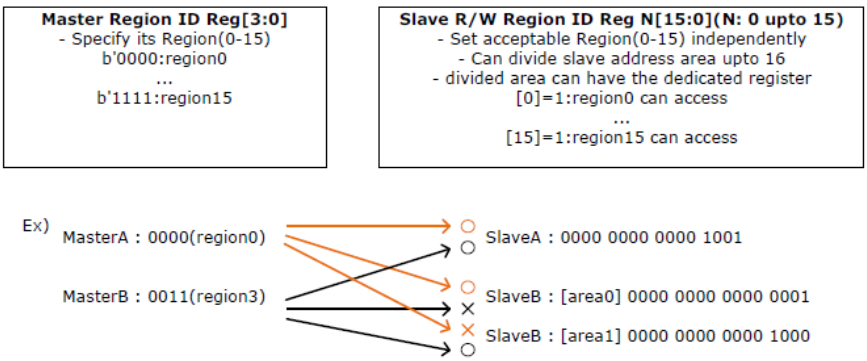



**Figure 18.12 A mechanism of Region ID Protection**

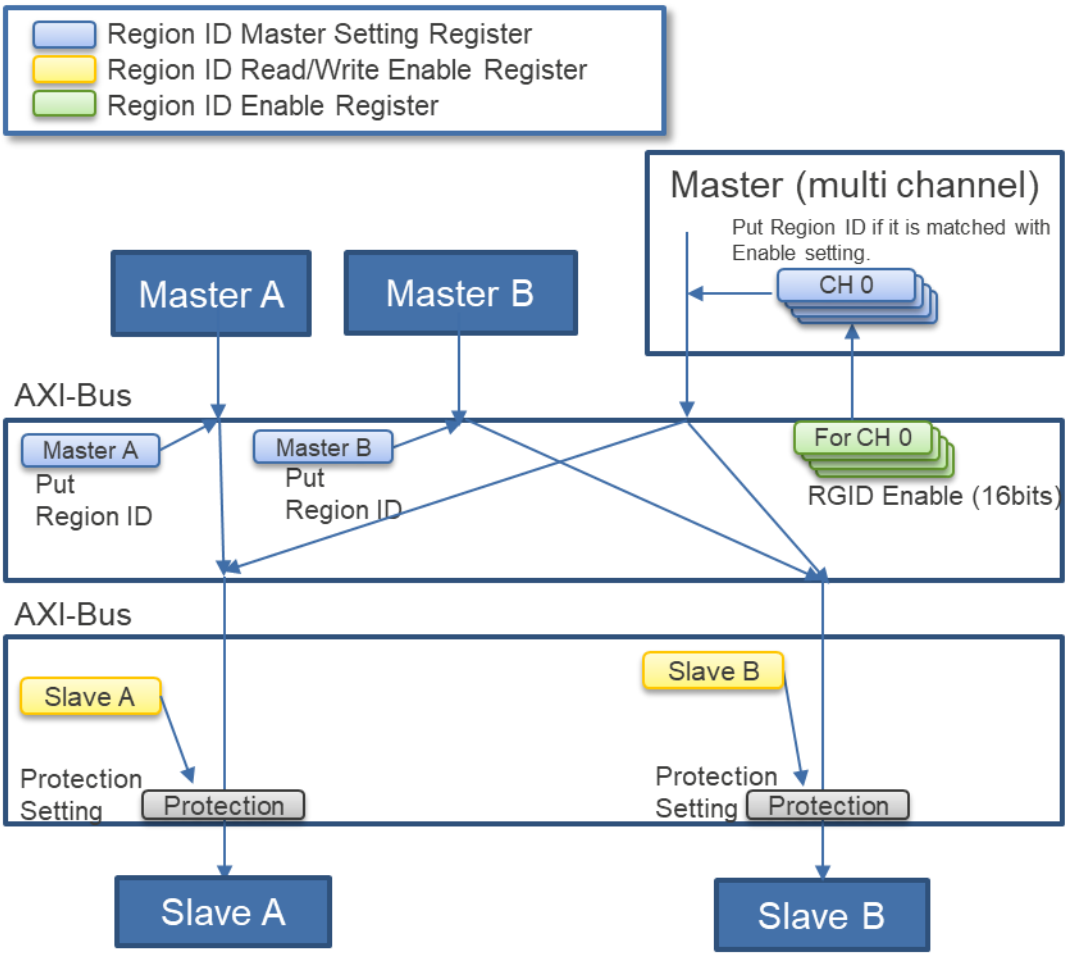



**Figure 18.13 Region ID protection structure**

**(3) Operation**

0. Initial Condition

All Region ID Master Setting Register (RGIDM_<Master>) is set as 0 (Region ID 0).
All Region ID Read Enable Register (RGIDR_<Slave>) and Regin ID Write Enable Register (RGIDW_<Slave>) allows only Region ID 0.
All Secure Group Setting Register (SEC_<Slave>) allows public access.
Secure Attribute Register for Real-time core (SECATTR_CR52_0) is set as 1 (secure).

1. Set SECATTR registers

Please set secure attribute for Realtime Core (0: public, 1: secure).

2. Set RGIDR and RGIDW registers

Please set allowed Region ID for the slave by RGIDR registers, RGIDW registers.
Bit width of these registers is 16bits and each bit correspond to each Region ID (bit n corresponds to Region ID n).
0 means “not allowed” and 1 means “allowed”.
For RT-VRAM, please set “allowed” for all Region ID.

3. Set SEC registers

Please set SEC registers for the slave that allows only secure access (e.g. ICUMX access).
Initial value allows both of public access and secure access.
For slave that needs to protect by secure, please set H’0 to the bit.

4. Set Region ID Enable for RT-VRAM

In RT-VRAM, there are registers of Region ID Enable.
Please refer section 36.

5. Set RGIDM registers

Please set Region ID number (from 0 to 15) to each IP’s RGIM register.

6. Set RGIDMEN registers

Registers of Region ID Master Setting for Sys-DMAC and RT-DMAC are in them.
DMAC Region ID is set in area map of each DMAC channel.
But, the area is protected by RGIDR and RGIDW registers of each DMAC channel.
To prevent security hole, AXI-Bus provides allowed Region ID to DMAC by RGIDMEN registers.
RGIDMEN registers are protected by RGIDR and RGIDW registers of AXI-bus registers.
Bit width of RGIDMEN is 16bits and each bit correspond to each Region ID (bit n corresponds to Region ID n).
0 means “not allowed” and 1 means “allowed”.

7. Set Region ID Master Setting for Sys-DMAC and RT-DMAC

For detail, please refer section 30.5.2.
These registers are protected by RGIDW and RGIDR registers of each DMAC channel, not RGIDMEN registers.
If set value is not allowed by RGIDMEN registers, register set is ignored.

8. Set Region ID Master Setting for AP-System Core

Registers of Region ID Master Setting for AP-System core are in it.
AP-System Core doesn't need setting of RGIDMEN registers.
Please refer section 5.

### 18.5.10 Bus Domain Protection

Bus Domain Protection is a function that provides independent Region Read/Write Enable function for one slave I/F by splitting address space. A target slave is listed below.

CPG/RESET/APMU/PFC/GPIO/PCIe/SYS-DMAC/RT-DMAC/Generic Counter/APSREG/AXCC/DBSC/SYSC/SYSRAM/DDR space

For some module on the list, it provides bit level access protection corporate with Bus Domain Protection. Figure 18.14 shows a structure of example.

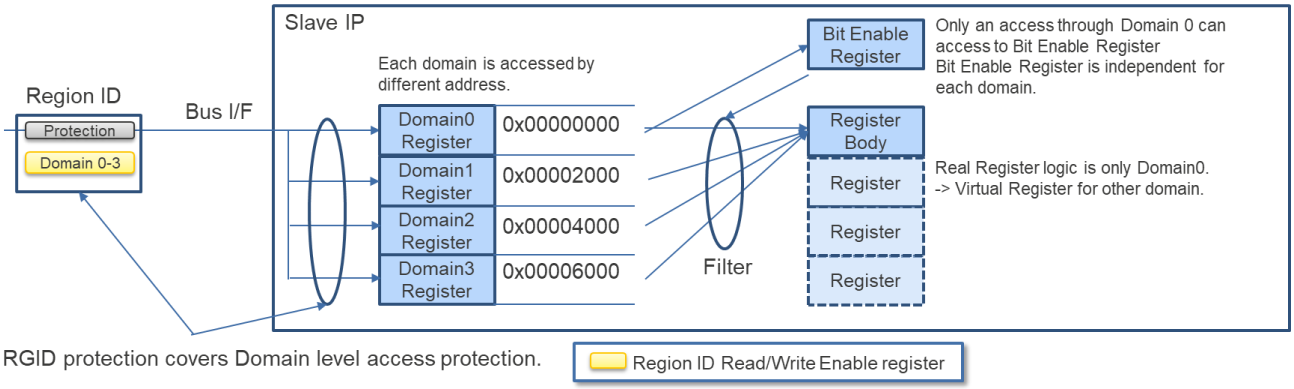



Figure 18.14 Overview of Bus Domain Protection

For SYSRAM and DDR space, area size of each domain is configurable by register setting. Figure 18.15 shows a structure of configuration of area size. A total number of configured areas is 64.

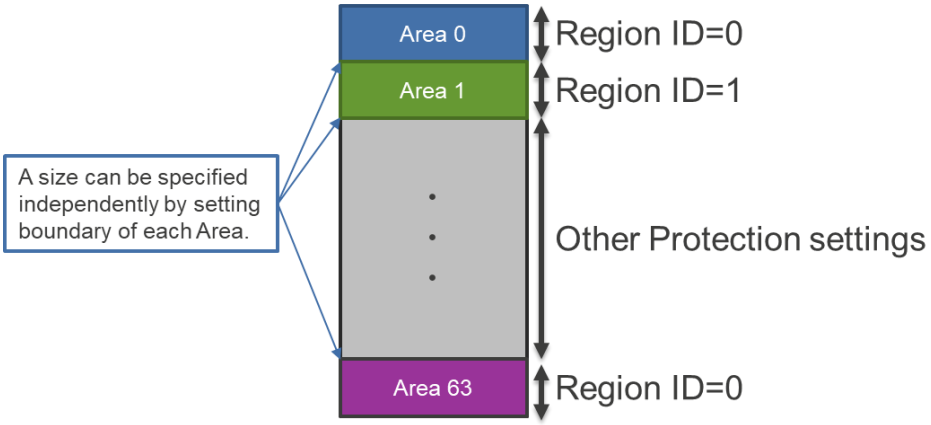



Figure 18.15 SYSRAM and DDR area configuration

### 18.5.11 EDC protection for QoS SRAM

#### (1) Overview

This safety mechanism checks register data in QoS SRAM by EDC. This QoS SRAM is in MM hierarchy.

#### (2) Hardware Description

Registers of protection target by this safety mechanism are QOSBW_FIX_QOS_BANK[n][k] and QOSBW_BE_QOS_BANK[n][k].

These registers are 64bits and EDC is generated from 64bits data to 8bits.

EDC is checked when SRAM is read from AXI-bus or QoS controller.

This error is notified to AXI-bus ECM module without register configuration.

An error status is captured in bit 13 of EDCSTMM0 register in AXI-bus ECM module.

#### (3) Operation

These registers shall be set by 64bits access to prevent wrong EDC error notification because 8bits EDC is generated from 64bit data.

# 19. IPMMU

## 19.1 Overview

The IPMMU is a Memory Management Unit (MMU) which provides address translation and access protection functionalities to processing units and interconnect networks.

- IPMMU cache and main IPMMU have 16 contexts that can be selected by the uTLBs
- Each context refers to two individual page tables, which can be set two base addresses by register setting
- IPMMU cache and main IPMMU context have a 1:1 relation
- All used contexts in main IPMMU as well as IPMMU cache need to be configured accordingly

### 19.1.1 Features

IPMMU includes the following main features.

- MMU architecture compatible with Armv8 VMSAv8-64 and VMSAv8-32 including the Large Physical Address Extension (LPAE), the Security Extensions and the Multiprocessing Extensions. IPMMU supports sixteen sets of TLBs and these can be managed independently.
- PMB (Privileged space Mapping Buffer) address translation functionality compatible with SH-4A. IPMMU supports sixteen domains. Four types of tables (16MB, 64MB, 128MB and 512MB) can be set.
- Caching recently used page table entries in translation look-aside buffer (TLB)
- Performance monitoring for Armv8 VMSAv8-64 and VMSAv8-32 LPAE
- DVM messages are not supported.
- Supported memory translation granule size is 4KB or 64KB. (16KB is not supported)
- 8-bit ASID size is only supported for 1st-starge and 2nd-stage address translation. (not support 16-bit size).

For more information about Armv8 VMSA, see Arm Architecture Reference Manual Armv8, for Armv8-A architecture profile.

In VMSAv8-64, supported memory translation granule size is 4KB and 64KB. 16KB granule size is not supported in IPMMU. In VMSAv8-32, 40bit IPA can be handled by IPMMU.

### 19.1.2 Block Diagram

Figure 19.1 shows the IPMMU block diagram. When a master (a processing unit or an interconnect) issues a transaction, a micro-TLB connected to the master sends a virtual address to the IPMMU cache which is connected to the same bus domain. If the virtual address is not cached in the IPMMU cache, the IPMMU performs a page table walk to the main IPMMU which is located at the main memory bus domain through an interconnect and returns a physical address to the micro-TLB.

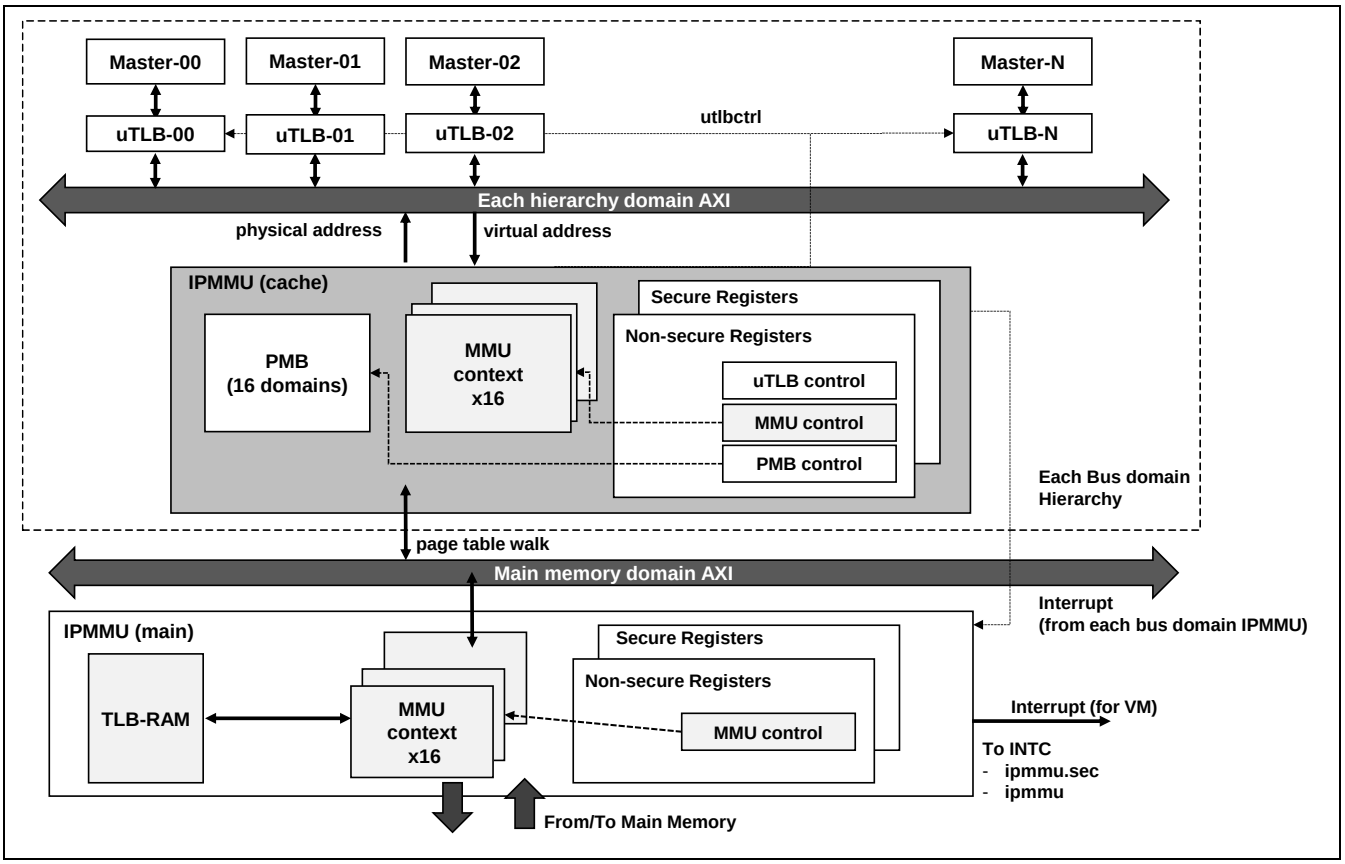



**Figure 19.1 IPMMU System Integration**

### 19.1.3 External Pins

There are no external pins in this module.

### 19.1.4 Connected Module

Table 19.1 shows the connected modules to IPMMU. As for the feature of the connected modules, refer to the chapter of each modules.

Table 19.1 Connected Modules

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| IPMMU | AXI-BUS | Bus |
| | SYSC | Clock Supply |
| | RST | Module Reset |
| | INTC | Interrupt handling |
| | ECM | Error Interrupt Controller |

### 19.1.5 Register configuration

Refer to R-CarV4M_UM_019_IPMMU_r0p81.xlsx and R-CarV4M_UM_019_IPMMU_long_descripter_r0p81.xlsx.

Table 19.2 IPMMU Base Address and Power Domain

| MMU | Power Domain | Base Address |
|---|---|---|
| IPMMU-VI0 | C4 | H'EEE8 0000 |
| IPMMU-VI1 | C4 | H'EEEC 0000 |
| IPMMU-VC | C4 | H'EEDC 0000 |
| IPMMU-IR | C4 | H'EED8 0000 |
| IPMMU-HC | C4 | H'EED4 0000 |
| IPMMU-RT0 | Always-On | H'EE48 0000 |
| IPMMU-RT1 | Always-On | H'EE4C 0000 |
| IPMMU-DS0 | Always-On | H'EED0 0000 |
| IPMMU-VIP0 | C4 | H'EEF0 0000 |
| IPMMU-3DG | C4 | H'EEE0 0000 |
| IPMMU-MM | Always-On | H'EEFC 0000 |

## 19.2 Register description

Refer to R-CarV4M_UM_019_IPMMU_r0p81.xlsx and R-CarV4M_UM_019_IPMMU_long_descripter_r0p81.xlsx.

## 19.3 Operation

### 19.3.1 Address Translation Sequence in micro-TLBs

Each micro-TLB has independent entries and can translate from a virtual address to a physical address without sending translation requests to the IPMMU if the virtual address is cached in the micro-TLB.

When a translation error occurred in address translation, the micro-TLB blocks subsequent transactions until a valid page entry is registered. Set IMUCTRn.FLUSH = 1 after the entry registration.

Figure 19.2 shows the micro-TLB address translation sequence.

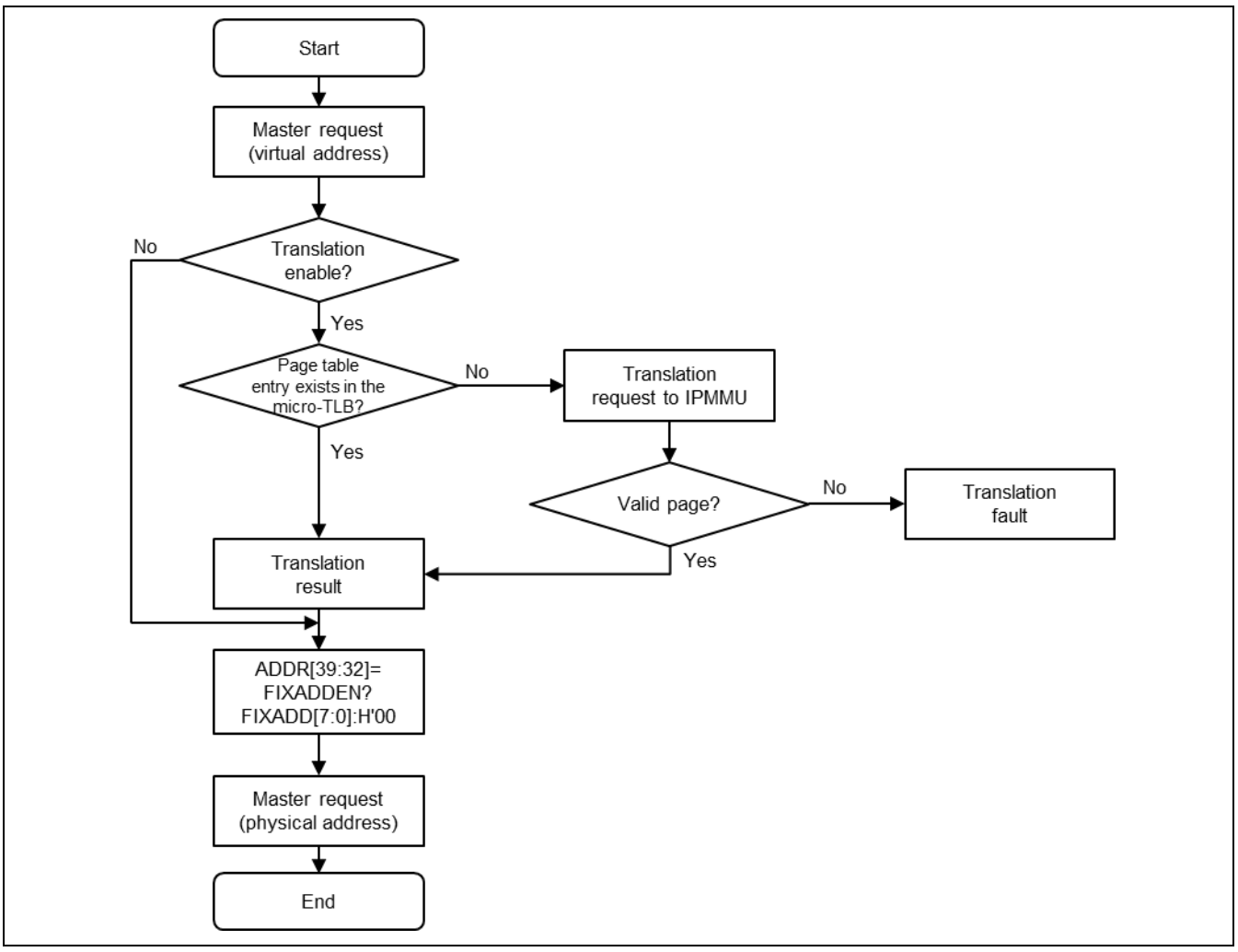



**Figure 19.2 micro-TLB Address Translation Sequence**

### 19.3.2 Address Translation Sequence in the IPMMU

#### (1) MMU Address Translation

The IPMMU has independent sixteen contexts that can select non-secure contexts or secure contexts with a page table for each context. It supports two page table formats, the Short-descriptor format and the Long-descriptor format. When the IPMMU receives an address translation request from a micro-TLB, the IPMMU starts address translation sequence based on the page table specified by IMUCTRn.TTSEL. In case that the micro-TLB is enabled and MMU is selected in IMUCTRn.TTSEL, then the corresponding IMCTRn.MMUEN also need to be set. In case that IMCTRn.MMUEN is disabled, the corresponding micro-TLB must not be enabled.

Figure 19.3 shows the MMU address translation sequence.

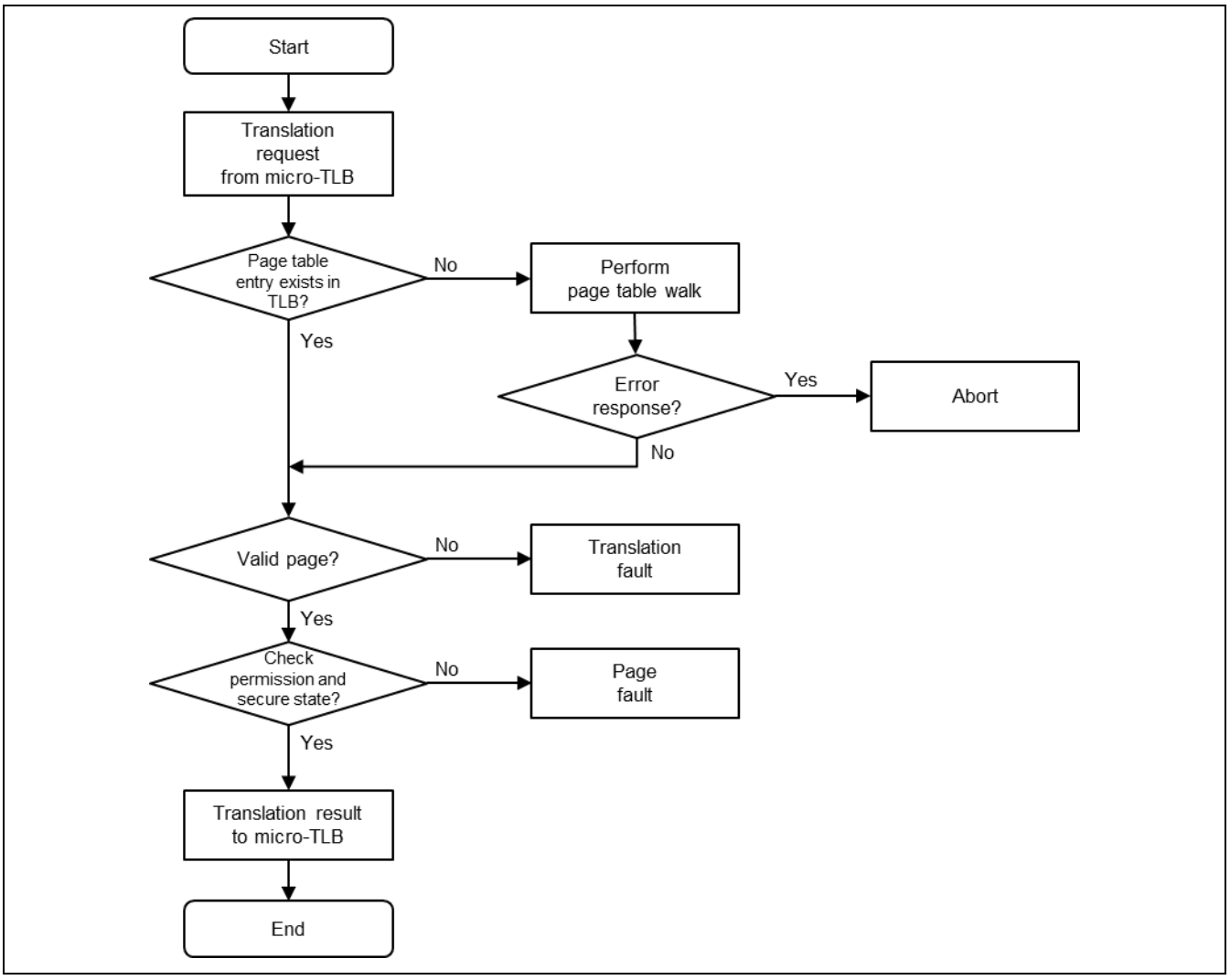



**Figure 19.3 MMU Address Translation Sequence**

### (2) PMB Address Translation

PMB address translation is performed in two ways. When PMB address translation is chosen in IMUCTRn.TTSEL, all virtual addresses are translated by PMB. In case that the micro-TLB is enabled and PMB is selected in IMUCTRn.TTSEL, then the corresponding IMPCTR.PMBEN also need to be set. In case that IMPCTR.PMBEN is disabled, the corresponding micro-TLB must not be enabled.

Figure 19.4 shows the PMB translation flow.

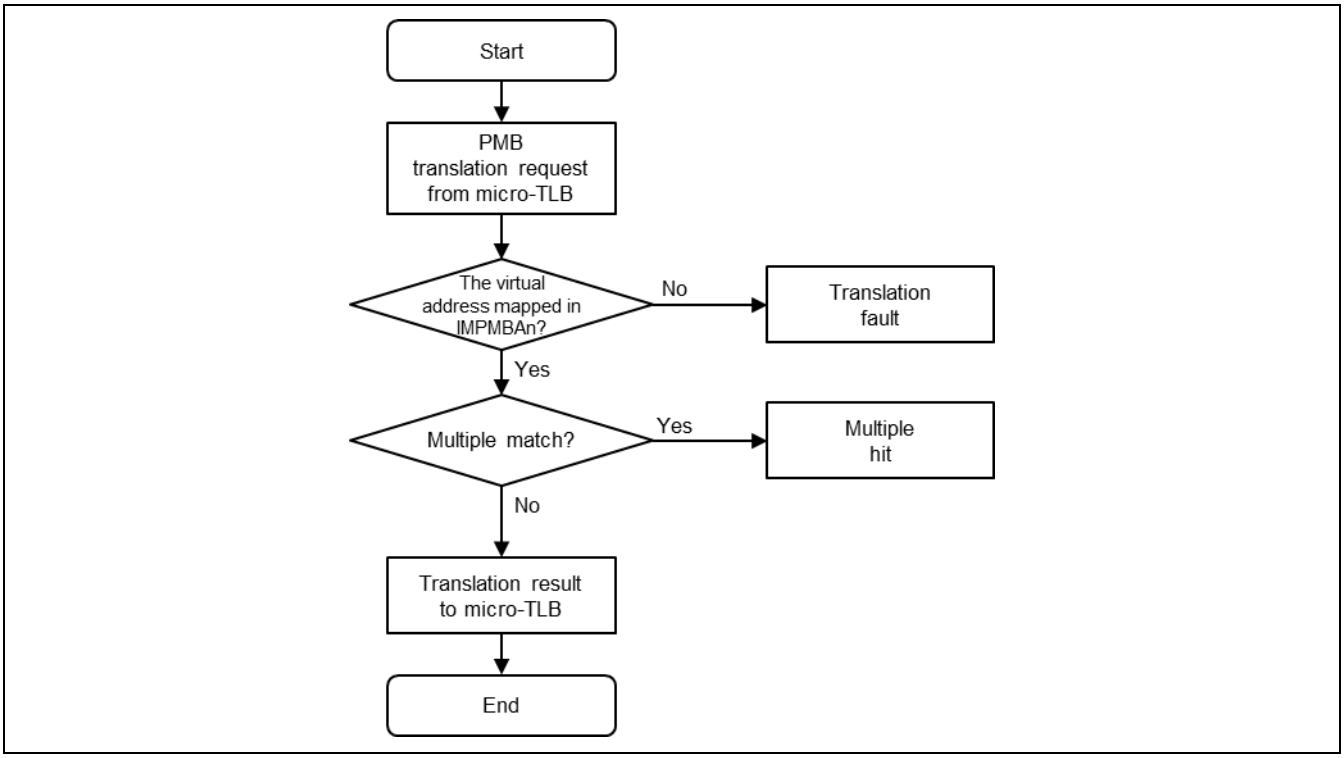



**Figure 19.4 PMB Translation Flow**

### 19.3.3 Address Space

#### (1) Short-descriptor

In the short-descriptor format, 32-bit virtual address space is translated to 32-bit physical address space managed by TTBR0/1.

Note: “TTBR0” is register bit of IMTTLBR0[n].TTBR[31:12].
“TTBR1” is register bit of IMTTLBR1[n].TTBR[31:12].
“TTBCR” is IMTTBCR[n] register.
n = 0 to 15 (IPMMU context)

Figure 19.5 shows the Short-descriptor address space.

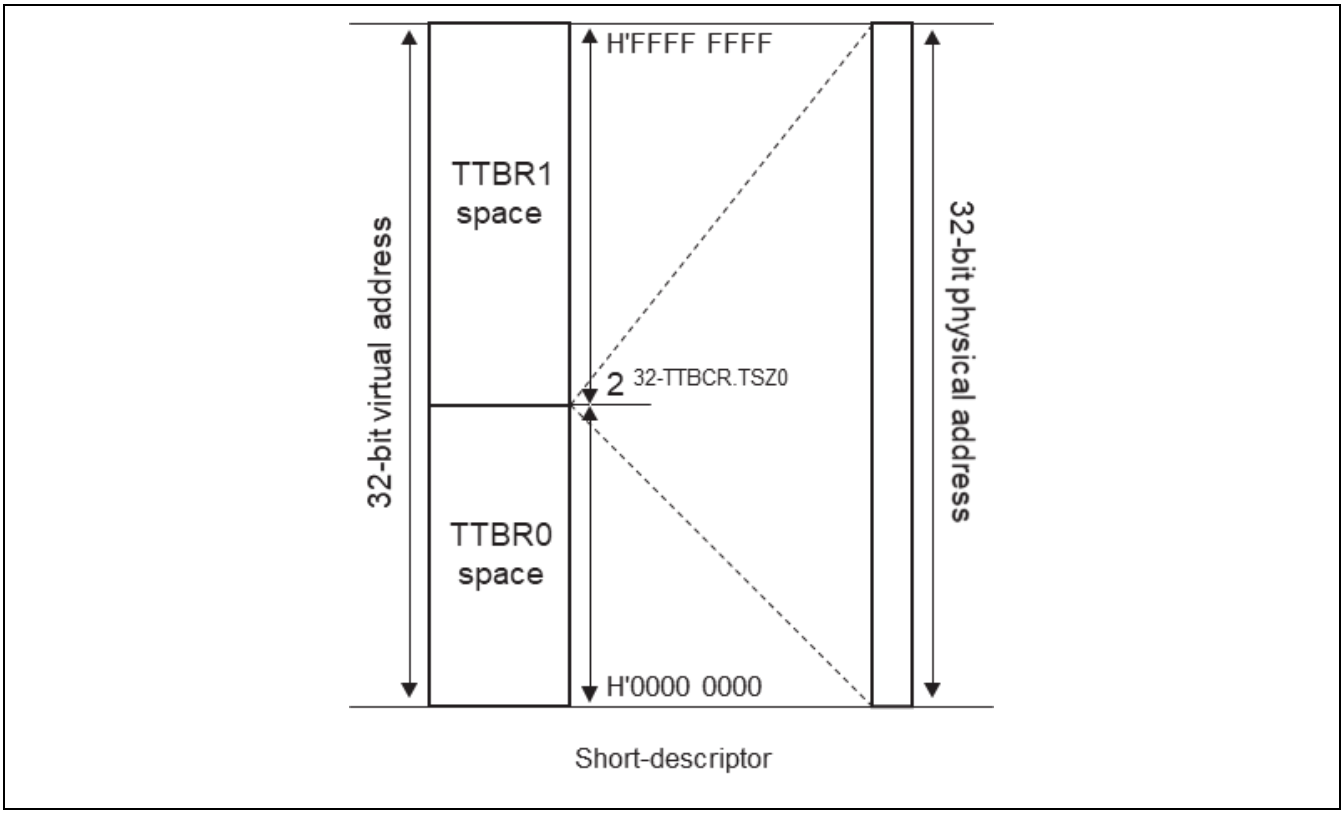



**Figure 19.5 Short-descriptor Address Space**

### (2) Long-descriptor

In the long-descriptor format, 32-bit and 40-bit virtual address space is translated to 40-bit physical address space managed by TTBR0/1. The IPMMU supports the Secure PL1 and PL0 stage 1 translation and the Non-secure PL1 and PL0 stage 1 translation.

Figure 19.6 and Figure 19.7 shows the Long-descriptor address space for VMSAv8-32 and VMSAv8-64.

Note: "TTBR0" is register bit of IMTTUBR0[n].TTBR[39:32] and IMTTLBR0[n].TTBR[31:12].
"TTBR1" is register bit of IMTTUBR1[n].TTBR[39:32] and IMTTLBR1[n].TTBR[31:12].
"TTBCR" is IMTTBCR[n] register.
n = 0 to 15 (IPMMU context)

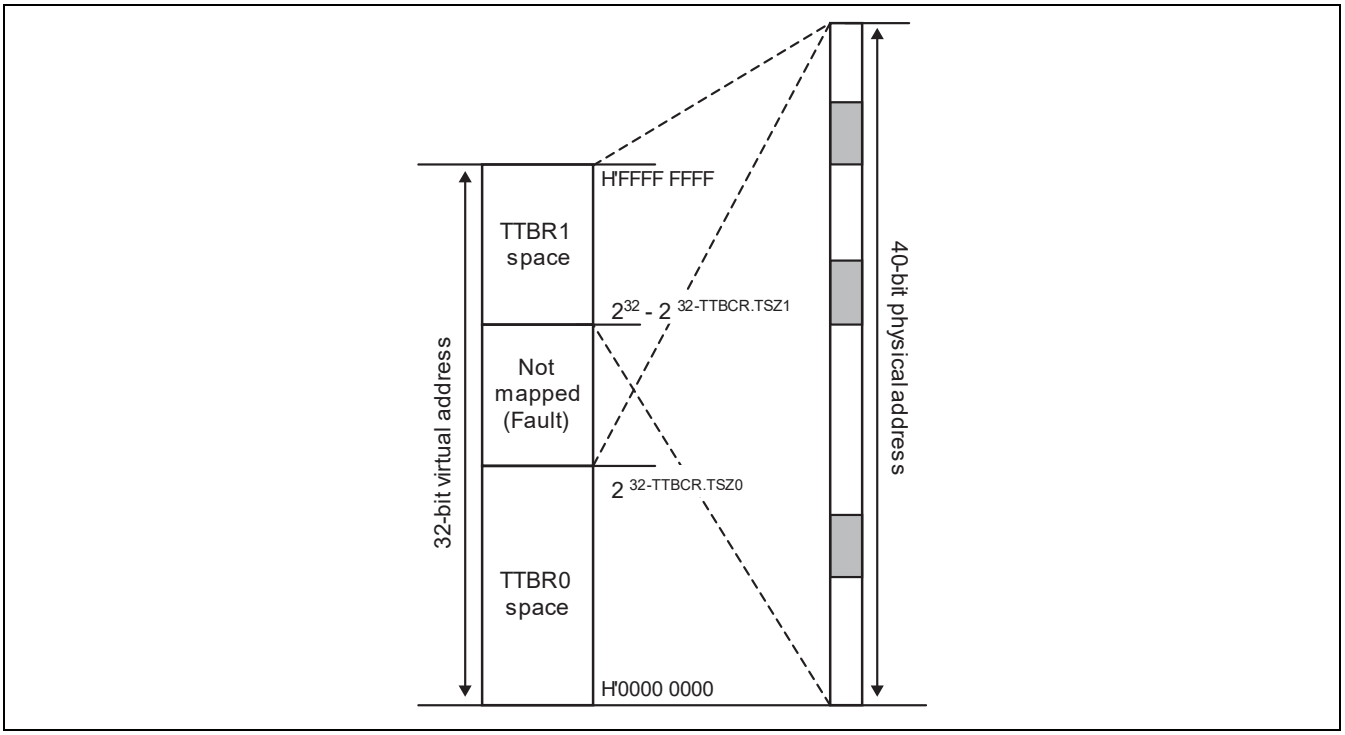



**Figure 19.6 Long-descriptor Address Space (VMSAv8-32)**

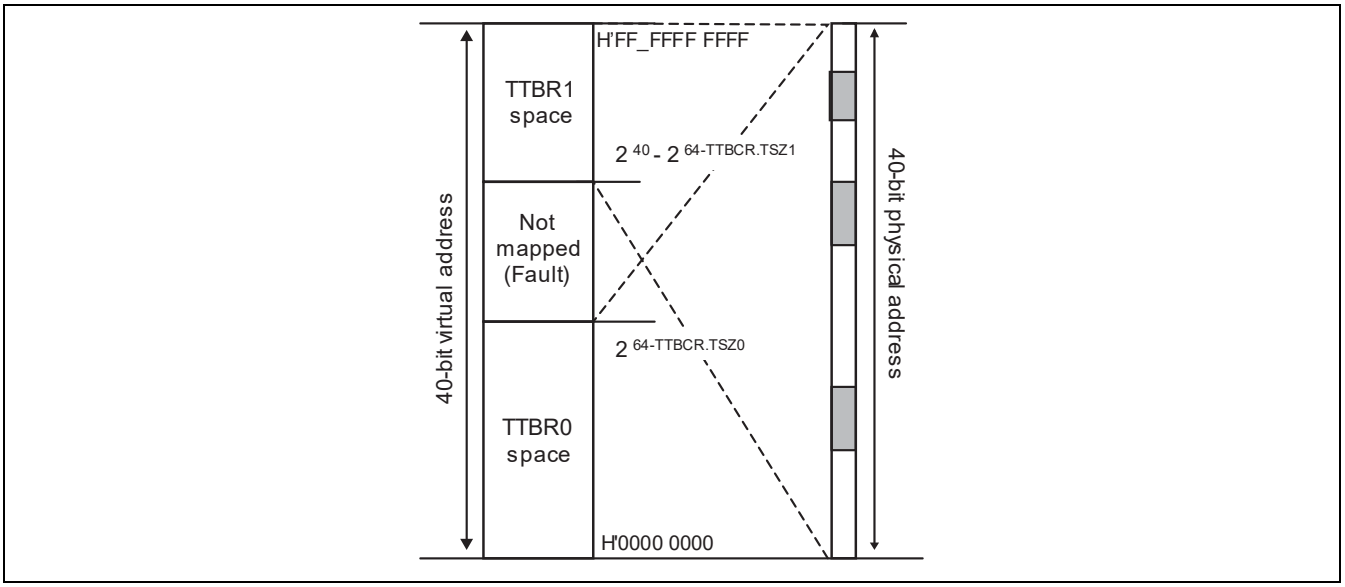



**Figure 19.7 Long-descriptor Address Space (VMSAv8-64)**

### (3) PMB

The IPMMU supports the PMB address translation system. Figure 19.8 shows the PMB address space.

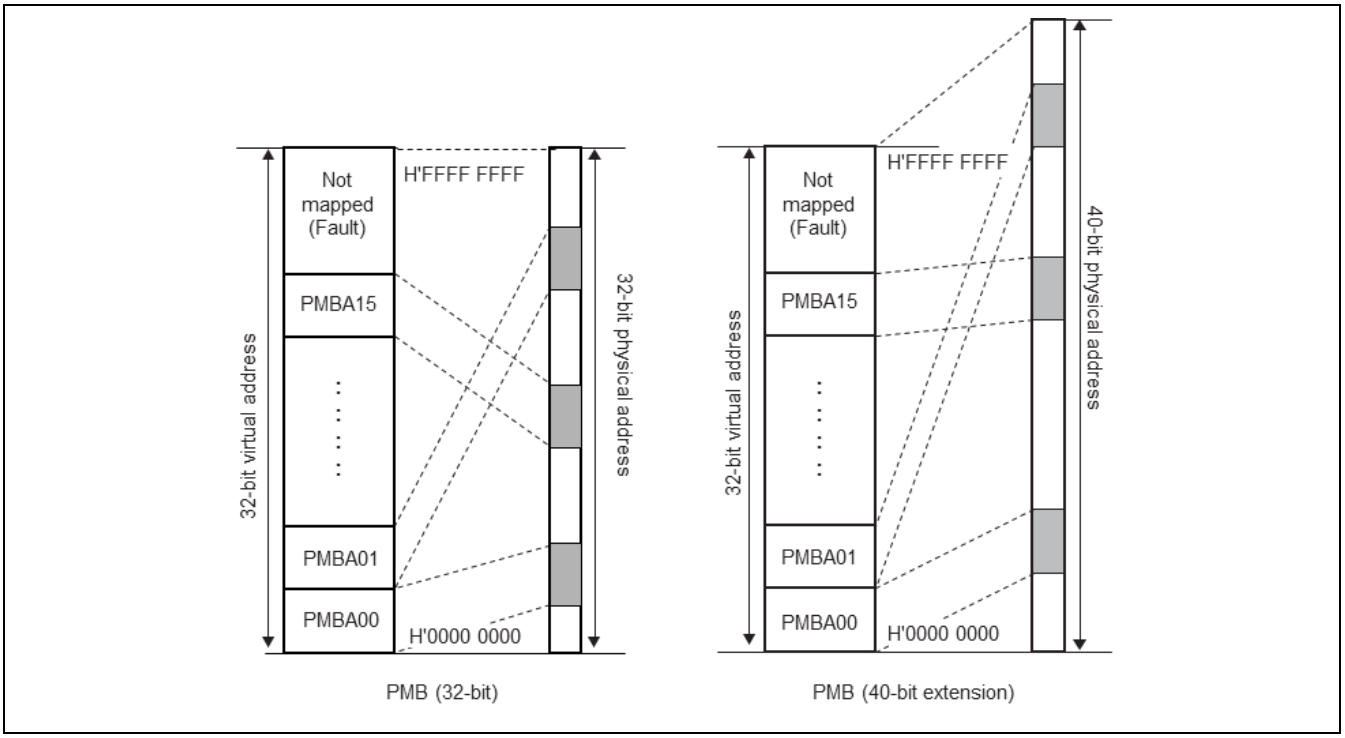



**Figure 19.8 PMB Address Space**

### (4) Stage 2 Translation

The IPMMU supports stage 2 translation. At the stage 1, 32-bit address space is translated to 32-bit or 40-bit intermediate physical address (IPA) space. At the stage 2, 32-bit or 40-bit IPA space is translated to 40-bit physical address space. In order to enable the stage 2 translation, set 1 to IMCTRn.TREN and set its stage 2 page table to IMCTRn.RTSEL. In default, IPMMU use stage 1 translation table format even if performing stage 2 translation. In case of using stage 2 translation table format in stage 2 translation, set 1 to IMSAUXCTLR.S2PTE. In some hypervisor system, only address translation from IPA to PA is supported by stage 2 translation. By setting 1 to IMTTBCRn.BYPEN, IPMMU doesn't perform stage 1 address translation and start stage 2 address translation.*

Figure 19.9 shows the stage 2 translation address space.

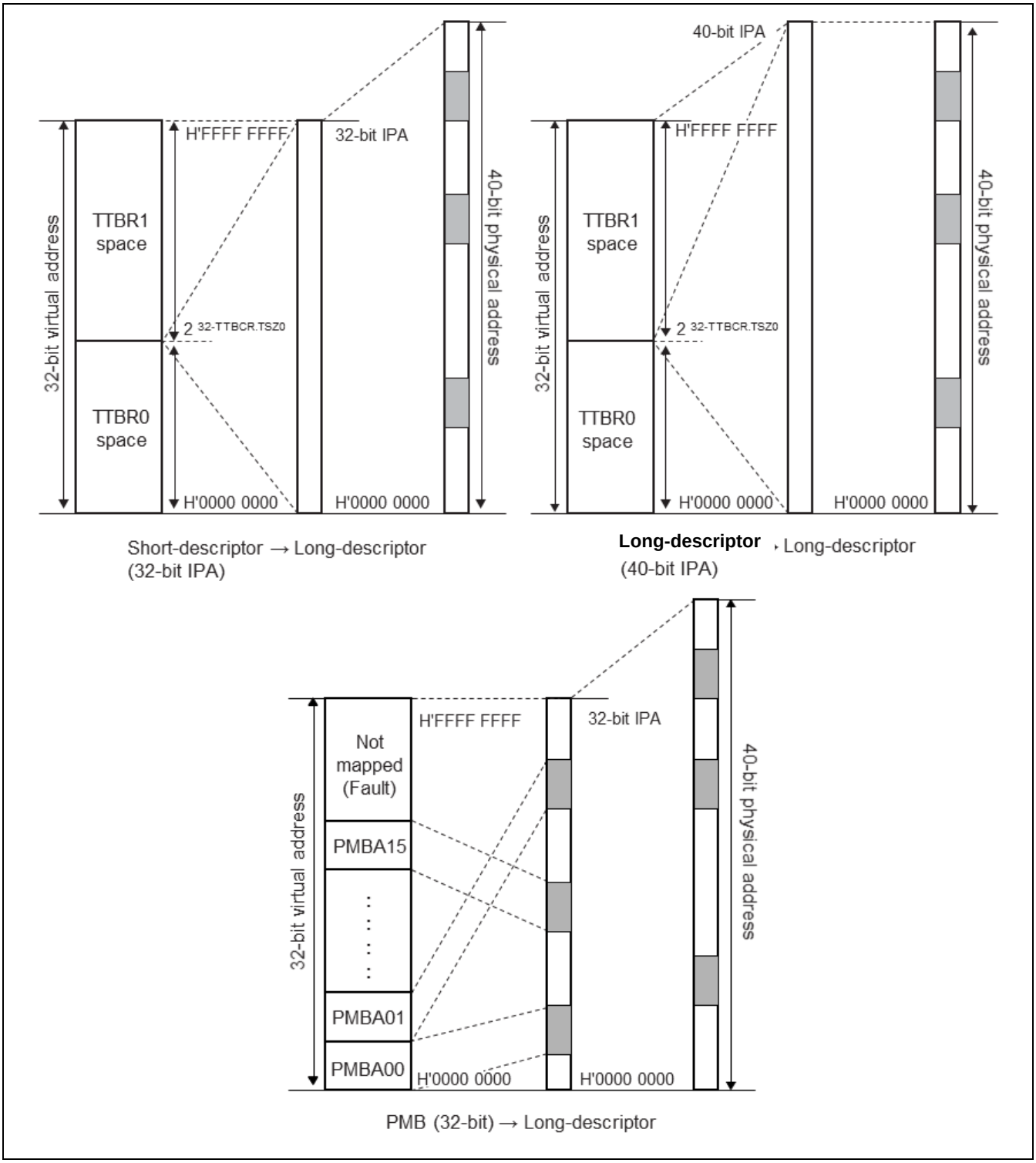



**Figure 19.9 Stage 2 Translation Address Space**

### 19.3.4 Initialization Sequence

This section describes the initialization sequence of the IPMMU.

**(1) Module Stop**

The IPMMU doesn't support module stop control. The IPMMU can be used whenever a master requires address translation through the IPMMU. When a master doesn't require address translation, the corresponding IMUCTRn MMUEN must be disabled.

**(2) micro-TLB**

Before a master access, the corresponding micro-TLB registers which includes the translation mode and ASIDs must be configured.

Figure 19.10 and Figure 19.11 shows the example of the procedure for using MMU and PMB function.

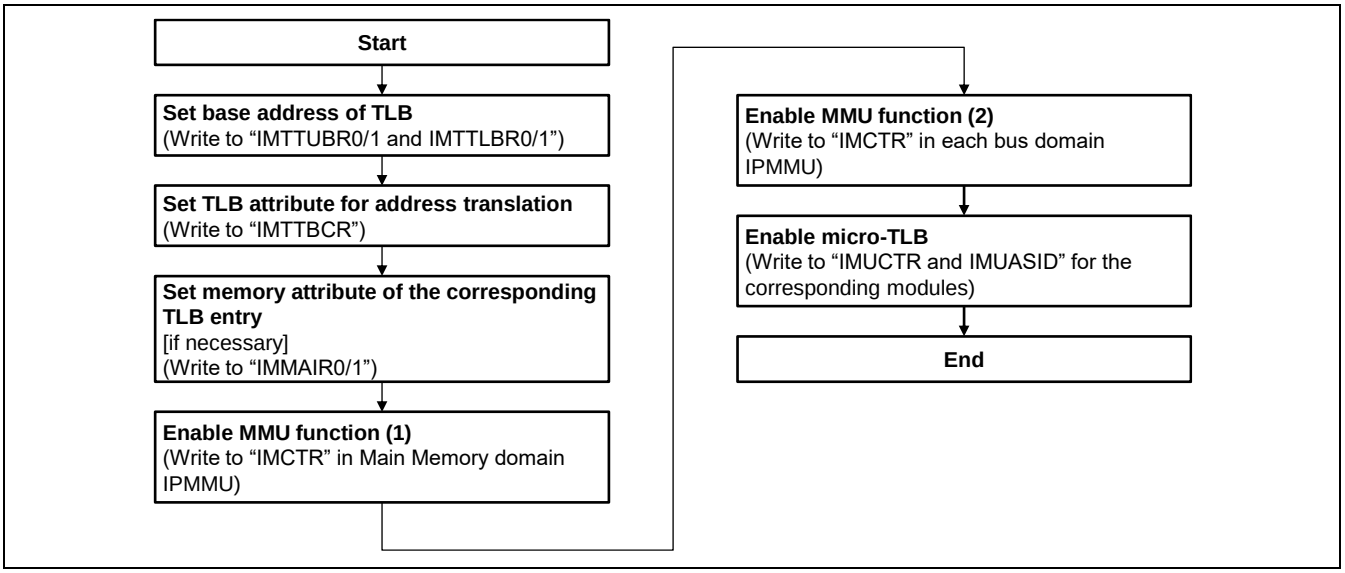



**Figure 19.10 initialization sequence for using MMU function**

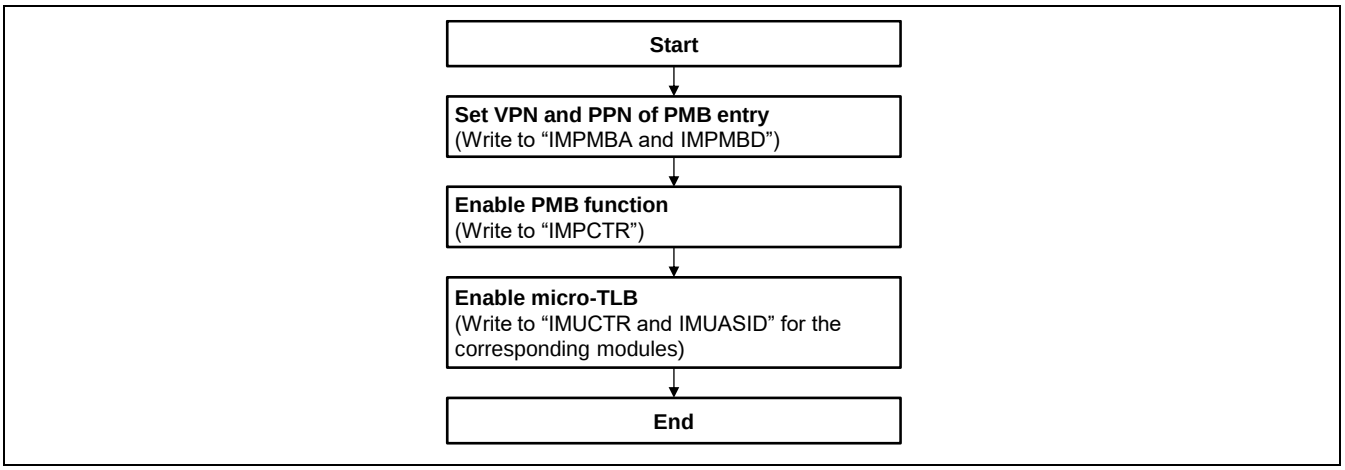



**Figure 19.11 initialization sequence for using PMB function**

### 19.3.5 TLB Maintenance

Some TLB maintenances between an IPMMU and the connected micro-TLBs are performed automatically by hardware. In case that the flush operation is performed via IMCTRn (IMCTRn FLUSH=1), all the micro-TLB which is related to the same MMU tables are applied the TLB flush and clear the all TLB entry in micro-TLB.

Table 19.3 TLB Maintenance

| Category | Software operation | TLB maintenance in IPMMU | TLB maintenance in micro-TLB |
|---|---|---|---|
| MMU Page Table | First entry registration | Not required | Not required |
| | Entry update | Invalidate the target TLB (IMCTRn.FLUSH = 1) | Automatically invalidated |
| | Entry release | Invalidate the target TLB (IMCTRn.FLUSH = 1) | Automatically invalidated |
| ASID | ASID update | Automatically invalidated | Automatically invalidated |
| MMU registers | MMU register values update | Invalidate the target TLB (IMCTRn.FLUSH = 1) | Automatically invalidated |
| PMB | PMB entry registration | Not required | Not required |
| | PMB entry update | Not required | Invalidate the target micro-TLB |
| | PMB entry release | Not required | Invalidate the target micro-TLB |

IPMMU-main cannot issue the flush operation to each IPMMU-cache and micro-TLB. In case of replacing the MMU table, the flush operation of both IPMMU-main and IPMMU-cache via IMCTRn(IMCTRn.FLUSH=1) is required. TLB invalidation is instantaneous so software doesn't need to check IMUCTRn.FLUSH or IMCTRn.FLUSH whether TLB invalidation is completed. Figure 19.12 shows the example of the procedure for updating TLB entry

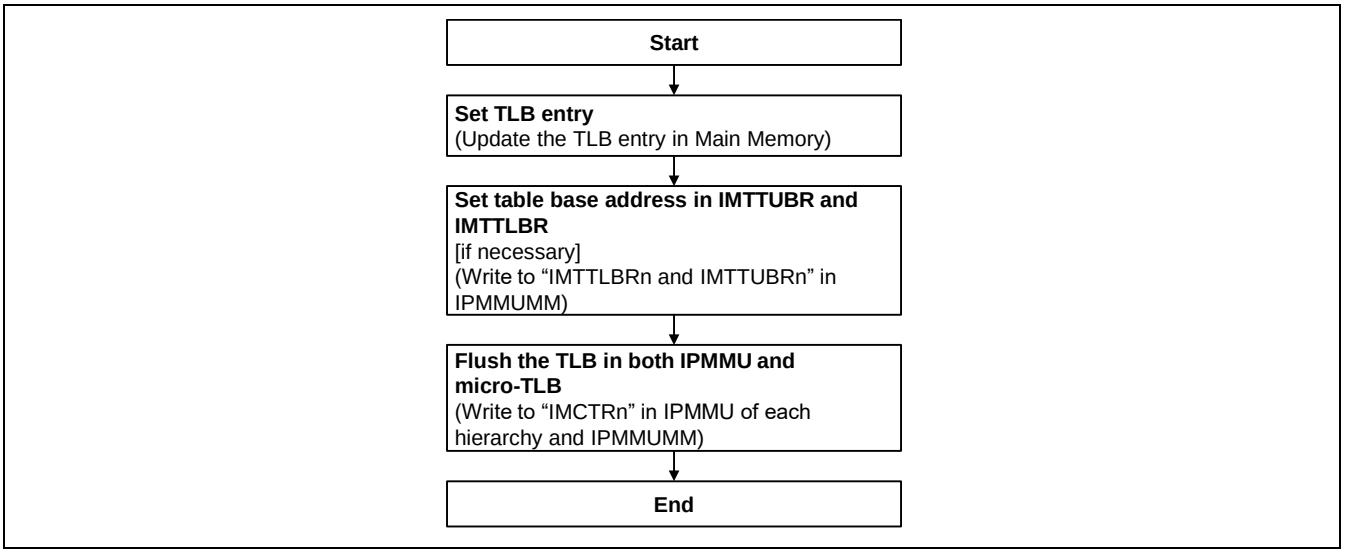



**Figure 19.12 update sequence of TLB entry**

### 19.3.6 Security Support

The IPMMU supports the security function based on Armv8 and additional extensions.

**(1) Banked Registers**

The IPMMU has two copies of a register at the same address. They are banked with secure and non-secure. A Non-secure access can read and write only non-secure registers and a secure access can read and write secure registers. A secure access can also read and write non-secure registers through the alias address space to non-secure registers (from H'4000 to H'7FFC).

**(2) Address Translation**

When a Secure/Accelerator master sends a translation request to an IPMMU through a micro-TLB, Secure MMU registers are referred to perform its page table walk. The Secure page table walk can read secure page table entries (NS = 0). In the case of a Non-secure translation request, Non-secure MMU registers are referred and the access to secure page table entries is ignored.

**(3) Secure Master**

Realtime Core can be secure master. And ICUMX is also secure master*. These modules use secure page table and Secure MMU registers.

Note: * ICUMX can be select the operation mode by the software. So it depends on the software whether the secure TLB and registers are used.

Note:** In case that secure master uses the address translation feature by IPMMU, use long descriptor format only is supported.

### 19.3.7 OS-ID support

IPMMU-HC, IPMMU-3DG and IPMMU-IR support the OS-ID which is used for specifying the operating system. In order to use this feature, the micro-TLB which corresponds to the OS-ID needs to be used. Table 19.4 and Table 19.5 show the micro-TLB assignment for each OS-ID.

Table 19.4 micro-TLB assignment for OS-ID support

| OS-ID | Micro-TLB assignment |
|---|---|
| ID 0 | uTLB0, uTLB8, uTLB16, uTLB24, uTLB32, uTLB40, uTLB48 and uTLB56 |
| ID 1 | uTLB1, uTLB9, uTLB17, uTLB25, uTLB33, uTLB41, uTLB49 and uTLB57 |
| ID 2 | uTLB2, uTLB10, uTLB18, uTLB26, uTLB34, uTLB42, uTLB50 and uTLB58 |
| ID 3 | uTLB3, uTLB11, uTLB19, uTLB27, uTLB35, uTLB43, uTLB51 and uTLB59 |
| ID 4 | uTLB4, uTLB12, uTLB20, uTLB28, uTLB36, uTLB44, uTLB52 and uTLB60 |
| ID 5 | uTLB5, uTLB13, uTLB21, uTLB29, uTLB37, uTLB45, uTLB53 and uTLB61 |
| ID 6 | uTLB6, uTLB14, uTLB22, uTLB30, uTLB38, uTLB46, uTLB54 and uTLB62 |
| ID 7 | uTLB7, uTLB15, uTLB23, uTLB31, uTLB39, uTLB47, uTLB55 and uTLB63 |

PCIe ch0/ch1 in IPMMU-HC have 16 OS-ID, the assignment is as below.

Table 19.5 micro-TLB assignment for 16 OS-ID support

| OS-ID | Micro-TLB assignment |
|---|---|
| ID 0 | uTLB0, uTLB16, uTLB32 and uTLB48 |
| ID 1 | uTLB1, uTLB17, uTLB33 and uTLB49 |
| ID 2 | uTLB2, uTLB18, uTLB34 and uTLB50 |
| ID 3 | uTLB3, uTLB19, uTLB35 and uTLB51 |
| ID 4 | uTLB4, uTLB20, uTLB36 and uTLB52 |
| ID 5 | uTLB5, uTLB21, uTLB37 and uTLB53 |
| ID 6 | uTLB6, uTLB22, uTLB38 and uTLB54 |
| ID 7 | uTLB7, uTLB23, uTLB39 and uTLB55 |
| ID 8 | uTLB8, uTLB24, uTLB40 and uTLB56 |
| ID 9 | uTLB9, uTLB25, uTLB41 and uTLB57 |
| ID 10 | uTLB10, uTLB26, uTLB42 and uTLB58 |
| ID 11 | uTLB11, uTLB27, uTLB43 and uTLB59 |
| ID 12 | uTLB12, uTLB28, uTLB44 and uTLB60 |
| ID 13 | uTLB13, uTLB29, uTLB45 and uTLB61 |
| ID 14 | uTLB14, uTLB30, uTLB46 and uTLB62 |
| ID 15 | uTLB15, uTLB31, uTLB47 and uTLB63 |

### 19.3.8 Error Handling

#### (1) IPMMU Interrupt Requests

Table 19.6 shows the error events which can cause interrupts. They are all errors which IPMMU supports.
To enable interrupt in MMU address translation, set IMCTRn.INTEN in IPMMU(main) before starting address translation. These errors are set in IMSTRn registers in IPMMU(main).
To enable interrupt in PMB address translation, set IMPCTRn.INTEN in each IPMMU(cache) before starting address translation. The errors are set in IMPSTR register in IPMMU(cache).

Table 19.6 IPMMU Error Events

| Category | Error | Description |
|---|---|---|
| Long-descriptor<br>Short-descriptor** | Translation Fault | A virtual address was not mapped in the region defined by TTBR0 and TTBR1.* |
| | | A descriptor was neither Table nor Block format |
| | Page Fault | AP (access permissions) / AF (access flag) bit mismatch occurred. |
| | | NS bit mismatch occurred. Non-secure master read a Secure page table entry. |
| | Abort | The IPMMU got an error response from an Interconnect during a page table walk. |
| | Multiple Hit (MHIT) | Multiple TLB hits occurred. |
| PMB*** | Translation Fault | A virtual address was not mapped in IMPMBAn. |
| | Multiple Hit (MHIT) | Multiple PMB hits occurred. |

Note: * This factor is only for long-descriptor.
** All of IPMMU error events in MMU translation are detected in IMSTRn registers in IPMMU(main).
*** Interrupts from IPMMU(cache) are caused only PMB address translation.

### (2) Error Handling Flow

When the micro-TLB cannot translate its virtual address because of some IPMMU errors, the micro-TLB stops to issue the following transactions. When the IPMMU detects an error, IMCTRn.FLUSH must be set after the IMSTRn register is cleared and page table entries are updated. After the TLB invalidation, the micro-TLB resumes to issue transactions. And when the IPMMU detects an error by PMB address translation, IMUCTRn.FLUSH must be set after the IMPSTRn register is cleared and PMB translation tables are updated.

Figure 19.13 shows the error handling flow in MMU address translation.

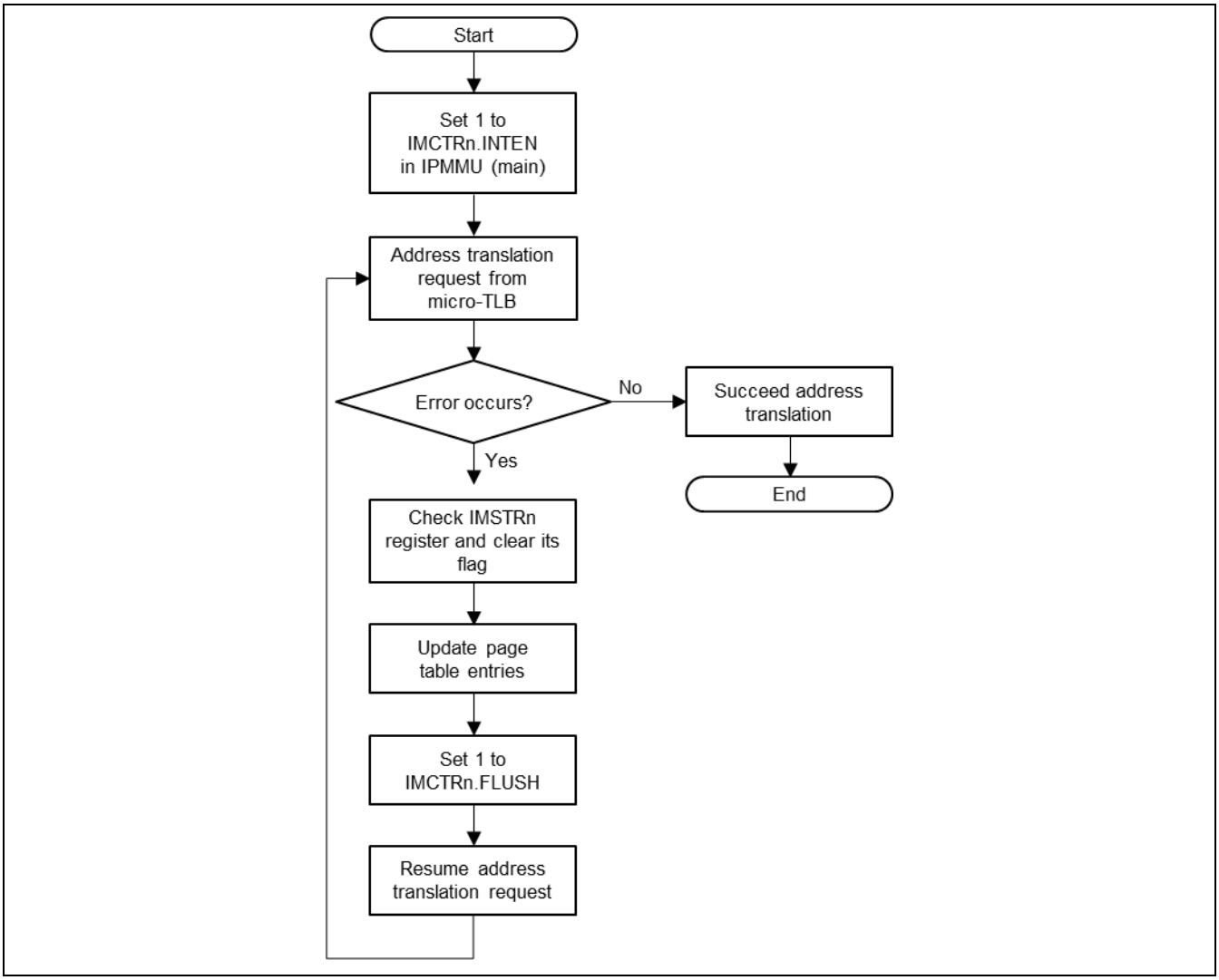



**Figure 19.13 Error Handling Flow in MMU address translation**

Figure 19.14 shows the error handling flow in PMB address translation.

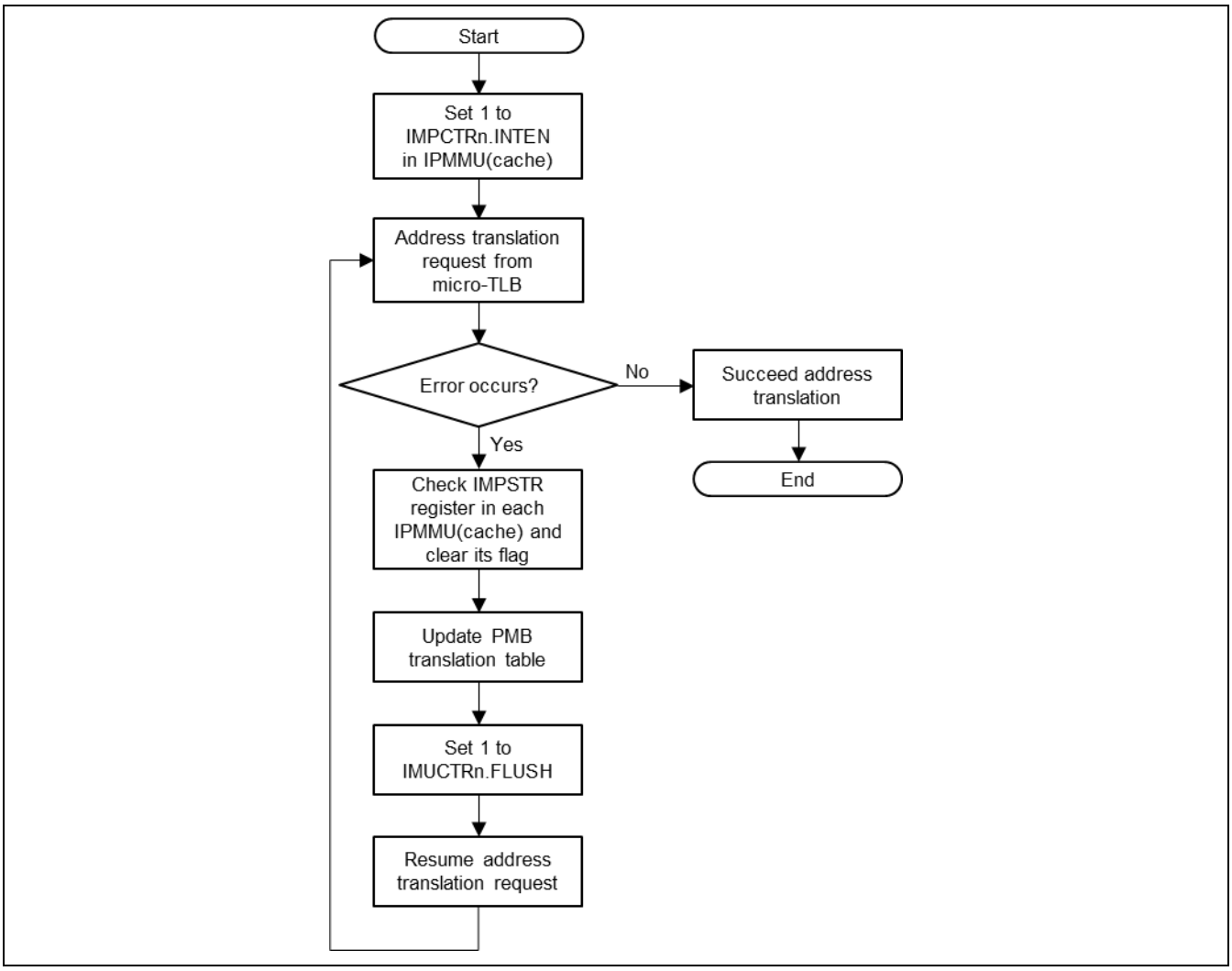



**Figure 19.14 Error Handling Flow in PMB address translation**

### 19.3.9 Performance monitor

In order to check the number of the page hit or miss, performance monitor is helpful for this purpose. When you use performance monitor, set the IMPFMCTR.MD[1:0] bits. After that set 1 to IMPFMCTR.EN bit. After you set 1 to IMPFMCTR.EN, IMPFMTOTAL, IMPFMHIT, IMPFML3MISS and IMPFML2MISS registers start counting.

Table 19.7 and Table 19.8 shows detail of performance monitor.

Note: n = 0 to 15 (IPMMU context)

**Table 19.7 Description of performance monitor in IPMMU cache**

| Register name | IMPFMCTR.MD[1:0] | description of monitor |
|---|---|---|
| IMPFMTOTAL | 2'b00 | The total number of translation requests |
| | 2'b11 | The total number of translation requests only in MMU[n] (specified by the IMPFMCTR.SEL[3:0] bits) |
| | 2'b01/10 | Reserved |
| IMPFMHIT | 2'b00 | The total number of TLB hit requests |
| | 2'b11 | The total number of TLB hit requests only in MMU[n] (specified by the IMPFMCTR.SEL[3:0] bits) |
| | 2'b01/10 | Reserved |
| IMPFMMISS | 2'b00 | The total number of miss requests |
| | 2'b11 | The total number of miss requests only in MMU[n] (specified by the IMPFMCTR.SEL[3:0] bits) |
| | 2'b01/10 | Reserved |

**Table 19.8 Description of performance monitor in main IPMMU**

| Register name | IMPFMCTR.MD[1:0] | description of monitor |
|---|---|---|
| IMPFMTOTAL | 2'b00 | The total number of translation requests of all 40-bit MMUs |
| | 2'b01 | The total number of translation requests of all 32-bit MMUs |
| | 2'b10 | Reserved |
| | 2'b11 | The total number of translation requests only in MMU[n] (specified by the IMPFMCTR.SEL[3:0] bits) |
| IMPFMHIT | 2'b00 | The total number of TLB hit requests (= 40-bit L3 TLB hit) |
| | 2'b01 | The total number of TLB hit requests (= 32-bit TLB hit) |
| | 2'b10 | Reserved |
| | 2'b11 | The total number of TLB hit requests only in MMU[n] (specified by the IMPFMCTR.SEL[3:0] bits) |
| IMPFML3MISS | 2'b00 | The total number of 40-bit L3 TLB miss requests (= 40-bit L2 TLB hit) |
| | 2'b01 / 2'b10 | Reserved |
| | 2'b11 | The total number of 40-bit L3 TLB miss requests only in MMU[n] (specified by the IMPFMCTR.SEL[3:0] bits) |
| IMPFML2MISS | 2'b00 | The total number of 40-bit L2 TLB miss requests not including 40-bit L1 miss (= Page Table Walk) |
| | 2'b01 | The total number of TLB miss requests (= Page Table Walk) |
| | 2'b10 | Reserved |
| | 2'b11 | The total number of TLB miss requests only in MMU[n] (specified by the IMPFMCTR.SEL[3:0] bits) |

### 19.3.10 micro-TLB Assignment

The assignment of micro-TLB. Refer to the following sections.

Table 19.9 micro-TLB Assignment

| | Master Module | | | | | | |
|---|---|---|---|---|---|---|---|
| **micro-TLB** | **IPMMU-HC** | **IPMMU-RT0** | **IPMMU-RT1** | **IPMMU-DS0** | **IPMMU-3DG** | **IPMMU-IR** | **IPMMU-VC** |
| 00 | Ether-AVB ch0 | - | RT-DMAC0 ch0 | SYS-DMAC1 ch0 | GSX I/F0 (OS_ID:0) | - | IMR2-LX6 ch0 |
| 01 | Ether-AVB ch1 | ICUMX | RT-DMAC0 ch1 | SYS-DMAC1 ch1 | GSX I/F0 (OS_ID:1) | - | IMR3-LX6 ch0 |
| 02 | Ether-AVB ch2 | ICUMX* | RT-DMAC0 ch2 | SYS-DMAC1 ch2 | GSX I/F0 (OS_ID:2) | - | - |
| 03 | - | - | RT-DMAC0 ch3 | SYS-DMAC1 ch3 | GSX I/F0 (OS_ID:3) | - | - |
| 04 | - | - | RT-DMAC0 ch4 | SYS-DMAC1 ch4 | GSX I/F0 (OS_ID:4) | - | IMR2-LX6 ch1 |
| 05 | - | - | RT-DMAC0 ch5 | SYS-DMAC1 ch5 | GSX I/F0 (OS_ID:5) | - | IMR3-LX6 ch1 |
| 06 | - | - | RT-DMAC0 ch6 | SYS-DMAC1 ch6 | GSX I/F0 (OS_ID:6) | - | - |
| 07 | - | - | RT-DMAC0 ch7 | SYS-DMAC1 ch7 | GSX I/F0 (OS_ID:7) | - | - |
| 08 | - | - | RT-DMAC0 ch8 | SYS-DMAC1 ch8 | - | FBIST RTT for IR 0 | IMR0-LX6 ch0 |
| 09 | - | - | RT-DMAC0 ch9 | SYS-DMAC1 ch9 | - | FBIST RTT for IR 1 | IMR1-LX6 ch0 |
| 10 | - | Realtime Core 0 | RT-DMAC0 ch10 | SYS-DMAC1 ch10 | - | FBIST RTT for IR 2 | |
| 11 | - | Realtime Core 1 | RT-DMAC0 ch11 | SYS-DMAC1 ch11 | - | FBIST RTT for IR 3 | |
| 12 | - | INTC-AP | RT-DMAC0 ch12 | SYS-DMAC1 ch12 | - | | - |
| 13 | - | Realtime Core 2 | RT-DMAC0 ch13 | SYS-DMAC1 ch13 | - | VDSP (128bit I/F) | - |
| 14 | - | - | RT-DMAC0 ch14 | SYS-DMAC1 ch14 | - | VDSP (256bit I/F) | iVCP1E |
| 15 | - | FBIST RTT for Application Core | RT-DMAC0 ch15 | SYS-DMAC1 ch15 | - | - | - |
| 16 | - | - | RT-DMAC1 ch0 | SYS-DMAC2 ch0 | - | IMP-X7 ch0 (osid=0) | - |
| 17 | - | - | RT-DMAC1 ch1 | SYS-DMAC2 ch1 | - | IMP-X7 ch0 (osid=1) | - |
| 18 | - | - | RT-DMAC1 ch2 | SYS-DMAC2 ch2 | - | IMP-X7 ch0 (osid=2) | - |
| 19 | - | - | RT-DMAC1 ch3 | SYS-DMAC2 ch3 | - | IMP-X7 ch0 (osid=3) | FBIST RTT for VC |
| 20 | - | - | RT-DMAC1 ch4 | SYS-DMAC2 ch4 | - | IMP-X7 ch0 (osid=4) | - |
| 21 | - | - | RT-DMAC1 ch5 | SYS-DMAC2 ch5 | - | IMP-X7 ch0 (osid=5) | - |
| 22 | - | - | RT-DMAC1 ch6 | SYS-DMAC2 ch6 | - | IMP-X7 ch0 (osid=6) | - |
| 23 | - | - | RT-DMAC1 ch7 | SYS-DMAC2 ch7 | - | IMP-X7 ch0 (osid=7) | - |

| | Master Module | | | | | | |
|---|---|---|---|---|---|---|---|
| **micro-TLB** | **IPMMU-HC** | **IPMMU-RT0** | **IPMMU-RT1** | **IPMMU-DS0** | **IPMMU-3DG** | **IPMMU-IR** | **IPMMU-VC** |
| 24 | - | - | RT-DMAC1 ch8 | - | - | IMP-X7 ch1 (osid=0) | - |
| 25 | - | - | RT-DMAC1 ch9 | - | - | IMP-X7 ch1 (osid=1) | - |
| 26 | - | - | RT-DMAC1 ch10 | - | - | IMP-X7 ch1 (osid=2) | - |
| 27 | - | - | RT-DMAC1 ch11 | - | - | IMP-X7 ch1 (osid=3) | - |
| 28 | - | - | RT-DMAC1 ch12 | - | - | IMP-X7 ch1 (osid=4) | - |
| 29 | - | - | RT-DMAC1 ch13 | - | - | IMP-X7 ch1 (osid=5) | - |
| 30 | - | - | RT-DMAC1 ch14 | - | - | IMP-X7 ch1 (osid=6) | - |
| 31 | - | - | RT-DMAC1 ch15 | - | - | IMP-X7 ch1 (osid=7) | - |
| 32 | PCIe ch0 (osid=0) | - | - | SDHI (MMC-IFch0) | - | - | - |
| 33 | PCIe ch0 (osid=1) | - | - | - | - | - | - |
| 34 | PCIe ch0 (osid=2) | - | - | - | - | - | - |
| 35 | PCIe ch0 (osid=3) | - | - | - | - | - | - |
| 36 | PCIe ch0 (osid=4) | - | - | - | - | - | - |
| 37 | PCIe ch0 (osid=5) | - | - | - | - | - | - |
| 38 | PCIe ch0 (osid=6) | - | - | - | - | - | - |
| 39 | PCIe ch0 (osid=7) | - | - | Flexray | - | - | - |
| 40 | PCIe ch0 (osid=8) | - | - | - | - | - | - |
| 41 | PCIe ch0 (osid=9) | - | - | - | - | - | - |
| 42 | PCIe ch0 (osid=10) | - | - | - | - | - | - |
| 43 | PCIe ch0 (osid=11) | - | - | - | - | - | - |
| 44 | PCIe ch0 (osid=12) | - | - | - | - | - | - |
| 45 | PCIe ch0 (osid=13) | - | - | - | - | - | - |
| 46 | PCIe ch0 (osid=14) | - | - | - | - | - | - |
| 47 | PCIe ch0 (osid=15) | - | - | - | - | - | - |
| 48 | - | - | - | - | - | - | - |
| 49 | - | - | - | - | - | - | - |

| | Master Module | | | | | | |
|---|---|---|---|---|---|---|---|
| **micro-TLB** | **IPMMU-HC** | **IPMMU-RT0** | **IPMMU-RT1** | **IPMMU-DS0** | **IPMMU-3DG** | **IPMMU-IR** | **IPMMU-VC** |
| 50 | - | - | - | - | - | - | - |
| 51 | - | - | - | - | - | - | - |
| 52 | - | - | - | - | - | - | - |
| 53 | - | - | - | - | - | - | - |
| 54 | - | - | - | - | - | - | - |
| 55 | - | - | - | - | - | - | - |
| 56 | - | - | - | - | - | - | - |
| 57 | - | - | - | - | - | - | - |
| 58 | - | - | - | - | - | - | - |
| 59 | - | - | - | - | - | - | - |
| 60 | - | - | - | - | - | - | - |
| 61 | - | - | - | - | - | - | - |
| 62 | - | - | - | - | - | - | - |
| 63 | - | - | - | - | - | - | - |

Note1: * Register setting for micro-TLB of IPMMURT0-02 must be set same as IPMMURT0-01.

| | Master Module | | |
|---|---|---|---|
| **micro-TLB** | **IPMMU-VI0** | **IPMMU-VI1** | **IPMMU-VIP0** |
| 00 | VIN ch0-7 | - | - |
| 01 | VIN ch8-15 | - | SMPO |
| 02 | - | - | SMPS |
| 03 | - | - | DOF |
| 04 | - | - | - |
| 05 | - | - | FBIST RTT for VIP |
| 06 | - | FCPVD ch0 | - |
| 07 | - | - | - |
| 08 | - | - | - |
| 09 | - | - | - |
| 10 | ISP0 ch0 | - | - |
| 11 | - | - | - |
| 12 | - | - | - |
| 13 | - | - | - |
| 14 | ISP0 ch1 | - | - |
| 15 | - | - | - |
| 16 | - | - | PAP-SubSystem |
| 17 | - | - | - |
| 18 | ISP0 ch2 | DSI ch0 | - |
| 19 | - | - | - |
| 20 | - | - | - |
| 21 | - | - | - |
| 22 | ISP0 ch3 | - | - |
| 23 | - | FBIST RTT for VIO | - |
| 24 | - | FCPVX ch0 | - |
| 25 | - | - | - |
| 26 | ISP0 ch4 | - | - |
| 27 | - | - | - |
| 28 | - | - | - |
| 29 | - | - | - |
| 30 | - | - | - |
| 31 | - | - | - |
| 32 | - | - | - |
| 33 | - | - | - |
| 34 | - | - | - |
| 35 | - | - | - |
| 36 | - | - | - |
| 37 | - | - | - |
| 38 | - | - | - |
| 39 | - | - | - |
| 40 | - | - | - |

| | Master Module | | |
|---|---|---|---|
| **micro-TLB** | **IPMMU-VI0** | **IPMMU-VI1** | **IPMMU-VIP0** |
| 41 | - | - | - |
| 42 | - | - | - |
| 43 | - | - | - |
| 44 | - | - | - |
| 45 | - | - | - |
| 46 | - | - | - |
| 47 | - | - | - |
| 48 | - | - | - |
| 49 | - | - | - |
| 50 | - | - | - |
| 51 | - | - | - |
| 52 | - | - | - |
| 53 | - | - | - |
| 54 | - | - | - |
| 55 | - | - | - |
| 56 | - | - | - |
| 57 | - | - | - |
| 58 | - | - | - |
| 59 | - | - | - |
| 60 | - | - | - |
| 61 | - | - | - |
| 62 | - | - | - |
| 63 | - | - | - |

## 19.4 Usage Notes

### 19.4.1 IPMMU Naming

Table 19.10 shows the correspondence between each IPMMU and each bus domain.

**Table 19.10 IPMMU naming**

| IPMMU | Bus Domain |
|---|---|
| IPMMU-VI0 | VIO (Video IO) domain AXI |
| IPMMU-VI1 | VIO (Video IO) domain AXI |
| IPMMU-VC | VC (Video Codec) domain AXI |
| IPMMU-IR | IMP domain AXI |
| IPMMU-RT0 | RT (Real Time) domain AXI |
| IPMMU-RT1 | RT (Real Time) domain AXI |
| IPMMU-DS0 | Peripheral domain AXI |
| IPMMU-HC | HC (High Communication) domain AXI |
| IPMMU-3DG | 3DG (3D-Graphics) domain AXI |
| IPMMU-VIP0 | VIP (Vision IP) domain AXI |
| IPMMU-MM | Main Memory domain AXI |

### 19.4.2 Restrictions for IPMMU

**(1) Page size and Section size usage in VMSAv8-32**

IPMMU cannot handle both 4KB page and 64KB page at the same MMU context in VMSAv8-32. In case of using the 64KB page, IMTTBCRn.PGSZ=1 is required. Also 1MB section and 16MB super-section cannot be handled. In case of using 16MB section, IMTTBCRn.SCSZ=1 is required.

**(2) 4KB page usage in VMSAv8-64**

Four level translation is not supported in IPMMU. In case of using 4KB page granule, we should set "0x1" to "SL" field. (start from 1st level) and use concatenated translation table. Architecturally, this table optimization is supported in stage 2 translation, but IPMMU supports this mode in stage 1 translation as well in order to reduce the number of page table walk. In table lookup, MSB of input address (IA[39]) is used for the page table entry selection for upper or lower table.

**(3) Error Handling in VMSAv8-64**

Architecturally, Address Size Fault is described in VMSAv8-64. But this fault is not handled in IPMMU. On the other hand, the NS bit in a Non-secure translation table descriptor need to be set in IPMMU although it is ignored in VMSAv8-64. In order to avoid “Secure Access Fault”, set “NS bit” in a Non-secure translation table descriptor.

**(4) Start Level for Page Table Walk**

Start Level for page table walk need to be set under below condition. Refer to Table 19.11.

Table 19.11 Start Level for Page Table Walk

| IMCTRn.VA64 = 1 IMTTBCRn.PGSZ = 0 (use 4KB page granule) | | |
|---|---|---|
| Look Up | Stage-1 Translation | Stage-2 Translation |
| Level 0 | No support | No Support |
| Level 1 | base address[39:x]: Input Address[y:30]<br>if IMTTBCRn.SL[1:0] = 2’b01<br>x = (37 – TSZn (n = 0, 1)) [24* <= TSZn <= 33] | base address[39:x]: Input Address[y:30]<br>if IMTTBCRn.SL[1:0] = 2’b01<br>x = (37 – TSZn (n = 0, 1)) [24* <= TSZn <= 33] |

| | | |
|---|---|---|
| | y = x+26<br>Other values<br>go to Level 2 | y = x+26<br>Other values<br>go to Level 2 |
| Level 2 | base address [39:x], Input Address[y:21]<br>if IMTTBCRn.SL[1:0] = 2'b00<br>x = (46 – TSZn (n = 0, 1)) [30 <= TSZn <= 39]<br>y = x +17<br>Other values<br>x = (46 -TSZn (n = 0,1)) [34 <= TSZn <= 39]<br>y = x + 17 | base address[39:x]: Input Address[y:21]<br>if IMTTBCRn.SL[1:0] = 2'b00<br>x = (46 – TSZn (n = 0,1)) [30 <= TSZn <= 39]<br>y = x+17<br>Other values<br>No support |
| IMCTRn.VA64 = 1 IMTTBCRn.PGSZ = 1 (use 64KB page granule) | | |
| Level 1 | No Support | No Support |
| Level 2 | base address [39:x], Input Address[y:29]<br>if IMTTBCRn.SL[1:0] = 2'b01<br>x = (38 – TSZn (n = 0, 1)) [24 <= TSZn <= 34]<br>y = x + 25<br>Other values<br>go to Level 3 | base address [39:x], Input Address[y:29]<br>if IMTTBCRn.SL[1:0] = 2'b01<br>x = (38 – TSZn (n = 0, 1)) [24 <= TSZn <= 34]<br>y = x + 25<br>Other values<br>go to Level 3 |
| Level 3 | base address [39:x], Input Address [y:16]<br>if IMTTBCRn.SL[1:0] = 2'b00<br>x = (51 – TSZn (n = 0, 1)) [31 <= TSZn <= 39]<br>y= x + 12<br>Other values<br>x = (51 – TSZn (n = 0, 1)) [35 <= TSZn <= 39]<br>y= x + 12 | base address [39:x], Input Address [y:16]<br>if IMTTBCRn.SL[1:0] = 2'b00<br>x = (51 – TSZn (n = 0, 1)) [31 <= TSZn <= 39]<br>y= x + 12<br>Other values<br>x = (51 – TSZn (n = 0, 1)) [35 <= TSZn <= 39]<br>y= x + 12 |
| IMCTRn.VA64 = 0 | | |
| Level 1 | base address[39:x]: Input Address[y:30]<br>if IMTTBCRn.SL[1] = 1'b1<br>x = (5 – TSZn (n = 0, 1)) [0 <= TSZn <= 1]<br>y = x + 26<br>Other values<br>go to Level 2 | base address[39:x]: Input Address[y:30]<br>if IMTTBCRn.SL[1] = 1'b1<br>x = (5 – TSZ0) [-2 <= TSZ0 <= 1]<br>y = x + 26<br>Other values<br>x = (5 – TSZ0) [-8 <= TSZ0 <= -3]<br>y = x + 26 |
| Level 2 | base address[39:x]: Input Address[y:21]<br>if IMTTBCRn.SL[1] = 1'b0<br>x = (14 – TSZn (n = 0, 1)) [2 <= TSZn <= 7]<br>y = x+17<br>Other values<br>No support | base address[39:x]: Input Address[y:21]<br>if IMTTBCRn.SL[1] = 1'b0<br>x = (14 – TSZ0) [-2 <= TSZ0 <= 7]<br>y = x+17<br>Other values<br>No support |

Note.* not support level 0 page table walk. Refer to 19.4.2(2) 4KB page usage in VMSAv8-64.

### (5) Virtual address space range

Virtual address space range cannot be configured less than or equal 128MB. In case of supporting this address range, set virtual address space range to 256MB by setting TSZ0/TSZ1 and unnecessary area is not mapped by MMU.

**(6) Disabling the Address Translation By IPMMU**

Even if a master IP is disabling the address translation by IPMMU, IMUCTRn.TTSEL[3:0] needs to be set unused MMU context number. It means that only fifteen MMU context can be used for the address translation by IPMMU in this case.

**(7) Restriction for PMB**

In order to perform address translation by IPMMU normally, it is limited not to use PMB mode in secure mode. Use PMB mode only in non-secure mode.

## 19.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 19.5.1 EDC for IP Memory Management Unit (IPMMU)

#### (a) Overview

EDC protection is applied to internal TLB RAM in IPMMU for main memory domain (IPMMU-MM).

TLB RAM is protected by SED-DED (single error detection, double error detection) EDC. Comparison logic for both EDC generator and EDC checker are implemented in IPMMU-MM as DCLS comparison error. Even if single error is detected, it doesn't correct the original data and doesn't update the data to main-IPMMU core. Also, error notification signal of the EDC error is asserted and notify this error to ECM. Both EDC generator and EDC checker are implemented in IPMMU-MM for each RAM macro.

TLB RAM support split-RAM structure in order to detect address bit failure.

#### (b) Hardware Description

Figure 19.15 shows a block diagram of the IPMMU protection for functional safety. Internal RAM is protected by EDC checking. TLB RAM is only provided in the MM domain.

When an error is detected, IPMMU issues error signal to ECM and records error information to the IMRAM0ERRSTR0/1 or IMRAM1ERRSTR0/1.

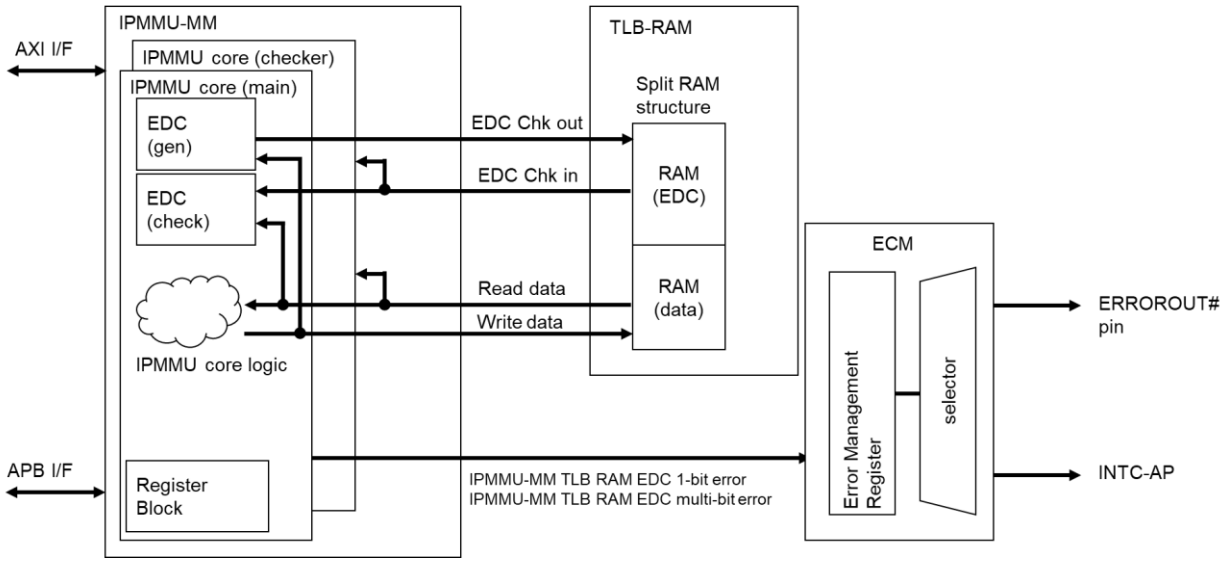



**Figure 19.15 Block diagram of the IPMMU protection for Functional Safety**

#### (c) Related Registers

Table 19.12 shows the each IPMMU base address. This table also shows which R-Car products have which IPMMUs. Basically, the IPMMU has one-to-one connections with each of the bus modules.

Table 19.12 IPMMU Base Address

| MMU | Base Address |
|---|---|
| IPMMU-MM | H'EEFC 0000 |

Table 19.13 Register Configuration

| Register Name | Abbreviation | R/W | Address Offset | Initial Value | Reset | Access Size |
|---|---|---|---|---|---|---|
| MMU RAM0 Error Control Register 0 | IMRAM0ERRCTR0 | R/W | H'0 05C0 | H'0000 0000 | Initialized | 32 |
| MMU RAM0 Error Control Register 1 | IMRAM0ERRCTR1 | R/W | H'0 05C4 | H'0000 0000 | Initialized | 32 |
| MMU RAM0 Error Status Register 0 | IMRAM0ERRSTR0 | R/WC1 | H'0 05C8 | H'0000 0000 | Initialized | 32 |
| MMU RAM0 Error Status Register 1 | IMRAM0ERRSTR1 | R/WC1 | H'0 05CC | H'0000 0000 | Initialized | 32 |
| MMU RAM1 Error Control Register | IMRAM1ERRCTR | R/W | H'0 05D0 | H'0000 0000 | Initialized | 32 |
| MMU RAM1 Error Status Register | IMRAM1ERRSTR | R/WC1 | H'0 05D4 | H'0000 0000 | Initialized | 32 |

Table 19.14 Supported Registers

| Register Abbreviation | IPMMU-Cache | IPMMU-MM |
|---|---|---|
| IMRAM0ERRCTR0 | - | √ |
| IMRAM0ERRCTR1 | - | √ |
| IMRAM0ERRSTR0 | - | √ |
| IMRAM1ERRCTR1 | - | √ |
| IMRAM1ERRCTR | - | √ |
| IMRAM1ERRSTR | - | √ |

About register description of these registers, refer to R-CarV4M_UM_019_IPMMU_r0p70.xlsx and R-CarV4M_UM_019_IPMMU_long_descripter_r0p70.xlsx.

**(d) Operation and Error Assignment of ECM**

When an error is detected, the error notified to ECM module that is reported to INTC or the external systems through ERROROUT# pin. For details, see “Section 17, Error Control Module (ECM)”.

[Initial Setting]
Set the corresponding bits of the ECMERRCTLRn register 1’b1 to be enable error detection.
And set corresponding bits of the ECMERRTGTRn register 1’b1 to inform INTC (Interrupt Controller) the error or set 1’b0 to inform the external device of the error via the ERROROUT# pin.

| ECMERRCTLRn | ECMERRTGTRn | Error Type |
|---|---|---|
| ECMERRCTLR21[22] = 1’b1 | ECMERRTGTR21[22] | IPMMU-MM TLB RAM EDC 1-bit error |
| ECMERRCTLR21[21] = 1’b1 | ECMERRTGTR21[21] | IPMMU-MM TLB RAM EDC multi-bit error |

### 19.5.2 BUS EDC Protection for IPMMU

#### (a) Overview

EDC protection is applied to AXI-bus and APB-bus in IPMMU for main memory domain (IPMMU-MM) and IPMMU for each hierarchy domain (IPMMU-Cache).

#### (b) Hardware Description

Figure 19.16 and Figure 19.17 show a block diagram of the AXI-bus and APB-bus in IPMMU protection for functional safety. AXI-bus and APB-bus in IPMMU are protected by EDC checking. And these EDC modules are protected by DCLS comparison structure.
EDC for AXI-bus and APB-bus in IPMMU are implemented in both slave side and master side of IPMMU-MM.
EDC for AXI-bus and APB-bus in IPMMU are implemented in both slave side and master side of IPMMU-Cache.

And these errors have the function of error injection.

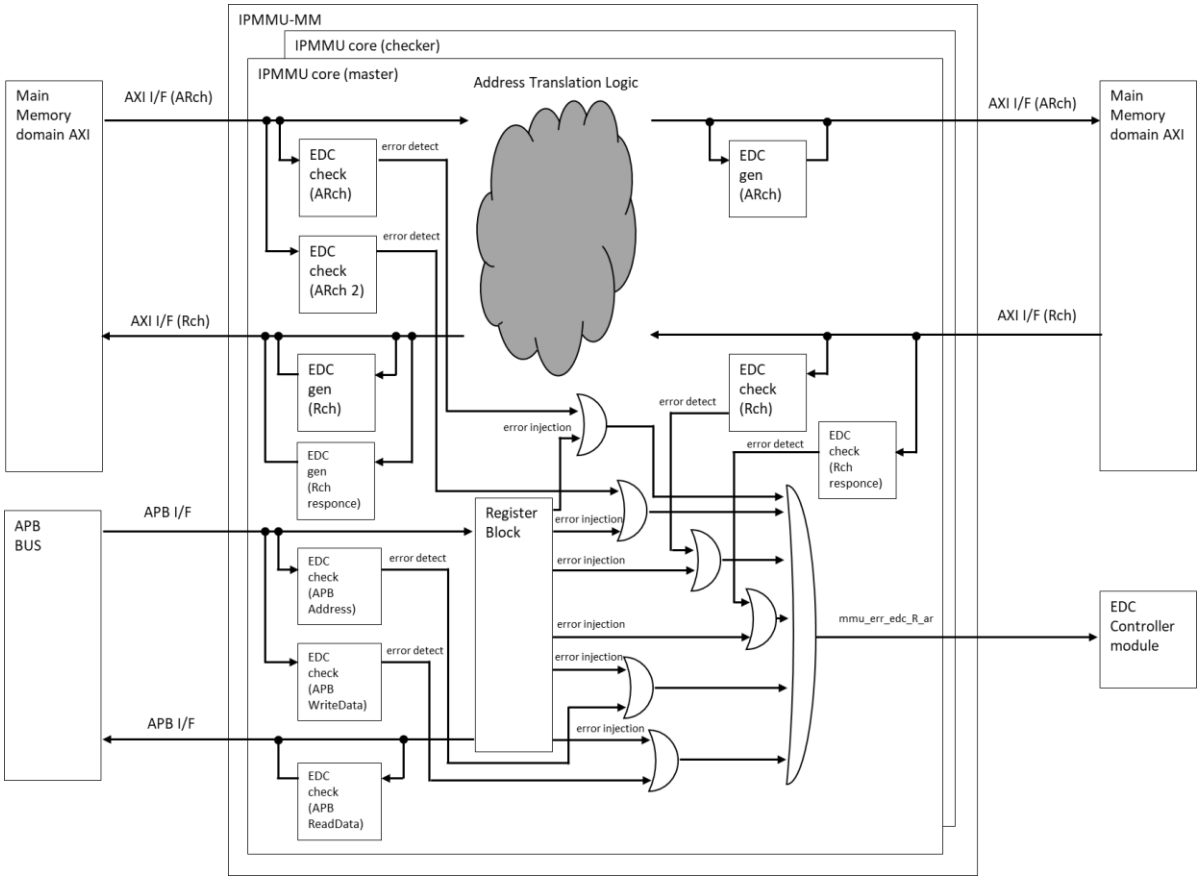



**Figure 19.16 Block diagram of EDC protection of IPMMU-MM**

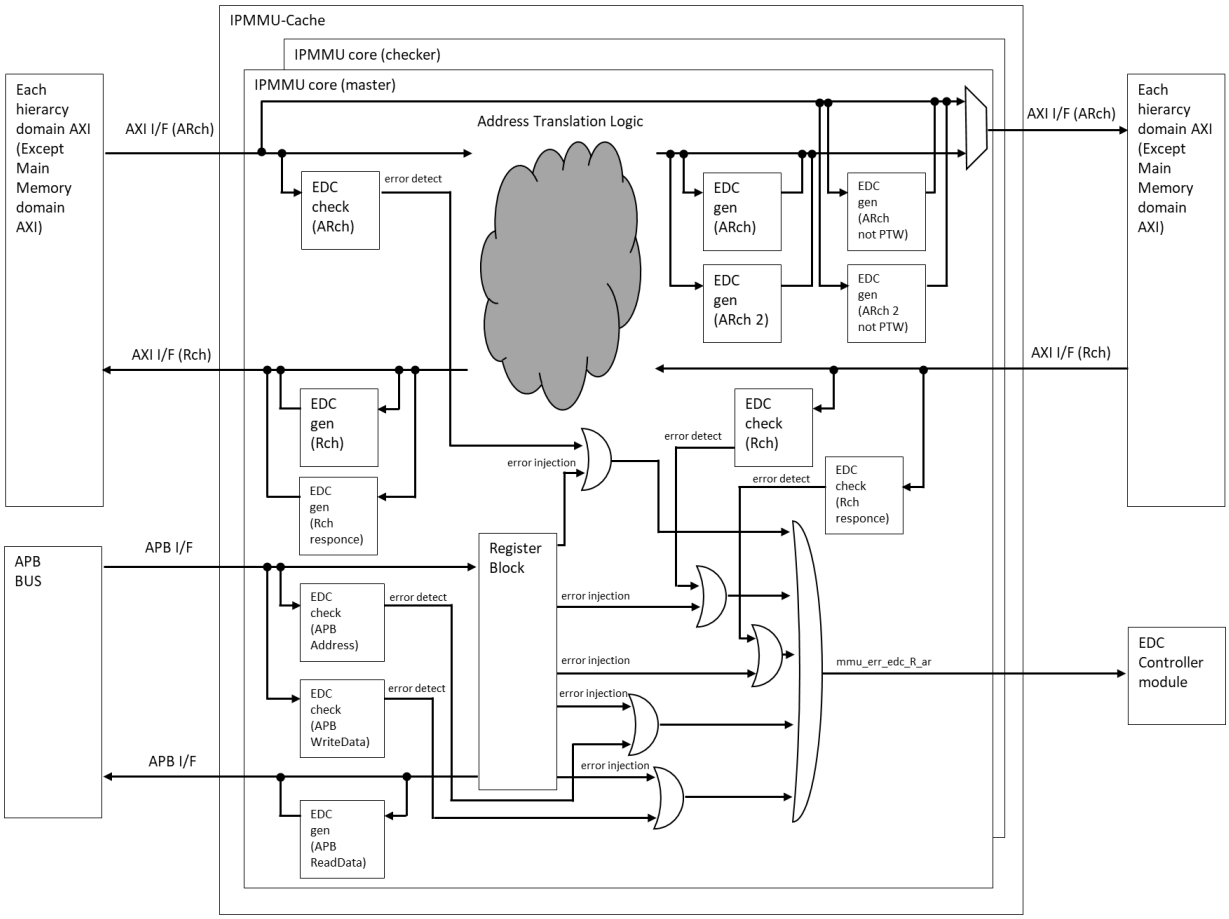



**Figure 19.17 Block diagram of EDC protection of IPMMU-Cache**

### (c) Related Registers

**Table 19.15 IPMMU Base Address**

| MMU | Base Address |
|---|---|
| IPMMU-MM | H'EEFC 0000 |
| IPMMU-RT0 | H'EE48 0000 |
| IPMMU-RT1 | H'EE4C 0000 |
| IPMMU-DS0 | H'EED0 0000 |
| IPMMU-HC | H'EED4 0000 |
| IPMMU-3DG | H'EEE0 0000 |
| IPMMU-IR | H'EED8 0000 |
| IPMMU-VC | H'EEDC 0000 |
| IPMMU-VI0 | H'EEE8 0000 |
| IPMMU-VI1 | H'EEEC 0000 |
| IPMMU-VIP0 | H'EEF0 0000 |

Table 19.16 Register Configuration

| Register Name | Abbreviation | R/W | Address Offset | Initial Value | Reset | Access Size |
|---|---|---|---|---|---|---|
| MMU System Error Injection Register | ISERRINJR | R/W | H'0554(IPMMU-MM)<br>H'1554(IPMMU-Cache) | H'0000 0000 | Initialized | 32 |

Table 19.17 Supported Registers

| Register Abbreviation | IPMMU-Cache | IPMMU-MM |
|---|---|---|
| ISERRINJR | √ | √ |

About register description of these registers, refer to R-CarV4M_UM_019_IPMMU_r0p70.xlsx and R-CarV4M_UM_019_IPMMU_long_descripter_r0p70.xlsx.

### 19.5.3 Hardware Redundancy for IPMMU

#### (a) Overview

Hardware redundancy is applied to the following IPMMUs in order to detect fault of functional logic.

— IPMMU-RT0 and IPMMU-RT1 in RT domain
— IPMMU-DS0 in PER0 domain
— IPMMU-MM in MM domain
— IPMMU-HC in HSC domain
— IPMMU-3DG in 3DG domain
— IPMMU-IR in IR domain
— IPMMU-VC in VC domain
— IPMMU-VI0 and IPMMU-VI1 in VIO domain
— IPMMU-VIP0 in VIP domain

#### (b) Hardware Description

Figure 19.18 shows the block diagram of IPMMU protection for functional safety. Not only the body of MMU logic but also the control registers are checked by lock-step operation. When an error related to the functional safety is detected, this error is reported through ECM and then to either INTC or the external system. Even if an error is detected, address translation operation can continue.

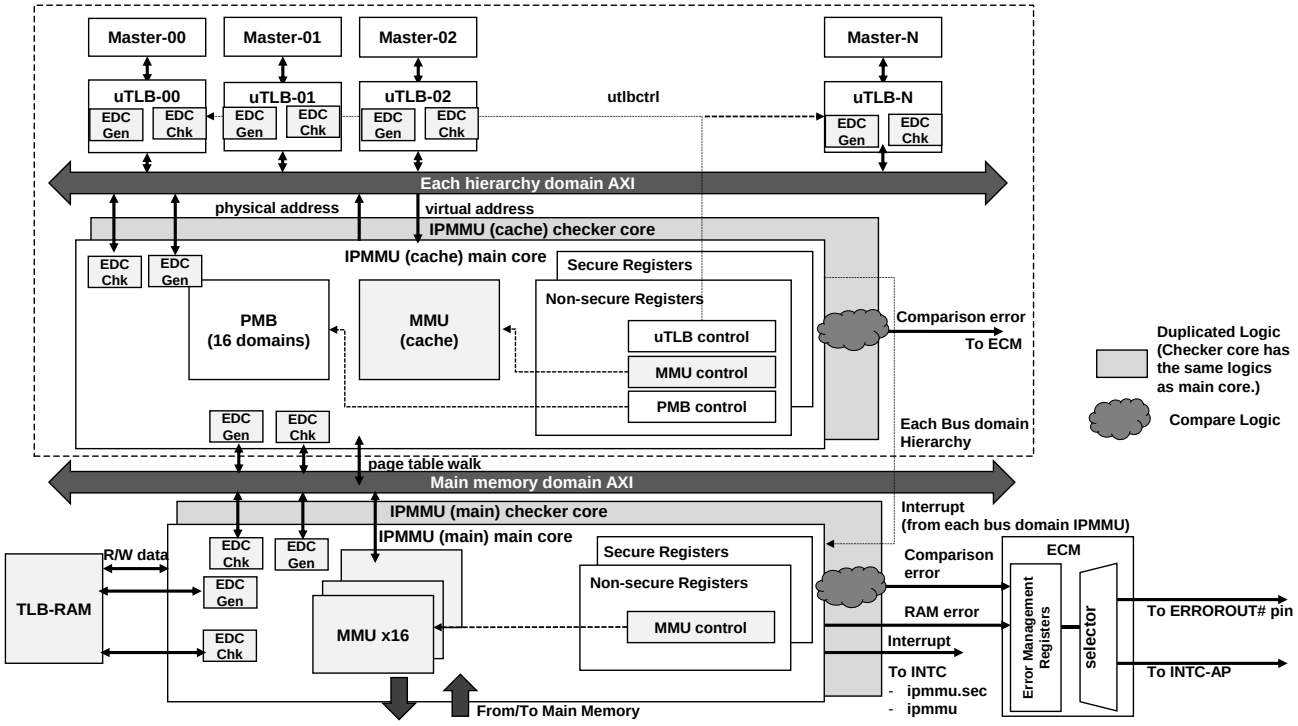



**Figure 19.18 Block diagram of Hardware Redundancy for IPMMU**

#### (c) Related Registers

Error by lock-step operation has the function of error injection.
This function could be controlled by a bit of MMU System Error Injection Register (ISERRINJR.ERRINJ_COMPFAIL). Refer MMU System Error Injection Register (ISERRINJR) in details.

#### (d) Operation and Error Assignment of ECM

When an error is detected, the error notified to ECM module that is reported to INTC (Interrupt Controller) or the external systems through ERROROUT# pin. For details, see “Section 17, Error Control Module (ECM)”.

[Initial Setting]
Set the corresponding bits of the ECMERRCTLRn register 1'b1 to be enable error detection.
And set corresponding bits of the ECMERRTGTRn register 1'b1 to inform INTC (Interrupt Controller) the error or set 1'b0 to inform the external device of the error via the ERROROUT# pin.

| ECMERRCTLRn | ECMERRTGTRn | Error Type |
|---|---|---|
| ECMERRCTLR20[4] = 1'b1 | ECMERRTGTR20[4] | IPMMU-DS0 checker core comparison error |
| ECMERRCTLR20[3] = 1'b1 | ECMERRTGTR20[3] | IPMMU-MM checker core comparison error |
| ECMERRCTLR20[2] = 1'b1 | ECMERRTGTR20[2] | IPMMU-RT0 checker core comparison error |
| ECMERRCTLR20[1] = 1'b1 | ECMERRTGTR20[1] | IPMMU-IR checker core comparison error |
| ECMERRCTLR21[31] = 1'b1 | ECMERRTGTR21[31] | IPMMU-HC checker core comparison error |
| ECMERRCTLR21[29] = 1'b1 | ECMERRTGTR21[29] | IPMMU-3DG checker core comparison error |
| ECMERRCTLR21[28] = 1'b1 | ECMERRTGTR21[28] | IPMMU-RT1 checker core comparison error |
| ECMERRCTLR21[27] = 1'b1 | ECMERRTGTR21[27] | IPMMU-VC checker core comparison error |
| ECMERRCTLR21[26] = 1'b1 | ECMERRTGTR21[26] | IPMMU-VI0 checker core comparison error |
| ECMERRCTLR21[25] = 1'b1 | ECMERRTGTR21[25] | IPMMU-VI1 checker core comparison error |
| ECMERRCTLR21[24] = 1'b1 | ECMERRTGTR21[24] | IPMMU-VIP0 checker core comparison error |

### 19.5.4 Region-ID setting for page table walk request

#### (a) Overview

Each IPMMU cache can control Region-ID in the request of page table walk.

#### (b) Hardware Description

・Table number is set for each micro-TLB by IMUCTR register and IMCTR register.
・Region ID is not related to the Context-ID (MMU-ID) which is used for page table walk from IPMMU. (IPMMU doesn't check the Region ID from each IP)
・The memory area which TLB is located need to be given an access permission based on Region-ID as Figure 19.19.

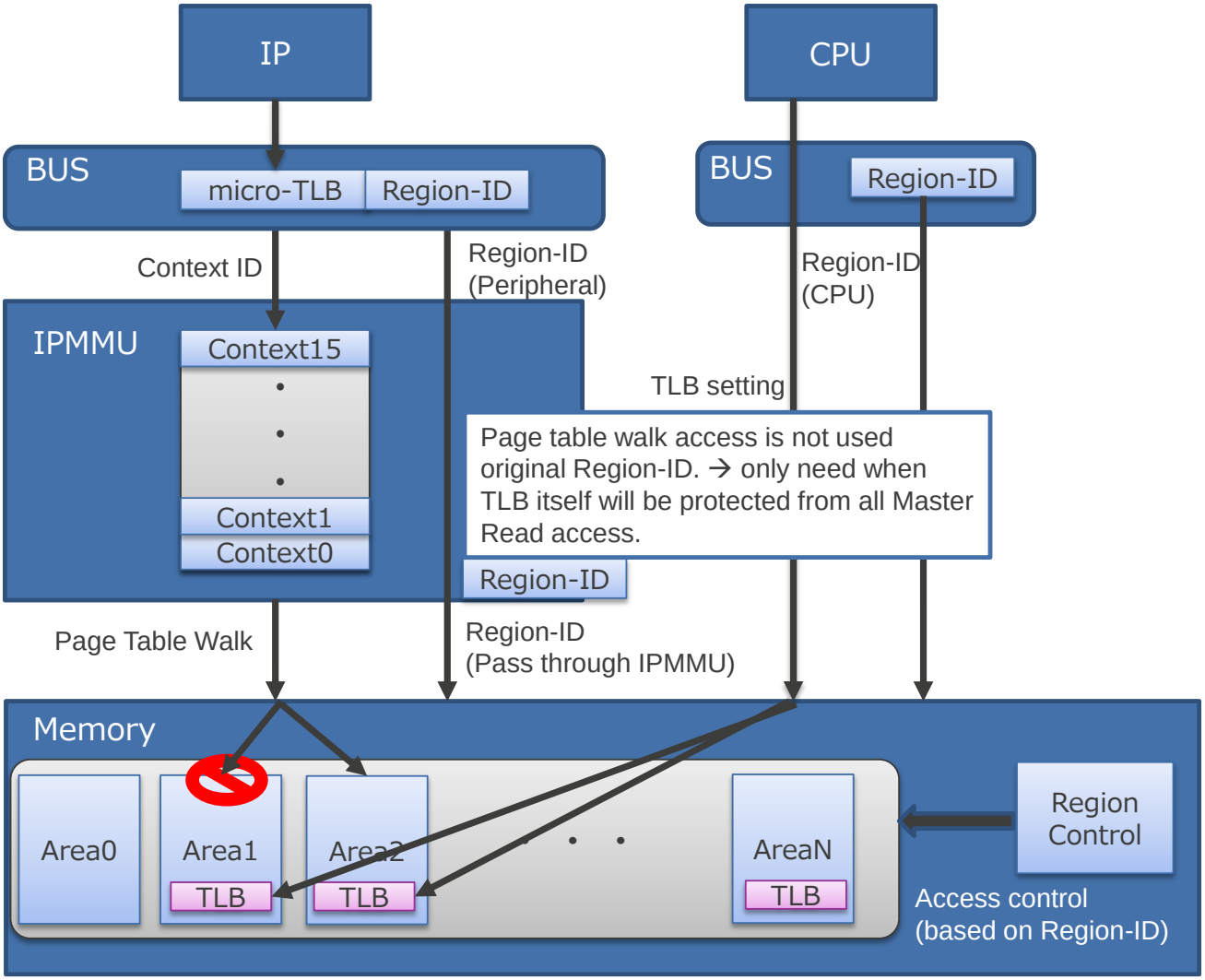



**Figure 19.19 Region-ID protection for page table walk request**

For example, Region-ID(CPU) is set to Area1 and Area2, but Region-ID (Peripheral) is set to Area2 only. In this case, TLB located in Area2 can be accessed by page table walk from Peripheral IP. But TLB located in Area1 cannot be accessed because access permission is not set.

**(c) Related Registers**

Region-ID in the request of page table walk can be set below registers in each IPMMU cache.

- MMU Region ID Register (IMRGID)
- MMU Region ID Enable Register (IMRGIDEN)
- MMU Secure Group Register (IMSECGRP)

In detail, refer to the register description.

# 30. Direct Memory Access Controller

This LSI includes a direct memory access controller for system (SYS-DMAC). The SYS-DMAC can be used in place of the CPU to handle high-speed data transfer to and from an external memory, the on-chip memory, memory-mapped external devices, or on-chip peripheral modules.

This LSI also includes a direct memory access controller in Real Time domain (RT-DMAC). The RT-DMAC can be used in place of the CPU to handle high-speed data transfer to and from an external memory, the on-chip memory, memory-mapped external devices.

## 30.1 Overview

### 30.1.1 Features

- Up to 32 DCLS channels are available due to 2 RT-DMAC instances.
- Up to 24 DCLS channels are available due to 2 SYS-DMAC instances.

- 1-T byte physical address space.
- Transfer data length: Byte, word (2 bytes), longword (4 bytes), 8 bytes, 16 bytes, 32 bytes, and 64 bytes.
- Maximum number of transfer times: 16,777,215.
- Address mode: Dual address mode.
- Transfer requests:
- Requests from on-chip peripheral modules or auto requests can be selected.
  - CRC can issue on-chip transfer requests to RT-DMAC.
  - I2C0~3, HSCIF0~3, SCIF0/1/3/4, TPU, MSIOF0~5, CANFD, SSI can issue on-chip transfer requests to SYS-DMAC.
- Selectable bus modes:
  - Normal speed mode or slow speed mode can be selected for each channel.
- Either fixed priority or round-robin arbitration can be selected for use in arbitration among the transfer channels.
- Interrupt request:
  - The DMAC can be set up to generate an interrupt request for the CPU upon completion of transfer under the control of one stage of the descriptor, at the end of the data transfer, in response to an MMU error, and in response to an address error.
- Descriptor function:
  - Up to 4096 sets of the settings for the source address register in external memory, destination address register, and transfer count register are available for use in setting up consecutive DMA transfers.
  - An infinite repeat mode is also available.
  - Event handling for descriptor suspension and execution for RT-DMAC.
- BUS I/F rate control
- FuSa:
  - All Channel FuSa DCLS supports (Core redundancy and comparison)
  - Bus protection and Memory protection (Security region assignment)
  - AXI and APB Bus EDC protection and Memory protection (Safety region assignment)

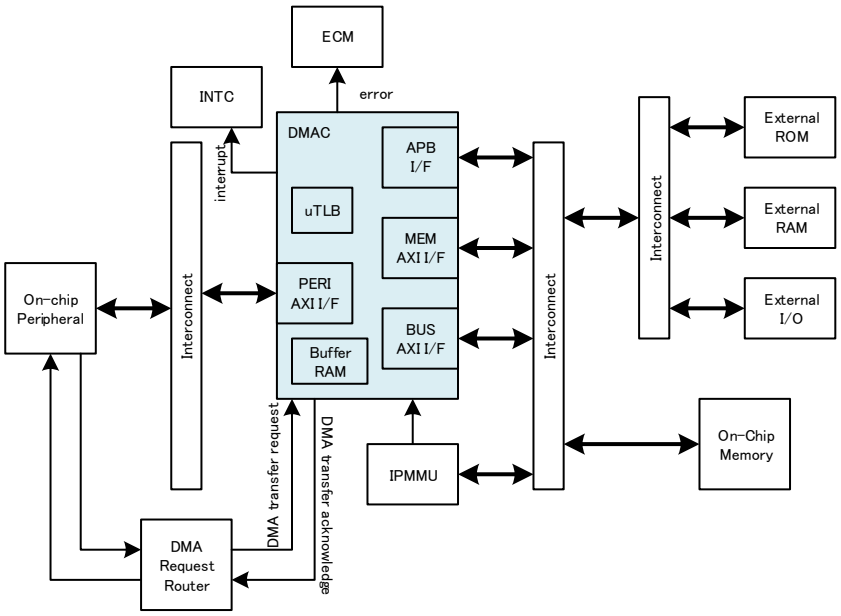



**Figure 30.1 Block Diagram of the DMAC**

### 30.1.2 Register configuration

Refer to R-CarV4M_UM_030_DMAC_r0p81.xlsx

## 30.2 Register Description

Refer to R-CarV4M_UM_030_DMAC_r0p81.xlsx

Table 30.1 SAR/DAR Address Restriction

| Resource Selection | Address Mode | Restriction |
|---|---|---|
| Auto request | Incremental | No restriction (byte boundaries) |
| | Others | Boundary corresponding to the DMA transfer size |
| On-chip peripheral module request Transmission/DAR, Reception/SAR | All | Boundary corresponding to the DMA transfer size |
| On-chip peripheral module request Transmission/SAR, Reception/DAR | Incremental | No restriction (byte boundaries) |
| | Others | Boundary corresponding to the DMA transfer size |

## 30.3 Operation

DMA transfer requests are basically generated in either the data transfer source or destination, but they can also be generated by on-chip peripheral modules that are neither the source nor the destination. Transfers can be requested in two modes: auto request, and on-chip peripheral module request. The request mode is selected for each channel in the DMRS.

### 30.3.1 DMA Transfer Requests

The transfer from/to SDRAM will use MEM I/F, and the transfer from/to peripheral will use PERI I/F, and other transfer source/destination will use BUS I/F

**(1) Auto-Request Mode**

When there is no transfer request signal from an external source, as in a memory-to-memory transfer or a transfer between memory and an on-chip peripheral module unable to request a transfer, auto-request mode allows the RT-DMAC to automatically generate a transfer request signal internally. When the DE bit in CHCR and the DME bit in DMOR are set to 1 for the target channel, the transfer begins so long as the CAE bit in CHCR is 0.

**(2) On-Chip Peripheral Module Request Mode**

This mode is in case of CHCR.RS=1000.

In this mode, a transfer is performed at the transfer request signal of an on-chip peripheral module. The source (on-chip peripheral module) of the DMA transfer request is specified by DMARS.

When this mode is selected, if the DMA transfer is enabled (DE = 1, DME = 1, TE = 0, CAE = 0), a transfer is performed upon the input of a transfer request signal.

When a transmit data empty transfer request of the peripheral is set as the transfer request, the transfer destination must be the peripheral transmit data register. Likewise, when receive data full transfer request of the peripheral is set as the transfer request, the transfer source must be the peripheral receive data register. These conditions also apply to the other on-chip peripheral modules.

The number of the receive FIFO triggers can be set as a transfer request depending on an on-chip peripheral module. Data needs to be read after the DMA transfer is ended, because data may be left in the receive FIFO when the receive FIFO trigger condition is not satisfied.

**Table 30.2 DMA Transfer Requests**

| DMAC | Instance | DMA Request | Direction | DMARS | Note |
|---|---|---|---|---|---|
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_tsip0_res | Peri->DMAC | 0x10 | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_tsip0_in | DMAC->Peri | 0x11 | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_tsip0_out | Peri->DMAC | 0x12 | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_tsip1_res | Peri->DMAC | 0x14 | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_tsip1_in | DMAC->Peri | 0x15 | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_tsip1_out | Peri->DMAC | 0x16 | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_crc0_res | Peri->DMAC | 0x18 | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_crc0_in | DMAC->Peri | 0x19 | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_crc0_out | Peri->DMAC | 0x1A | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_crc0_res | Peri <-> DMAC | 0x1B | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_kcrc4_res | Peri->DMAC | 0x1C | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_kcrc4_in | DMAC->Peri | 0x1D | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_kcrc4_out | Peri->DMAC | 0x1E | |
| RT-DMAC (0ch t0 31ch) | wcrc0 | wcrc0_kcrc4_res | Peri <-> DMAC | 0x1F | |

| | | | | | |
|---|---|---|---|---|---|
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_tsip2_res | Peri->DMAC | 0x40 | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_tsip2_in | DMAC->Peri | 0x41 | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_tsip2_out | Peri->DMAC | 0x42 | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_tsip3_res | Peri->DMAC | 0x44 | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_tsip3_in | DMAC->Peri | 0x45 | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_tsip3_out | Peri->DMAC | 0x46 | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_crc1_res | Peri->DMAC | 0x48 | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_crc1_in | DMAC->Peri | 0x49 | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_crc1_out | Peri->DMAC | 0x4A | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_crc1_res | Peri <-> DMAC | 0x4B | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_kcrc5_res | Peri->DMAC | 0x4C | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_kcrc5_in | DMAC->Peri | 0x4D | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_kcrc5_out | Peri->DMAC | 0x4E | |
| RT-DMAC (0ch t0 31ch) | wcrc1 | wcrc1_kcrc5_res | Peri <-> DMAC | 0x4F | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_tsip4_res | Peri->DMAC | 0x80 | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_tsip4_in | DMAC->Peri | 0x81 | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_tsip4_out | Peri->DMAC | 0x82 | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_tsip5_res | Peri->DMAC | 0x84 | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_tsip5_in | DMAC->Peri | 0x85 | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_tsip5_out | Peri->DMAC | 0x86 | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_crc2_res | Peri->DMAC | 0x88 | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_crc2_in | DMAC->Peri | 0x89 | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_crc2_out | Peri->DMAC | 0x8A | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_crc2_res | Peri <-> DMAC | 0x8B | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_kcrc6_res | Peri->DMAC | 0x8C | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_kcrc6_in | DMAC->Peri | 0x8D | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_kcrc6_out | Peri->DMAC | 0x8E | |
| RT-DMAC (0ch t0 31ch) | wcrc2 | wcrc2_kcrc6_res | Peri <-> DMAC | 0x8F | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_tsip6_res | Peri->DMAC | 0xC0 | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_tsip6_in | DMAC->Peri | 0xC1 | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_tsip6_out | Peri->DMAC | 0xC2 | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_tsip7_res | Peri->DMAC | 0xC4 | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_tsip7_in | DMAC->Peri | 0xC5 | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_tsip7_out | Peri->DMAC | 0xC6 | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_crc3_res | Peri->DMAC | 0xC8 | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_crc3_in | DMAC->Peri | 0xC9 | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_crc3_out | Peri->DMAC | 0xCA | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_crc3_res | Peri <-> DMAC | 0xCB | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_kcrc7_res | Peri->DMAC | 0xCC | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_kcrc7_in | DMAC->Peri | 0xCD | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_kcrc7_out | Peri->DMAC | 0xCE | |
| RT-DMAC (0ch t0 31ch) | wcrc3 | wcrc3_kcrc7_res | Peri <-> DMAC | 0xCF | |
| SYS-DMAC (0ch t0 23ch) | tpu0 | tpu0_tx | Peri->DMAC | 0x17 | Support only Stepping-Motor Control Mode |
| SYS-DMAC (0ch t0 23ch) | hscif0 | hscif0_rx | Peri->DMAC | 0x30 | |
| SYS-DMAC (0ch t0 23ch) | hscif0 | hscif0_tx | DMAC->Peri | 0x31 | |
| SYS-DMAC (0ch t0 23ch) | hscif1 | hscif1_rx | Peri->DMAC | 0x32 | |
| SYS-DMAC (0ch t0 23ch) | hscif1 | hscif1_tx | DMAC->Peri | 0x33 | |
| SYS-DMAC (0ch t0 23ch) | hscif2 | hscif2_rx | Peri->DMAC | 0x34 | |
| SYS-DMAC (0ch t0 23ch) | hscif2 | hscif2_tx | DMAC->Peri | 0x35 | |
| SYS-DMAC (0ch t0 23ch) | hscif3 | hscif3_rx | Peri->DMAC | 0x36 | |
| SYS-DMAC (0ch t0 23ch) | hscif3 | hscif3_tx | DMAC->Peri | 0x37 | |
| SYS-DMAC (0ch t0 23ch) | msi0 | msi0_rx | Peri->DMAC | 0x40 | |
| SYS-DMAC (0ch t0 23ch) | msi0 | msi0_tx | DMAC->Peri | 0x41 | |
| SYS-DMAC (0ch t0 23ch) | msi1 | msi1_rx | Peri->DMAC | 0x42 | |
| SYS-DMAC (0ch t0 23ch) | msi1 | msi1_tx | DMAC->Peri | 0x43 | |
| SYS-DMAC (0ch t0 23ch) | msi2 | msi2_rx | Peri->DMAC | 0x44 | |
| SYS-DMAC (0ch t0 23ch) | msi2 | msi2_tx | DMAC->Peri | 0x45 | |
| SYS-DMAC (0ch t0 23ch) | msi3 | msi3_rx | Peri->DMAC | 0x46 | |
| SYS-DMAC (0ch t0 23ch) | msi3 | msi3_tx | DMAC->Peri | 0x47 | |
| SYS-DMAC (0ch t0 23ch) | msi4 | msi4_rx | Peri->DMAC | 0x48 | |
| SYS-DMAC (0ch t0 23ch) | msi4 | msi4_tx | DMAC->Peri | 0x49 | |
| SYS-DMAC (0ch t0 23ch) | msi5 | msi5_rx | Peri->DMAC | 0x4A | |
| SYS-DMAC (0ch t0 23ch) | msi5 | msi5_tx | DMAC->Peri | 0x4B | |
| SYS-DMAC (0ch t0 23ch) | scif0 | scif0_rx | Peri->DMAC | 0x50 | |

| | | | | | |
|---|---|---|---|---|---|
| SYS-DMAC (0ch t0 23ch) | scif0 | scif0_tx | DMAC->Peri | 0x51 | |
| SYS-DMAC (0ch t0 23ch) | scif1 | scif1_rx | Peri->DMAC | 0x52 | |
| SYS-DMAC (0ch t0 23ch) | scif1 | scif1_tx | DMAC->Peri | 0x53 | |
| SYS-DMAC (0ch t0 23ch) | scif3 | scif3_rx | Peri->DMAC | 0x56 | |
| SYS-DMAC (0ch t0 23ch) | scif3 | scif3_tx | DMAC->Peri | 0x57 | |
| SYS-DMAC (0ch t0 23ch) | scif4 | scif4_rx | Peri->DMAC | 0x58 | |
| SYS-DMAC (0ch t0 23ch) | scif4 | scif4_tx | DMAC->Peri | 0x59 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_cf0_rx | Peri->DMAC | 0x70 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_cf1_rx | Peri->DMAC | 0x72 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_cf2_rx | Peri->DMAC | 0x74 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_cf3_rx | Peri->DMAC | 0x76 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_rf0_rx | Peri->DMAC | 0x80 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_rf1_rx | Peri->DMAC | 0x82 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_rf2_rx | Peri->DMAC | 0x84 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_rf3_rx | Peri->DMAC | 0x86 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_rf4_rx | Peri->DMAC | 0x88 | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_rf5_rx | Peri->DMAC | 0x8A | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_rf6_rx | Peri->DMAC | 0x8C | |
| SYS-DMAC (0ch t0 23ch) | canfd | canfd_rf7_rx | Peri->DMAC | 0x8E | |
| SYS-DMAC (0ch t0 23ch) | i2c0 | i2c0_mst_rx | Peri->DMAC | 0x90 | |
| SYS-DMAC (0ch t0 23ch) | i2c0 | i2c0_mst_tx | DMAC->Peri | 0x91 | |
| SYS-DMAC (0ch t0 23ch) | i2c0 | i2c0_slv_rx | Peri->DMAC | 0xA0 | |
| SYS-DMAC (0ch t0 23ch) | i2c0 | i2c0_slv_tx | DMAC->Peri | 0xA1 | |
| SYS-DMAC (0ch t0 23ch) | i2c1 | i2c1_mst_rx | Peri->DMAC | 0x92 | |
| SYS-DMAC (0ch t0 23ch) | i2c1 | i2c1_mst_tx | DMAC->Peri | 0x93 | |
| SYS-DMAC (0ch t0 23ch) | i2c1 | i2c1_slv_rx | Peri->DMAC | 0xA2 | |
| SYS-DMAC (0ch t0 23ch) | i2c1 | i2c1_slv_tx | DMAC->Peri | 0xA3 | |
| SYS-DMAC (0ch t0 23ch) | i2c2 | i2c2_mst_rx | Peri->DMAC | 0x94 | |
| SYS-DMAC (0ch t0 23ch) | i2c2 | i2c2_mst_tx | DMAC->Peri | 0x95 | |
| SYS-DMAC (0ch t0 23ch) | i2c2 | i2c2_slv_rx | Peri->DMAC | 0xA4 | |
| SYS-DMAC (0ch t0 23ch) | i2c2 | i2c2_slv_tx | DMAC->Peri | 0xA5 | |
| SYS-DMAC (0ch t0 23ch) | i2c3 | i2c3_mst_rx | Peri->DMAC | 0x96 | |
| SYS-DMAC (0ch t0 23ch) | i2c3 | i2c3_mst_tx | DMAC->Peri | 0x97 | |
| SYS-DMAC (0ch t0 23ch) | i2c3 | i2c3_slv_rx | Peri->DMAC | 0xA6 | |
| SYS-DMAC (0ch t0 23ch) | i2c3 | i2c3_slv_tx | DMAC->Peri | 0xA7 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid00_p | Peri->DMAC | 0x6E | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid00_p | DMAC->Peri | 0x6F | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid01_p | Peri->DMAC | 0x6C | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid01_p | DMAC->Peri | 0x6D | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid02_p | Peri->DMAC | 0x6A | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid02_p | DMAC->Peri | 0x6B | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid03_p | Peri->DMAC | 0x68 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid03_p | DMAC->Peri | 0x69 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid04_p | Peri->DMAC | 0x66 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid04_p | DMAC->Peri | 0x67 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid05_p | Peri->DMAC | 0x64 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid05_p | DMAC->Peri | 0x65 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid06_p | Peri->DMAC | 0x62 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid06_p | DMAC->Peri | 0x63 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid07_p | Peri->DMAC | 0x60 | |
| SYS-DMAC (0ch t0 23ch) | ssi | ssi_ssid07_p | DMAC->Peri | 0x61 | |

Table 30.3 Data Length of DMA Transfer for Each of the On-Chip Peripheral Modules

| Module* | 1 Byte | 2 Bytes | 4 Bytes | 8 Bytes | 16 Bytes | 32 Bytes | 64 Bytes |
|---|---|---|---|---|---|---|---|
| TPU | | √ | | | | | |
| HSCIF0/1/2/3 | √ | | | | | | |
| MSIOF0/1/2/3/4/5 | √ | √ | √ | | | | |
| SCIF0/1/3/4 | √ | | | | | | |
| CANFD | | | | | | √ | |
| I2C0/1/2/3 | √ | | | | | | |
| SSI | | | √ | | | | |
| CRC | | | | √ | √ | √ | √ |

### 30.3.2 Channel Priority

When the RT-DMAC receives simultaneous transfer requests on two or more channels, it transfers data according to a predetermined priority. Two modes (fixed mode and round-robin mode) are selected by the PR[1:0] bits in DMOR.

**(a) Fixed Mode**

In this mode, the priority levels among the channels remain fixed.

CH0 > CH1 > ... > CH14 > CH15.
CH0 > CH1 > ... > CH6> CH7.(SYS-DMAC 2 only)

**(b) Round-Robin Mode**

In round-robin mode each time data of one transfer unit (byte, word, longword, 8-byte or 16-byte unit) is transferred on one channel, the priority is rotated. The channel on which the transfer was just finished rotates to the bottom of the priority. The priority of round-robin mode is CH0 > CH1 > ... > CH14 > CH15 immediately after reset.

### 30.3.3 Slow Speed Mode

DMAC has 2 kinds of Rate control (Slow Speed mode) function.

**(a) SLM Bit in CHCRB**

In the slow speed mode, DMA transfer is operated once in multiple clock cycle specified by SLM bit in CHCRB described at DMA channel control register B for RT-DMAC [m] in Channel[n](RDMCHCRB_[m]_[n])(m=0-1)(n=0-15), for SYS-DMAC 1 in Channel[n](SDMCHCRB_1_[n])(n=0-15) and for SYS-DMAC 2 in Channel[n](SDMCHCRB_2_[n])(n=0-7) in section 30.2 Register Description. It is possible whether DMA transfer is operated in slow speed mode or not for each DMA channel respectively. When some transfer channel in slow speed mode is kept waiting, it is possible for the other transfer channel in slow speed mode to operate DMA transfer during the waiting time.

**(b) CNT Bit in RATE_RD and RATE_WR**

In the rate access control mode, the performance of DMA transfer will decrease depending on value of the rate access counter which is set up by CNT bits in RATE_RD register or RATE_WR register. When to be enable rate access bit which is set up to 1 by EN bit in RATE_RD register or RATE_WR register, the rate access counter starts to count. Whenever value of CNT bits matches with value of counter register, then enable control bit will be toggled after 3 clock cycles.

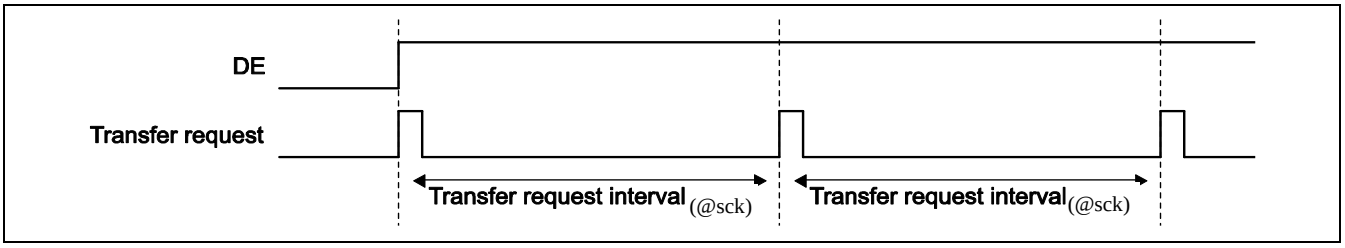



Figure 30.2 Slow Speed Mode

**Table 30.4 Operation specification of Slow Speed Mode**

| | Operation for Slow mode | Support interface | Setting possible range (Transfer request interval) |
|---|---|---|---|
| SLB in CHCRB | Read and Write | Memory, Bus and Peripheral | 0000: Normal mode<br>1000: 256 clock cycles<br>1001: 512 clock cycles<br>1010: 1024 clock cycles<br>:<br>1111: 32768 clock cycles |
| RATE_RD | Read | Bus | 3-255: valid setting<br>(CNT + 1) * 2 + 3 (clock cycle) |
| RATE_WR | Write | | |

### 30.3.4 Descriptor

Descriptor is enabled by setting B’01, B'10 or B'11 to DPM[1:0] bit in CHCR. When TCR decrease to 0, and DMA transfer is stopped, Descriptor load next configuration from Descriptor and set to the register of the channel. Maximum number of steps is 256 when single channel is operated. Descriptor maximum step can extend to 4096 for external memory.

Set below in order to use Descriptor.

- Set base address of Descriptor which configurations of DMA transfer are written to DPBASE.
- Set 1 to DRST bit in CHCRB in order to reset Descriptor.
- Set the number of steps to DCNT bit in CHCRB. For the maximum step can be 4096, the DCNT bit in DPCR is used to set the upper bits for the number of steps. For example, if using 256 as the number of steps, 0xFF need to set to DCNT in CHCRB. If using 4096 as the number of steps, not only 0xFF need to set to DCNT in CHCRB, but also 0xF need to set to DCNT in DPCR.

Descriptor is shared by all DMA channels. Therefore, it is necessary to eliminate crossover of address spaces which are used by each DMA channel. And, arrange each step of Descriptor in 16-byte boundary.

There are two different methods to start Descriptor.

- Set configuration of first DMA transfer to SAR, DAR, TCR register and configurations of following DMA transfer to Descriptor, then set 0 to DPB bit in CHCR in order to start Descriptor. In this case, first DMA transfer is operated according to the configuration of SAR, DAR, TCR register and configuration of following DMA transfer is loaded when preceding DMA transfer ended. Then configurations of next DMA transfer are loaded from Descriptor and

second DMA transfer start. But if Descriptor is operated as repeat mode, configuration of SAR, DAR, TCR register are not loaded and Descriptor repeatedly load configuration from the head of Descriptor.

- Write configurations of DMA transfer to Descriptor, then write 1 to DPB bit in CHCR in order to start Descriptor. In this case, DMA transfer start from head of descriptor.

There are three different operating mode of Descriptor. Operating mode is specified by DPM[1:0] bit in CHCR register.

Details of each operating mode are described below.

**(1) Setting Updating Flow By Descriptor**

The setting updating flow by Descriptor can be specified by the RPT bit of CHCR.

If the setting updating flow by Descriptor is completed, the DPTR bit of CHCRB is incremented 1. When DPTR value equals to DCNT value, DPTR value returns to 0. DPTR also can be extends to 4096 by using DPTR bits in DPCR.

This flow is automatically processed with hardware.

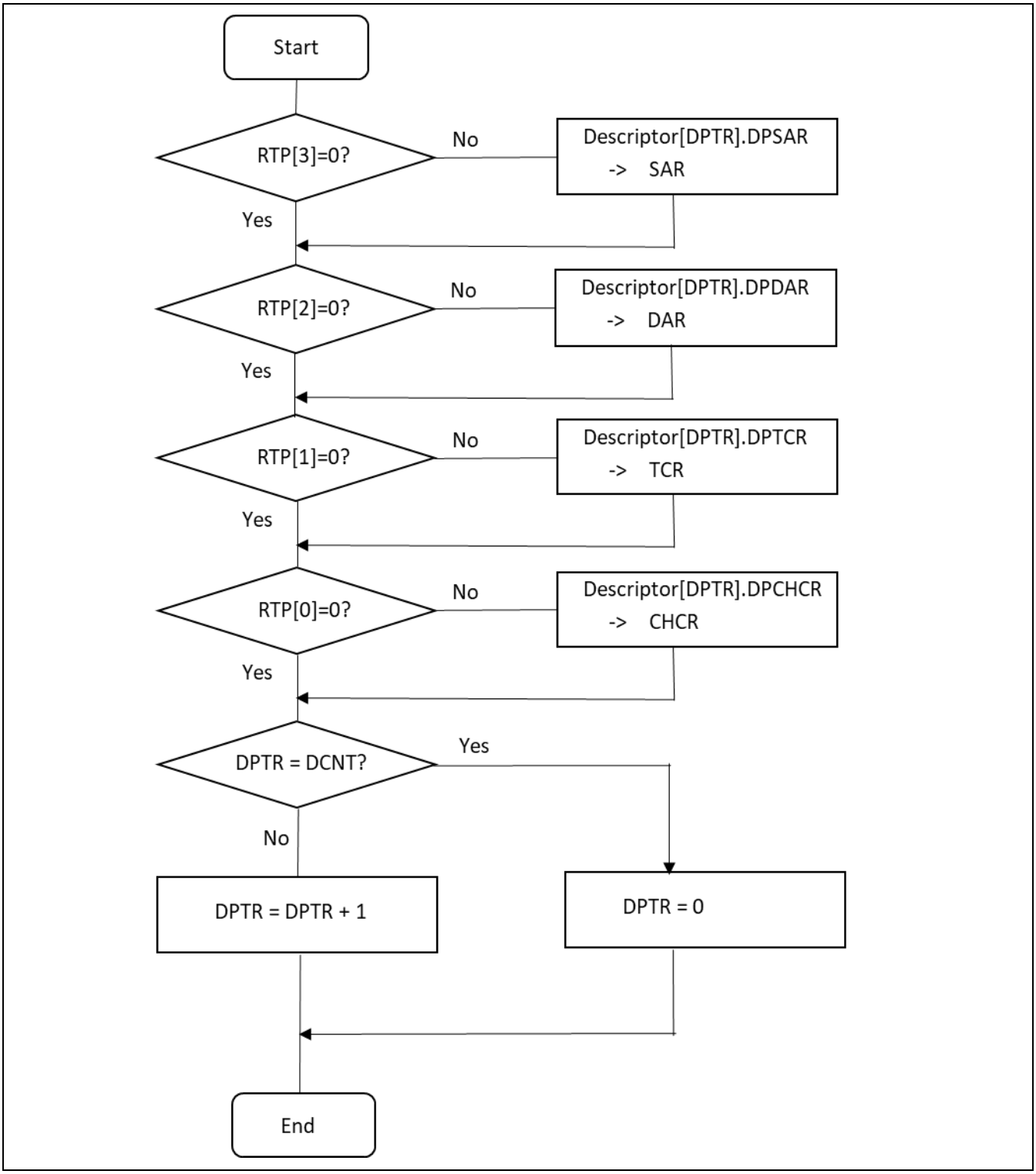



Figure 30.3 Setting Updating Flow by Descriptor

**(2) Descriptor Operational Mode 1**

The descriptor operational mode 1 (descriptor normal operation mode) sets 1 to TE bit of CHCR after the descriptor number-of-step transmission set as DCNT of a CHCRB register, and a DMA transfer ends. To select this operational mode, set B'01 to DPM of CHCR. Continuously transferring by a descriptor can be performed by setting to this operational mode.

When DSIE is set as 1, a DSE interrupt occurs for every termination of one step of descriptor. If the next DSE interrupt occurs without performing DSE interrupt processing, the following descriptor will not be read but a DMA transfer will stop. If a DSE bit is cleared, only interruption generated first will be cleared and the following descriptor will be read.

These are examples when the number of steps of Descriptor is set as four steps.

When the Descriptor operational mode 1 and TE/DSE is enabled, the example of transfer operation is shown in Figure 30.4.

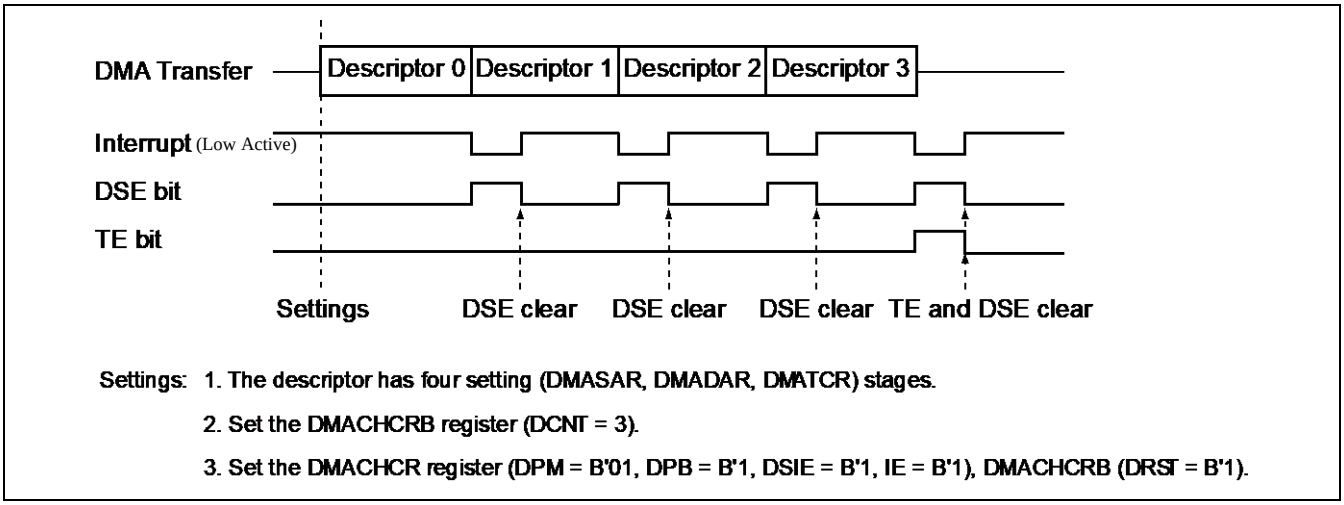



Figure 30.4 Descriptor Operational Mode 1 (Example 1)

When the Descriptor operational mode 1 and TE/DSE is enabled, if DSE interruption handling is delayed, the example of transfer operation is shown in Figure 30.5.

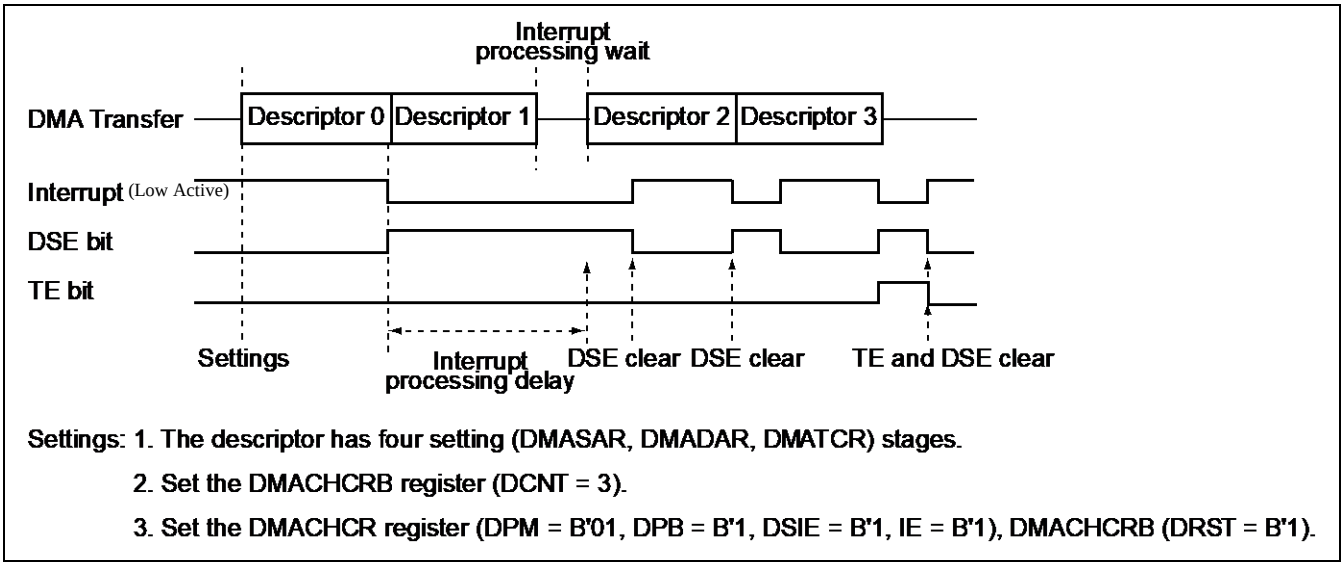



Figure 30.5 Descriptor Operational Mode 1 (Example 2)

When the Descriptor operational mode 1 and TE is enabled, the example of transfer operation is shown in Figure 30.6.

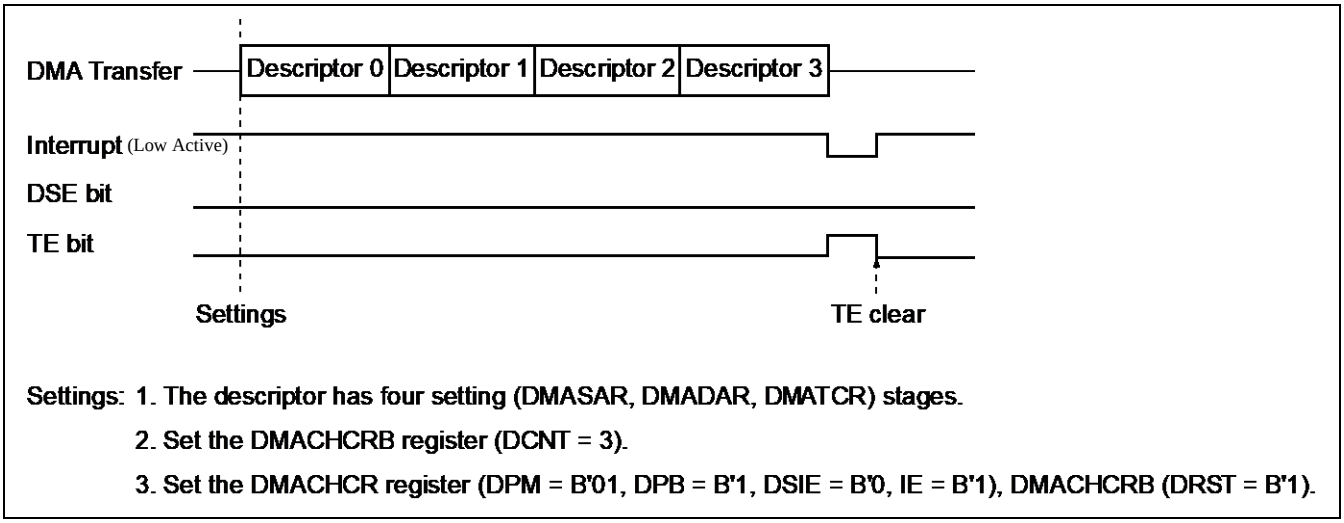



**Figure 30.6 Descriptor Operational Mode 1 (Example 3)**

### (3) Descriptor Operational Mode 2

The descriptor operational mode 2 (repeat mode) sets 1 to TE bit of CHCR after the descriptor number-of-step transmission set as DCNT of CHCRB and repeats a DMA transfer again from the head of a descriptor. To select this operational mode, set B'10 to DPM of CHCR. Continuously transferring by a descriptor can be performed by setting to this operational mode.

When DSIE is set as one, a DSE interrupt occurs for every termination of one step of descriptor. If the next DSE interrupt occurs without performing DSE interrupt processing, the following descriptor will not be read but a DMA transfer will stop. If a DSE bit is cleared, only interruption generated first will be cleared and the following descriptor will be read.

When DSIE is set as 0, 1 is set to TE bit of CHCR after all the descriptor number-of-step transmission, and TE interrupt occurs. If the next TE interrupt occurs without performing TE interrupt processing, the following descriptor will not be read but a DMA transfer will stop. If TE bit is cleared, only interruption generated first will be cleared and the following descriptor will be read.

Change into the descriptor operational mode 1 in TE interrupt processing to end the descriptor operational mode 2.

By changing into the descriptor operational mode 1, a DMA transfer is completed to the following TE interruption occurrence time.

These are examples when the number of steps of Descriptor is set as four steps.

When the Descriptor operational mode 2 and TE/DSE is enabled, the example of transfer operation is shown in Figure 30.7.

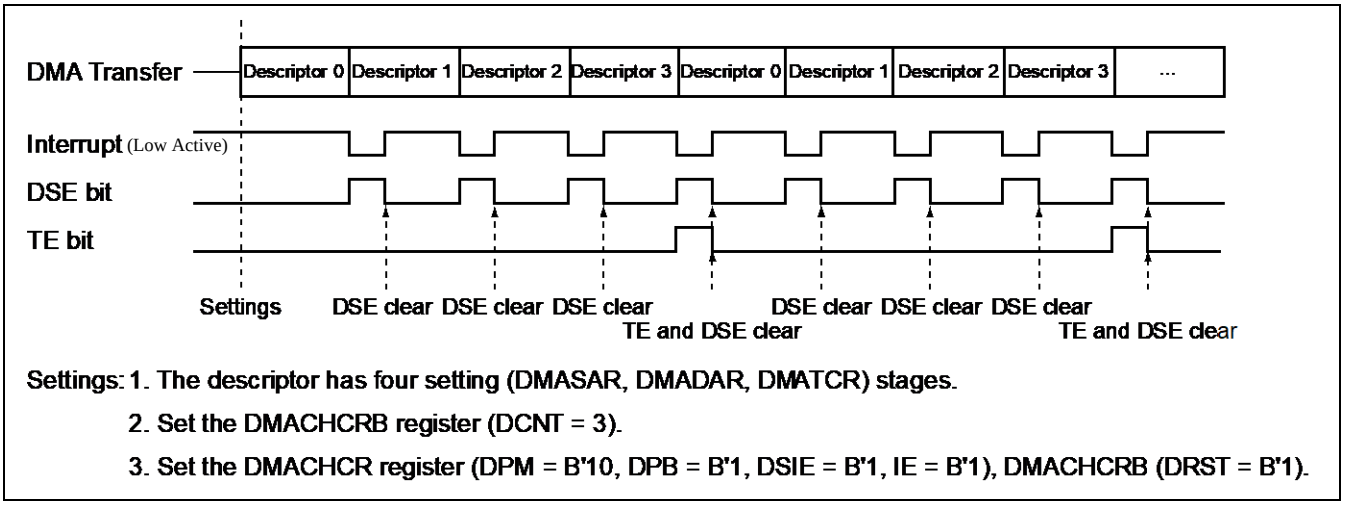



**Figure 30.7 Descriptor Operational Mode 2 (Example 1)**

When the Descriptor operational mode 2 and TE/DSE is enabled, if DSE interruption handling is delayed, the example of transfer operation is shown in Figure 30.8.

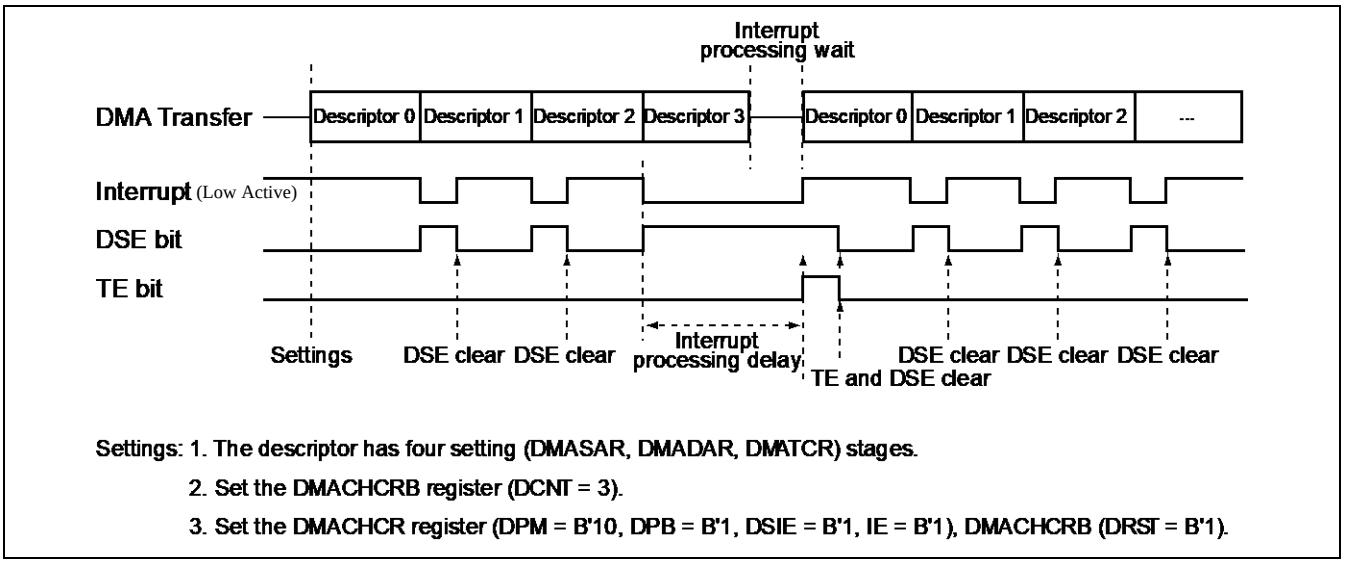



**Figure 30.8 Descriptor Operational Mode 2 (Example 2)**

When the Descriptor operational mode 2 and TE is enabled, the example of transfer operation is shown in Figure 30.9.

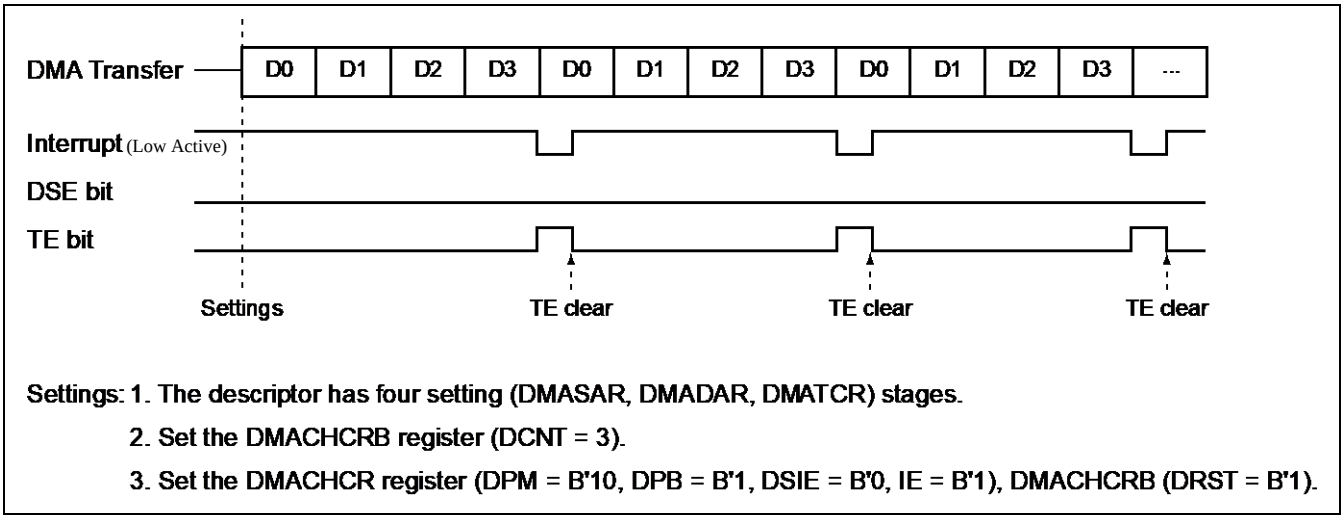



**Figure 30.9 Descriptor Operational Mode 2 (Example 3)**

When the Descriptor operational mode 2 and TE is enabled, if TE interruption handling is delayed, the example of transfer operation is shown in Figure 30.10.

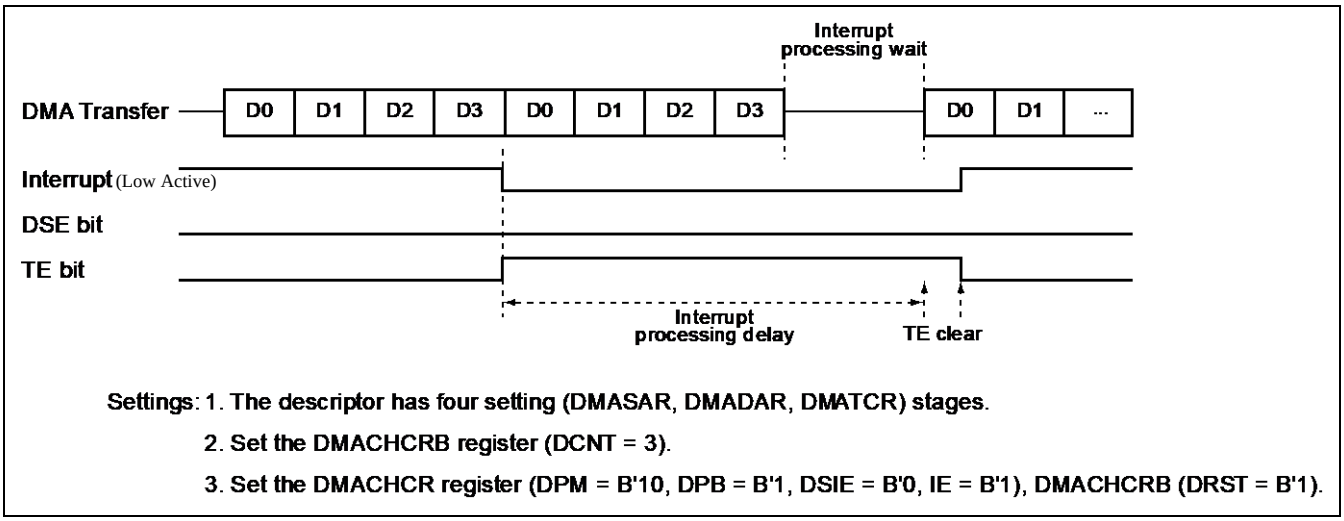



**Figure 30.10 Descriptor Operational Mode 2 (Example 4)**

### (4) Descriptor Operational Mode 3

Change to infinite repeat mode and a descriptor read-out interrupt mode is possible for the descriptor operational mode 3 by the setting existence of DSIE and IE bit.

Descriptor infinite repeat mode performs the DMA transfer repeatedly set as the descriptor after completing transmission of the number of steps set as DCNT of CHCRB until it stops a DMA transfer repeatedly from the head of a descriptor again.

A DSE interrupt generates a descriptor read-out interrupt mode after reading the descriptor set as the DIPT register of DPCR. A descriptor can be started from a continuation by setting up to the following number of sections in DSE interrupt processing and clearing DSE. Similarly, it repeats from the head of a descriptor again after termination to the number of sections set as DCNT of CHCRB.

After a DMA transfer is completed during DSE interrupt processing, a DMA transfer is stopped without reading the following descriptor. When DSE interruption is cleared, the following descriptor is read, and transmission is resumed.

To select descriptor infinite repeat mode of the descriptor operational mode 3, set B'11 to the DPM bits and 0 to the DSIE and IE bits of CHCR register.

This operational mode is repeated from the head of a descriptor after the descriptor number-of-step transmission set as DCNT of CHCRB.

Perform a DMA transfer interruption process to stop this operational mode.

When performing the descriptor read-out interrupt mode of the descriptor operational mode 3, the number of sections of the descriptor which set the descriptor CHCR range to the DCNT register of CHCRB and was set as the DIPT bit of a DPCR register is set up. Set B'11 to the DPM bit of a CHCR register, set 1 to a DSIE bit, and start a DMA transfer.

A descriptor can be continuously started by a DSE interrupt occurring, setting up the following descriptor in interrupt treatment, and clearing interruption.

A descriptor number of sections is an example at the time of setting it as four steps.

When the Descriptor operational mode 3 and the example of infinite repeat mode transfer operation, the example of transfer operation is shown in Figure 30.11.

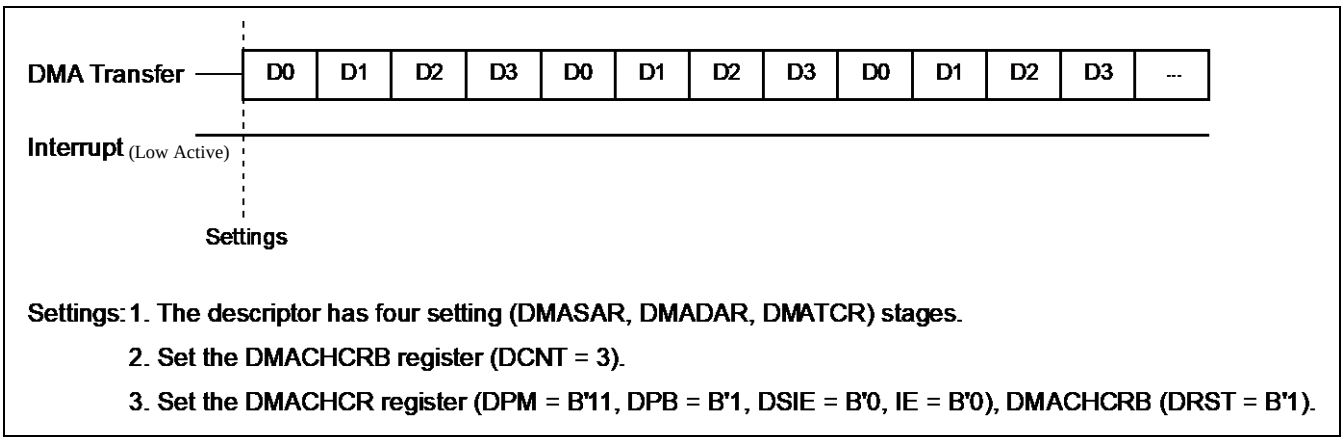
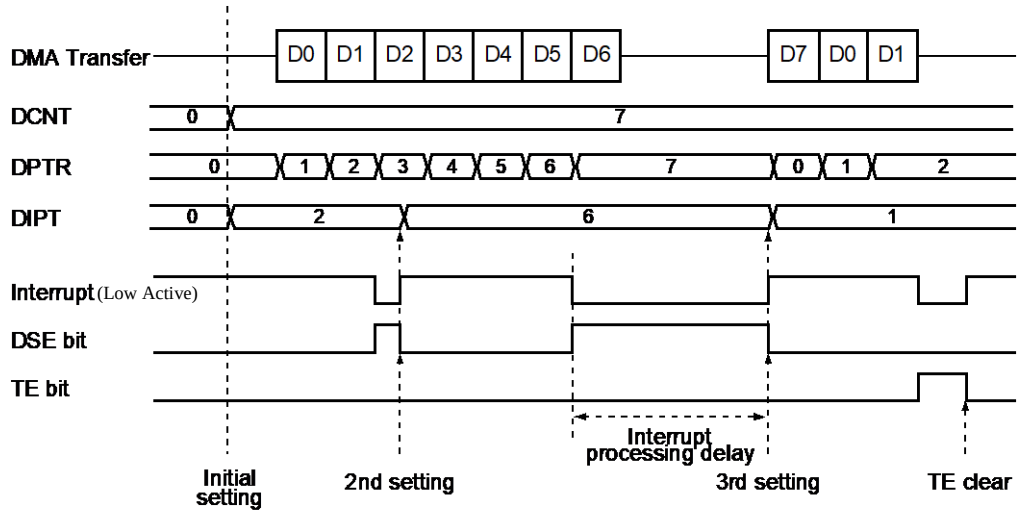



**Figure 30.11 Descriptor Operational Mode 3 (Infinite Repeat)**

When the descriptor operational mode 3 and the example of descriptor read-out interrupt mode, the example of transfer operation is shown in Figure 30.12.




Initial setting:
1. Set the DMACHCRB register (DCNT = 7).
2. The descriptor has three setting (DMASAR, DMADAR, DMATCR) stages.
3. Set the DMADPCR register (DIPT = 2).
4. Set the DMACHCR register (DPM = B'11, DPB = B'1, DSIE = B'1, IE = B'0), DMACHCRB (DRST = B'1)

2nd setting:
1. The descriptor has four setting (DMASAR, DMADAR, DMATCR) stages.
2. Set the DMADPCR register (DIPT = 6).
(3 stages + 4 stages -1) mod 8 = 6
3. Clear DSE.

3rd setting: (Transfer end)
1. The descriptor has three setting (DMASAR, DMADAR, DMATCR) stages.
2. Set the DMADPCR register (DIPT =1).
(3 stages + 4 stages + 3 stages -1) mod 8 = 1
3. Clear the DSE bit and set the DMACHCR register (DPM = B'11, DPB = B'0, DSIE = B'0, IE = B'1).

**Figure 30.12 Descriptor Operational Mode 3 (Descriptor Read-Out Interrupt Mode)**

### (5) The Double Buffer Using of Descriptor

To use Descriptor for double buffering, set the number of Descriptor steps as two steps, and write each configuration of buffer in a descriptor. Then start it as the descriptor operational mode 2. Interruption uses only DSE interruption.

To end the use of double buffering, disable the descriptor operational mode. It stops after completing the transmission under execution.

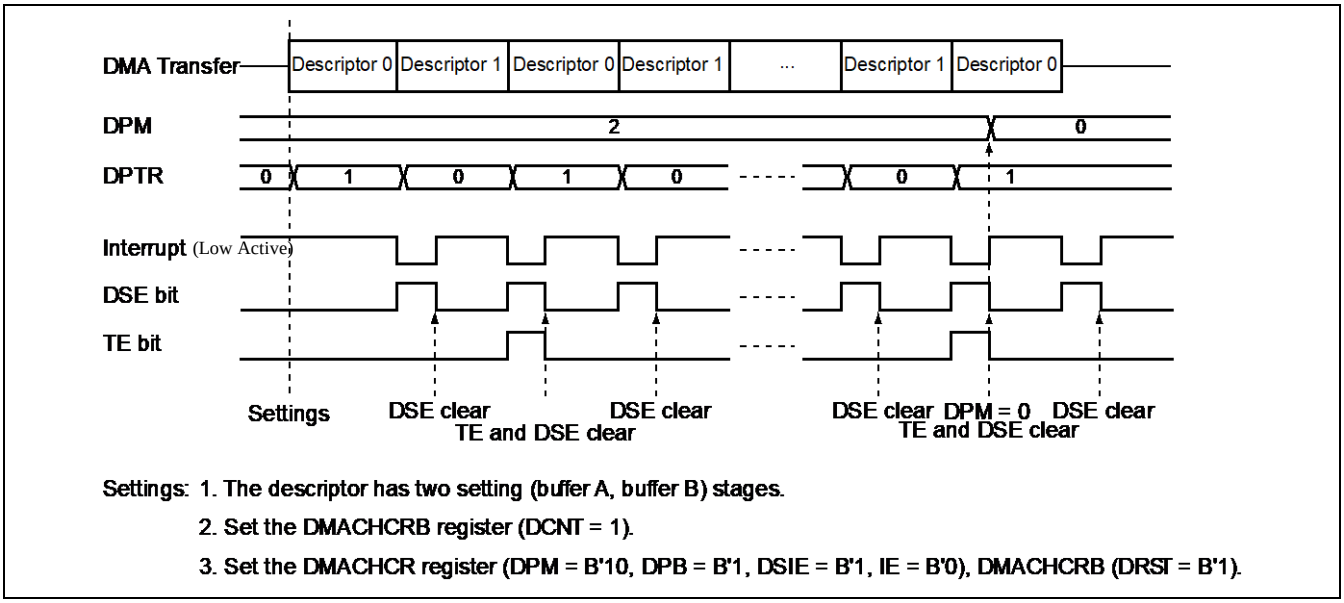
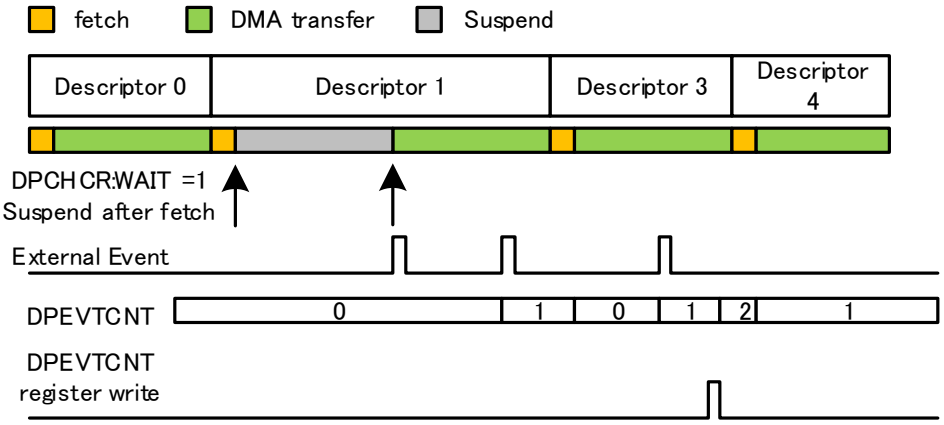



**Figure 30.13 The Double Buffer by a Descriptor**

### (6) Descriptor Suspend and Resume

To adapt the peripheral data collection speed, a descriptor processing suspend and resume scheme is introduced in to S4. Descriptor processing in every channel can suspend after its DPCHCR fetched into DMAC and the WAIT bit in DPCHCR is set to 1, and the suspending descriptor processing can resume by the selected external event trigger edge or by register DPEVTCNT write.




Notice: if user uses DPCHCR updating, RPT[0] in CHCR must be set.

The write access (regardless of value) of DPEVTCNT will trigger a pseudo-event.

The DPEVTCNT records how many external events or pseudo-events happen, if events come, the DPEVTCNT register bit EVTCNT will increment. If the DPEVTCN.EVTCNT is 0x7F and new event comes, overflow happens and the DPEVTCNT register bit OV set, DMISTA register bit OV [15:0] will set too corresponding bit that behalf current channel. The interrupt of AE will assert as well. When overflow happens, the event cannot resume the suspending descriptor processing any more until overflow is reset and new event come.

When descriptor processing is suspending, if the DPEVTCNT.EVTCNT is not 0 and not overflow, the descriptor processing will resume. If the DPEVTCNT.EVTCNT is 0, the suspending keeps until new event comes. If the DPEVTCNT.OV is 1, the suspending keeps until OV is reset by DPEVTCR.RST or CHCLR.CLR set.

When descriptor processing is suspending, the WAIT status can be checked by reading CHCR register bit WAIT.

and if the descriptor processing resumes, the DPEVTCNT will decrement.

Typical operation that wait event:

1. Prepare descriptors in the external memory, mark 1 to the WAIT bit of DPCHCR in corresponding descriptor which want to be suspend.
2. Configure DMAC channel that will use the above descriptors, set DPEVTCR register bit EXEN be 1 to enable external event input and set DPEVTCR register bit EVTID [4:0] for current channel to select which external event to use.
3. Start descriptor processing. DMAC will stop after it fetched the descriptor and found that DPCHCR.WAIT is set.
4. The descriptor processing suspends and is waiting event comes
5. The selected external event comes, and DMAC detect the rising edge of this event, DPEVTCNT.EVTCNT increments.
6. Suspending descriptor processing resumes and start its DMA transfer, DPEVTCNT.EVTCNT decrements.

The step 5 above can also be replaced by “write DPEVTCNT, DPEVTCNT.EVTCNT increments”

Typical operation that event outstanding:

1. Prepare descriptors in the external memory, mark 1 to the WAIT bit of DPCHCR in corresponding descriptor which want to be suspend.
2. Configure DMAC channel that will use the above descriptors, set DPEVTCR register bit EXEN be 1 to enable external event input and set DPEVTCR register bit EVTID [4:0] for current channel to select which external event to use.
3. Start descriptor processing. DMAC will stop after it fetched the descriptor and found that DPCHCR.WAIT is set.
4. Before descriptor processing suspends, the selected events come multiple times and DPEVTCNT.EVTCNT increment by each rising edge of the event.
5. The descriptor processing suspends and resumes immediately, DPEVTCNT.EVTCNT decrements.

The step 4 above can also be replaced by “write DPEVTCNT multiple times, DPEVTCNT.EVTCNT increments”

Typical operation that event overflow:

1. Prepare descriptors in the external memory, mark 1 to the WAIT bit of DPCHCR in corresponding descriptor which want to be suspend.
2. Configure DMAC channel that will use the above descriptors, set DPEVTCR register bit EXEN be 1 to enable external event input and set DPEVTCR register bit EVTID [4:0] for current channel to select which external event to use.
3. Start descriptor processing. DMAC will stop after it fetched the descriptor and found that DPCHCR.WAIT is set.

4. Before descriptor processing suspends, the selected events come too many times and DPEVTCNT.EVTCNT increment and finally overflow. DPEVTCNT.OV bit set and AE interrupt assert.
5. The descriptor processing suspends.
6. ECM detects the interrupt and interrupt processing master checks DMISTA.OV[15:0] to find OV in which channel happens.
7. Interrupt processing master set DPEVTCR.RST in corresponding channel to reset DPEVTCNT register, or just set CHCLR.CLR to reset the channel itself.
8. New event comes and resume the suspending descriptor processing.

### 30.3.5 Transmission Flow

Write transfer condition setup in DMA source address register (SAR), DMA destination address register (DAR), DMA transfer count register (TCR), DMA Channel control register (CHCR), DMA operation register (DMOR) and a DMA extended resource selector (DMRS), and DMAC transmits data in following order.

- A transfer request occurs and confirm whether to be a transfer permission state for (DE = 1, DME = 1, TE = 0, DSE = 0, CAE = 0). In the case of the auto request mode, start transmission automatically.
- Check whether updating by Descriptor is required.
  When DPB of CHCR is set to 1 or Descriptor is enabled, if TCR is set to 0, updating by Descriptor is operated at the state where transmission is completed.
  About updating by Descriptor, refer to a section 30.3.4, Descriptor.
- Check whether the address translation by IPMMU is required.
  When DE of CHCR is enabled, if updating of transfer setting by Descriptor is operated and address exceeds the effective size of address translation, address translation by IPMMU is operated.
- If transfer request is generated, one transfer unit (specified by TS [3:0]) will be transmitted. TCR will be decremented 1 at every termination of DMA transfer.
- When transmission of the number of times having been specified is finished (the value of TCR is set to 0), transmission is ended normally. At this time if IE bit of CHCR is set 1, TE interruption will be generated to CPU. When Descriptor is enabled, processing changes with each Descriptor operational modes. For details, see section 30.3.4, Descriptor.
- Transmission is terminated when the address error occurs in DMA transfer. In addition, transmission is terminated even if DE bit of CHCR or the DME bit of DMOR is set to 0.

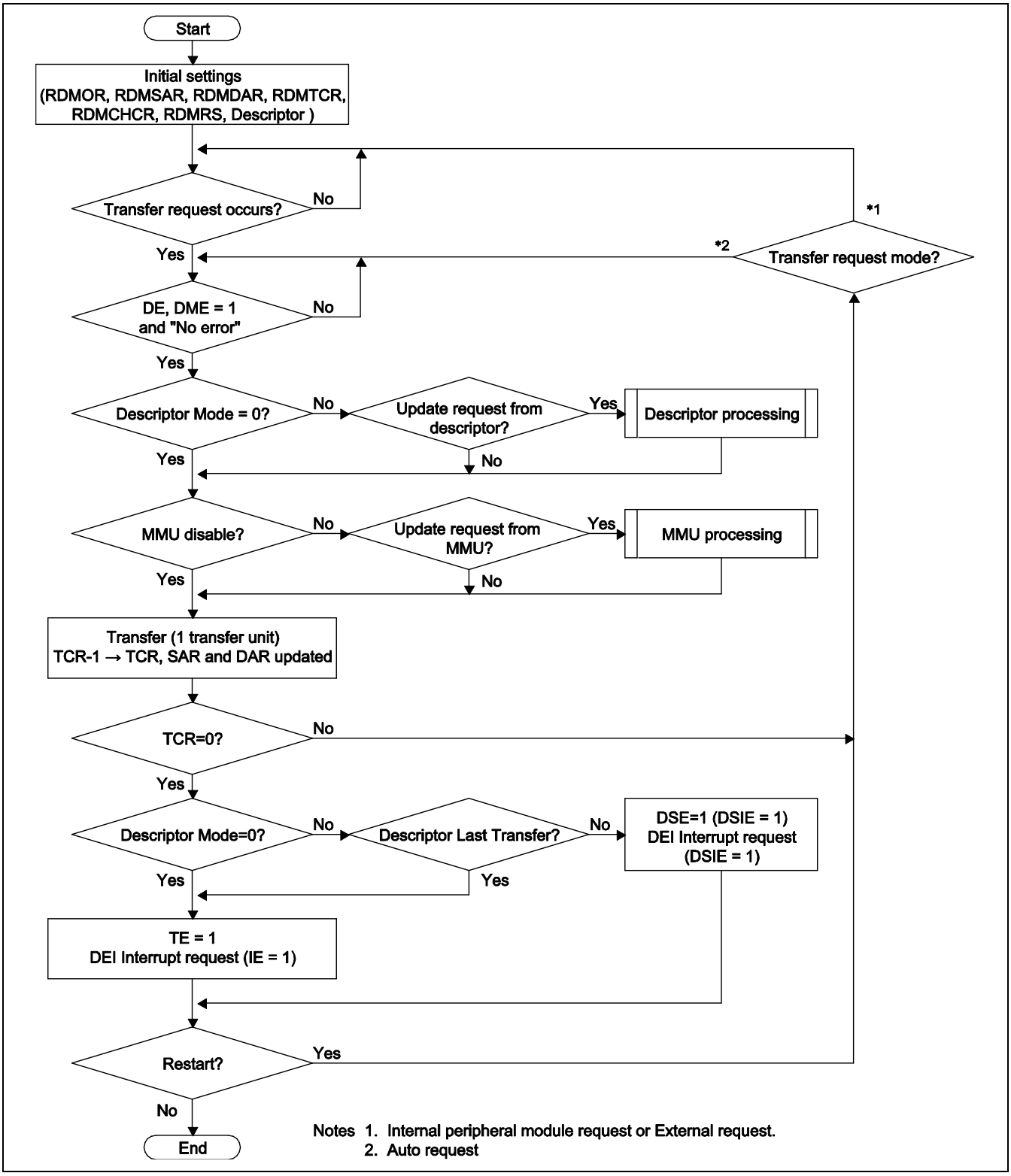



**Figure 30.14 Transmission Flow**

### 30.3.6 Total Size Transmission

Total size transmission can be set to TSR by byte size regardless of DMA transfer size (CHCR.TS). Thereby, arbitrary sizes can be transmitted with big DMA transfer size.

In order to use this function, use the address for Total size transmission of SAR, DAR, TSR, TSRB and CHCR (Refer to section 30.2 Register Description for the register for Total size transmission).

Total size is set in the TCR (TSR) of Descriptor and 1 is set in CHCRB.DTS at the time of Total size transmission use.

### 30.3.7 DREQ Outstanding and DMA Outstanding

To adapt the peripheral w/ high throughput and extract the maximum performance of RT-DMAC, DREQ outstanding and DMA outstanding option feature can be used in such case.

#### 30.3.7.1 DREQ Outstanding

Normally, DMAC channel receives a DREQ (DMA transfer REQuest) from peripheral, then start DMA transfer. And after DMA transfer finish, DACK (DMA transfer ACKnowledge) will send back to the peripheral. So the peripheral must wait DACK and then assert new DREQ.

User can set CHCRB register bit DREQOSEN of current channel to enable DREQ outstanding feature.

In DREQ outstanding enable mode, current DMAC channel receives DREQ from peripheral, and store it to built-in DREQ outstanding buffer, then return the DACK immediately to the peripheral, and if the peripheral has the capability to send the next DREQ, DMAC will store it again and return the DACK, and so on until the peripheral has no capability to send new DREQs. The DREQ outstanding buffer has 8bit depth and can store up to 255 DREQ, but it supposes that no peripheral has capability to send such many DREQ, so no overflow flag for DREQ outstanding buffer.

Typical operation flow:

1. When configure DMAC channel, set CHCRB register bit DREQOSEN to enable DREQ outstanding feature. (don’t set this register when in DMA transfer. The action will not be guaranteed)
2. Start DMA transfer. The DREQ and DACK handshake will automatically proceed.
3. Current DREQ outstanding status (the number of current DREQ outstanding) can be checked by reading DREQOS register.
4. After all DMA transfer finish such like TE bit set, the peripheral can also check the current DREQ outstanding status if there are DREQ remains or not. If the peripheral concerns that the remaining DREQ outstanding will affect the next DMA transfer, it can be cleared by clearing CHCRB.DREQOSEN and setting CHCRB.DREQOSEN again, or just by setting CHCLR.CLR to reset the channel itself.

#### 30.3.7.2 DMA Outstanding

Normally, DMA outstanding capability of each channel is determined by the maximum space that each channel can use in buffer RAM. Buffer RAM’s space(total 8KB) divides equally to 16 and correspond to each channel. So each channel can use maximum 512Bytes. And each channel can run maximum 256Byte for every burst transfer, so the outstanding of each channel is 2.

To improve the performance for DMA transfer, DMAC adds a partial outstanding improve method to get better performance.

Register BUFCR.MBU and BUFCR.ULB is used to set maximum burst size and maximum space usage. So in normal, MBU is max 256 and ULB is max 512.

User can set BUFMODE register bit BUFMODE to enable DMA Partial outstanding feature.

In Partial Outstanding mode, Channel 8~15 maximum space is decreased and the saving space is assigned to Channel 0~3.

Normal

| CH | Max Burst | Max Space | Outstanding |
|---|---|---|---|
| CH0~15 | 256B | 512B | 2 |

Partial Outstanding

| CH | Max Burst | Max Space | Outstanding |
|---|---|---|---|
| CH0~3 | 256B | 1024B | 4 |
| CH4~7 | 256B | 512B | 2 |
| CH8~15 | 128B | 256B | 2 |

To set Max space for 1024B, the BUFCR.ULB extend to 11bit and change its initial value to 1024. And adds guard logic to prevent wrong setting in BUFCR.ULB and BUFCR.MBU.

Typical operation flow:

1. When configure DMAC, set BUFMODE register bit BUFMODE to enable DMA Partial outstanding feature. (don't set this register when in DMA transfer. The action will not be guaranteed)

2. Start DMA transfer. The DMA outstanding will be partial outstanding.

User can use CH0~3 for heavy traffic.

## 30.4 Usage Notes

Pay attentions to the following notes when the DMAC is used.

### 30.4.1 Module Stop

While the DMAC is in operation, the DMAC should not be stopped by the module stop register (Refer to section "9.2.3 Module stop control register") setting. When the DMAC is stopped, transfer contents cannot be guaranteed.

### 30.4.2 Address Error

When a DMA address error is generated, set the register of the error-ridden channel again and then start a transfer.

### 30.4.3 DMA Transfer Interruption

When make the DMA transfer interrupted, the interruption disable should be set and set the DE bit in DMA channel control register (CHCR) as the DMA transfer disable. If the TE is generated during the DMA transfer interruption, the TE should be initialized. The generation timing after the transfer interruption for an asynchronous assumed following three cases.

1. A case that after setting for the interruption disable, TE/DSE is changed to 1 before the CHCR initialization. The TE/DSE is initialized by the processing DMA transfer initialization.
2. A case that after setting for the interruption disable, the TE/DSE is changed to 1 by ignoring the transfer interruption. After the DMA transfer initialization and the DE bit become 0, check the TE/DSE bit. If the TE/DSE bit is 1, the DMA transfer initialization process should be performed.
3. A case that after setting for the interruption disable, the TE/DSE is not changed to 0 even though the transfer interruption is occurred. After the DMA transfer interruption, the TE/DSE is not generated because the transfer data is remained.

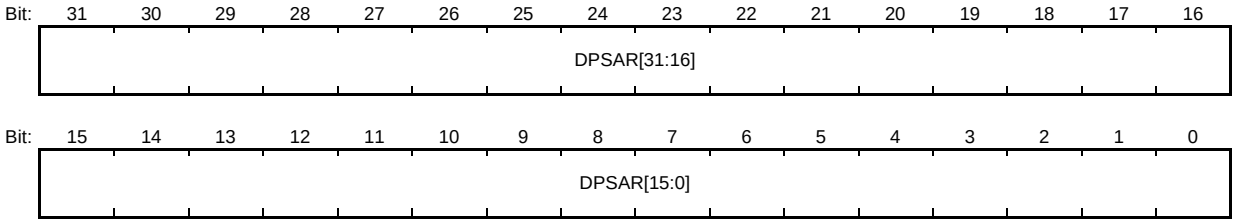
Note: In a case that the DMA transfer initialization is performed while the last transfer is executing, causes TE/DSE generation delay. So, the dummy read should be performed.

### 30.4.4 Descriptor Format

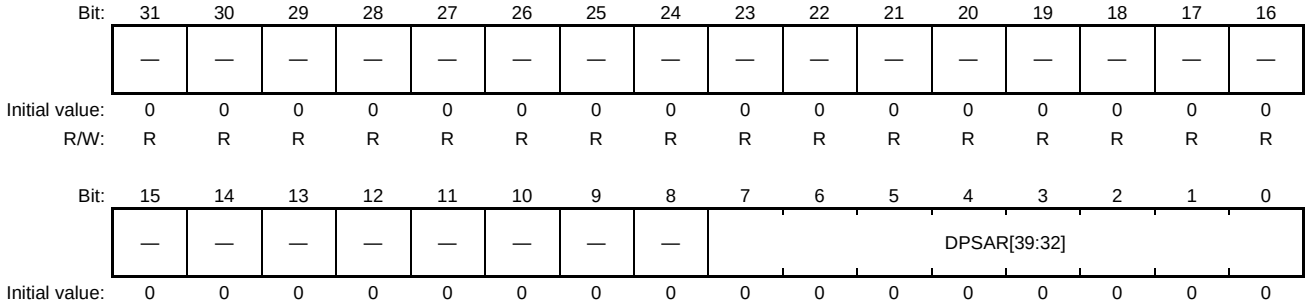
Descriptor format expands for 40-bit address support,

| | 40-bit mode | 32-bit mode |
|---|---|---|
| 0x00 | DPSAR[31:0] | DPSAR[31:0] |
| 0x04 | DPSAR[39:32] | DPDAR[[31:0] |
| 0x08 | DPDAR[[31:0] | DPTCR[31:0] |
| 0x0C | DPDAR[[39:32] | DPCHCR[31:0] |
| 0x10 | DPTCR[31:0] | DPSAR[31:0] |
| 0x14 | - | DPDAR[[31:0] |
| 0x18 | DPCHCR[31:0] | DPTCR[31:0] |
| 0x1C | - | DPCHCR[31:0] |

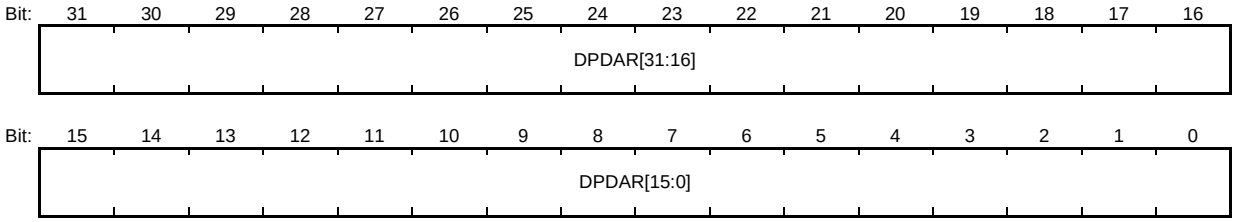
### (1) Descriptor Memory Source Address Registers (DPSAR[31:00])



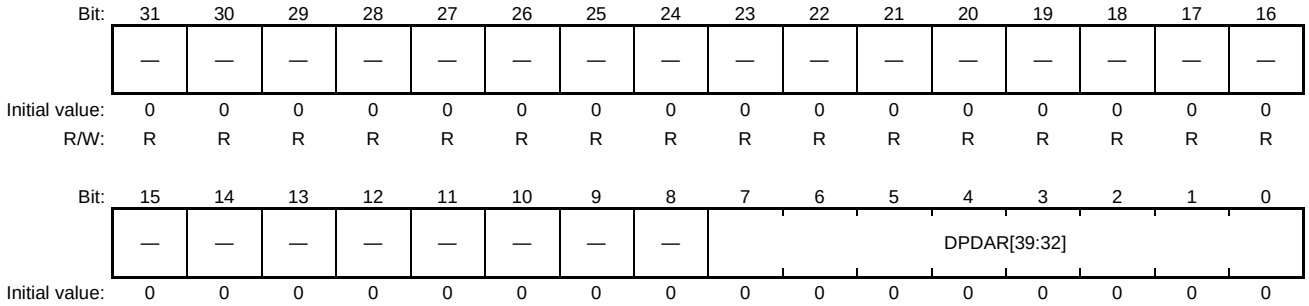

### (2) Descriptor Memory Source Address Registers (DPSAR[39:32])



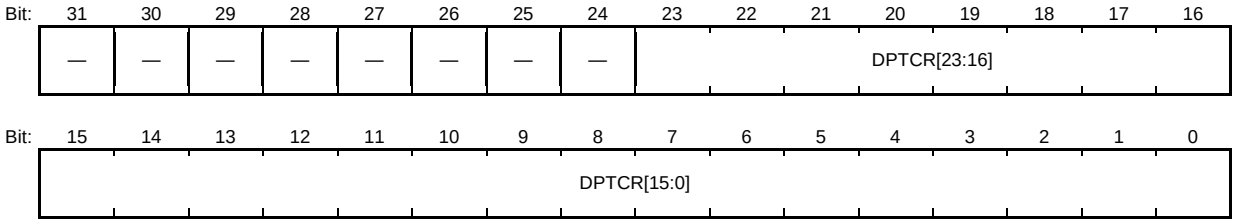

### (3) Descriptor Memory Destination Address Registers (DPDAR[31:00])



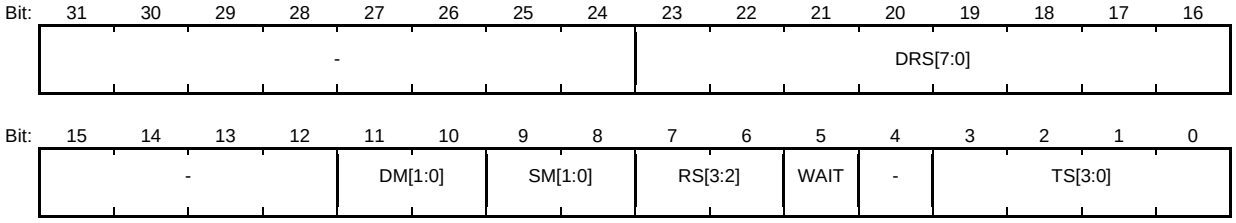

### (4) Descriptor Memory Destination Address Registers (DPDAR[39:32])



### (5) Descriptor Memory Transfer Count Registers (DPTCR)




### (6) Descriptor Memory Channel Control Register (DPCHCR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 23 to 16 | DRS[7:0] | 0 | R/W | resource select for DMA request mode |
| 11 to 10 | DM[1:0] | 0 | R/W | Destination Address Mode |
| 9 to 8 | SM[1:0] | 0 | R/W | Source Address Mode |
| 7 to 6 | RS[3:2] | 0 | R/W | Resource Selection |
| 5 | WAIT | 0 | R/W | Descriptor execution wait<br>0: normal<br>1: wait<br>If WAIT bit is set, execution of the entry is paused during OV==1 \|\| EVENT_COUNT[6:0]== 0.<br>If OV==0 && EVENT_COUNT!=0, this descriptor entry is executed and EVENT_COUNT is decremented. |
| 4 | - | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 3 to 0 | TS[3:0] | 0 | R/W | Transfer size |

## 30.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 30.5.1 Secure Function

Security/Non-secure of each channel of RT-DMAC is controlled by DMSEC register. Refer description of DMA secure control register for each channel

In case a channel is in secure mode, when Non-secure CPU access to that channel, the access must be denied, then RT-DMAC do these actions:

(1) Issue error status and store into ERRDET register.
(2) Log error ID of the first access into ERRPID register.
(3) Log error address of the first access into ERRADR register.
(4) Active error signal to ECM module. (Refer module ECM manual)

### 30.5.2 Safety Protection

As AXI master, DMAC can use REGIONID register in each channel to decide which region the channel will access. And the region that permitted to use for each channel are determined by region ID enable input
REGION ID of each channel only can be set to the value that external 16bit Enable signal allowed. There is Enable signal(*) correspond to each channel.
Each bit of Enable signal determine that which ID (0~15) can be set, if the corresponding bit is zero, the corresponding ID cannot be set by software.
For example, Enable bit is 16'b0010_0001_0000_1000 for current channel , only 3, 8, 13 can be set to REGIONID register of current channel. Other values that written to REGIONID register will be ignored.(Register values are keep.)

As APB slave, DMAC depends on bus bridge to distinguish the correct safety access or not to each channel register. If bus bridge pass an APB access to DMAC, DMAC treat it as valid access and do it. Regarding to PUBLIC registers, it suppose that only secure master can access it.

Note: The Enable signal reflects the following register settings of the BUS function.
Region ID register

RGIDMEN_RTDM{0-1}_{0-15}
RGIDMEN_SYDM1_{0-15}, RGIDMEN_SYDM2_{0-7}

For detail, see the BUS chapter (18.5.9 Region ID protection )for a description of Region ID protection.

# 31. Boot

## 31.1 Overview

This product allows selection of a master boot processor through the MD7 and MD6 pins. When Cortex-R52 or ICUMX is selected as a master boot processor, setting the MD[4:1] pins enables on-chip ROM boot or XIP(eXecute In Place) mode boot.

### 31.1.1 Features

- Either Cortex-R52 or ICUMX can be selected as a master boot processor through the MD7 and MD6 pins.
  Selection of a master boot processor is described in Table 31.2.

- The following boot devices are supported through MD pins.
  Selection of a boot device is described in Table 31.1.

  — HyperFlash (Max. 320Mbps per pin) Octal SPI Flash and Serial Flash
  — eMMC (12.5MHz x8 bus widths)
  — SCIF and HSCIF as serial downloader

- The boot sequence of HyperFlash, Octal SPI Flash, Serial Flash and eMMC.

  Cortex-R52 or ICUMX master boot processor executes the instructions in on-chip ROM.

  Data including Boot ROM parameters, address information and data size information of IPL* are transferred from the HyperFlash, Octal SPI Flash, Serial Flash and eMMC to RT-VRAM. Then the amount of data specified by data size information is transferred to RT-VRAM. After the transfer, the processor jumps to the address specified by address information of IPL.

- XIP(eXecute In Place) mode of Hyper Flash boot is supported if master boot processor is Cortex-R52.

- Error message output through SCIF port when no boot error or verification error occurs at secure boot.

- Secure boot is supported (option).

  — Secure boot is supported if a master boot processor is ICUMX.
  The boot image for each processor is authorized by ICUMX.
  — Encrypted boot image is supported.
  — Debugging is disabled at booting, and it can be enabled with secure debugging authentication.
  ICUMX can only support one-time authentication by using the Challenge and Response integrity check.
  — Boot status register setting is supported.
  — XIP mode of Hyper Flash boot is restricted.

Note: * IPL: Initial Program Loader.

### 31.1.2 External Pins

#### 31.1.2.1 MD pins

Boot device, master boot processor can be selected through MD pins.

**Table 31.1 Mode Pin Configuration (1)**

| Name | Pin Name | I/O | Function |
|---|---|---|---|
| Mode pin | MD4 to MD1 | Input | Select a boot device. |

| MD[4:1] | Description |
|---|---|
| B’0000 | Reserved |
| B’0001 | Reserved |
| B’0010 | HyperFlash boot at 160MHz using DMA |
| B’0011 | HyperFlash boot at 80MHz using DMA |
| B’0100 | Serial Flash boot at single read 40MHz using DMA |
| B’0101 | Reserved |
| B’0110 | Serial Flash boot using DMA.*2 |
| B’0111 | Octal SPI Flash 160/80MHz using DMA.*2 |
| B’1000 | Reserved |
| B’1001 | Reserved |
| B’1010 | HyperFlash at 160MHz(320Mbps) using XIP mode |
| B’1011 | HyperFlash at 80MHz using XIP mode |
| B’1100 | Reserved |
| B’1101 | eMMC boot at 12.5MHz x8 bus widths using DMA |
| B’1110 | Reserved |
| B’1111 | SCIF/HSCIF download mode*1 |

Notes:
*1 SCIF or HSCIF download mode can be selected through MD[32:31] pins when MD[4:1] pins equal to B’1111.
*2 After reset sequence, RPC makes transfer with READ command (40MHz single bit SDR and dummy cycle 0) until Boot parameters loaded. Frequency and transfer mode can be selected by CLKSELR setting. For CLKSELR, see chapter 31.3.3.1. Dummy cycle can be set by DMCYC at MD[4:1]==B'0110 and CLKSELR=B'0000010x. For DMCYC, see chapter 31.3.3.2.

**Table 31.2 Mode Pin Configuration (2)**

| Name | Pin Name | I/O | Function |
|---|---|---|---|
| | MD32 and MD31 | Input | Select SCIF/HSCIF download mode<br>**MD[32:31]** / **Description**<br>B'00 – SCIF download (*1)<br>B'01 – HSCIF download 921,600 bps (*1)<br>B'10 – HSCIF download 1,843,200 bps (*2)<br>B'11 – Reserved.<br>Notes :<br>(*1) IO voltage 3.3V<br>(*2) IO voltage 1.8V |
| Mode Pin | MD5 | Input | Reserved<br>**MD5** / **Description**<br>0 – Reserved for security.<br>1 – Normal boot. |
| Mode pin | MD7 and MD6 | Input | Select a master boot processor.<br>**MD[7:6]** / **Description**<br>B'00 – Reserved<br>B'01 – Reserved<br>B'10 – Booted through ICUMX<br>B'11 – Booted through Cortex-R52 |

#### 31.1.2.2 HyperFlash boot pins

**Table 31.3 HyperFlash Pin Configuration**

| Name | Pin Name | I/O | Function |
|---|---|---|---|
| Serial clock | QSPI0_SPCLK | Output | Differential clock output+ |
| Serial clock | QSPI1_SPCLK | Output | Differential clock output- |
| Chip select | QSPI0_SSL | Output | Chip select signal. |
| Data0 | QSPI0_MOSI_IO0 | I/O | Data 0 |
| Data1 | QSPI0_MISO_IO1 | I/O | Data 1 |
| Data2 | QSPI0_IO2 | I/O | Data 2 |
| Data3 | QSPI0_IO3 | I/O | Data 3 |
| Data4 | QSPI1_MOSI_IO0 | I/O | Data 4 |
| Data5 | QSPI1_MISO_IO1 | I/O | Data 5 |
| Data6 | QSPI1_IO2 | I/O | Data 6 |
| Data7 | QSPI1_IO3 | I/O | Data 7 |
| RDS | QSPI1_SSL | Input | Read write data strobe |
| Flash Reset | RPC_RESET_N | Output | Reset signal output |

#### 31.1.2.3 Serial Flash boot pins

**Table 31.4 Serial Flash pin Configuration**

| Name | Pin Name | I/O | Function |
|---|---|---|---|
| Serial clock | QSPI0_SPCLK | Output | Clock output. |
| Chip select | QSPI0_SSL | Output | Chip select signal |
| Transmit data | QSPI0_MOSI_IO0 | Output | Outputs transmit data |
| Receive data | QSPI0_MISO_IO1 | Input | Inputs receive data |
| Flash Reset | RPC_RESET_N | Output | Reset signal output |

#### 31.1.2.4 SCIF and HSCIF download mode pins

**Table 31.5 SCIF0 Pin Configuration**

| Name | Pin Name | I/O | Function |
|---|---|---|---|
| Transmit data | TX0 | Output | Transmit data output |
| Receive data | RX0 | Input | Receive data input |

**Table 31.6 HSCIF0 Pin Configuration**

| Name | Pin Name | I/O | Function |
|---|---|---|---|
| Transmit data | HTX0 | Output | Transmit data output |
| Receive data | HRX0 | Input | Receive data input |

#### 31.1.2.5 eMMC boot pins

**Table 31.7 MMC pin Configuration**

| Name | Pin Name | I/O | Function |
|---|---|---|---|
| Clock | MMC_CLK | Output | MMC clock |
| Command/response | MMC_CMD | I/O | Command/response |
| Data0 | MMC_D0 | I/O | Data [bit 0] |
| Data1 | MMC_D1 | I/O | Data [bit 1] |
| Data2 | MMC_D2 | I/O | Data [bit 2] |
| Data3 | MMC_D3 | I/O | Data [bit 3] |
| Data4 | MMC_D4 | I/O | Data [bit 4] |
| Data5 | MMC_D5 | I/O | Data [bit 5] |
| Data6 | MMC_D6 | I/O | Data [bit 6] |
| Data7 | MMC_D7 | I/O | Data [bit 7] |

### 31.1.3 Boot ROM

The capacity is 160 Kbytes. Cortex-R52 and ICUMX codes are included.

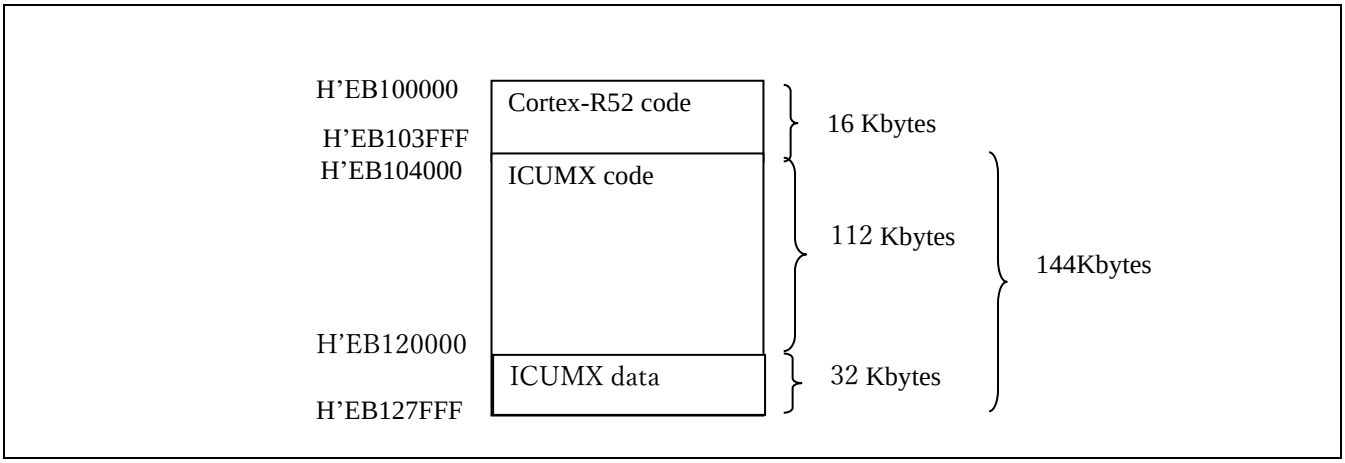



**Figure 31.1 Boot ROM memory map**

The ICUMX code area can only be read from ICUMX instruction fetch access.
For other accesses, the read value of the ICUMX code area is all '0'.

## 31.2 Register Description

No description on this section.

## 31.3 Operation

### 31.3.1 Boot Operation overview

The boot address of the master boot processor is controlled by RESET module. It depends on boot device selection through MD[4:1] pins.

If XIP mode boot is selected, the processor can directly fetch from the target device. If the boot device except for XIP mode is selected, the processor boots from on-chip Boot ROM and executes the boot sequence described "HyperFlash/Octal SPI Flash/Serial Flash using DMA Boot sequence" and "eMMC using DMA boot sequence" section. The header data are transferred from the target boot device to RT-VRAM. Then IPL is transferred to RT-VRAM. Then the processor jumps to the specified address.

If MD[4:1] reserved setting is selected, error message can be outputted via SCIF.

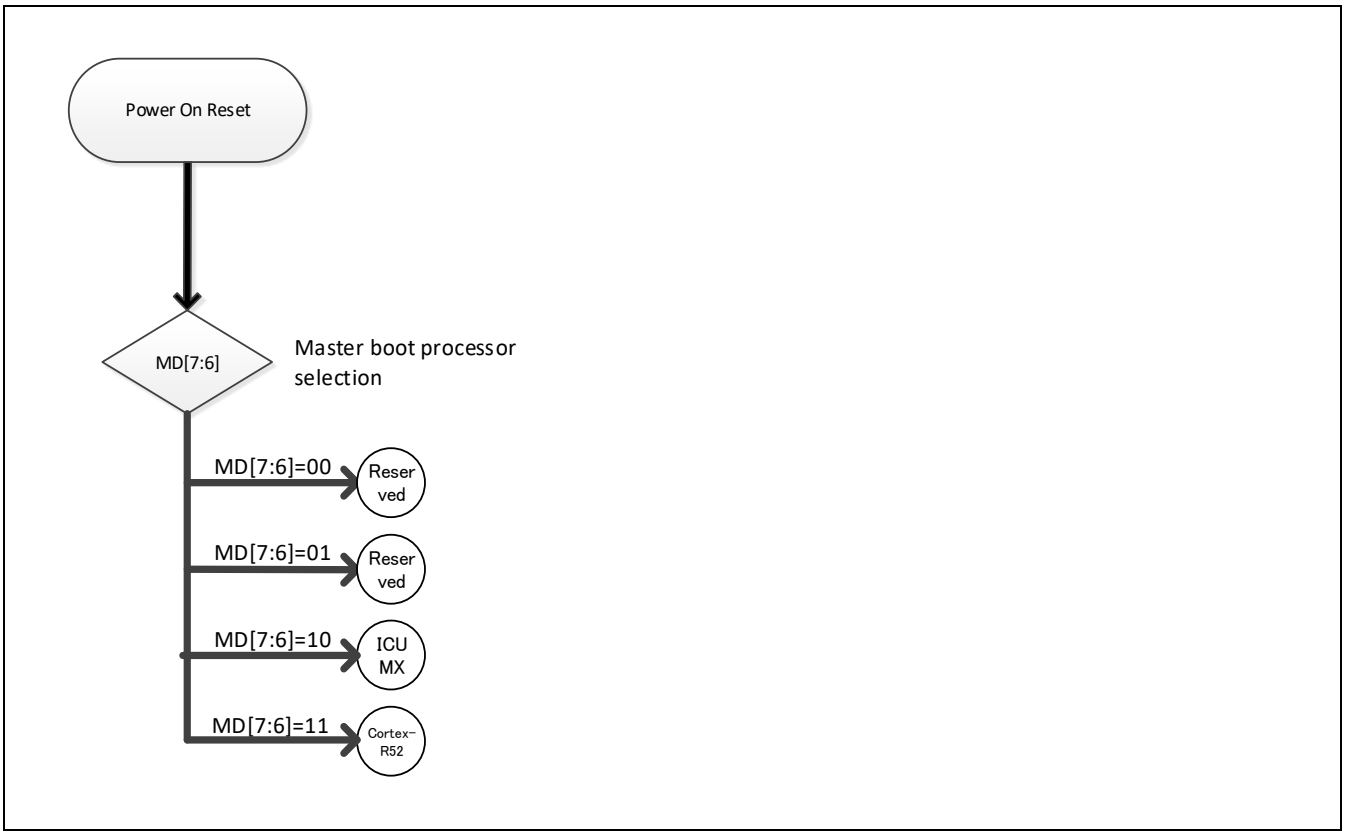



**Figure 31.2 Master Boot processor selection**

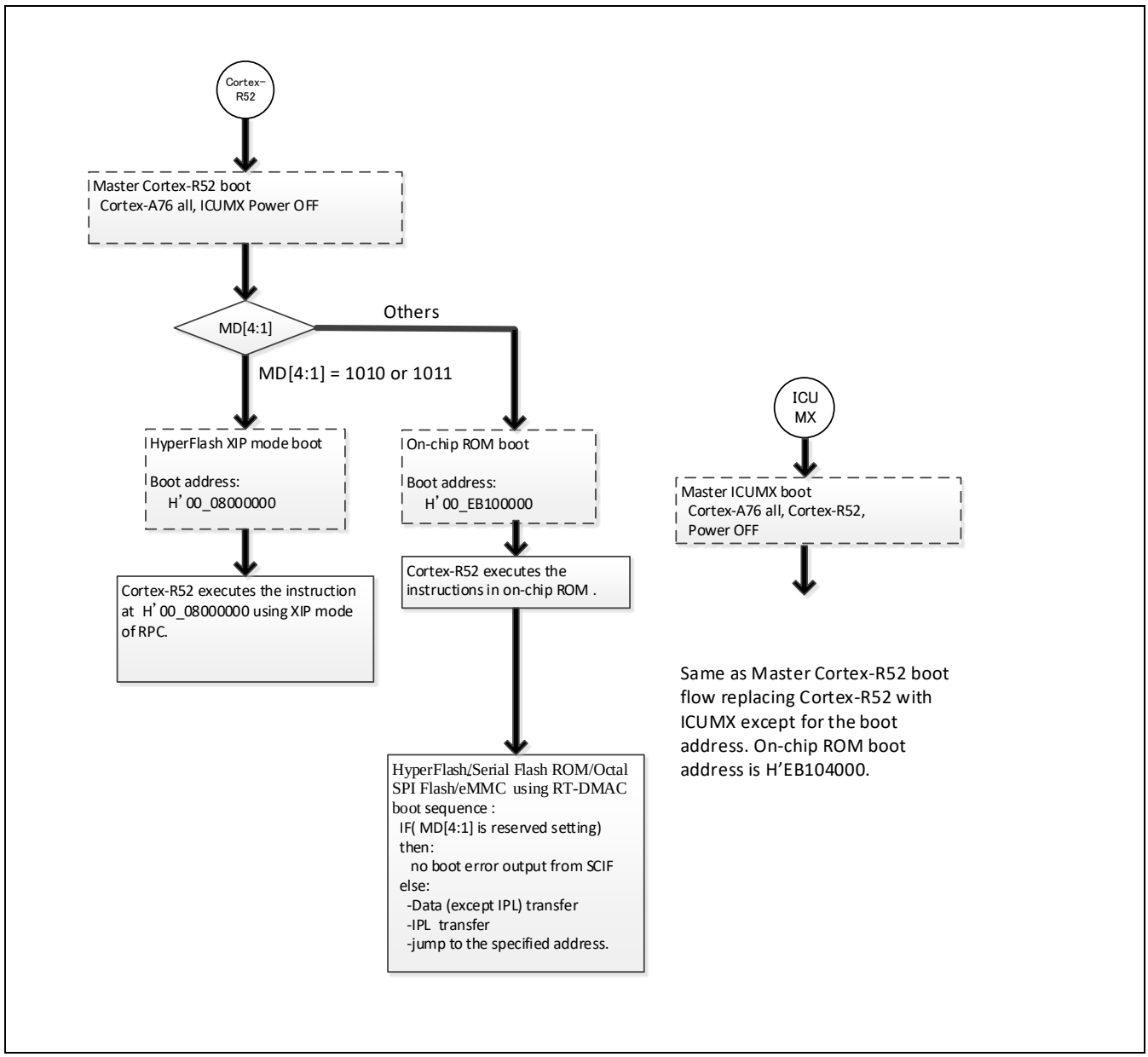



**Figure 31.3 Cortex-R52, ICUMX boot operation overview**

### 31.3.2 Boot Address control

Boot address is controlled by APMU module. If Cortex-R52 boot is selected through MD[7:6] pins, APMU.CR52BAR register function is used to specify the boot address. If ICUMX boot selected, RESET.ICUMXBAR register function is used to specify the boot address.

### 31.3.3 Boot parameters

#### 31.3.3.1 Flash Clock parameters ( External Flash ROM Address H'00000004 )

Flash Clock parameters are 4bytes and it should be at on H'00000004 address of serial flash. It controls Octal SPI Flash clock frequency. The lower 8-bit value of this parameters is copied in BOOT status register bit15-8 in MFIS by Boot ROM program.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | | | | CLKSELR | | | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | - | All 0 | - | Reserved |
| 7 to 0 | CLKSELR | - | - | QSPI Flash clock setting at MD[4:1]=B'0110<br>B'00000101: 80MHz Quad I/O<br>B'00000100: 133MHz Quad I/O<br>B'00000011: 80MHz Quad output read<br>B'00000010: 133MHz Quad output read<br>B'00000001: Reserved<br>B'00000000: 80MHz Single fast read<br>Others: 40MHz Single fast read<br><br>Octal SPI Flash clock setting at MD[4:1]=B'0111<br>B'00000101: 160MHz Octal SPI for Macronix's products<br>B'00000100: 80MHz Octal SPI for Macronix's products<br>B'00000011: 160MHz Octal SPI for Micron's products<br>B'00000010: 80MHz Octal SPI for Micron's products<br>Others: Setting prohibited |

### 31.3.3.2 Dummy Cycle Parameters (External Flash ROM Address H'00000008)

The 32bit value of this parameters is copied in RPC_DRDMCR[31:0] by Boot ROM program.

When the Quad I/O mode (MD[4:1]==B'0110 and CLKSELR=B'0000010x), these parameters are enabled. Also, if this value is 0xFFFFFFFF, RPC_DRDMCR.DMCYC is forcibly set 0x09 by Boot ROM program.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | DMCYC | | | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | - | 27'b 0 | - | Reserved<br>The write value should always be 0. |
| 4 to 0 | DMCYC[4:0] | 5'b 0 | - | Number of Dummy Cycles Setting. (*1)<br>B'00000: 1 cycle<br>B'00001: 2 cycle<br>B'00010: 3 cycle<br>:<br>B'10010: 19 cycle<br>B'10011: 20 cycle |

(*1) Ignore bit except for MD pin= B'0110 and Quad I/O mode.

### 31.3.4 HyperFlash/Octal SPI Flash/Serial Flash using DMA boot sequence

1. The master boot processor jumps to H'EB10 0000 (Cortex-R52) or H'EB10 4000 (ICUMX). The processor executes the instructions in Boot ROM and 30KB data(except IPL image) are transferred from HyperFlash, Octal SPI Flash, Serial Flash to RT-VRAM using RT-DMAC.

2. These data contain Flash Clock parameters, address information and data size information of IPL.

3. The amount of data specified by data size information is transferred to RT-VRAM. After the transfer, the master processor jumps to the address specified by address information of IPL.

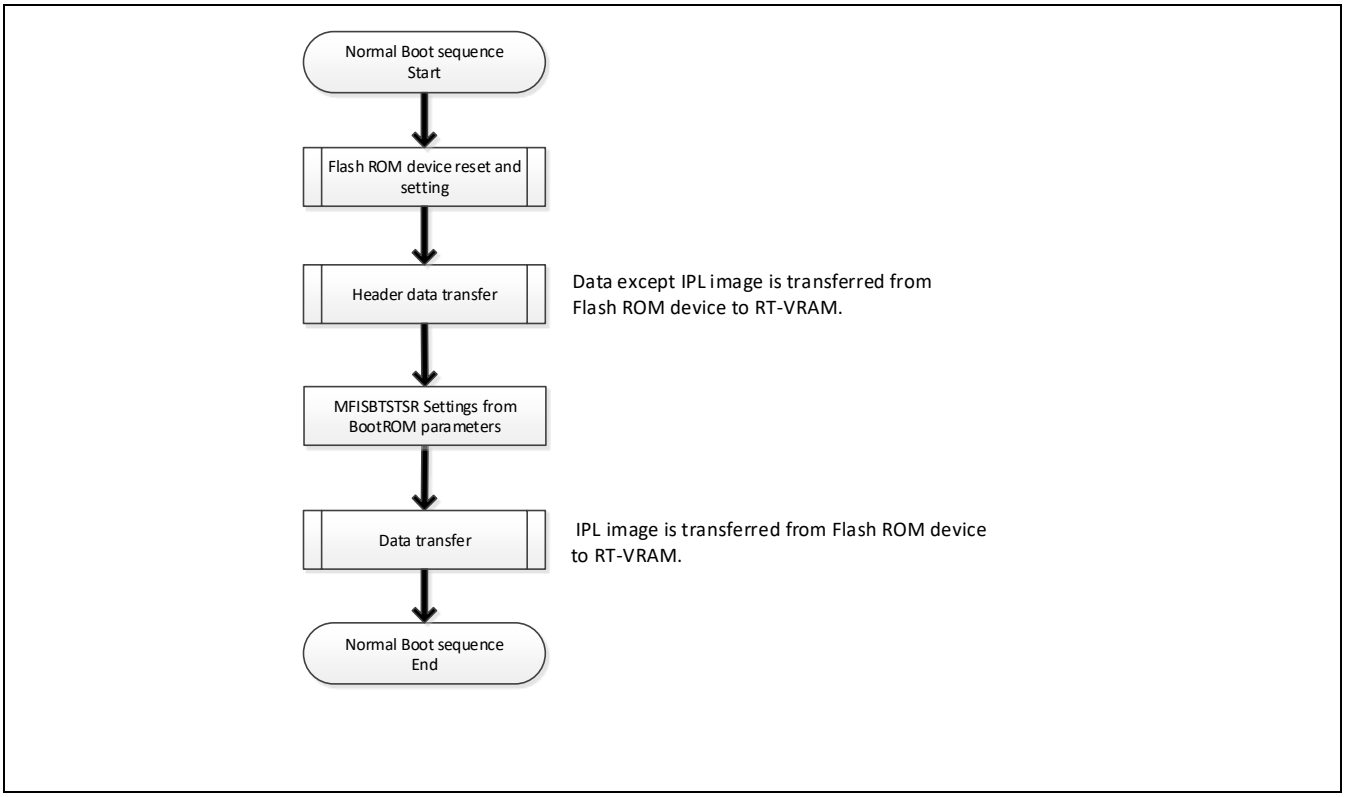



**Figure 31.4 HyperFlash/Octal SPI Flash/Serial Flash using RT-DMAC boot sequence**

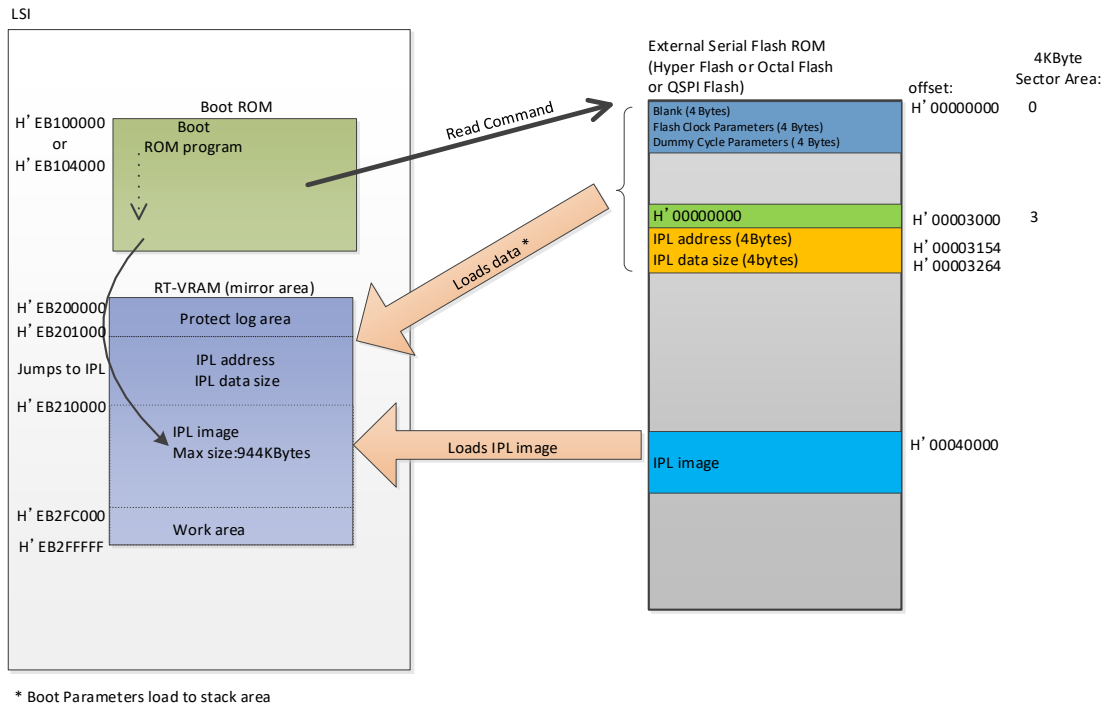



**Figure 31.5 IPL transfer and Program image**

### 31.3.5 eMMC using DMA boot sequence

1. [Cortex-R52 boot]
   The master boot processor jumps to H'EB10_0000. The processor executes the instructions in Boot ROM and 13.5KB data (except IPL image) are transferred from eMMC to RT-VRAM using MMC's DMA with 512-Byte transfer size.

   [ICUMX boot]
   The master boot processor jumps to H' EB10_4000. The processor executes the instructions in Boot ROM and 26.5KB data (except IPL image) are transferred from eMMC to RT-VRAM using MMC's DMA with 512-Byte transfer size.

2. These data contain address information and data size information of IPL.

3. The amount of data specified by data size information is transferred to RT-VRAM. After the transfer, the master processor jumps to the address specified by address information of IPL.

Note1: eMMC device must meet the following conditions.

- eMMC device that conforms to JEDEC eMMC standard 4.4 or higher
- eMMC device's IO voltage should be 1.8V.

Note2: eMMC device must meet the following conditions if master boot processor is Cortex-R52.

Set the EXT_CSD register byte179, PARTITION_CONFIG bit[6]= H'0( No boot acknowledge sent ).

- Set the EXT_CSD register byte179, PARTITION_CONFIG bit[5:3]= H'1 ( Boot Area partition1 ).
- Set the EXT_CSD register byte177, BOOT_BUS_CONDITIONS bit[4:3]= H'1( 50MHz SDR ).
- Set the EXT_CSD register byte177, BOOT_BUS_CONDITIONS bit[1:0]= H'2( x8 bus widths ).

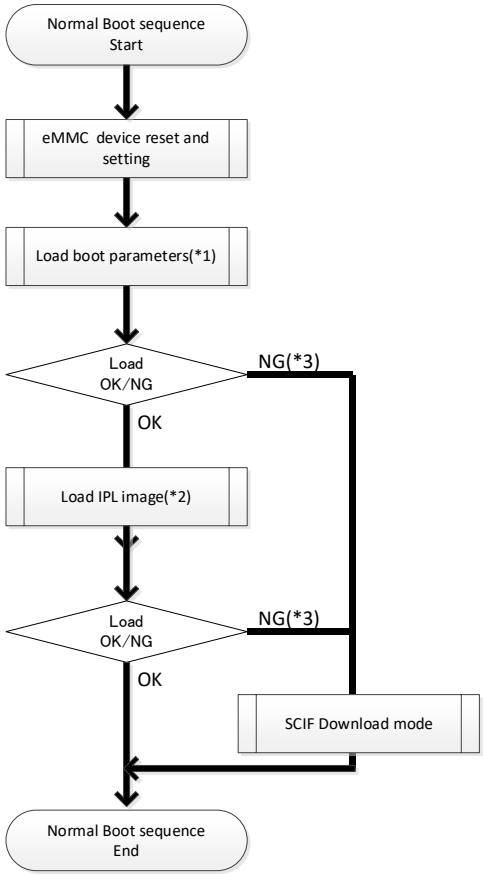



**Figure 31.6 eMMC using DMA boot sequence**

If master boot processor is Cortex-R52:

(*1) CMD0 is issued three times to transfer boot parameters, with argument + H'F0F0F0F0, + H'FFFFFFFA, + H'00000000.

(*2) As with (*1), CMD0 is also used during IPL image transfer.

(*3) In case of the error detection of SDHI module or IPL address range is invalid, judged to be NG.

If master boot processor is ICUMX:

(*1) Use the following command to initialize the device (IDLE state --> Transfer state).
CMD0 with argument + H'00000000,CMD1,CMD2,CMD3,CMD9,CMD7,CMD16,CMD8.
CMD6, CMD13 are used for changing frequency and bus width.
When loading boot parameters, CMD23,CMD18,and CMD13 are used .

(*2) CMD23,CMD18,and CMD13 are used.

(*3) In case of the error detection of SDHI module or IPL address range is invalid, judged to be NG.

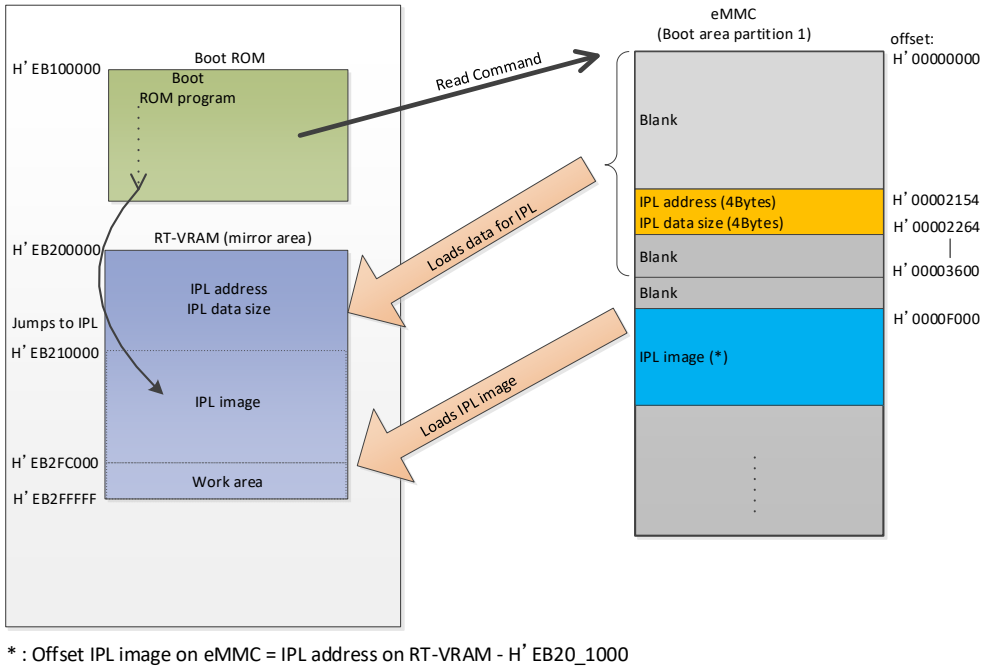



**Figure 31.7 IPL transfer and Program image(eMMC, Cortex-R52 Boot)**

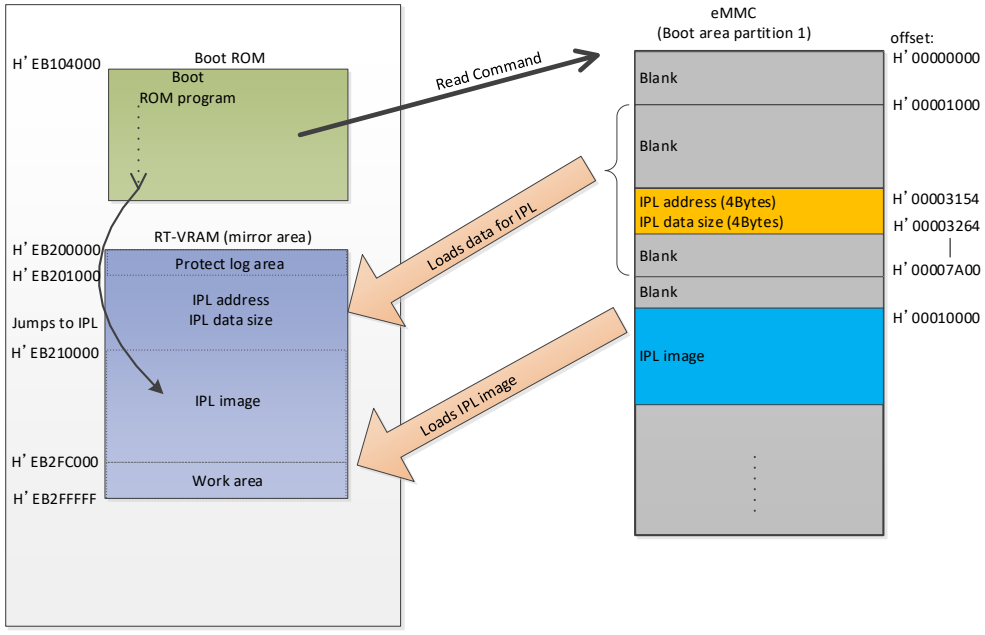



**Figure 31.8 IPL transfer and Program image(eMMC, ICUMX Boot)**

### 31.3.6 SCIF download mode

SCIF download mode uses SCIF0 serial ports. Data from Host PC is transferred to RT-VRAM. The transfer specification is below.

- Baud rate : 115200bps
- Data length : 8bit
- Parity : none
- Stop bit : 1bit
- Flow control : none
- Data format : Motorola S-record

The download sequence and data format are below.

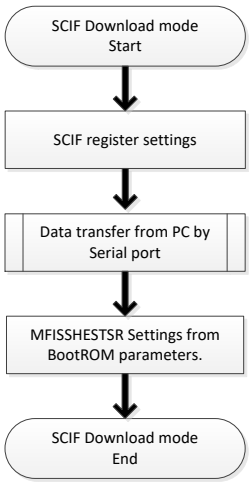



**Figure 31.9 SCIF download mode sequence**

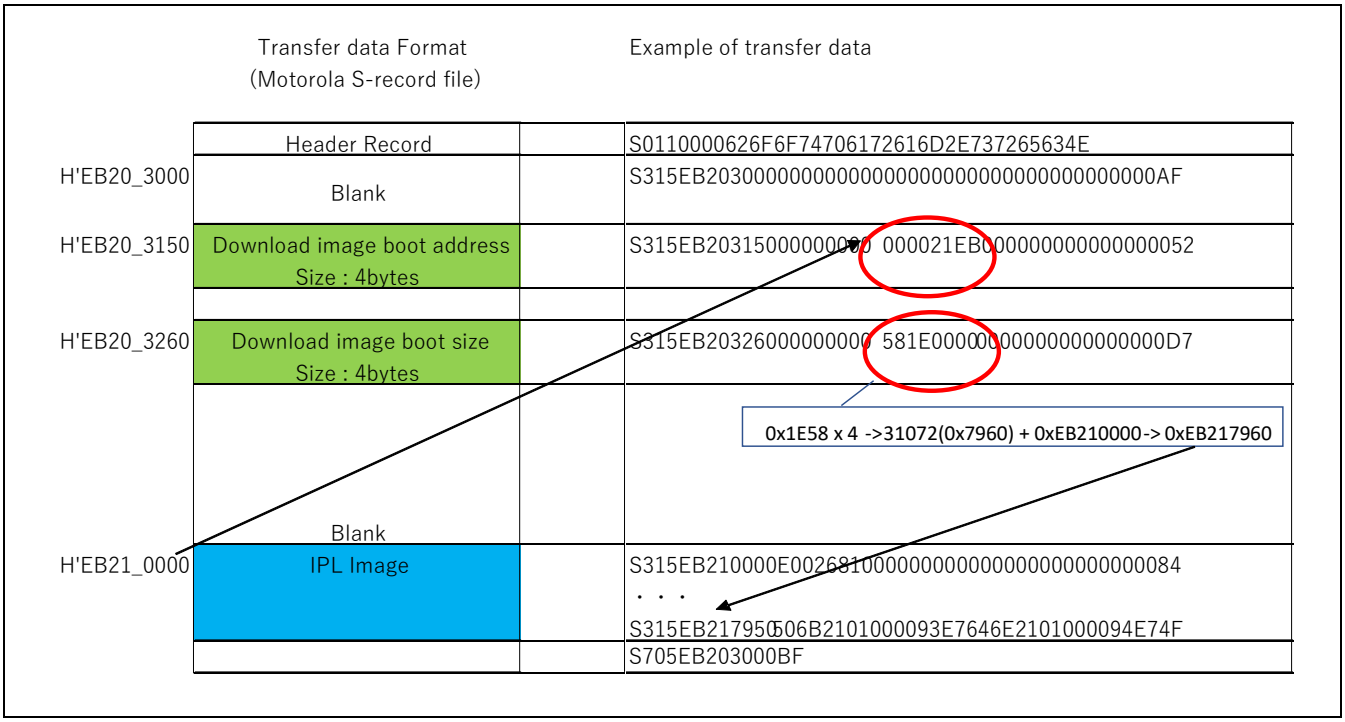



**Figure 31.10 SCIF data format for transfer data**

### 31.3.7 HSCIF download mode

HSCIF download mode uses HSCIF0 serial ports. Data from Host PC is transferred to RT-VRAM. The transfer data format is the same as SCIF download mode's.

The baud rate can be selected with MD[32:31] setting. Refer to Table 31.2.

### 31.3.8 Error message

The Boot ROM program outputs an error message via the serial port, if:

1. invalid (reserved) boot mode selected

2. failed verification of data in secure boot mode

3. invalid parameter(IPL address and size) input

## 31.4 Usage Note

### 31.4.1 IPL address and data size limitation

The IPL end address can be calculated by IPL address and data size. It should be lower than the address below.

- H'EB2F_BFFF

The IPL address requires 512-byte alignment.

### 31.4.2 Cleanup

Before jumping to the IPL, Boot ROM program cleans hardware registers used for loading the IPL and the master boot processor, as listed below.

Cortex-R52

- RT-DMAC registers are set to hardware initial value.
- Disable I-cache. Disable branch prediction setting.

ICUMX

- RT-DMAC registers are set to hardware initial value.
- FE level maskable interrupt (FEINT) and EI level maskable interrupt (EIINT) are always disabled.

Notes

- The PFC setting is not cleared after jumping to the IPL.
- The RPC module registers are not cleaned by Boot ROM program.
  For Serial Flash (MD[4:1]=B'0110) and Octal SPI Flash(MD[4:1]=B'0111) boot, RPC registers are configured by Boot ROM program. Refer to 31.4.6 for register settings after Boot ROM program execution.
- In the case of eMMC boot, initialize the SDHI interface registers.
- IRQ and FIQ are disabled.
- MPU of Cortex-R52 is not cleaned. Refer to chapter 6. Realtime Core for MPU setting.

### 31.4.3 Pull-up/down settings

Boot ROM program sets pull-up/down to QSPI pins. The below table shows pull-up/down settings in Serial Flash and Octal SPI Flash boot mode.

**Table 31.8 Pull-up/down settings**

| MD[4:1] | Descriptions | QSPI0_IO2/QSPI0_IO3 | QSPI1_SPCLK | QSPI1_MOSI_IO0/<br>QSPI1_MISO_IO1/<br>QSPI1_IO2/QSPI1_IO3 |
|---|---|---|---|---|
| B'0100 | Serial Flash 40MHz | Pull-up | Pull-down | Pull-up |
| B'0110 | Serial Flash 40MHz<br>to 80MHz<br>(CLKSELR=0) | Pull-up | Pull-down | Pull-up |
| | Serial Flash 40MHz<br>to Quad Output/IO<br>(CLKSELR=2~5) | Pull-up(single)<br>to<br>No Pull-up/down (Quad) | Pull-down | Pull-up |
| | Serial Flash 40MHz<br>(CLKSELR=others, except<br>CLKSELR=1 (reserved)) | Pull-up | Pull-down | Pull-up |
| B'0111 | Octal SPI Flash 80/160MHz<br>(CLKSELR=2~5) | Pull-up(single)<br>to<br>No Pull-up/down (octal) | Pull-down(single)<br>to<br>No Pull-up/down (octal) | Pull-up(single)<br>to<br>No Pull-up/down (octal) |

### 31.4.4 Dual channel Serial Flash(Dual QSPI) boot

Boot ROM program supports booting from single channel serial flash and does NOT support booting from dual channel serial Flash. Dual channel serial Flash boot can be selected by setting the BSZ bits of CMNCR register in RPC to B'01 after IPL loading.

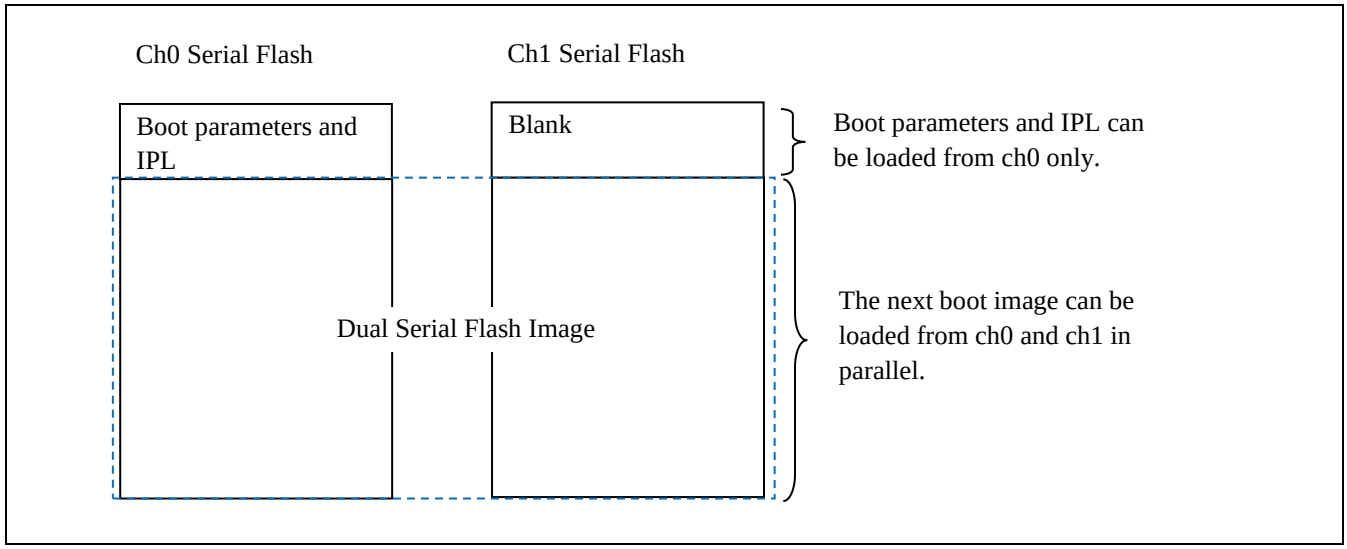



**Figure 31.11 Flash image of dual channel Serial Flash booting**

### 31.4.5 Serial Flash Reset commands

Serial Flash Reset commands are listed below.

**Table 31.9 Serial Flash Reset commands**

| Command | Description |
|---|---|
| H'F0 | RESET ( selected by E-fuse setting ) |
| H'66 | RSTEN ( selected by E-fuse setting ) |
| H'99 | RST ( selected by E-fuse setting ) |

### 31.4.6 Register settings after on-chip ROM boot

#### 31.4.6.1 Serial Flash boot (MD[4:1] = B'0110)

Table 31.10 shows the register settings after on-chip ROM boot in Serial Flash (MD[4:1]=B’0110).

**Table 31.10 Register settings after on-chip ROM boot in Serial Flash boot (MD[4:1]=B’0110)**

| Boot Parameter | CLKSELR (*1) | B'00000000 | B'000000010 | B'000000011 | B'000000100 | B'00000101 | Others (*6) |
|---|---|---|---|---|---|---|---|
| Specification | Operating clock | 80MHz | 133MHz | 80MHz | 133MHz | 80MHz | 40MHz |
| | Bit width (*2) | 1-1-1 | 1-1-4 (*3) | 1-1-4 (*3) | 1-4-4 (*3) | 1-4-4 (*3) | 1-1-1 |
| | Command | H'0B | H'6B | H'6B | H'EB | H'EB | H'0B |
| | Address width | 24 bit | 24 bit | 24 bit | 24 bit | 24 bit | 24 bit |
| | Dummy cycle | 8 cycle | 8 cycle | 8 cycle | (*4) | (*4) | 8 cycle |
| Register Setting | CMNCR | H'01555300 | H'01555300 | H'01555300 | H'01555300 | H'01555300 | H'01555300 |
| | DRCR | H'001F0100 | H'001F0100 | H'001F0100 | H'001F0100 | H'001F0100 | H'001F0100 |
| | DRCMR | H'000B0000 | H'006B0000 | H'006B0000 | H'00EB0000 | H'00EB0000 | H'000B0000 |
| | DRENR | H'0000C700 | H'0002C700 | H'0002C700 | H'0202C700 | H'0202C700 | H'0000C700 |
| | DRDMCR | H'00000007 | H'00000007 | H'00000007 | (*4) | (*4) | H'00000007 |
| | DRDRENR | H'00000000 | H'00000000 | H'00000000 | H'00000000 | H'00000000 | H'00000000 |
| | PHYCNT | H'08078260 | H'08078260 | H'08078260 | H'08078260 | H'08078260 | H'08078260 |
| | PHYOFFSET1 | (*5) | (*5) | (*5) | (*5) | (*5) | (*5) |
| | PHYINT | (*5) | (*5) | (*5) | (*5) | (*5) | (*5) |
| | RPCCKCR | H'00000013 | H'00000019 | H'00000013 | H'00000019 | H'00000013 | H'00000017 |

(*1) Refer to section 31.3.3.1.

(*2) Command-Address-Data

(*3) Some devices require the Non-Volatile register settings to enable the Quad I/O and Quad Output operation.

(*4) Depends on DMCYC[4:0] in Dummy Cycle Parameters, refer to section 31.3.3.2.

(*5) Same as initial value, refer to Table 112.3.

(*6) except CLKSELR=B’00000001(reserved)

#### 31.4.6.2 Octal SPI Flash boot (MD[4:1] = B'0111)

Table 31.11 shows the register settings after on-chip ROM boot in Octal SPI Flash (MD[4:1]=B’0111).

**Table 31.11 Register settings after on-chip ROM boot in Octal SPI Flash boot (MD[4:1]=B’0111)**

| Boot Parameter | CLKSELR (*1) | B'00000010 | B'00000011 | B'00000100 | B'000000101 |
|---|---|---|---|---|---|
| Specification | Operating clock | 80MHz DDR | 160MHz DDR | 80MHz DDR | 160MHz DDR |
| | Bit width (*2) | 8-8-8 | 8-8-8 | 8-8-8 | 8-8-8 |
| | Command | H'0B, H'0B | H'0B, H'0B | H'EE, H'11 | H'EE, H'11 |
| | Address width | 32 bit | 32 bit | 32 bit | 32 bit |
| | Dummy cycle | 16 cycle | 16 cycle | 20 cycle | 20 cycle |
| Register Setting | CMNCR | H'01FF5301 | H'01FF5301 | H'01FF5301 | H'01FF5301 |
| | DRCR | H'001F0100 | H'001F0100 | H'001F0100 | H'001F0100 |
| | DRCMR | H'000B000B | H'000B000B | H'00EE0011 | H'00EE0011 |
| | DRENR | H'8202CC00 | H'8202CC00 | H'8202CC00 | H'8202CC00 |
| | DRDMCR | H'0000000F | H'0000000F | H'00000013 | H'00000013 |
| | DRDRENR | H'00005101 | H'00005101 | H'00005101 | H'00005101 |
| | PHYCNT | H'08B78261 | H'08B78261 | H'08F78261 | H'08F78261 |
| | PHYOFFSET1 | H'21511144 | H'21511144 | H'21511144 | H'21511144 |
| | PHYINT | (*4) | (*4) | (*4) | (*4) |
| | DREAR (*3) | H'00000000 | H'00000000 | H'00000000 | H'00000000 |
| | RPCCKCR | H'00000013 | H'00000011 | H'00000013 | H'00000011 |

(*1) Refer to section 31.3.3.1.

(*2) Command-Address-Data

(*3) If reading 32MB or more in the IPL software, set the EAC[2:0] = 001.

(*4) Same as initial value, refer to Table 112.3.

## 31.5 Safety Mechanisms

No description on this section.

# 33. External Bus Controller for SDRAM (DBSC5)

## 33.1 Overview

The external bus controller (DBSC5) supports LPDDR4X/LPDDR5 SDRAM. DBSC5 contains AXI bus controller, cache, scheduler, device controller, and register controller. The AXI bus controller receives requests from an AXI bus and performs protocol conversion to suit the bus in DBSC5. The scheduler makes schedules for the requests stored in the cache block. The device controller converts requests that follow the internal bus protocol to command-level requests and reads from and writes to the SDRAM in response to commands. The register controller reads from and writes to the control registers that set the operations of DBSC5.

DBSC5 enables the maximum use of SDRAM bus bandwidth using the following functions:

(1) Multibank operation that improves the page hit rate.
(2) Preceding execution of the bank activate commands for subsequent requests

### 33.1.1 Definition of Terms and Symbols

The following terms and symbols are used throughout the DBSC5 specifications:

- In some cases, LPDDR4X/LPDDR5 SDRAM is abbreviated as SDRAM
- ceil($x$) is the smallest integer which is greater than or equal to real number $x$.
- floor($x$) is the greatest integer which is less than or equal to real number $x$.

### 33.1.2 Functions

Table 33.1 DBSC5 Functions.

| Item | Function |
|---|---|
| Bank/Bank group mode | Supports 8 banks and no bank group for LPDDR4X<br>Supports 4 banks and 4 bank groups (BG mode) for LPDDR5<br>Supports 16 banks and no bank group for LPDDR5 |
| External bus width | 16 bits x 2 channels x 1 instances |
| Preceding activate functions | Determines the content of subsequent requests in a request queue and performs preceding activate processing for the bank to be accessed during an empty command cycle upon page-miss. |
| Operating modes | Burst length:16 (fixed)<br>Burst type: Sequential (fixed) |
| Power-down mode | Supports Idle power down mode, Self-refresh mode, Self-refresh-power down mode and<br>Deep sleep mode.<br>Combined use of the power-down mode and address translation function allows the SDRAM partial array self-refresh function to be effectively implemented. |
| Refreshes | Average interval and the minimum/maximum pulling-in count are set by the register. |
| ZQ calibration operation | Supports the automatic ZQ calibration function. The frequency of issuing ZQ calibration function can be specified through register settings in units of the auto-refresh count. |
| PHY interface | Supports DFI5.0. |
| Memory to be connected | LPDDR4X-SDRAM compliant with JEDEC JESD209-4C and JEDEC JESD209-4-1A.<br>(One LPDDR4X-SDRAM can be connected to one channel. two 16bit channels within a pair must have same sizes. Support memory with sizes from 2Gbits to 16Gbits per 16bit channel per rank.)<br>2 ranks/channel support. (Two ranks must have same sizes.)<br>LPDDR5-SDRAM compliant with JEDEC JESD209-5B<br>(One LPDDR5-SDRAM can be connected to one channel. two 16bit channels within a pair must have same sizes. Support memory with sizes from 2Gbits to 32Gbits per 16bit channel per rank.)<br>2 ranks/channel support. (Two ranks must have same sizes.) |
| DBI (Data Bus Inversion) | Supports. |
| Cache | 64KB x 1, one cache of which is shared by two 16bit channels. |
| Restriction | Unsupports the following LPDDR5 functions.<br>• Data Copy Low Power Fucntion<br>• Write X Operation<br>Supports the memory bit rate.<br>• 6400Mbps(max)@1rank / 5500Mbps(max)@2rank (T.B.D) |

### 33.1.3 Block Diagram

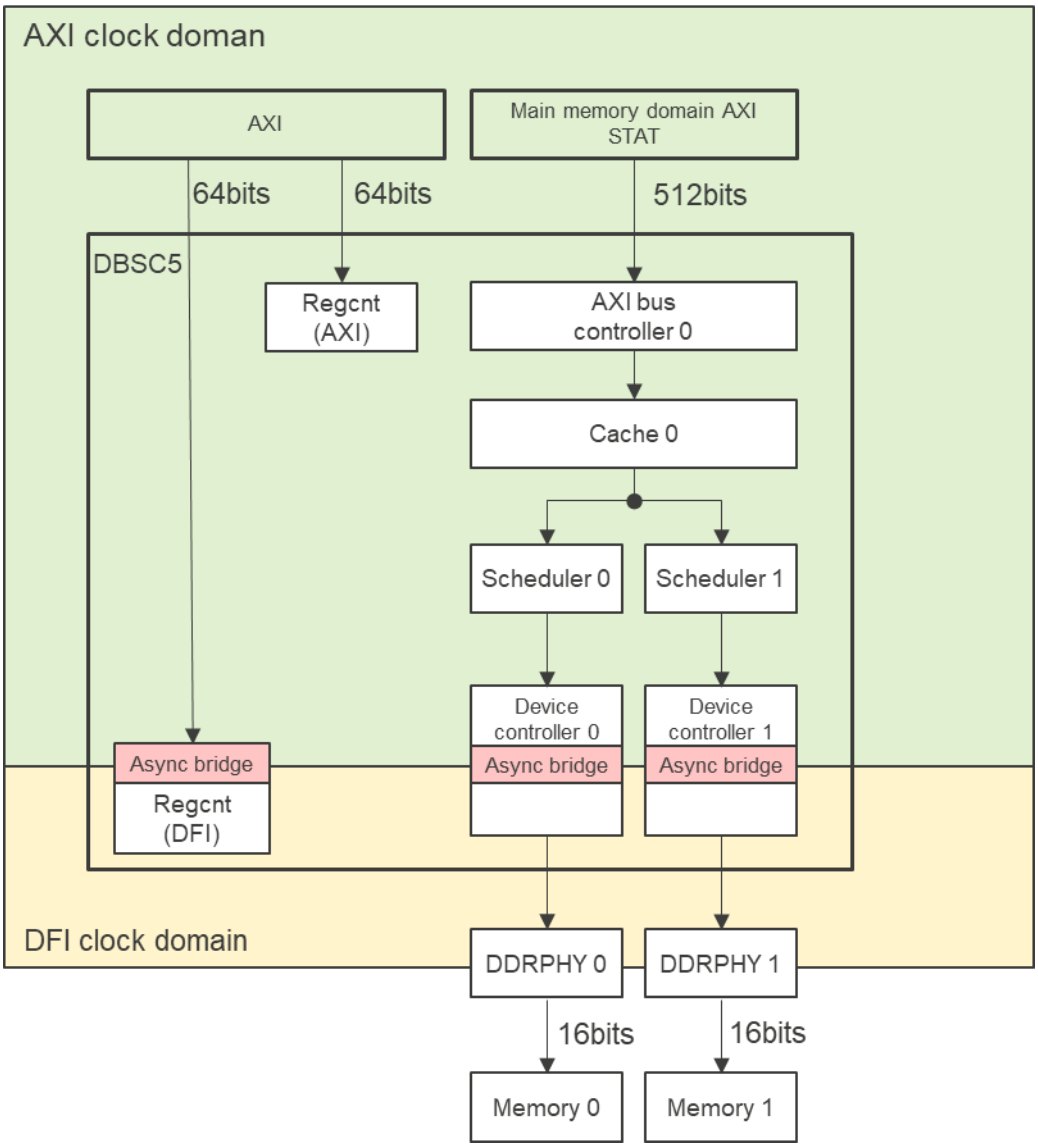



Figure 33.1 Block Diagram of DBSC5

### 33.1.4 Register Configuration

Refer to sheet “Register Configuration” of R-CarV4M_UM_033_DBSC5_r0p81.xlsx

Notes:

1. Access dbsc5 registers 32 bits at a time. Operations cannot be guaranteed if access is attempted other than 32 bits.
2. Do not access to any address other than listed above. Operations cannot be guaranteed if access is attempted to addresses other than listed above.
3. Writing to listed above register (except DB[0/1]SYSCNT0 and DB[0/1]SYSCNT0A ) must be follow the following procedure.
   DB0SYSCNT0.REGLOCK <=0x1234
   DB0SYSCNT0A.REGLOCKA <= 0x1234
   If setting values other than the above are written, it can not be written to listed above register.
4. Register access is prohibited when clock is not supplied to dbsc5.
5. Register access to DDR memory/DDR-PHY is prohibited when clock is not supplied to DDR memory/DDR-PHY.
6. If a setting value or calculation formula is defined for additional document for application note, etc., those document contents will be given priority.

## 33.2 Register Description

Refer to sheet “Register Description” of R-CarV4M_UM_033_DBSC5_r0p81.xlsx

Table 33.2 Manual Command Issue Functions.

| OPC | Code | Operation | Function of the CH Bits | Function of the RANK Bits | Function of the ARG Bits | LPDDR4X | LPDDR5 |
|---|---|---|---|---|---|---|---|
| 0000 | Wait | Issue "Device Deselected". | Don't care. | Don't care | Don't care | √ | √ |
| 0001 | Reset | Control the dfi_reset_n signal. | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | 0: Assertion<br>1: Negation | √ | √ |
| 0010 | Lock | Disable issuing a command other than manual commands. | Don't care. | Don't care. | 0: Releasing from being disabled<br>1: Disabling | √ | √ |
| 0100 | PreA | Issue "Precharge All". | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | Don't set a value other than H'0010. | √ | √ |
| 0101 | Ref | Issue "Refresh". | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | Don't set a value other than H'0010. | √ | √ |
| 0111 | CAS | Issue "CAS". | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | B'10_0000_0000: CAS(WCK2CK FAST SYNC)<br>B'11_1000_0000: CAS(WCK2CK SYNC OFF)<br>B'10_1000_0000: CAS(WCK SUSPEND) | — | √ |
| 1000 | PD | Issue "Power Down Entry/Exit". | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | 0: Enter<br>1: Exit | √ | √ |
| 1010 | SR | ssue "Self-Refresh/Self-Refresh-Power-Down/Deep-Sleep-Mode Entry/Exit". | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | B'00: SR Enter<br>B'01: SR/SRPD/DSM Exit<br>B'10: SRPD Enter(LPDDR5 olny)<br>B'11: DSM Enter(LPDDR5 only) | √ | √ |
| 1101 | MPC | Issue MPC | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | B'0100_1111: ZQCal Start(LPDDR4 only)<br>B'0101_0001: ZQCal Latch(LPDDR4 only)<br>B'1000_0101: ZQCal Start(LPDDR5 only)<br>B'1000_0110: ZQCal Latch(LPDDR5 only) | √ | √ |
| 1110 | MRW | Issue MRW. | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | Set the value of MA[7:0] in ARG[15:8] and of OP[7:0] in ARG[7:0]. | √ | √ |
| 1111 | MRR | Issue MRR. | A particular physical channel is specified for the operation. | A particular rank is specified for the operation. | Set the value of MA[7:0] in ARG[15:8]. | √ | √ |

## 33.3 Operation

No description on this section.

## 33.4 Usage Notes

Before writing to DBSC register, it must be set to "0x00001234" for the following registers.
After writing to DBSC register, it must be set to a value other than "0x00001234" for the following registers.

*DB0SYSCNT0A*
*DB0SYSCNT0*

### 33.4.1 Activation Sequence

**(1) Initialization Sequence**

Refer to the following document.

R-Car Series, 4th Generation Initialization sequence of LPDDR4X/LPDDR5 interface.

**(a) Overview**

1. After stable power supply, reset release
2. Disable ECM
3. Disable DCLS error interrupt for memory channel 0 and 1.
4. DDRPHY register initialization
5. DDRPHY of PLL / DLL lock start
6. DDRPHY of PLL / DLL lock
7. DDRPHY training operation start
8. DDRPHY training Exit
9. DBSC register initialization
10. SDRAM register initialization
11. Auto-refresh operation start
12. Initialization Exit
13. Clear safety error interrupt signal for clk_dbsc domain
14. Enable DCLS error interrupt for memory channel 0 and 1.
15. Enable ECM

Note: Do not write DB[n]RFEN and DB[n]ACEN before 9.

**(2) Self-Refresh Sequence**

Refer to the following document.

R-Car Series, 4th Generation Initialization sequence of LPDDR4X/LPDDR5 interface.

**(a) Overview**

1. Disable DRAM access
2. Precharge all DRAM Banks
3. Issue Self-Refresh Entry Command
4. Disable Auto-Refresh

Use the following procedure to release self-refresh mode.

1. Enable Auto-Refresh
2. Issue Self-Refresh Exit Command
3. Enable DRAM access

**(3) SDRAM Power-Supply Backup Mode**

Refer to the following document.

R-Car Series, 4th Generation Initialization sequence of LPDDR4X/LPDDR5 interface.

**(a) Overview**

1. Disable DRAM access
2. Precharge all DRAM Banks
3. Issue Self-Refresh Entry Command
4. Disable Auto-Refresh
5. Set the M*BKUP signal to high level. Wait for at least 1us.
6. You can turn off the unnecessary power, other than the DDR-IO power.

**(4) Recovery from SDRAM Power-Supply Backup Mode**

Refer to the following document.

R-Car Series, 4th Generation Initialization sequence of LPDDR4X/LPDDR5 interface.

**(a) Overview**

1. After stable power supply, reset release
2. Disable ECM
3. Disable DCLS error interrupt for memory channel 0 and 1.
4. DDRPHY register initialization
5. DDRPHY of PLL / DLL lock start
6. DDRPHY of PLL / DLL lock
7. Set the M*BKUP signal to low level.
8. DDRPHY training operation start
9. DDRPHY training Exit
10. DBSC register initialization
11. SDRAM register initialization
12. Auto-refresh operation start
13. Recovery Exit
14. Clear safety error interrupt signal for clk_dbsc domain
15. Enable DCLS error interrupt for memory channel 0 and 1.
16. Enable ECM

Note: Do not write DB[n]RFEN and DB[n]ACEN before 10.

### 33.4.2 Dynamic Refresh Rate Control Sequence

Refresh period setting in STAT shall be constant, and the longest period in the operation shall be set.

**(1) For Change from Longer Period to Shorter Period**

1. QoS setting in STAT shall be changed for the shorter period.
2. Refresh period setting in DBSC shall be changed.
3. Enable new refresh period setting in DBSC.

**(2) For Change from Shorter Period to Longer Period**

1. Refresh period setting in DBSC shall be changed.
2. Enable new refresh period setting in DBSC.
3. QoS setting in STAT shall be changed for the longer period.

## 33.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 33.5.1 DCLS Mechanisms

#### (1) Overview

In case of DBSC, dbsc5rcarv4m_core (dbsc5rcarv4m_master) is a target of DCLS. This is an integration spec. of DCLS (Dual Core Lock Step) for DBSC. This module has two clock domain such as clk_axim and clk_dbsc. Since these clock domain are asynchronous, it's necessary to implement "comparator" as block diagram (Figure 33.2).

#### (2) Related Registers

Table 33.3 and Table 33.4 show the register configuration for DBSC module related to DCLS functional safety.

Table 33.3 Related Registers (ECM Register Mapping)

*n = 0*

| Register Name | Abbreviation | Bit | Error content in ECM |
|---|---|---|---|
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 31 | DCLS comparator error interrupt indication for AXI-domain (excluding SRAM related). |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 30 | DCLS comparator error interrupt indication for AXI-domain (SRAM related) |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 31 | DCLS comparator error interrupt indication for memory channel 1 (DDR1) in DFI clock domain |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 15 | DCLS comparator error interrupt indication for memory channel 0 (DDR0) in DFI clock domain |

**Table 33.4 Related Registers**

*n = 0*

| Register Name | Abbreviation |
|---|---|
| Fusa Interrupt Enable Register 0 (for AXI domain) | DB[*n*]FSINTENB00A |
| Fusa Injection Mode Register 0 (for AXI domain) | DB[*n*]FSINJECT00A |
| Fusa Interrupt Enable Register 1 (for DFI domain) | DB[*n*]FSINTENB01D |
| Fusa Injection Mode Register 1 (for DFI domain) | DB[*n*]FSINJECT01D |

### (3) Hardware Description

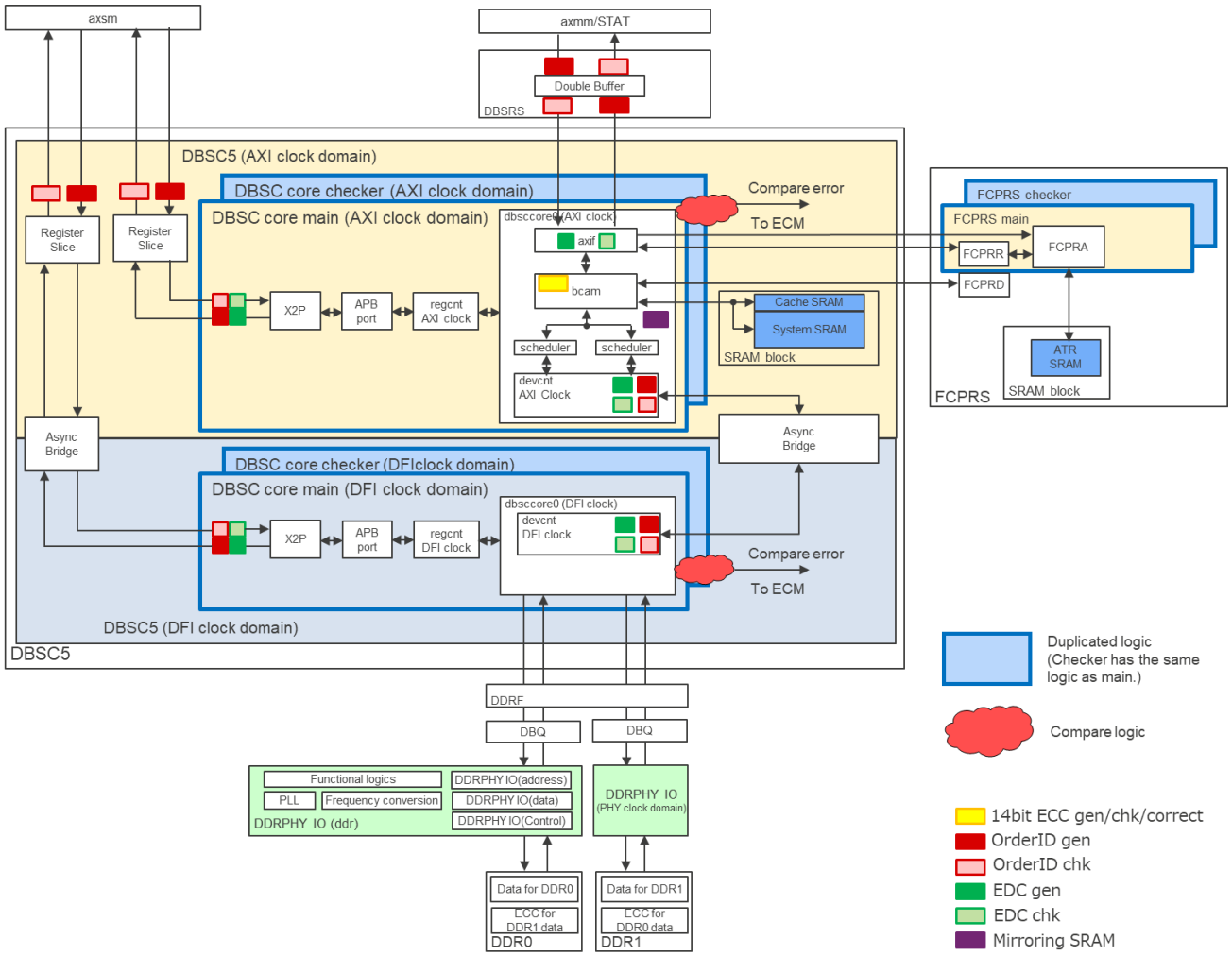



**Figure 33.2 Block diagram of the DCLS protection**

2 cycle delay: Two clock cycle delay for input signals of Checker Core and output signals from Master Core to Compare module is implemented.

**Requirement of DCLS protection**

- All logics except AXI64 access, SRAM and asynchronous bridge are duplicated.
- AXI64 is protected by EDC and other SMs, EDC chk/gen and other SMs are duplicated in DCLS core.
- Following SRAM and all of asynchronous bridge are located outside of DCLS core.
  - Data SRAM
  - 64bit/14bit ECC SRAM
- Compare outputs except following signals of DCLS core between DCLS master instance and DCLS checker instance.
  - AXI512 read data.
  - DFI write data.
  - Data SRAM write data.

- Data of asynchronous bridge.

- Compare logic is separated to some portions to avoid congestion.

- APB access converted from AXI64 is protected by DCLS.

### 33.5.2 ECC for DRAM and System RAM

**(1) Overview**

In DBSC, **ECC protection is applied to protect both SRAM and DRAM access**

**(2) Related Registers**

Table 33.5 Related Registers (ECM Register Mapping)

*n = 0*

| Register Name | Abbreviation | Bit | Error content in ECM |
|---|---|---|---|
| Fusa Interrupt Indication Register 1 (for AXI domain) | DB[*n*]FSINTXXX01A | 5 | Cache RAM duplication error interrupt indication for dbsccore0 |

Table 33.6 Related Registers

*n = 0*

| Register Name | Abbreviation |
|---|---|
| Fusa Interrupt Indication Register 2 (for AXI domain) | DB[*n*]FSINTXXX02A |
| Fusa Interrupt Indication Register 3 (for AXI domain) | DB[*n*]FSINTXXX03A |
| Fusa Interrupt Indication Register 4 (for AXI domain) | DB[*n*]FSINTXXX04A |
| Fusa Interrupt Indication Register 8 (for AXI domain) | DB[*n*]FSINTXXX08A |
| Fusa Interrupt Enable Register 1 (for AXI domain) | DB[*n*]FSINTENB01A |
| Fusa Interrupt Enable Register 2 (for AXI domain) | DB[*n*]FSINTENB02A |
| Fusa Interrupt Enable Register 3 (for AXI domain) | DB[*n*]FSINTENB03A |
| Fusa Interrupt Enable Register 4 (for AXI domain) | DB[*n*]FSINTENB04A |
| Fusa Interrupt Enable Register 8 (for AXI domain) | DB[*n*]FSINTENB08A |
| Fusa Injection Mode Register 2 (for AXI domain) | DB[*n*]FSINJECT02A |
| Fusa Injection Mode Register 3 (for AXI domain) | DB[*n*]FSINJECT03A |
| Fusa Injection Mode Register 4 (for AXI domain) | DB[*n*]FSINJECT04A |
| Fusa Interrupt Counter Register 3 (for AXI domain) | DB[*n*]FSINTCNT3A |
| Fusa Interrupt Counter Register 4 (for AXI domain) | DB[*n*]FSINTCNT04A |
| Fusa AXI Configuration Register0 (for AXI domain) | DB[*n*]FSCONFAXI0 |
| Fusa ECC Injection Check Register (for AXI domain) | DB[*n*]FSECCIJTCHK |

| | |
|---|---|
| Fusa ECC Injection For Address (LSB side) Register (for AXI domain) | DB[*n*]FSECCIJTADRL0 |
| Fusa ECC Injection For Address (MSB side) Register (for AXI domain) | DB[*n*]FSECCIJTADRH0 |
| Fusa ECC Protection Area Setting (for Rank 0) Register0 (for AXI domain) | DB[*n*]FSDRAMECCAREA00 |
| Fusa ECC Protection Area Setting (for Rank 1) Register0 (for AXI domain) | DB[*n*]FSDRAMECCAREA01 |
| Fusa CAM Contorol Register 0 (for AXI domain) | DB[*n*]FSCTRLBCAM0A |
| Fusa ECC Injection Seeting (Low bit side) Register (for AXI domain) | DB[*n*]FSECCIJTERRL |
| Fusa ECC Injection Seeting (Middle bit side) Register (for AXI domain) | DB[*n*]FSECCIJTERRM |
| Fusa ECC Injection Seeting (High bit side) Register (for AXI domain) | DB[*n*]FSECCIJTERRH |
| Fusa ECC Injection For Data (LSB side) Register (for AXI domain) | DB[*n*]FSECCIJTDATL0 |
| Fusa ECC Injection For Data (MSB side) Register (for AXI domain) | DB[*n*]FSECCIJTDATH0 |
| Fusa Status Register 2 (for AXI domain) | DB[*n*]FSSTAT02A |

### (3) Hardware Description

For ECC gen/ check implementation position in DBSC, refer below figure:

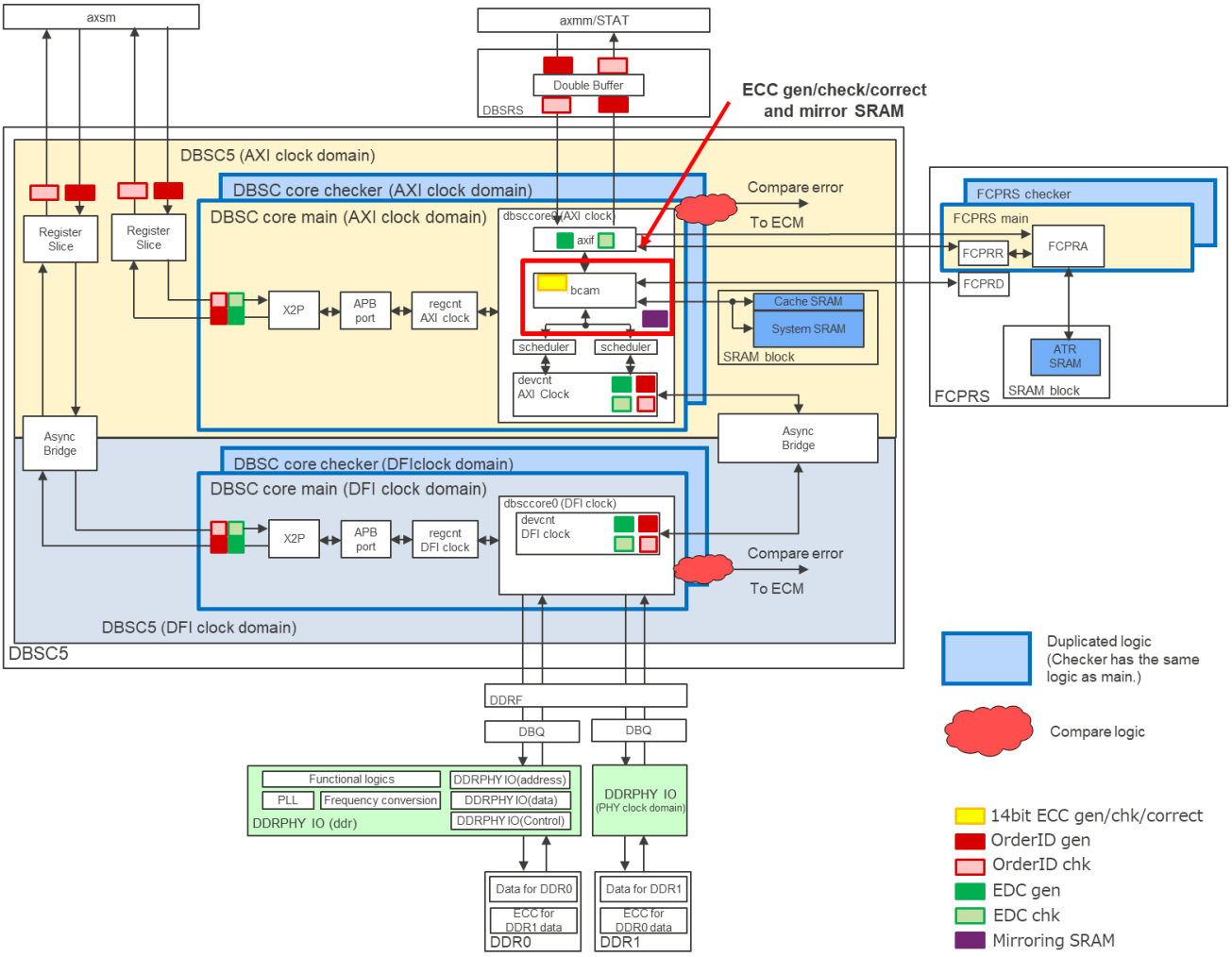



Figure 33.3 ECC protection position

**Policy of 14bit ECC**
DBSC use a single-error correction, double-error detection (SECDED). 14bit check bits has 2^14 = 16384 different possible syndromes, 1 syndrome indicates "no error", 64-bits data +14bit check bits = 78 syndromes indicate "1bit fault"(correction possible) , 16384-78-1 = 16305 syndromes indicate "multi bit fault" 16305/16384 = 99.52% of all multi bit faults are correctly identified. Select ECC (enable correction) mode or EDC (detection only) mode by register. If gen_din is all 0, gen_chkout is not all 0 so in DDR initialization, it can detect an error which all data and check bit are 0.
ECC decoder/encoder are placed into DCLS structure.
ECC protect for both SRAM and DDR.

**ECC gen:**
14bit check bits are generated from 64bit data in imb (internal master bus) write phase.
Some input data (din[3:2]) are inverted to avoid outputting all 0 check bit when input data are all 0.

**ECC chk:**
Check fault in 64bits data and 14bit check bits in imb read phase.
Use ECC mode (DB[N]FSCONFAXI[0] (errcrctdis) = 0). (errcrctdis bit can change to EDC mode (errcrctdis = 1)).
Use ECC gen because a check bit is recomputed from data.
ECC chk outputs 2 error signals (err_detect, err_multpl) to Error Injection chk.
Channel of Data & channel of ECC/EDC are different.
ECC protection is enabled by register setting. Default setting is disable

**Policy of Error injection for ECC chk:**
Inject before ECC chk because of high BER (Bit Error Rate). There are FFs which store wrong DATA/check bit pair. CPU writes 64bit data to a register in regcnt, and ECC gen generates 14bit check bits. 78bit DATA/check bit corruption register can select any bits (but no more than 2 bits) to be inverted. Read request to specific address reads the wrong DATA/check bit pair. Normal address and specific address paths are divided by specific address register ECC chk can detect the injected error. After that, it outputs err_detect and err_multpl to Error injection check. Address compare receives read address and compares with specific address. If the read address matches specific address, Address compare informs Error injection chk that this error is injected error.

**Protection policy**
*16bit x 2ch DDR mode*
All entries of SystemRAM are protected by 64bit/14bit ECC
Specied area of DRAM (set by register) is protected by 64bit/14bit ECC
Data of partial write to DRAM (data write is not full 64bit unit) is protected by data mirroring until ECC is regenerated.
When parital write access to SystemRAM (data write is not full 64bit unit) comes, ram_mw execute Read-Modify-Write.

**ECC address generation**

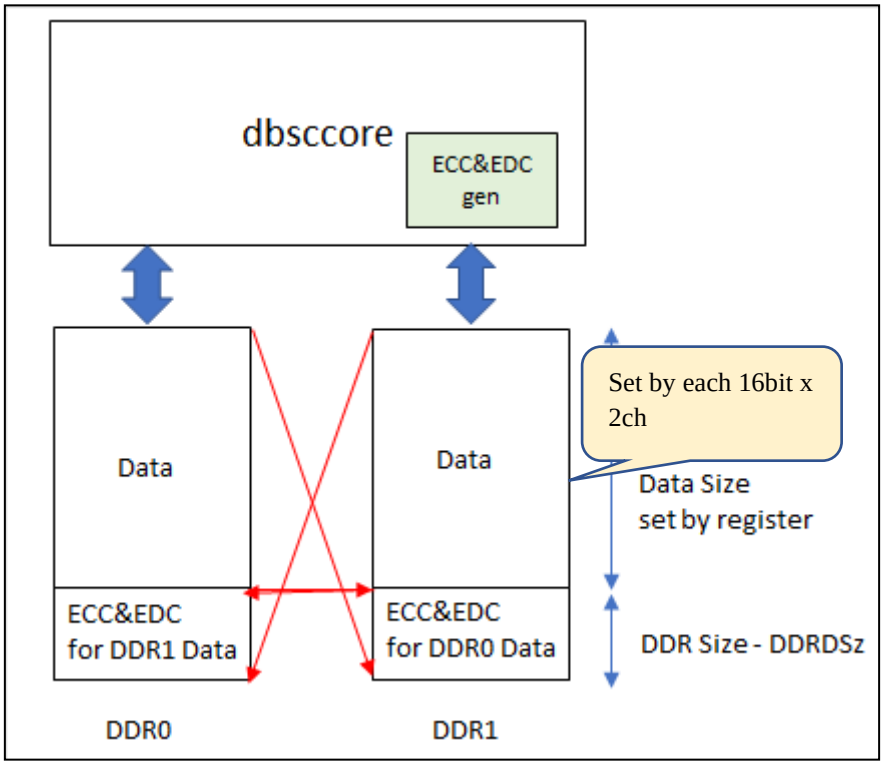



Figure 33.4 Image of DATA and ECC storage

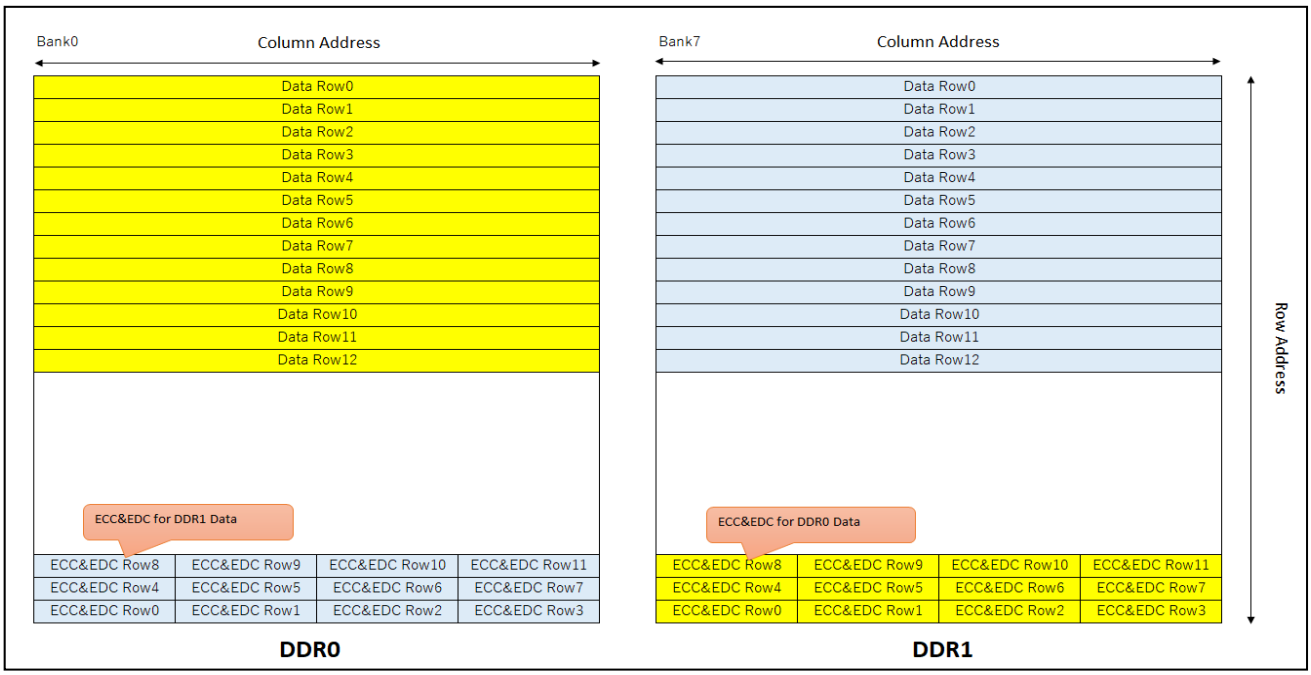



**Figure 33.5** Image of ECC/EDC mapping

Bank of Data & bank of ECC are same.

Start row address of ECC is end of DDR row address.

In 1 row address, there are ECC for data of 4 row address.

Data size of target of ECC can bet set by register.

During ECC protection is disabled by register, can access to all DRAM area.

Cache data and ECC packet generated at AXI write phase are written together to DDR.

Refer to sheet "DRAM ADDRESS MAP" of 033_DRAM_address_map_r0p70.xlsx for DRAM address map(DATA/ECC).

### 33.5.3 EDC/OrderID protection for register access

#### (1) Overview

In DBSC, register access use EDC and OrderID to protect error in signals and prevent from data duplication or lost as below figure:

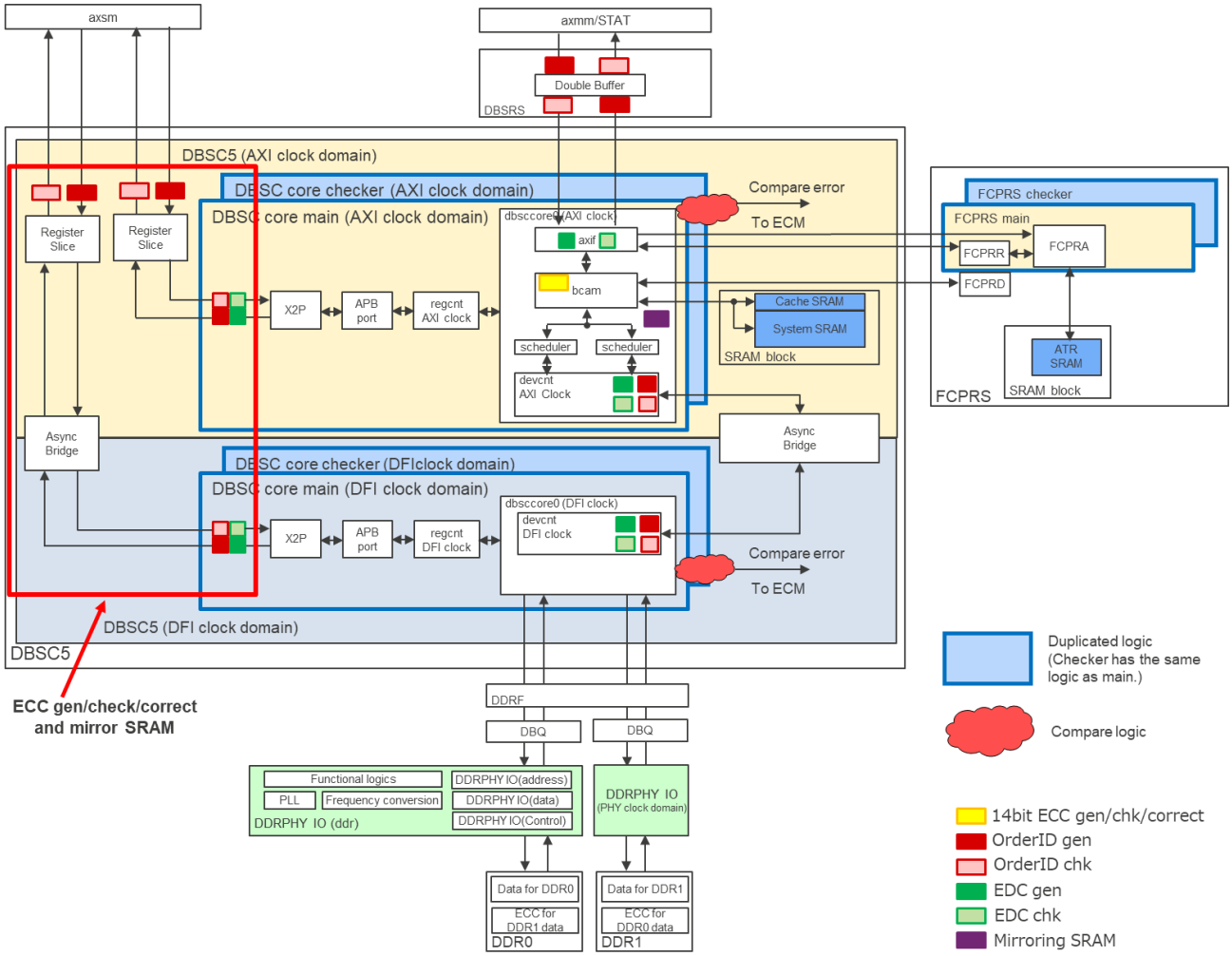



**Figure 33.6** EDC/OrderID register access protection

#### (2) Related Registers

**Table 33.7 Related Registers (ECM Register Mapping)**

*n = 0*

| Register Name | Abbreviation | Bit | Error content in ECM |
|---|---|---|---|
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 10 | OrderID error in AXI64 Bch clk_dbs side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 9 | OrderID error in AXI64 Rch clk_dbs side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 8 | OrderID error in AXI64 Bch clk_axi side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 7 | OrderID error in AXI64 Rch clk_axi side Interrupt indication |

| | | | |
|---|---|---|---|
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 6 | OrderID error in AXI64 Wch clk_axi side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 5 | OrderID error in AXI64 AWch clk_axi side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 4 | OrderID error in AXI64 ARch clk_axi side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 2 | EDC error in AXSM W Ch clk_axi domain Interrupt enable bit |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 1 | EDC error in AXSM AW Ch clk_axi domain Interrupt enable bit |
| Fusa Interrupt Indication Register 0 (for AXI domain) | DB[*n*]FSINTXXX00A | 0 | EDC error in AXSM AR Ch clk_axi domain Interrupt enable bit. |
| Fusa Interrupt Indication Register 0 (for DFI domain) | DB[*n*]FSINTXXX00D | 6 | OrderID error in AXI64 Wch clk_axi side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for DFI domain) | DB[*n*]FSINTXXX00D | 5 | OrderID error in AXI64 AWch clk_axi side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for DFI domain) | DB[*n*]FSINTXXX00D | 4 | OrderID error in AXI64 ARch clk_axi side Interrupt indication |
| Fusa Interrupt Indication Register 0 (for DFI domain) | DB[*n*]FSINTXXX00D | 2 | EDC error in AXSM W Ch clk_axi domain Interrupt indication |
| Fusa Interrupt Indication Register 0 (for DFI domain) | DB[*n*]FSINTXXX00D | 1 | EDC error in AXSM AW Ch clk_axi domain Interrupt indication |
| Fusa Interrupt Indication Register 0 (for DFI domain) | DB[*n*]FSINTXXX00D | 0 | EDC error in AXSM AR Ch clk_axi domain Interrupt indication |

**Table 33.8 Related Registers**

*n = 0*

| Register Name | Abbreviation |
|---|---|
| Fusa Interrupt Enable Register 0 (for AXI domain) | DB[*n*]FSINTENB00A |
| Fusa Injection Mode Register 0 (for AXI domain) | DB[*n*]FSINJECT00A |
| Fusa Interrupt Counter Register 0 (for AXI domain) | DB[*n*]FSINTCNT0A |
| Fusa Interrupt Enable Register 0 (for DFI domain) | DB[*n*]FSINTENB00D |
| Fusa Injection Mode Register 0 (for DFI domain) | DB[*n*]FSINJECT00D |
| Fusa Interrupt Counter Register 0 (for DFI domain) | DB[*n*]FSINTCNT00D |

**(3) Hardware Description**

In AXI I/F between axsm and dbsc, it has EDC code on AxUSER and WUSER for EDC check and EDC code on RUSER for EDC generate. EDC check/gen are located into DCLS core

OrderID is used for register access to detect request packet lost and duplication. OrderID counter is increased when AR, AW or W ch packet is VALID & READY. Add OrderID counter value between axsm and Register slice to packet's attribute. When counter value of OrderID counter in DCLS core, and packet's attribute are different, it will report error.

OrderID is used for register access to detect response packet lost and duplication. OrderID counter is prepared on between DCLS port and AXI to APB Bridge in DCLS core, and on between axsm and Register slice. OrderID counter is increased when R or B ch packet is VALID & READY. OrderID counter value is added between DCLS port and AXI to APB Bridge to packet's attribute. Add 1 bit to MSB of packet's attribute for identification of clock domain. When counter value of OrderID counter between axsm and Register slice, and packet's attribute are different, it will report error.

There is error injection to inject an intentional error to each error flag to regcnt.

### 33.5.4 Protection between axmm & dbsc

**(1) Overview**

Interface between axmm and DBSC5 is protected by EDC gen/check and address check.

Double buffer of DBSRS module located between axmm and DBSC5 is protected by Order ID.

**(2) Related Registers**

**Table 33.9 Related Registers**

*n = 0*

| Register Name | Abbreviation |
|---|---|
| Fusa Interrupt Indication Register 1 (for AXI domain) | DB[*n*]FSINTXXX01A |
| Fusa Interrupt Enable Register 1 (for AXI domain) | DB[*n*]FSINTENB01A |
| Fusa Injection Mode Register 1 (for AXI domain) | DB[*n*]FSINJECT01A |
| Fusa Interrupt Counter Register 1 (for AXI domain) | DB[*n*]FSINTCNT1A |
| Fusa Monitor EDC Error SrcID Register 0 (for AXI domain) | DB[*n*]FSMNEDCSID0A |
| Fusa Monitor Nomem Error Address (LSB side) Register 0 (for AXI domain) | DB[*n*]FSMNDEA0LA |
| Fusa Monitor Nomem Error Address (MSB side) Register 0 (for AXI domain) | DB[*n*]FSMNDEA0HA |

### (3) Hardware Description

All EDC gen check for axmm and dbsc interface are located inside DCLS as below figure.

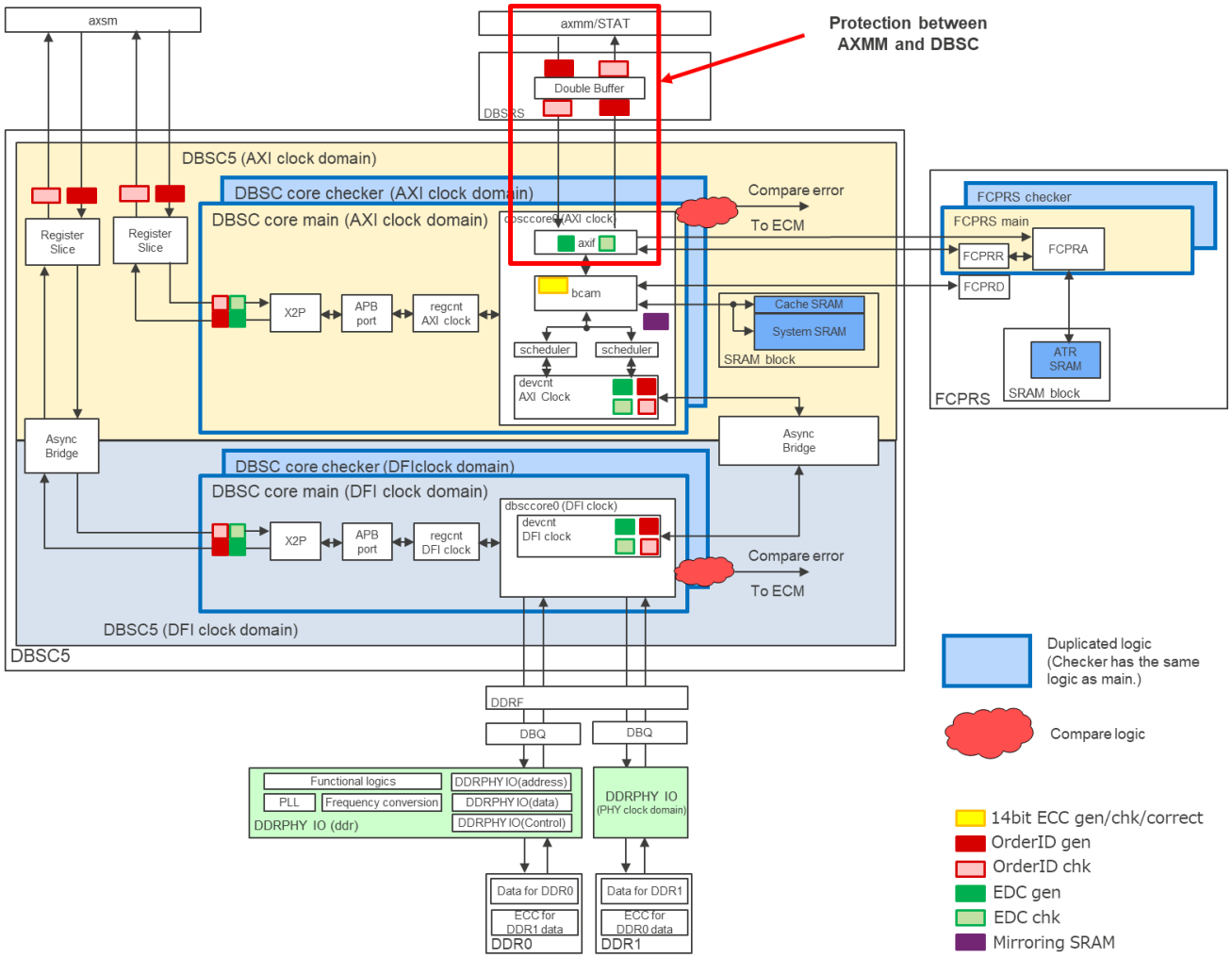



**Figure 33.7 EDC protection between DBSC and AXMM**

*Operation:*

EDC gen/check:

For request (address / write data), check EDC code generated by hierarchical bus in axiif module. For read response, EDC code is generated in axiif module, and added to AXI RUSER field. There is error injection to inject an intentional error flag to regcnt.

Address check:

Check whether the requested address is within valid space or not. If address is not in valid space, it will output error and set error status to register. Error address will be stored in register for first address error.

Burst ID gen/chk:

For read response, Burst ID is added to AXI RUSER field. For write data, check Burst ID to detect lost/interleave error in a burst.

Source ID is stored in a register for the first error.

There is error injection to inject an intentional error flag to regcnt.

### 33.5.5 Asynchronous protection

**(1) Overview**

Bus protocol of asynchronous bridge of DBSC are protected by EDC and OrderID. Other signals are protected by specific method such as inversion duplication, operation checks.

**(2) Related Registers**

Table 33.10 Related Registers

*n = 0*

| Register Name | Abbreviation | Bit | Error content in ECM |
|---|---|---|---|
| Fusa Interrupt Indication Register 9 (for AXI domain) | DB[*n*]FSINTXXX09A | 9 | OrderID error of isbus async read ch. interrupt indication for memory channel 1 |
| Fusa Interrupt Indication Register 9 (for AXI domain) | DB[*n*]FSINTXXX09A | 8 | EDC error of isbus async read ch. interrupt indication for memory channel 1 |
| Fusa Interrupt Indication Register 9 (for AXI domain) | DB[*n*]FSINTXXX09A | 1 | OrderID error of isbus async read ch. interrupt indication for memory channel 0 |
| Fusa Interrupt Indication Register 9 (for AXI domain) | DB[*n*]FSINTXXX09A | 0 | EDC error of isbus async read ch. interrupt indication for memory channel 0 |
| Fusa Interrupt Indication Register10 (for AXI domain) | DB[*n*]FSINTXXX10A | 15:5 | Error status of injected errors (just reflect config of registers) |
| Fusa Interrupt Indication Register10 (for AXI domain) | DB[*n*]FSINTXXX10A | 4 | Error status of synchronizing the write leveling acknowledge from DBSC to AXI domain |
| Fusa Interrupt Indication Register10 (for AXI domain) | DB[*n*]FSINTXXX10A | 3 | Error status of synchronizing the acknowledge for external refresh request from DBSC to AXI domain |
| Fusa Interrupt Indication Register10 (for AXI domain) | DB[*n*]FSINTXXX10A | 2 | Error status of synchronizing the data scramble seed to AXI domain |
| Fusa Interrupt Indication Register10 (for AXI domain)) | DB[*n*]FSINTXXX10A | 1 | Error status of synchronizing the devcnt empty info to AXI domain |
| Fusa Interrupt Indication Register10 (for AXI domain) | DB[*n*]FSINTXXX10A | 0 | Error status of synchronizing to AXI domain of info from devcnt to schcnt |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 25 | OrderID error of isbus async write ch. interrupt indication for memory channel 1 |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 24 | OrderID error of isbus async address (command queue) ch. interrupt indication for memory channel 1 |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 23 | EDC error of isbus async write ch. interrupt indication for memory channel 1 |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 22 | EDC error of isbus async address (command queue) ch. interrupt indication for memory channel 1 |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 9 | OrderID error of isbus async write ch. interrupt indication for memory channel 0 |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 8 | OrderID error of isbus async address (command queue) ch. interrupt indication for memory channel 0 |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 7 | EDC error of isbus async write ch. interrupt indication for memory channel 0 |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D | 6 | EDC error of isbus async address (command queue) ch. interrupt indication for memory channel 0 |
| Fusa Interrupt Indication Register 3 (for DFI domain) | DB[*n*]FSINTXXX03D | 15:2 | Error status for injected error (just reflect config of registers) |
| Fusa Interrupt Indication Register 3 (for DFI domain) | DB[*n*]FSINTXXX03D | 1 | Error status of synchronizing the request queue empty info to DFI domain |
| Fusa Interrupt Indication Register 3(for DFI domain) | DB[*n*]FSINTXXX03D | 0 | Error status of synchronizing the scheduler busy status info to DFI domain |

**Table 33.11 Related Registers**

*n = 0*

| Register Name | Abbreviation |
|---|---|
| Fusa Interrupt Indication Register 9 (for AXI domain) | DB[*n*]FSINTXXX09A |
| Fusa Interrupt Indication Register 10 (for AXI domain) | DB[*n*]FSINTXXX10A |
| Fusa Interrupt Enable Register 9 (for AXI domain) | DB[*n*]FSINTENB09A |
| Fusa Interrupt Enable Register 10 (for AXI domain) | DB[*n*]FSINTENB10A |
| Fusa Injection Mode Register 9 (for AXI domain) | DB[*n*]FSINJECT09A |
| Fusa Injection Mode Register 10 (for AXI domain) | DB[*n*]FSINJECT10A |
| Fusa Interrupt Counter Register 7 (for AXI domain) | DB[*n*]FSINTCNT07A |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D |
| Fusa Interrupt Indication Register 3 (for DFI domain) | DB[*n*]FSINTXXX03D |
| Fusa Interrupt Enable Register 1 (for DFI domain) | DB[*n*]FSINTENB01D |
| Fusa Interrupt Enable Register 3 (for DFI domain) | DB[*n*]FSINTENB03D |
| Fusa Injection Mode Register 1 (for DFI domain) | DB[*n*]FSINJECT01D |
| Fusa Injection Mode Register 3 (for DFI domain) | DB[*n*]FSINJECT03D |
| Fusa Interrupt Counter Register 1 (for DFI domain) | DB[*n*]FSINTCNT01D |
| Fusa Interrupt Counter Register 3 (for DFI domain) | DB[*n*]FSINTCNT03D |

### (3) Hardware Description

Below figure show position of EDC/OrderID for asynchronous bridge in DBS.

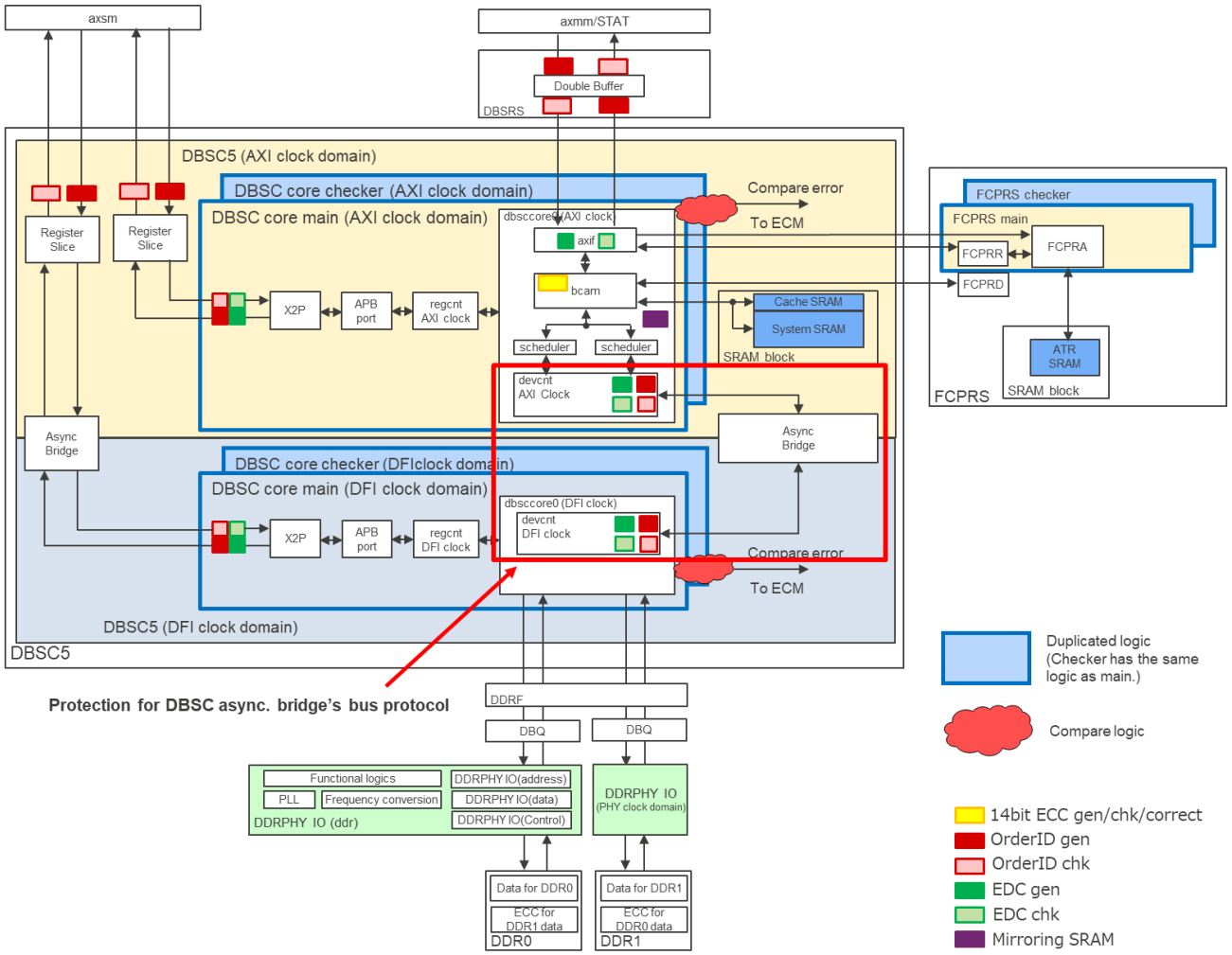



**Figure 33.8 EDC/OrderID protection for DBSC's asynchronous bridge**

**For bus protcol:**
EDC checker /generator is in DCLS core. All address and data are protected by EDC.
OrderID counter will detect packet lost and duplication. On before and after of asynchronous bridge, prepare OrderID counter. OrderID counter of before asynchronous bridge is increased when asynchronous bridge receives the packet. OrderID counter of after asynchronous bridge is increased when asynchronous bridge sends the packet. OrderID counter value of before asynchronous bridge is added to packet's attribute. When OrderID counter value of after asynchronous bridge and packet's attribute are different, report error. OrderID counter and checker are located into DCLS core.

**For other sideband signals:**

sy_ref_req & sy_wtraining protection

- When input request is high, but ack is low over 63 cycles, it will issue error. From request de-assertion, if ack is still high over 63 cycles, it will issue error. There is ack assertion but there is no request assertion before, it will issue error

Policy of devcnt status info to schcnt asynchronous path protection

- Protect by inversion duplication signal, and by OrderID

Policy of asdb_dv_empty path protection

- Protect by inversion duplication signal. The empty signal is asserted when the synced dv_empty asserted from 8 cycles or above. And, if empty signal generated from inversion signals and empty signal generated from normal signal are not equal during 16cycles, it will output error.

Policy of sy_dbscrm path protection

- Prepare status monitor register for sy_dbscrm (after async), user read the register and check read data by software. During accen=1, check sy_dbscrm is not change.

Policy of scxx_busy (busy signal of scheduler) path protection

- Signal duplication with inversion. Output signal is asserted when synced signal assert 8cycles or above. If signal generated from inversion signal is mismatch over 16 cycles with output signal, there will be error.

Policy of dvxx_rq_empty path protection

- Signal duplication with inversion. If generated signal from normal signal or inversion signal don’t change during 64 cycles, and mismatch signal generated from normal signal and signal generated from inversion signals between 16cylcles and 32cycles ago, it will output error.

Policy of dvxx_stop_valid, dvxx_stop_rq_ack path protection

- Signal duplication with inversion and FSM, if async state from normal signal and async state from inversion signal is not matched after 31 cycles, it will output error.

Policy of dvxx_stop_cs, dvxx_stop_bg, dvxx_stop_ba, dvxx_stop_pb path protection

- Signal duplication with inversion.

Policy of other asynchronous path protection: No protection. Target signals type is as follows.

- Power down request from CPG
- Global monitor acknowledgement from CPU
- Performance monitor control from STAT

### 33.5.6 Self-test for Async, DBP, Hier-boundary

#### (1) Overview

In order to test data before DBSC initialization, ensure data is correct when data go through asynchronous bridges: It is in devcnt. There are 2 modules to protect data: POST gen (placed before asynchronous bridge) and POST check module (placed after asynchronous bridge). POST gen generates 2 packets in each clock cycle: n'hAAAA and n'h5555. POST check will checks that all bits are not tied to 0 or 1.

#### (2) Related Registers

**Table 33.12 Related Registers**

*n = 0*

| Register Name | Abbreviation |
|---|---|
| Fusa Interrupt Indication Register 1 (for AXI domain) | DB[*n*]FSINTXXX01A |
| Fusa Interrupt Enable Register 1 (for AXI domain) | DB[*n*]FSINTENB01A |
| Fusa Injection Mode Register 1 (for AXI domain) | DB[*n*]FSINJECT01A |
| Fusa AXI Contorol Register 0 (for AXI domain) | DB[*n*]FSCTRLAXI0 |
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D |
| Fusa Interrupt Enable Register 1 (for DFI domain) | DB[*n*]FSINTENB01D |
| Fusa Injection Mode Register 1 (for DFI domain) | DB[*n*]FSINJECT01D |
| Fusa Control Register 0 (for DFI domain) | DB[*n*]FSCTRLDBS0 |

### (3) Hardware Description

***Block diagram:***

Structure of asynchronous bridge in devcnt: Protect data for ach, wch, rch at devcnt module.

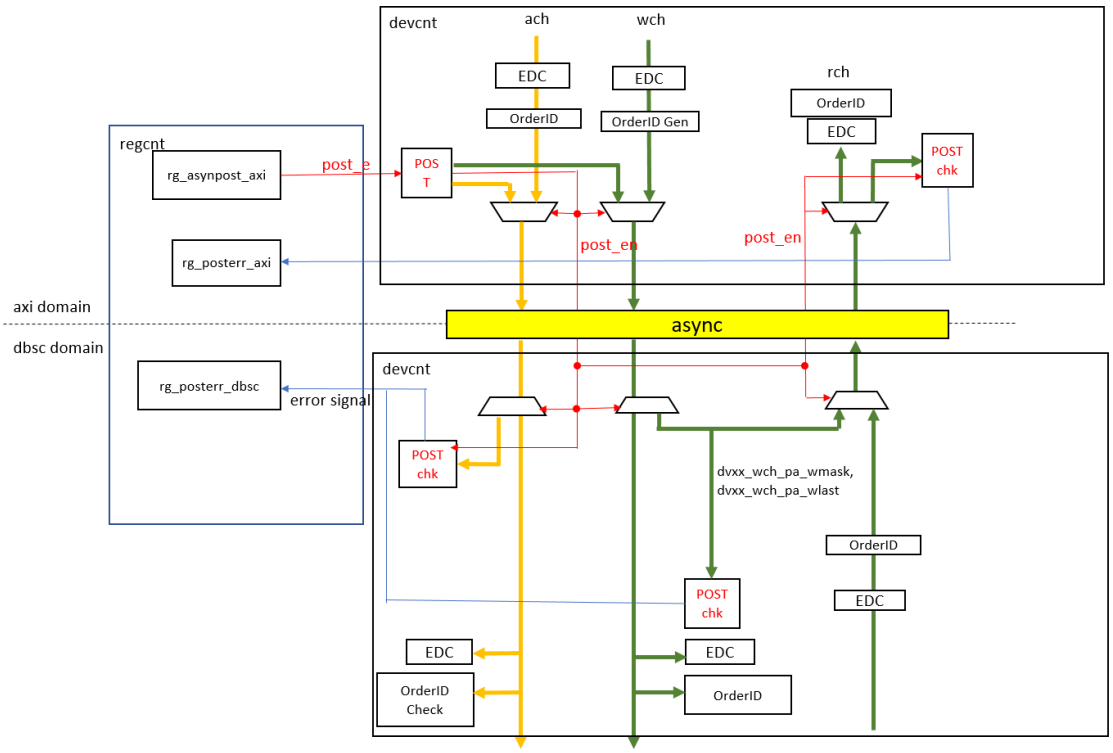



**Figure 33.9 POST check for asynchronous bridge in devcnt**

**Operation:**

Self-test is started by setting register (dbsc register). Timing of self-test sequence is 2 cycles continually. In self-test sequence, data between post_gen module and post_check module will be replaced by self-test data (n'hAAAA and n'h5555), data after post_check module will be mask zero during self-test sequence (Some DFI signals are needed to keep value: cke, cs, reset, wrdata_cs, rddata_cs, parity data, dll_rst_n).

Support error injection register: Inject an intentional error to each error flag by register

Self-test must be started before DBSC initialization to avoid pseudo error related to DFI protocol and another FUSA error (EDC, Parity Check). Please refer to below self-test sequence

1. Disable protection enable (EDC) and parity enable
2. Start Self-Test sequence
3. Confirm end of Self-Test sequence
4. Start DBSC initialization
5. Assert protection enable and parity enable

### 33.5.7 Parity check and generation for DDR-PHY register interface

**(1) Overview**

Generate parity (odd) for write data and register selection on DDR-PHY register interface in DBSC

Check parity (odd) for read data on DDR-PHY register interface and issue interrupt as fusa support in DBSC

**(2) Related Registers**

**Table 33.13 Related Registers**

*n = 0*

| Register Name | Abbreviation |
|---|---|
| Fusa Interrupt Indication Register 1 (for DFI domain) | DB[*n*]FSINTXXX01D |
| Fusa Interrupt Enable Register 1 (for DFI domain) | DB[*n*]FSINTENB01D |
| Fusa Injection Mode Register 1 (for DFI domain) | DB[*n*]FSINJECT01D |
| Fusa Interrupt Counter Register 2 (for DFI domain) | DB[*n*]FSINTCNT02D |
| Fusa Interrupt Counter Register 4 (for DFI domain) | DB[*n*]FSINTCNT04D |

**(3) Hardware Description**

**Block diagram**

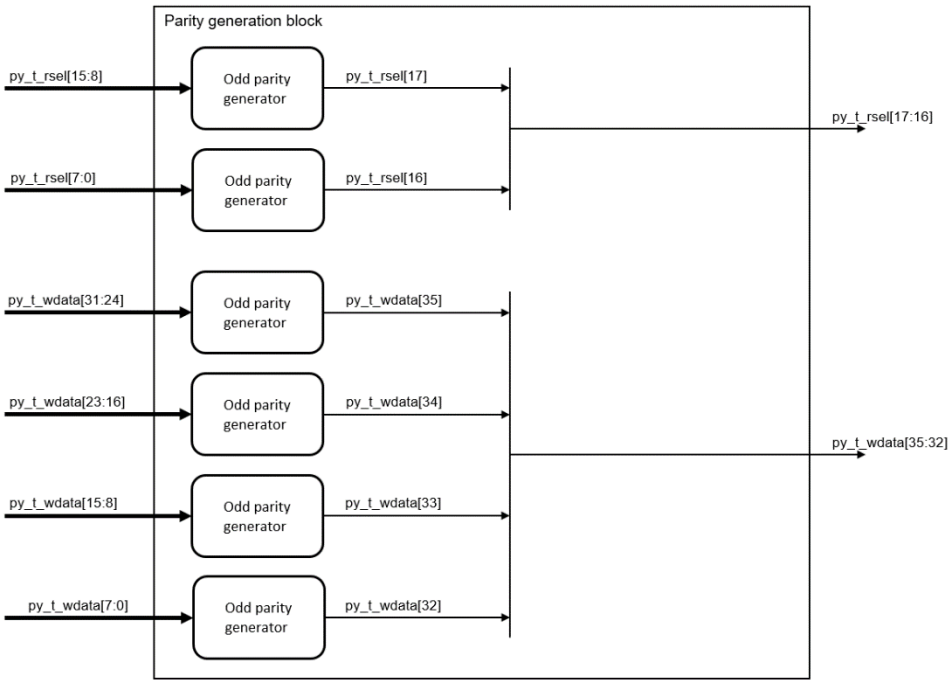



Figure 33.10 Parity generation block

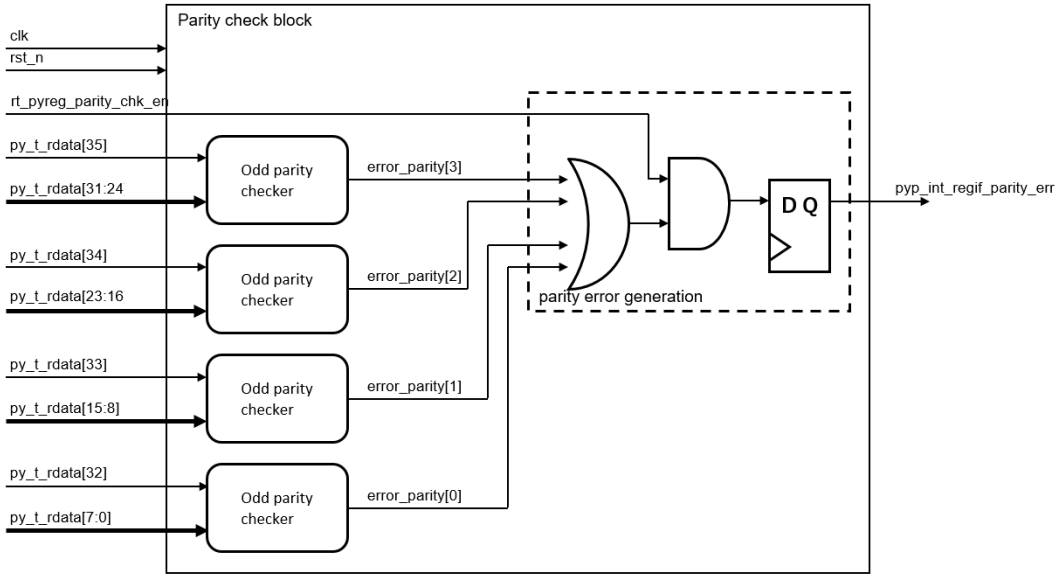



Figure 33.11 Parity check block

**Operation**

Parity check is started by setting register (dbsc register, active high) and run during DBSC operation. Parity generate module will generate parity (odd) for write data and register selection on DDR-PHY register interface in DBSC4. Check module will check parity (odd) for read data on DDR-PHY register interface and issue interrupt as fusa support in DBSC.

Support error injection register: Inject an intentional error to each error flag by register.

Parity check must run after self-test to avoid pseudo error. Please refer to 10.5.6 for detail information.

# 34. System RAM

## 34.1 Overview

R-Car series 4th generation products incorporate System RAM.

### 34.1.1 Features

• Memory size: 1024KBytes.

• The data of System RAM is undefined on power-on reset and retained on other resets.

• System RAM takes charge of buffer between CPUs.

### 34.1.2 Block Diagram

System RAM is embedded in DBSC5.

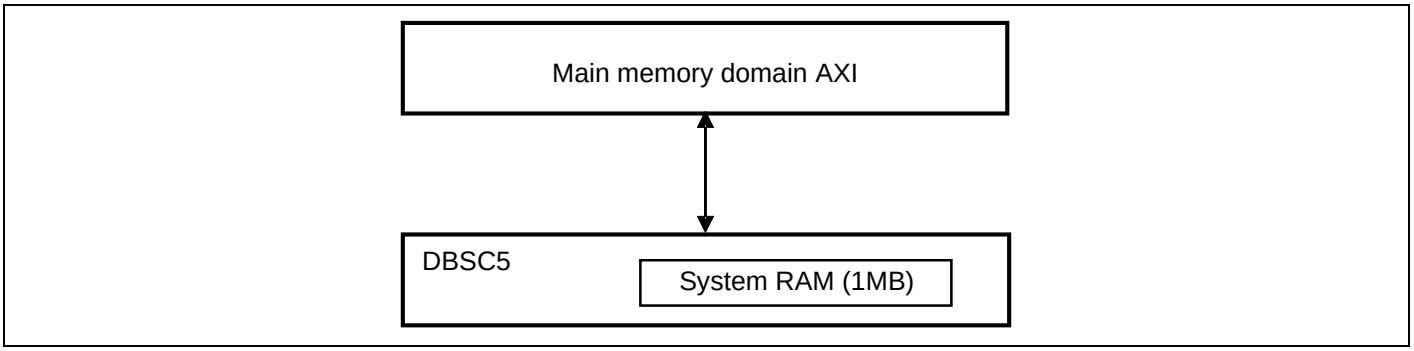



Figure 34.1 Block Diagram

### 34.1.3 Register Configuration

System RAM does not have the register.

## 34.2 Register Description

System RAM does not have the register.

## 34.3 Operation

No description on this section.

## 34.4 Usage Notes

Address of System RAM is shown in the table below.

Table 34.1 Address of System RAM

| Address | Notice |
|---|---|
| H'E630 0000 to H'E63F FFFF | - |

Note: Do not write to any addresses other than listed above. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed above are undefined.

## 34.5 Safety Mechanisms

Refer to section 33.5 for safety mechanisms of System RAM.

# 36. RT-VRAM

RT-VRAM is 1M byte RAM in compatible mode and 28M byte cache in extended mode on Real Time domain. There is an instruction area of Cortex-R52 or ICUMX, and communication area.

## 36.1 Overview

### 36.1.1 Features

- Memory size: 1M bytes
- Number of modules: 2 modules (total 2M bytes)
- Memory initialization function: all memory value set to zero.
- Safety Region (up to 16 region) protect.
- Access protection for secure regions.
- Support exclusive access: 16 exclusive access monitors
- DCLS support.
- RAM EDC (single-bit, multi-bit) support.
- Cache size: Maximum 28M bytes (4/8/12/16/20/24/28MB selectable)
- Cache 4-WAY/8-WAY configurable
- 32-bit CRC protection per 256-byte boundary unit.
- DDR-RAM mapping configurable
- Data Scramble support

### 36.1.2 Block diagram

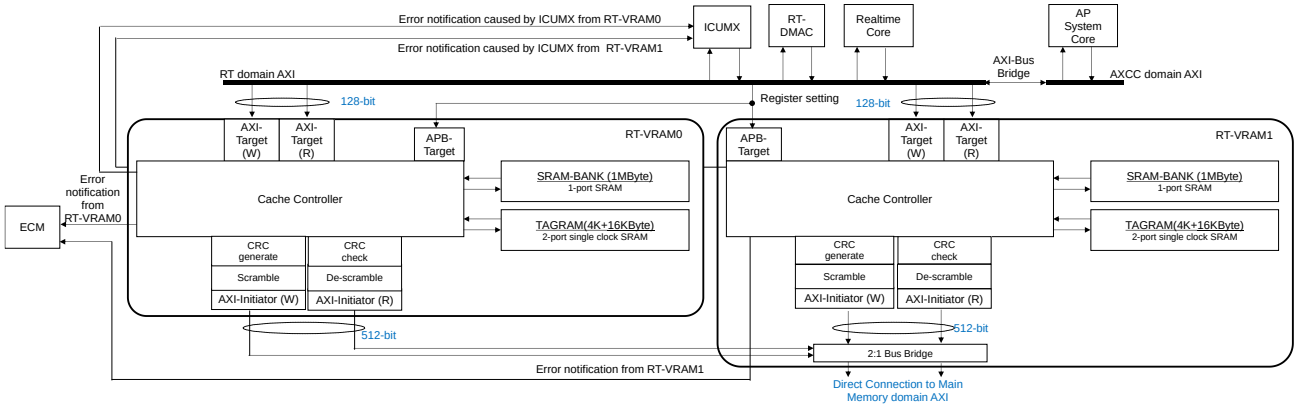



**Figure 36.1 Block diagram**

### 36.1.3 External Pins

The RT-VRAM does not have any external pins.

### 36.1.4 Connected Module

Table 36.1 shows the connected modules to RT-VRAM. As for the feature of the connected modules, refer to the chapter of each module.

Table 36.1 Connected Modules

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| RT-VRAM | AXI-BUS | AXI-Bus, APB-Bus |
| | CPG | Clock Supply<br>SRAM Initialization control |
| | RST | Module Reset |
| | ECM | Error Interrupt Controller |
| | ICUMX | Hardware security module |

### 36.1.5 Register configuration

Refer to R-CarV4M_UM_036_RTVRAM_r0p81.xlsx.
Note: Register address spaces of FFE9_E000 to FFE9_FFFF and FFEC_E000 to FFEC_FFFF are reserved area.

## 36.2 Register description

Refer to R-CarV4M_UM_036_RTVRAM_r0p81.xlsx.

## 36.3 Operation

### 36.3.1 Address of RT-VRAM

Address of RT-VRAMs are below:

RT-VRAM0: 0xE000_0000 to 0xE1BF_FFFF (up to 28MB)

RT-VRAM1: 0xE200_0000 to 0xE3BF_FFFF (up to 28MB)

Note:

Address space is depending on RT-VRAM mode. Compatible mode supports 1MB from source address. For Extended mode these areas are extended up to 28MB area.

### 36.3.2 Compatible mode/Extended mode

#### (1) Compatible mode

Compatible mode can use 1MB of SRAM area (Default mode), All functions of conventional are available in this mode.

RT-VRAM doesn't issue any AXI transaction to DRAM.

Only in RT-VRAM0, 1MB of SRAM area (0xE000_0000 to 0xE00F_FFFF) are mirrored other area (0xEB20_0000 to 0xEB2F_FFFF).

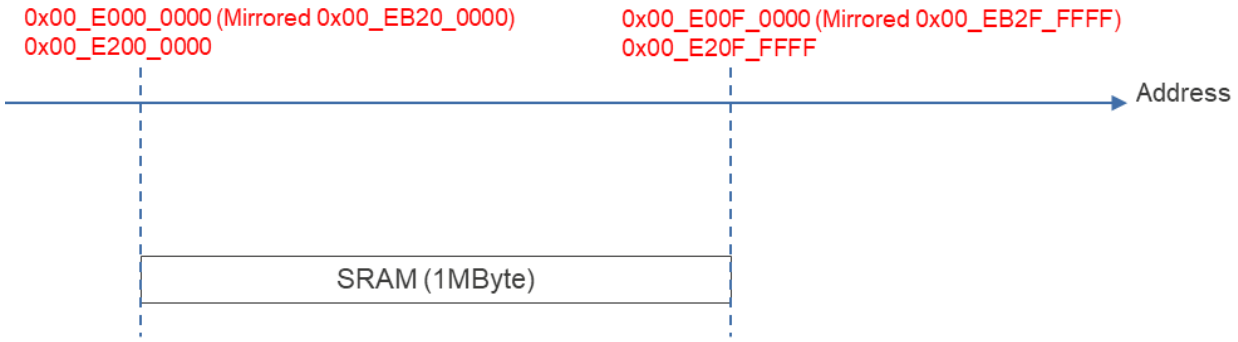



Figure 36.2 Address area of RT-VRAM in Compatible mode

### (2) Extended mode

Buffer size is extended up to 28 MB (4/8/12/16/20/24/28MB selectable). It is a virtual buffer.

Actual data is allocated on DRAM, and it is protected by 32-bit CRC at 256-byte boundary unit.

The first 1MB area can mix SRAM and the virtual buffer, but this area doesn't support cache flush function. In details, please refer Section 36.3.4. *

RT-VRAM acts as a cache to access data on the virtual buffer.

The size of SRAM area is configurable. It depends on the size of the virtual buffer and the number of way of data cache. If IP access the out of the virtual buffer, RT-VRAM returns SLVERR to IP.

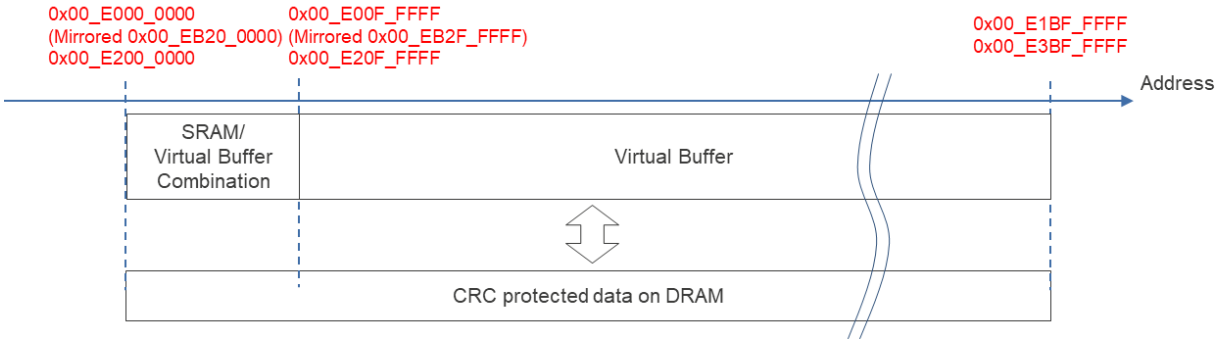



**Figure 36.3 Address area of RT-VRAM in Extended mode**

Cache in RT-VRAM has the following functions:

- Set Associative, 4 or 8-way configurable, 256 bytes line, 256 sets.
- Write policy: write back to DRAM when the line is modified and pushed out by replacement operation
- Replacement policy: Pseudo Least Recently Used (pLRU)
- There are 256 valid bits for each tag, and all bits become 0 when switching to extended mode. If the bit is 0, the cache uses the default value for tag instead of reading from TAGRAM. When the tag is updated, the corresponding bit is set to 1.
- Forcibly writes back of cache data to DRAM can be performed.

Note: * The size of SRAM area and Virtual buffer area in the first 1MB area in Extended Mode are decided as Figure 36.4. The size of the virtual buffer and the number of way of data cache are set by VBUF_CFG register.

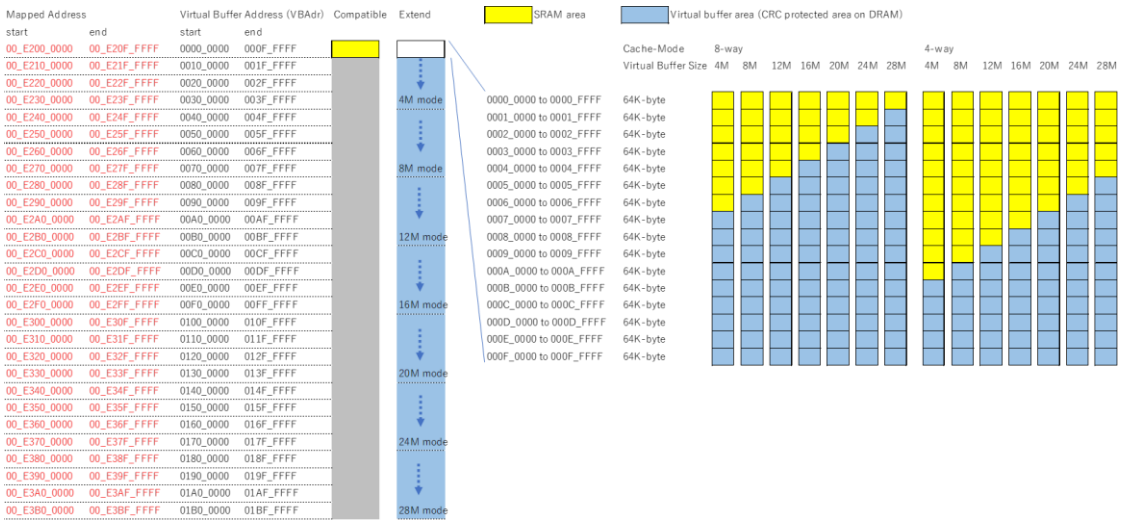



**Figure 36.4 The size of SRAM area and Virtual buffer area in the first 1MB area in Extended Mode**

### 36.3.3 Switching compatible mode to Extended Mode

#### (1) Procedure of switching Compatible Mode to Extended Mode

1. Wait memory initialization has completed (MEM_INIT.INIT)
2. Set VBUF_BADDRn register (Base address of each 4M-byte area of virtual buffer[39:16], n = 0 to 6)
3. Set VBUF_CFG register
4. Set 1’b1 to MODE bit in EXT_MODE register

#### (2) Restrictions in Extended Mode

Switch Compatible Mode to Extended Mode by setting to Extended Mode register EXT_MODE. It cannot switch back.

If the Extended Mode register is set to Extended Mode, all valid bits of TAG are set to 0 only.

The data continues to stay in SRAM-BANK and will be transferred to DRAM by write back processing of cache.

It is prohibited to set duplicated each 4MB area by BASE_ADDRn[23:0] bits in each VBUF_BADDRn register.

### 36.3.4 Cache Flush Function

RT-VRAM supports cache flush function.

The flush target area is one of 16 divided areas specified by access policy registers.

The target area is configured by register, and different areas can be flushed sequentially.

Note: it is prohibited to flush the area which contains first 1 MB (40’h00_E000_0000 to 40’h00_E00F_FFFF, 40’h00_E200_0000 to 40’h00_E20F_FFFF).

The flush operation is executed only when the access authority of the IP that issued the flush instruction matches the read permission of the area to be flushed.

If the execution is rejected by access policy, it is notified as PSVERR of the flush operation register access.

Cache data remains valid after flush operation.

Any IP cannot access to the flush target area during flush operation.

When an IP accesses, the access is rejected and notified as SLVERR.

The completion of the flush process is indicated by the status register.

Note that cache flush is available when extended mode (EXT_MODE.MODE = 1). If the register is set to perform the flush in compatible mode (EXT_MODE.MODE = 0), it is canceled and returns PSLVERR.

### 36.3.5 Software reset for RT-VRAM

RT-VRAM starts the initialization process of SRAM-BANK and TAGRAM both immediately after reset.

RT-VRAM has MEM_INIT register to execute the initialization process, but it is available only when Compatible Mode (EXT_MODE.MODE = 0). If the register is set to perform the initialization in the extended mode (EXT_MODE.MODE = 1), the initialization is canceled and return PSLVERR.

There is basically no use for MEM_INIT register because it breaks all processing that uses RT-VRAM. Above function is just for software compatibility.

RT-VRAM provide a reset sequence by software as RT-VRAM. To write 1 to INIT bit of MEM_INIT register, RT-VRAM can set all RAM data to zero.

INIT bit in Memory initialization Register (MEM_INIT) is set to "1" during initialization.

When initialization finished, INIT bit is cleared automatically.

By checking INIT bit, it is possible to check the initialization status via software.

After Power On Reset, if software wants to initialize memory manual, it shall follow the steps below:

1. Polling MEM_INIT register by READ until the read out value is 0x0000_0000.
2. READ MEM_INIT register to confirm the value is 0x0000_0000.
3. WRITE MEM_INIT register INIT(bit 0) by 1.
4. Polling MEM_INIT register by READ until the read out value is 0x0000_0000.
5. READ MEM_INIT register to confirm the value is 0x0000_0000.

Note: It is recommended that before running software reset, running flush operation to all using access protection of protect area defined by SECDIV[n] register to remain cache data to DDR memory.

### 36.3.6 Access protection for secure/safety regions

#### (1) Memory protection

RT-VRAM has memory protected area function. This function as shown Figure 36.5 is that RT-VRAM area can be divided into 16 areas and each area can be set access privilege to 2 secure groups and 16 safety groups.

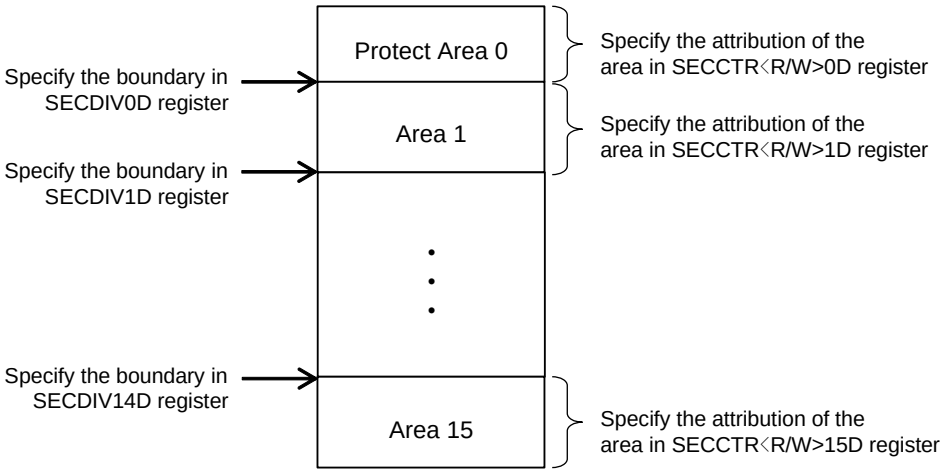



Figure 36.5 Memory Protected Area Function Concept

Table 36.2 Secure ID

| Group | Group ID |
|---|---|
| Secure group | 2'b00 |
| Public group | 2'b10 |

Table 36.3 Region ID

| Group | Region ID | Group | Region ID |
|---|---|---|---|
| Region 0 | 4'b0000 | Region 8 | 4'b1000 |
| Region 1 | 4'b0001 | Region 9 | 4'b1001 |
| Region 2 | 4'b0010 | Region 10 | 4'b1010 |
| Region 3 | 4'b0011 | Region 11 | 4'b1011 |
| Region 4 | 4'b0100 | Region 12 | 4'b1100 |
| Region 5 | 4'b0101 | Region 13 | 4'b1101 |
| Region 6 | 4'b0110 | Region 14 | 4'b1110 |
| Region 7 | 4'b0111 | Region 15 | 4'b1111 |

### 36.3.7 Exclusive access

RT-VRAM supports exclusive access from at most 16 masters. Exclusive marked block size is configurable up to 1 byte. For example, marked block size is 32 bytes, please write H'FFFFFFE0 to EX_ADR_MASKL register.

Exclusive access is performed in a pair of "Load-Exclusive" and "Store-Exclusive" of the identical address.

RT-VRAM always returns "EXOKAY" response to "Load-Exclusive".

If "Store-Exclusive" is success, RT-VRAM returns "EXOKAY" response and updates RAM.

If "Store-Exclusive" is fail (ex. by the interference of other processors), RT-VRAM returns "OKAY" response and don't update RAM.

Please refer to "Arm Architecture Reference Manual Armv8 (B2.2 Atomicity in the Arm architecture)" about details of exclusive access.

NOTE: Because Cortex-A76+AXCC changes ARch transaction ID and AWch transaction ID, so please don't use Store-Exclusive and Load-Exclusive instruction to access RT-VRAM from Cortex-A76. But Cortex-R52 has no this kind of issue. So Cortex-R52 can use Store-Exclusive and Load-Exclusive instruction to access RT-VRAM.

### 36.3.8 Data Scramble

RT-VRAM can set whether to scramble data in 4M bytes unit of virtual buffer.

A subset version of AES is used for encryption.

Apply two rounds processing.

A Key is 128 bits and it is set by secure register in RT-VRAM.

Block length is 256 bytes and the counter mode (CTR) is used for block cipher mode.

The Nonce is 64 bits and it is set by secure register in RT-VRAM.

Input of encryption is concatenated data of the Nonce (64-bit), start address of corresponding 256 bytes data in virtual buffer (32-bit) and incremental counter in units of 16 bites within the 256 bytes data (32-bit).

Only ICUMX can set the Key and Nonce only once after reset. After setting, no IP can access.

## 36.4 Usage Notes

### 36.4.1 Initialization waiting of RT-VRAM

If MEMINIT_ENABLE_SYSRAM_ALL is 1b on releasing hardware reset, software reset for RT-VRAM data (refer to 36.3.5) will run after releasing hardware reset. Initialization takes about 41.7us after releasing hardware reset when AXI clock is 400MHz.

While RT-VRAM running initialization, must not to access RT-VRAM. To read RT-VRAM will not finish until initialization ends, so it influences the performance.

### 36.4.2 Access protection regions configuration

It is prohibited that DIVADDR of SECDIV[n]D (n=1-14) register is set smaller than lower DIVADDR, access protection of Protect Area [n]. For example, it is prohibited that DIVADDR of SECDIV0D register is set H'20 and DIVADDR of SECDIV1D register is set H'10. It is not prohibited that DIVADDR of SECDIV0D register is set H'20 and DIVADDR of SECDIV1D register is set H'20 (not smaller but same value) and size of Area 1 will be 0 in this example.

When setting the SECDIV[n]D register, set it in order from the SECDIV0D register. Otherwise, the access protection settings do not take effect properly.

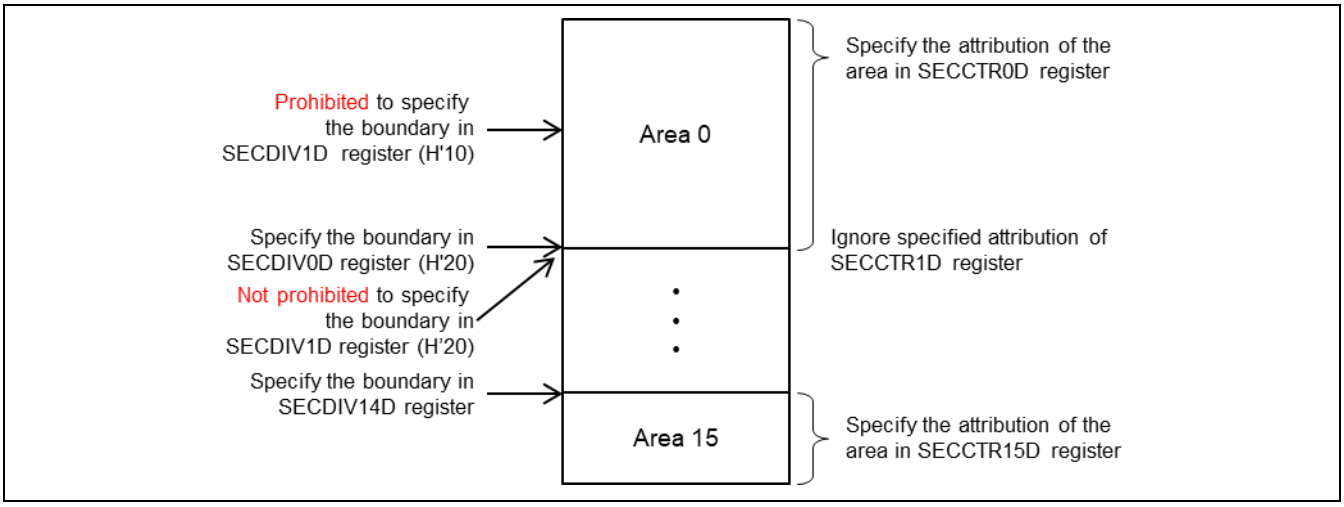



Figure 36.6 Access protection register configuration sample

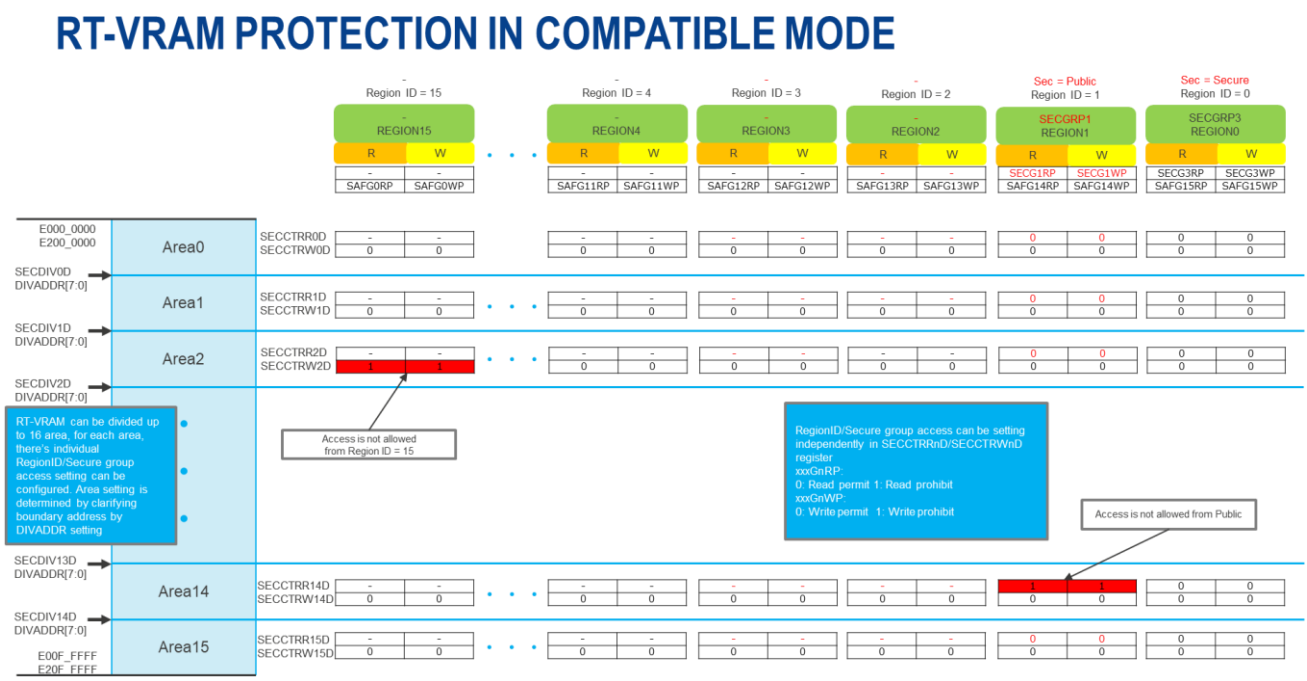



Figure 36.7 Access protection register configuration sample in Compatible Mode

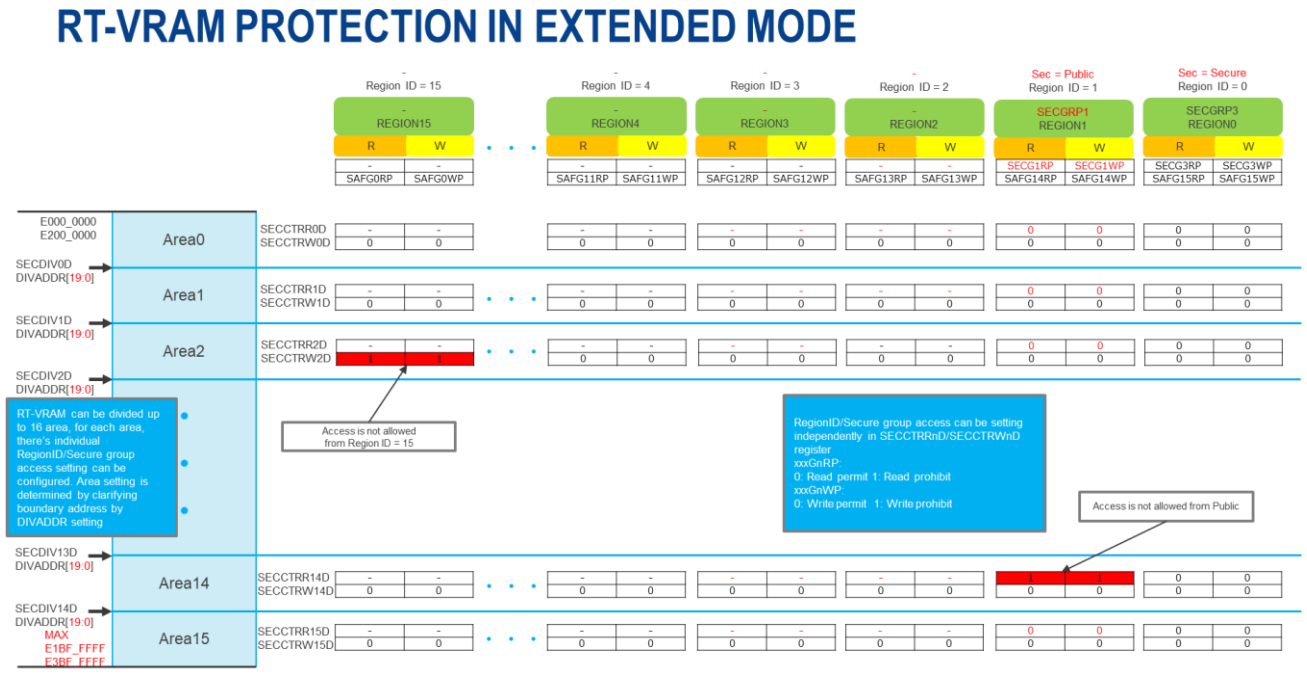



Figure 36.8 Access protection register configuration sample in Extended Mode

The figures above show a sample that how RT-VRAM protect the Memory area from Safety/Security access.

The total memory can be divided up to 16 area for safety/security access. The areas are determined by boundary address setting in SECDIV[n]D:DIVADDR registers. Each area has individual safety (i.e. RegionID) /security read/write access control setting register to permit/prohibit safety/security read/write access.

In SECCTRR[m]D/SECCTRW[m]D (m = 0 to 15)registers, safety/security group read/write access can be setting for each area. When the corresponding bit set to 0, it means the corresponding access is permitted. When the corresponding bit set to 1, it means the corresponding access is prohibited.

### 36.4.3 Timeout Detection for DRAM Access

RT-VRAM measures the period from the output of a transaction to the receipt of the corresponding response. If the period is longer than the value set in the register, RT-VRAM detects as a timeout error.

The error is transmitted to ECM module.

When multiple transactions are output to the main bus, RT-VRAM measures the duration of the oldest transaction.

When the measurement of the period of one transaction is completed, RT-VRAM switches to the measurement of the next oldest transaction. In that case, the start of the measurement period is not when the transaction is output but when the transaction to be measured is switched.

The threshold is set in units of 256 cycles at 400 MHz. The condition to timeout is as follow:

( ( counter to measure the period ) >> 8 ) > (THRESHOLD)

The minimum period is 640 nsec and the maximum period is about 40 msec.

### 36.4.4 Restriction for Cache Function

RT-VRAM does not support cache coherency function with DDR memory.

## 36.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 36.5.1 EDC for RT-VRAM

#### (a) Overview

EDC protection is applied to the SRAM for multi-bank internal memory. 8-bit error correction code is applied to each 64-bit data.

#### (b) Hardware Description

If the EDC error is detected on any bank of internal memory, RT-VRAM issues an interrupt and records error information to error status registers. An interrupt is issued to ICUMX or an error notification is issued to ECM, because RT-VRAM classifies ICUMX and other AXI master.

EDC_CFG register can control whether EDC error detection is enable or not.

RT-VRAM has error injection function to test the interrupt. While the corresponding bit of DUMMY_ERROR register is being set, RT-VRAM issues an interrupt. RT-VRAM doesn't update error information by error injection.

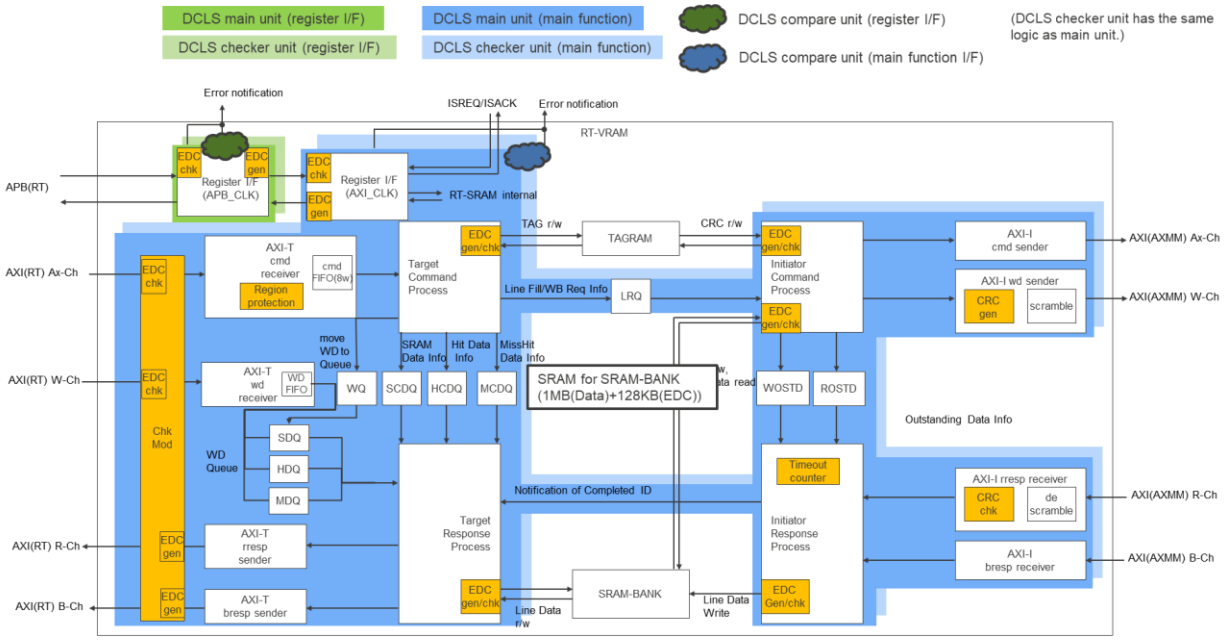



Figure 36.9 Protection for Internal memory

Table 36.4 Related Register

| Register Name | Abbreviation |
|---|---|
| EDC error status Register | ERRBUSCAUSE |
| ICU EDC error status Register | ICUERRBUSCAUSE |
| EDC error control Register | EDC_CFG |
| Dummy Error Insertion Register | DUMMY_ERROR |
| ICU Dummy Error Insertion Register | ICUDUMMY_ERROR |

About register description of these registers, refer to R-CarV4M_ UM_036_RTVRAM_r0p81.xlsx

Table 36.5 Description of interrupt signals

| Interrupt signal | Description |
|---|---|
| ERR_RTSRAM_ICUDUPLEX | DCLS error notification to ICUMX |
| ERR_RTSRAM_ICU_SEC | Security error notification to ICUMX |
| ERR_RTSRAM_ICU_SAFE | Safety error notification to ICUMX |
| ERR_RTSRAM_ICU_EDC_RAM_DBIT | EDC error (DED) notification to ICUMX |
| ERR_RTSRAM_ICU_EDC_RAM | EDC error (SEC/SED) notification to ICUMX |
| ERR_RTSRAM_ICU_EDC_BUS | EDC error notification to ICUMX<br>Note: This signal notifies to ICUMX only EDC error of APB caused by ICUMX |
| ERR_RTSRAMEDC_RAM_DBIT | EDC error (DED) notification to ECM |
| ERR_RTSRAM_TIMEOUT | Timeout error notification to ECM |
| ERR_RTSRAM_EXT | CRC error notification to ECM |
| ERR_RTSRAM_DUPLEX | DCLS error notification to ECM |
| ERR_RTSRAM_SEC | Security error notification to ECM |
| ERR_RTSRAM_SAFE | Safety error notification to ECM |
| ERR_RTSRAM_EDC_RAM_DBIT | EDC error (DED) notification to ECM |
| ERR_RTSRAM_EDC_RAM | EDC error (SEC/SED) notification to ECM |
| ERR_RTSRAM_EDC_BUS | EDC error notification to ECM<br>Note: This signal notifies to ECM not only EDC errors by other AXI master but also EDC errors of AXI Ax-Ch and AXI W-Ch caused by ICUMX. |

### 36.5.2 EDC for RT-VRAM AXI interface

#### (a) Overview

EDC protection is applied to AXI interface of RT-VRAM. 8-bit error detection code is applied to each 64-bit data.

#### (b) Hardware Description

If the EDC error is detected on AXI interface of RT-VRAM, RT-VRAM issues an interrupt and records error information to error status registers. An interrupt is issued to ICUMX or an error notification is issued to ECM, because RT-VRAM classifies ICUMX and other AXI master.
RT-VRAM has error injection function to test the interrupt. While the corresponding bit of DUMMY_ERROR register is being set, RT-VRAM issues an interrupt. RT-VRAM doesn't update error information by error injection.
Note *: RT-VRAM can't classify whether EDC errors of AXI Ax-Ch and AXI W-Ch are detected by the request from ICUMX or other AXI master. But RT-VRAM can classify EDC errors of APB. So, an interrupt signal in Table 36.6 "ERR_RTSRAM_ICU_EDC_BUS" notifies to ICUMX only EDC error of APB caused by ICUMX. And an interrupt signal in Table 36.6 "ERR_RTSRAM_EDC_BUS" notifies to ECM not only EDC errors caused by other AXI master but also EDC errors of AXI Ax-Ch and AXI W-Ch caused by ICUMX.

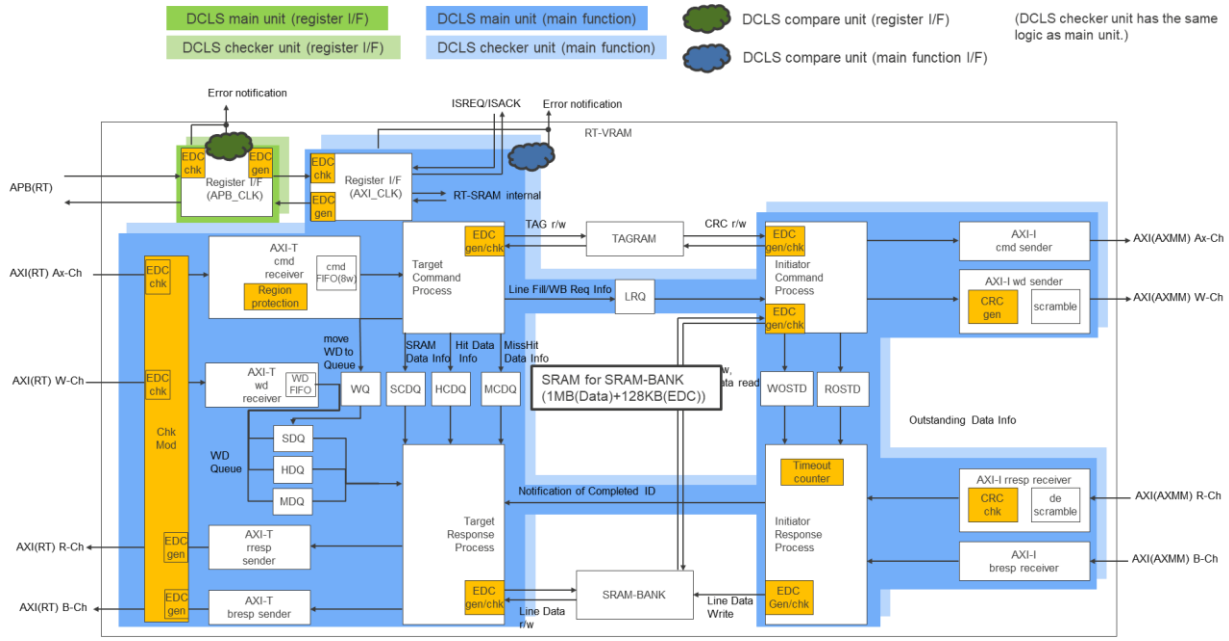



Figure 36.10 EDC protection for RT-VRAM

Table 36.6 Related Register

| Register Name | Abbreviation |
|---|---|
| EDC error status Register | ERRBUSCAUSE |
| ICU EDC error status Register | ICUERRBUSCAUSE |
| EDC error control Register | EDC_CFG |
| Dummy Error Insertion Register | DUMMY_ERROR |
| ICU Dummy Error Insertion Register | ICUDUMMY_ERROR |

About register description of these registers, refer to R-CarV4M_ UM_036_RTVRAM_r0p81.xlsx

Table 36.7 Description of interrupt signals

| Interrupt signal | Description |
|---|---|
| ERR_RTSRAM_ICUDUPLEX | DCLS error notification to ICUMX |
| ERR_RTSRAM_ICU_SEC | Security error notification to ICUMX |
| ERR_RTSRAM_ICU_SAFE | Safety error notification to ICUMX |
| ERR_RTSRAM_ICU_EDC_RAM_DBIT | EDC error (DED) notification to ICUMX |
| ERR_RTSRAM_ICU_EDC_RAM | EDC error (SEC/SED) notification to ICUMX |
| ERR_RTSRAM_ICU_EDC_BUS | EDC error notification to ICUMX<br>Note: This signal notifies to ICUMX only EDC error of APB caused by ICUMX |
| ERR_RTSRAMEDC_RAM_DBIT | EDC error (DED) notification to ECM |
| ERR_RTSRAM_TIMEOUT | Timeout error notification to ECM |
| ERR_RTSRAM_EXT | CRC error notification to ECM |
| ERR_RTSRAM_DUPLEX | DCLS error notification to ECM |
| ERR_RTSRAM_SEC | Security error notification to ECM |
| ERR_RTSRAM_SAFE | Safety error notification to ECM |
| ERR_RTSRAM_EDC_RAM_DBIT | EDC error (DED) notification to ECM |
| ERR_RTSRAM_EDC_RAM | EDC error (SEC/SED) notification to ECM |
| ERR_RTSRAM_EDC_BUS | EDC error notification to ECM<br>Note: This signal notifies to ECM not only EDC errors by other AXI master but also EDC errors of AXI Ax-Ch and AXI W-Ch caused by ICUMX. |

### 36.5.3 Hardware Redundancy for RT-VRAM

#### (a) Overview

Hardware redundancy is applied to whole RT-VRAM except RAM bank.

#### (b) Hardware Description

RT-VRAM has Hardware Redundancy.

Figure 36.11 shows the block diagram of Hardware Redundancy of RT-VRAM. If the Redundancy error is detected, RT-VRAM issues an interrupt and records error information to error status registers. An error notification is issued to ECM.

RT-VRAM has error injection function to test the interrupt. While the corresponding bit of DUMMY_ERROR register is being set, RT-VRAM issues an interrupt. RT-VRAM doesn't update error information by error injection.

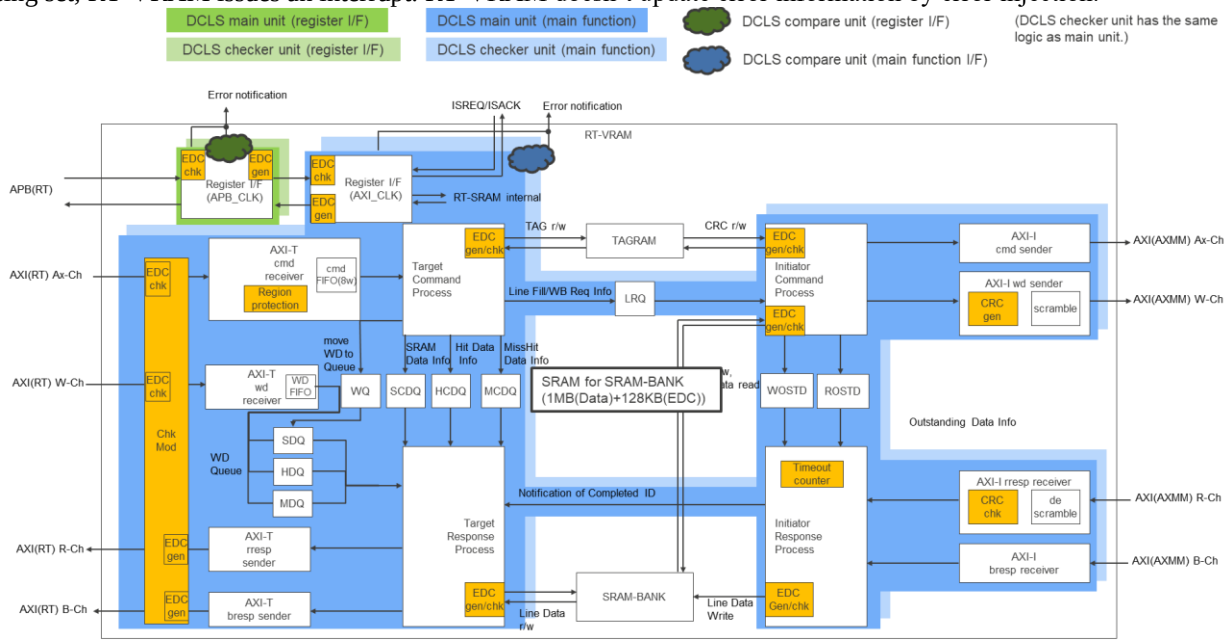



Figure 36.11 Block Diagram of Hardware Redundancy for RT-VRAM

Table 36.8 Description of interrupt signals

| Interrupt signal | Description |
|---|---|
| ERR_RTSRAM_ICUDUPLEX | DCLS error notification to ICUMX |
| ERR_RTSRAM_DUPLEX | DCLS error notification to ECM |

### 36.5.4 Access protection for secure/safety regions

**(a) Overview**

Area protection function is applied to RT-VRAM in order to detect the non-permitted access to the specified memory area that can be set by secure group and Region-ID which is defined by AXI-BUS module. Access control level can be defined for each area. It can be limited the modules which can access the area by access control level defined for each area.

**(b) Hardware Description**

**(1) Memory protection**

RT-VRAM has memory protected area function. This function as shown Figure 36.12 is that RT-VRAM area can be divided into 16 areas and each area can be set access privilege to 2 secure groups and 16 safety groups.

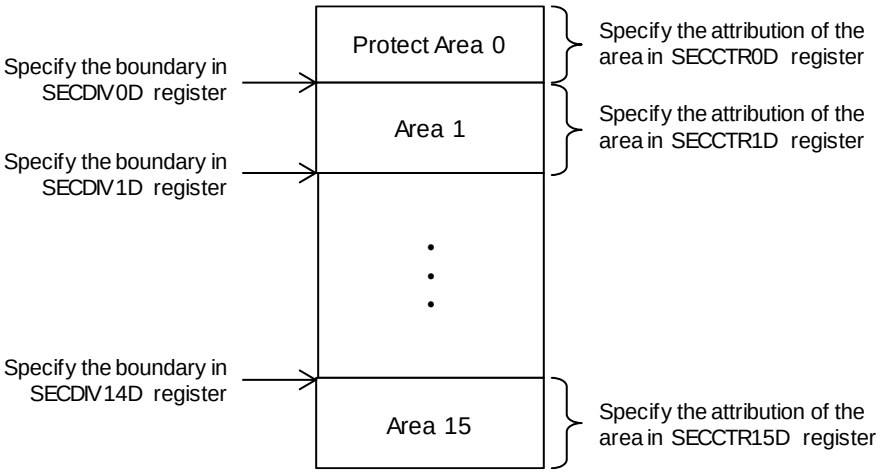



Figure 36.12 Memory Protected Area Function Concept

Table 36.9 Secure ID

| Group | Group ID |
|---|---|
| Secure group | 2'b00 |
| Public group | 2'b10 |

Table 36.10 Region ID

| Group | Region ID | Group | Region ID |
|---|---|---|---|
| Region 0 | 4'b0000 | Region 8 | 4'b1000 |
| Region 1 | 4'b0001 | Region 9 | 4'b1001 |
| Region 2 | 4'b0010 | Region 10 | 4'b1010 |
| Region 3 | 4'b0011 | Region 11 | 4'b1011 |
| Region 4 | 4'b0100 | Region 12 | 4'b1100 |
| Region 5 | 4'b0101 | Region 13 | 4'b1101 |
| Region 6 | 4'b0110 | Region 14 | 4'b1110 |
| Region 7 | 4'b0111 | Region 15 | 4'b1111 |

### (2) Memory protect function

#### 1) RT-VRAM Protect function

The RT-VRAM is divided into 16 areas and access settings for each area specific to the safety group are possible.
The areas can be specified by addresses within a 1Mbyte range from H'E200 0000 to H'E20F FFFF in compatible mode or 28Mbyte range from H'E200 0000 to H'E3BF FFFF in extended mode.
Separate access settings specific to the safety group reading and writing are possible.
Access to a protected area returns an error response. Writing to a protected area is ignored and reading from a protected area always returns 0.

#### a) Set limit to a register note and function.

Write protect function only by the specific safety group.

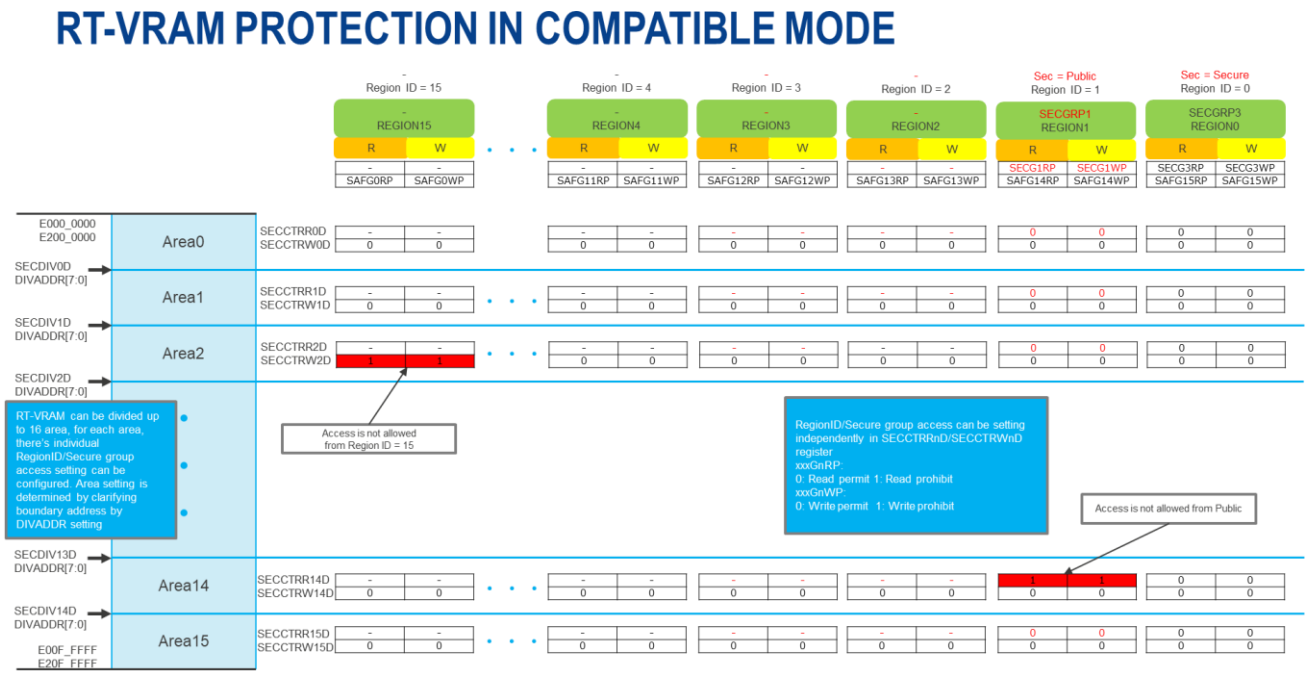



Figure 36.13 Access protection register configuration sample in Compatible Mode

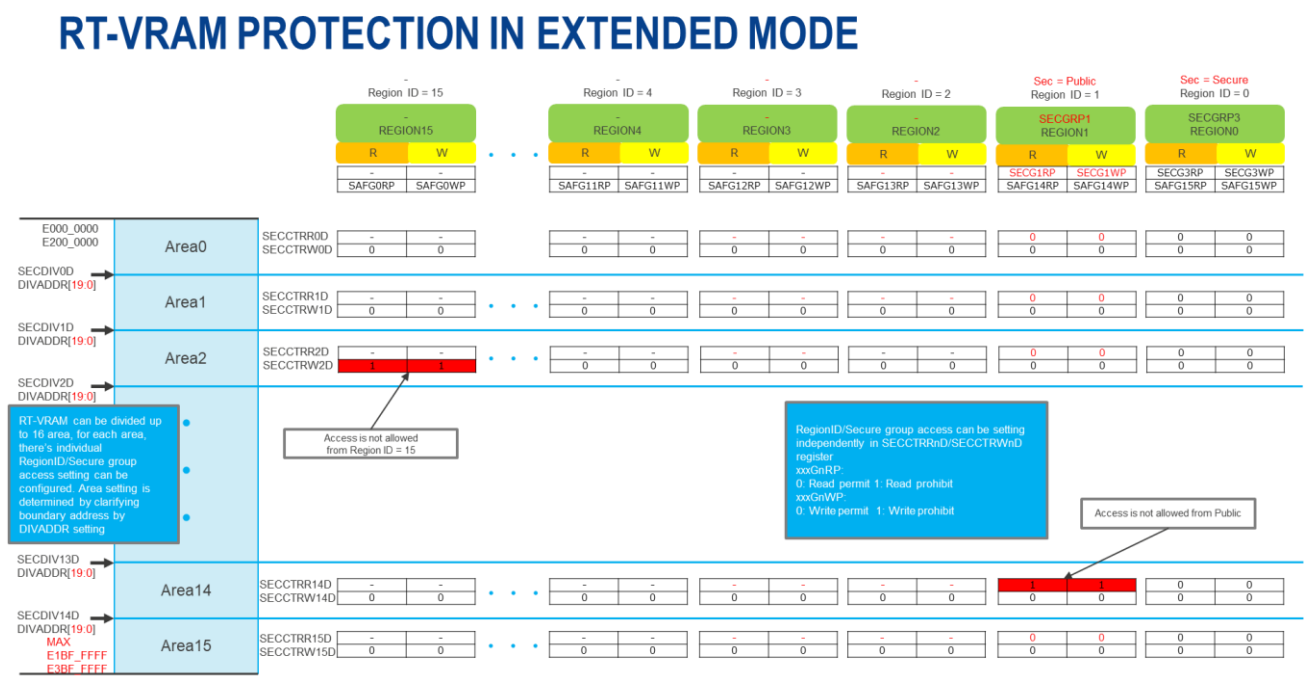



**Figure 36.14 Access protection register configuration sample in Extended Mode**

The figures above show a sample that how RT-VRAM protect the Memory area from Safety/Security access.

The total memory can be divided up to 16 area for safety/security access. The areas are determined by boundary address setting in SECDIV[n]D:DIVADDR registers. Each area has individual safety (i.e. RegionID) /security read/write access control setting register to permit/prohibit safety/security read/write access.

In SECCTRR[m]D/SECCTRW[m]D (m = 0 to 15)registers, safety/security group read/write access can be setting for each area. When the corresponding bit set to 0, it means the corresponding access is permitted. When the corresponding bit set to 1, it means the corresponding access is prohibited.

### 36.5.5 CRC for DDR on RT-VRAM

#### (a) Overview

In Extended mode, the buffer size of RT-VRAM is virtually extended up to 28M bytes. Actual buffer is allocated on DRAM, and the data in DRAM is protected by CRC (End-to-End protection).

RT-VRAM generates 32-bit CRC and stores in dedicated SRAM in RT-VRAM when writing 256 bytes data to DRAM. If RT-VRAM reads 256 bytes data from DRAM, it uses the corresponding CRC to check that the data is correct.

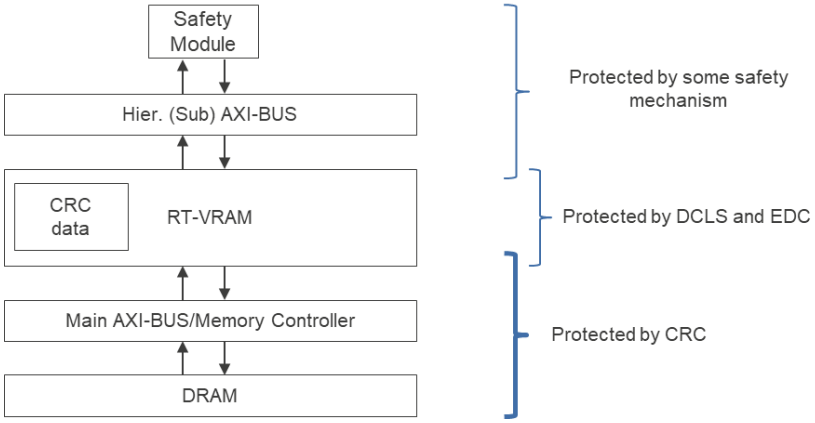



Figure 36.15 Overview of CRC protection for DRAM data

#### (b) Hardware Description

32-bit CRC is calculated every 256 bytes data. It is generated when 256 bytes data is stored to DRAM and checked when corresponding 256 bytes data is read from DRAM. The generated CRC is stored dedicated area of TAGRAM and SRAM-BANK. The maximum size of the protected area on DRAM is determined to 28M bytes by the size of CRC stored area on TAGRAM and SRAM-BANK. The CRC polynomial is H'04C1 1DB7, and initial value of reminder is H'FFFF FFFF.

CRC error detection function distinguishes whether it occurred in the transaction issued by ICUMX or not. And the function only notifies only CRC error due to non-ICUMX issued transactions. In other words, CRC error caused by ICUMX only updates the status.

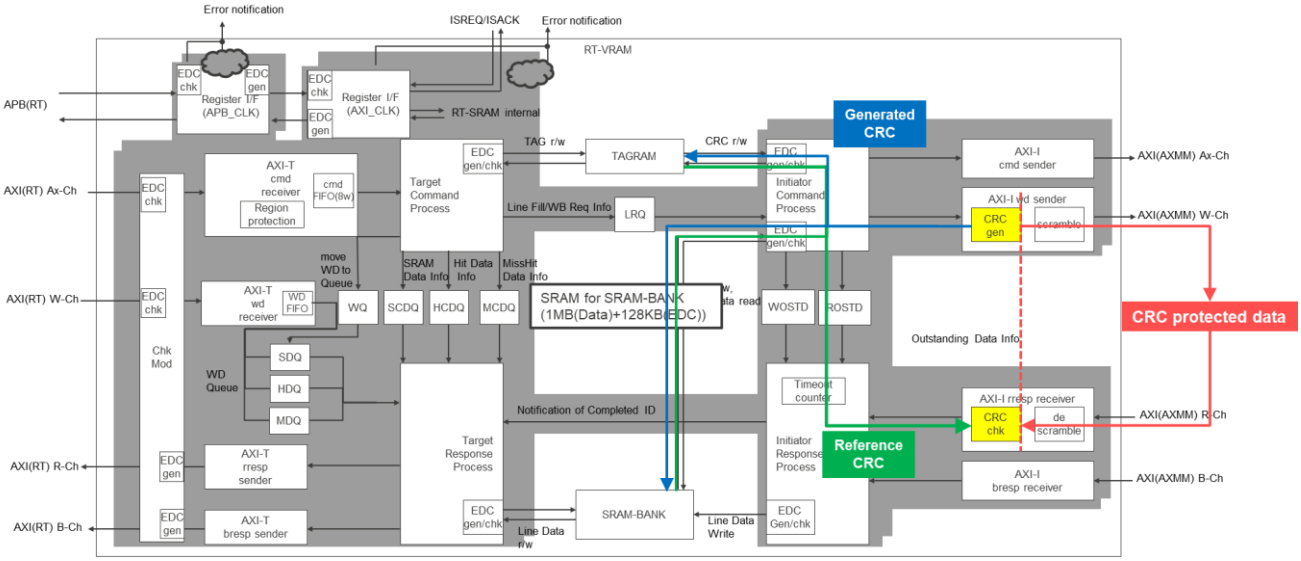



Figure 36.16 Block diagram of CRC for DDR in RT-VRAM

Table 36.11 Related Register

| Register Name | Abbreviation |
|---|---|
| CRC Configuration Register | CRC_CFG |
| CRC Error Status Register | CRC_ERRCAUSE |
| CRC Error Address Register | CRC_ERRVADDR |
| ICU_CRC Error Status Register | ICU_CRC_ERRCAUSE |
| ICU CRC Error Address Register | ICU_CRC_ERRVADDR |
| Dummy Error Insertion Register | DUMMY_ERROR |

About register description of these registers, refer to R-CarV4M_ UM_036_RTVRAM_r0p81.xlsx

Table 36.12 Description of interrupt signals

| Interrupt signal | Description |
|---|---|
| ERR_RTSRAM_EXT | CRC error notification to ECM |

### 36.5.6 Timeout Counter in RT-VRAM

#### (a) Overview

Timeout Counter in RT-VRAM has counters for each transaction, and it can monitor all of transaction in parallel. The oldest counter is always compared with threshold, if the counter value exceeds the threshold, then timeout error will be notified to ECM.
When new request transaction is issued from RT-VRAM to DRAM, then counting starts. When request from DRAM is arrived, then counting stops. If one of request is not arrived within predefined period, then timeout error will be occurred. Thus, by setting expected the worst DRAM delay, the delay and loss between RT-VRAM and DRAM can be detectable.

Timeout counter error detection function distinguishes whether it occurred in the transaction issued by ICUMX or not. And the function only notifies only Timeout counter due to non-ICUMX issued transactions. In other words, Timeout counter caused by ICUMX only updates the status.

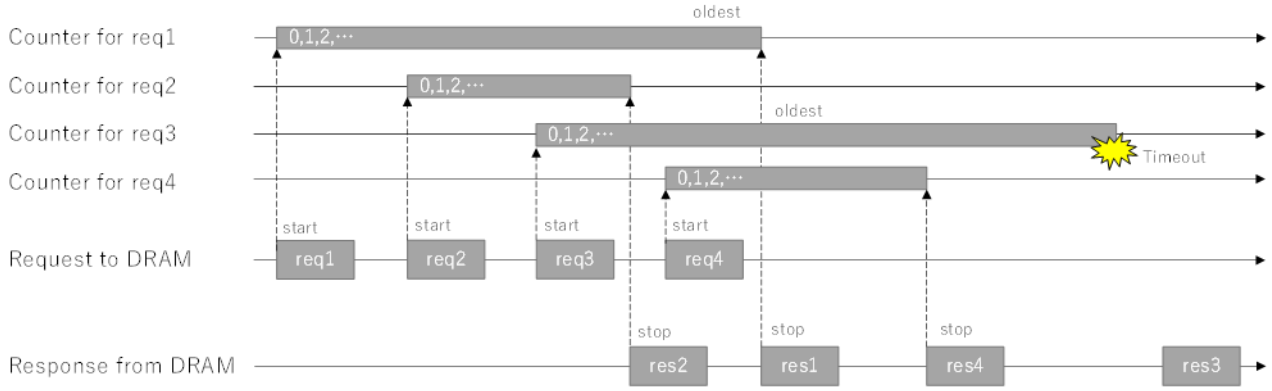



Figure 36.17 Timeout monitoring by timeout counter in RT-VRAM

#### (b) Hardware Description

Refer to section 36.4.3 Timeout Detection for DRAM Access.
Timeout Detection is supported fault injection function by FAULT_INJECTION register.

Table 36.13 Related Register

| Register Name | Abbreviation |
|---|---|
| Timeout detection configuration Register | TIMEOUT_CFG |
| Time-out error Status Register | TIMEOUT_ERR |
| ICU Time-out error Status Register | ICUTIMEOUT_ERR |
| Dummy Error Insertion Register | DUMMY_ERROR |
| Fault Injection to SM Register | FAULT_INJECTION |

About register description of these registers, refer to R-CarV4M_ UM_036_RTVRAM_r0p81.xlsx

Table 36.14 Description of interrupt signals

| Interrupt signal | Description |
|---|---|
| ERR_RTSRAM_TIMEOUT | Timeout error notification to ECM |

# 37. Back-up Buffer (BKBUF)

The Back-up Buffer (BKBUF) is a module with space for storing 1024 bytes of data. Data stored in the Back-up Buffer are not deleted as long as the power is supplied to the given LSI chip. This section gives an overview and describes the usage of the Back-up Buffer.

## 37.1 Overview

### 37.1.1 Features

- The Back-up Buffer has a capacity of 1024 bytes.
  Address range to which the Back-up Buffer is mapped: H'EB18 0000 to H'EB18 03FC
- The power domain of the Back-up Buffer is the always-on domain of application domain*1, which is beyond control over shutting off the supply of power.
  The AXI bus for use by the Back-up Buffer is connected to the RT domain AXI*2.
- Power-on resets are applicable to the Back-up Buffer.
- The 1024 bytes of data storage in the Back-up Buffer are not initialized after a reset.
- The power-on self-test is not applicable to the Back-up Buffer.

Notes: 1. For details, refer to section 11.1.2, Power Domain Structure, in section 11, System Controller (SYSC).
2. For details, refer to section 18, AXI-bus.

### 37.1.2 Block Diagram

Figure 37.1 shows a block diagram of the Back-up Buffer.
The Data Storage is made from Flip Flop (F/F) not used SRAM, use the Back-up Buffer function in accordance with the electrical characteristics.

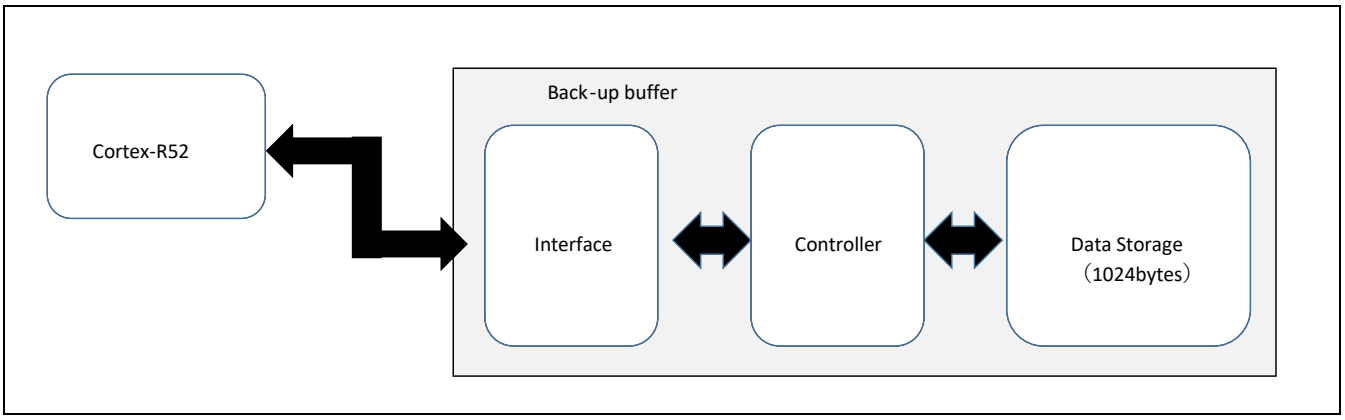



Figure 37.1 Block Diagram of the Back-up Buffer

### 37.1.3 External Pins

The Back-up Buffer does not have any external pins.

### 37.1.4 Connected module

Table 37-1 Connected Modules

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| Back-up Buffer | CPG | Clock Generation<br>Configuration signal controller |
| | AXI-bus | AXI Bus |

### 37.1.5 Register configuration

The Back-up Buffer has two address space, the data storage area, and the register area of access protection.

Table 37-2 Address space for Back-up Buffer

| Address | Notice |
|---|---|
| H'EB18 0000 to H'EB18 03FC | The data storage area |
| H'E615 0280 to H'E615 0283 | The register area of access protection |

Refer to R-CarV4M_UM_037_BKBUF_r0p50.xlsx.

## 37.2 Register description

Refer to R-CarV4M_UM_037_BKBUF_r0p50.xlsx.

## 37.3 Operation

For access to the Back-up Buffer, disable protection of the data storage beforehand. On completion of the required access, re-enable protection of the stored data. The procedure for using the Back-up Buffer is given below.

For details on the register other than the BKBAPR register described here (in section 37.3.1), refer to section 8.2, Register Description, in section 8, Clock Pulse Generator (CPG).

### 37.3.1 Access to the Back-up Buffer

To use the Cortex-R52 processor to access the Back-up Buffer, proceed with steps 1 to 3 below.

1. Disable protection of the data storage in the Back-up Buffer.
   (1) Write H'FFFF FFFE to the CPG write protect register (CPGWPR).
   (2) Write H'0000 0001 to the Back-up Buffer access protect register (BKBAPR) of the CPG.
2. Use the Cortex-R52 processor to access the data storage in the Back-up Buffer.

On completion of the required access to the Back-up Buffer, re-enable protection of the stored data through the steps below.

3. Enable protection of the data storage in the Back-up Buffer.
   (1) Write H'FFFF FFFF to the CPG write protect register (CPGWPR).
   (2) Write H'0000 0000 to the Back-up Buffer access protect register (BKBAPR) of the CPG.

### 37.3.2 Module Reset

**Back-up buffer is using un-reset F/F.**

- Contents of F/F should be clear by write "0" at initialization

**Back-up data is not guaranteed when:**

- Software Reset is asserted during write transaction to Back-up Buffer
- POST(Power On Self-Test) is executed while protection of the stored data is Disable (BKBAPR.BKBAP bit = 1).

## 37.4 Usage Notes

— The 1024 bytes of data storage in the Back-up Buffer can only be initialized by using software to write initial values to it.

— Access to the Back-up Buffer from processors other than the Cortex-R52 is prohibited.

— Access to the Back-up Buffer from DMAC is prohibited.

— Calculate CRC in the Cortex-R52.

— Address space assigned to Back-up Buffer (H'00_EB18_0000 to H'00_EB1F_FFFF) is mirrored at each 1024 bytes address boundary.

## 37.5 Safety Mechanisms

No description on this section.

# 50. 3D Graphics Processsing Unit（GSX）

## 50.1 Overview

- R-Car V4M integrates Imagination Technologies IMG BXS-4-64 GPU.

### 50.1.1 Features

- ZGφ
- 4 pixels/clk
- 64 FLOPs/clk (FP32)
- System Level Cache provides cashing of all types of workload data. 64 KB capacity.
- OpenGL ES 3.2 supported
- ‘per module automatic clock gating’ supported

### 50.1.2 Block Diagram

Figure 50.1 shows the block diagram of IMG BXS-4-64 and the corresponding power-domains (A23DGB, A33DGA and C4_U3DG).

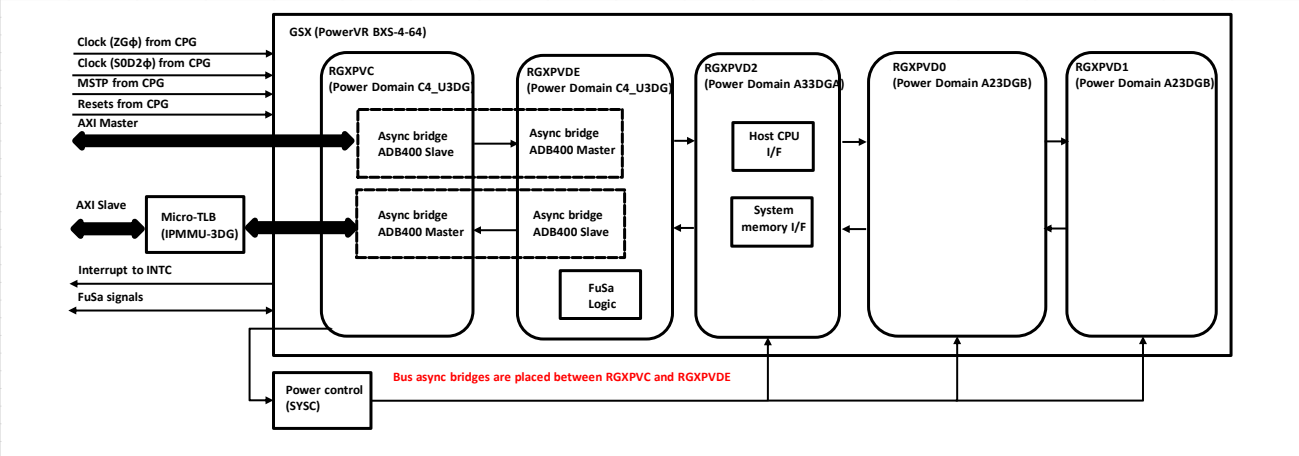



*Figure 50.1 Block diagram of IMG BXS-4-64*

## 50.2 Register Configuration

Base address of registers is allocated to H’00 FD00 0000. The address range is 8192 KB (H’00 FD00 0000 - H’00 FD7F FFFF).

## 50.3 Operation

### 50.3.1 Power and Clock Supply

The power and clock need to be supplied before IMG BXS-4-64 is used. In the low power mode, all C4 modules are powered OFF, so in order to use IMG BXS-4-64, it must be in full run mode by the following power on sequence: C4 >> A33DGA >> A23DGB. Vice versa, the power off sequence is: A23DGB >> A33DGA >> C4. Figure 50.2 shows the example of initial setting.

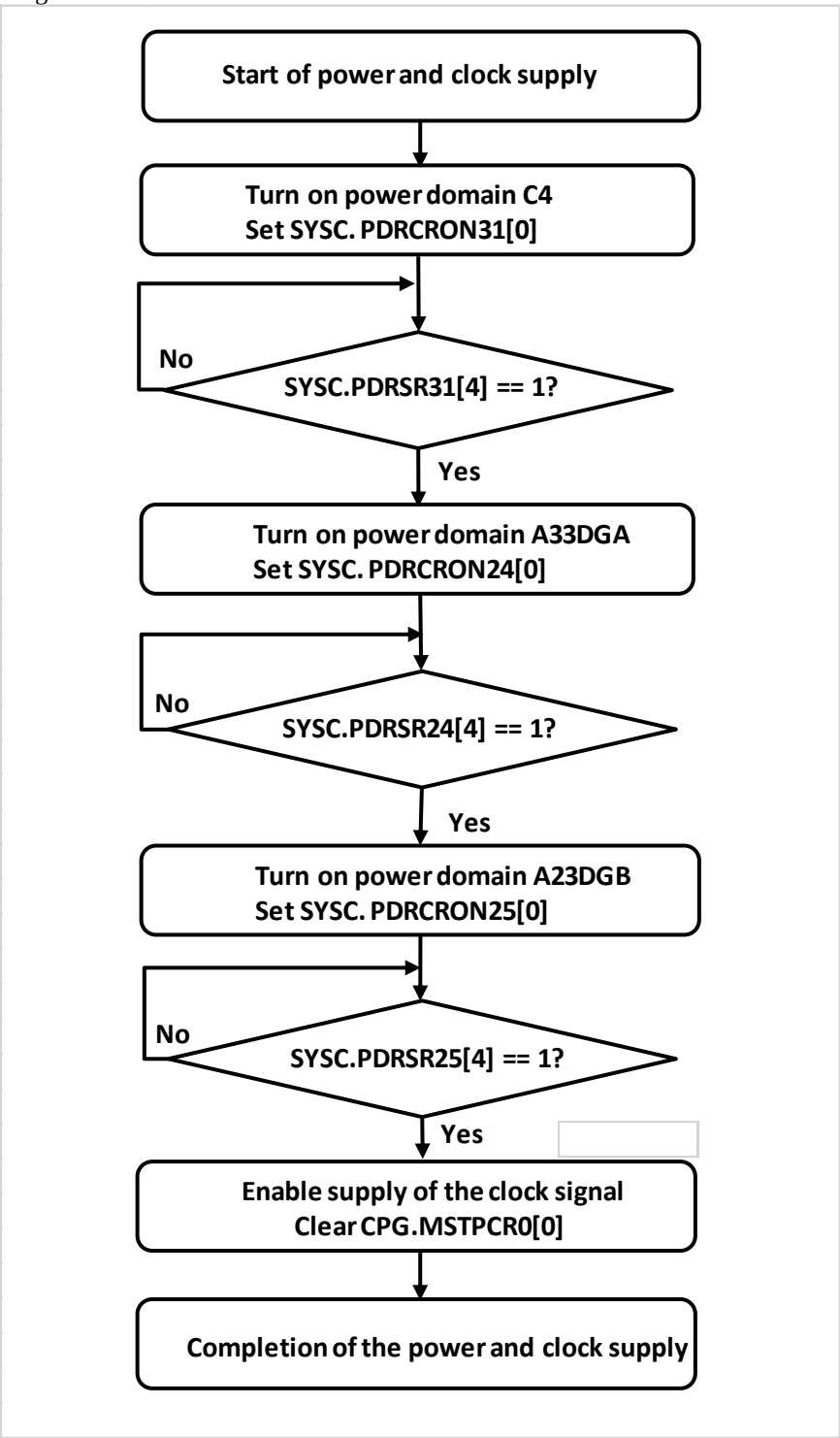



*Figure 50.2 IMG BXS-4-64 power and clock supply flowchart*

## 50.4 Usage Notes

Nothing special to note on the usage.

## 50.5 Safety Mechanisms

There is some safety mechanisms in the bus bridge.

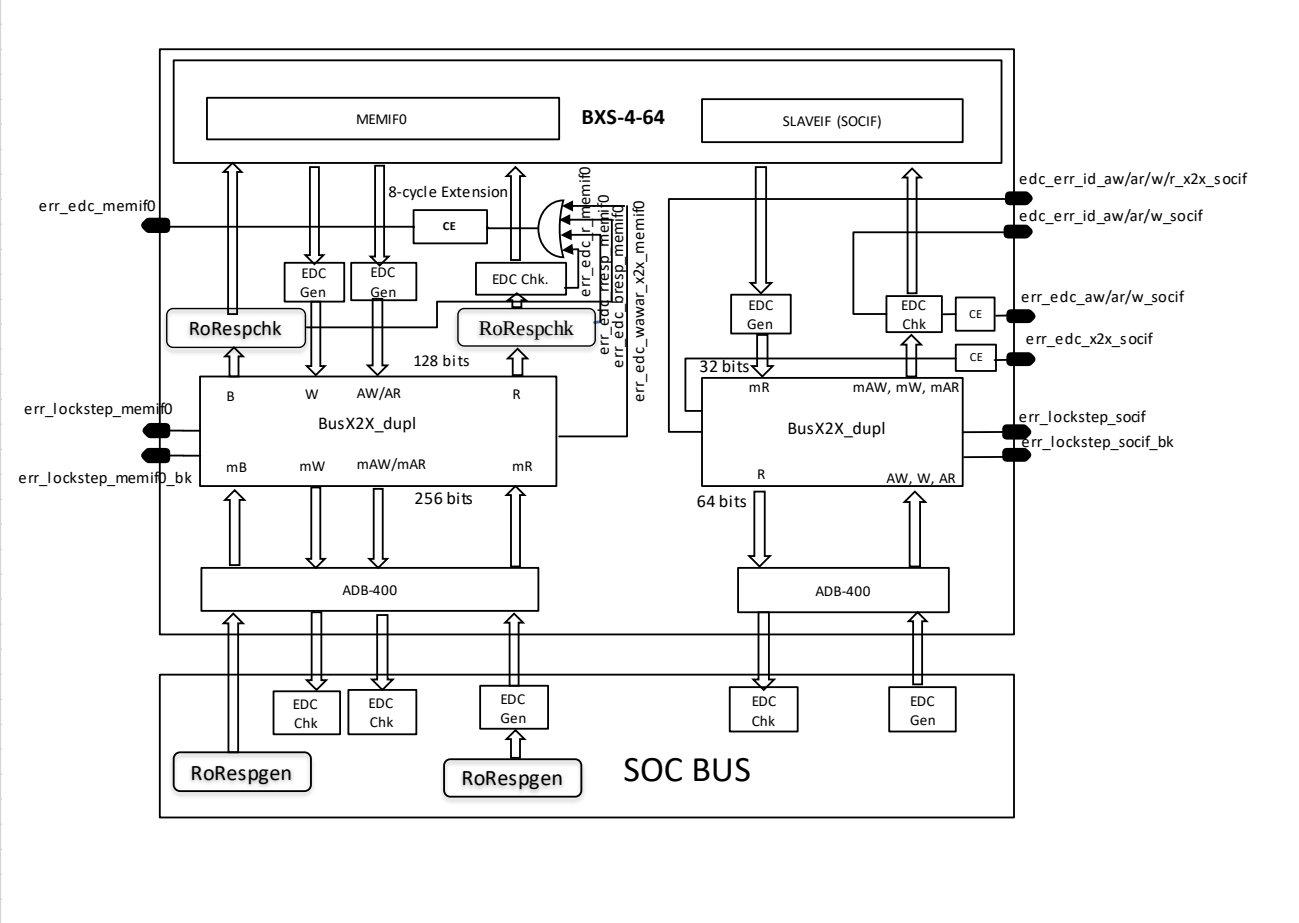



*Figure 50.3 Safety mechanisms of IMG BXS-4-64*

# 53. CSI2

## 53.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The CSI2 is an MIPI CSI-2 (Camera Serial Interface 2) receiver module.

The R-Car V4M has two CSI2 sets. Two 4-lane D-PHY.

The received signals are input to the PHY modules; the LINK modules receive the byte data output from the PHY modules; and the PHY modules extract the image data (embedded data included), data type, and channel information before outputting the data to the following ISP/CS modules. Here, the LINK modules generate the vertical Blanking Packet Indicator(VD), Horizontal Blanking Packet Indicator(HD), field signals (FLD),data enable signals (PE), byte enable signals (PEB) for each channel, and outputs them to the ISP/CS modules.

### 53.1.1 Features

The CSI2 module has the following features.

**(1) MIPI CSI-2 Version 2.0**

**(2) MIPI D-PHY Version 2.1**
**Up to 2.5-Gbps transfer rate per data lane**
**Maximum throughput is 10-Gbps with 4 data lanes**

**(3) 4 RX data lanes on D-PHY.**

**(4) Deskew for PHY-Ex-IF.**

**(5) Dynamically configurable multi-lane merging**

**(6) Long and Short packet decoding**

**(7) Timing accurate signaling of Frame and Line synchronization packets**

**(8) Several frame formats**

- General Frame or Digital Interlaced Video with or without accurate sync timing
- Data Type (Packet or Frame level) and virtual channel interleaving

**(9) All primary and secondary data formats**

- YUV420 8bit /10bit
- YUV422 8bit / 10bit / Y-12bit,UV-8bit ( Transferred as YUV422 10bit )
- RGB888 / RGB565
- RAW8 / RAW10 / RAW12 / RAW14 / RAW16 / RAW20 / RAW24

**(10) Error detection and correction**

- PHY level
- Packet level
- Line level
- Frame level

Support 16 Virtual Channel

Note) The optional features of MIPI CSI-2 and D-PHY not described in this section are not supported.

### 53.1.2 Block Diagram

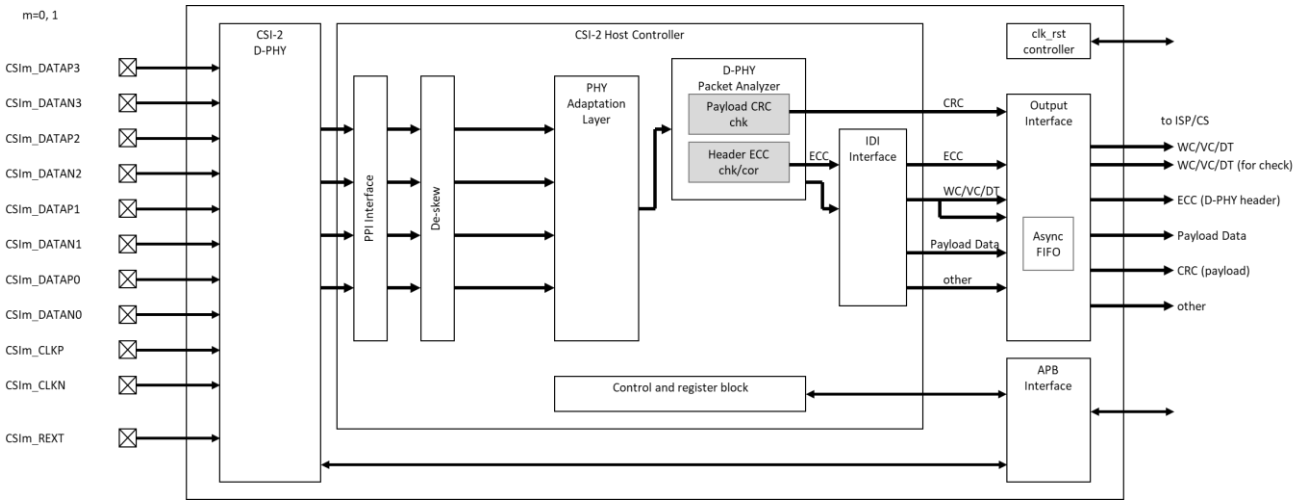



Figure 53.1 Block Diagram

### 53.1.3 External Pins

Table 53.1 External Pins

| Pin Name | Abbreviation | I/O | Function |
|---|---|---|---|
| CSI2 0 lane0 data | CSI0_DATAP0 | Input | CSI2 0 :<br>D-PHY data lane 0 positive terminal. |
| CSI2 0 lane0 data | CSI0_DATAN0 | Input | CSI2 0 :<br>D-PHY data lane 0 negative terminal. |
| CSI2 0 lane1 data | CSI0_DATAP1 | Input | CSI2 0 :<br>D-PHY data lane 1 positive terminal. |
| CSI2 0 lane1 data | CSI0_DATAN1 | Input | CSI2 0 :<br>D-PHY data lane 1 negative terminal. |
| CSI2 0 lane2 data | CSI0_DATAP2 | Input | CSI2 0 :<br>D-PHY data lane 2 positive terminal. |
| CSI2 0 lane2 data | CSI0_DATAN2 | Input | CSI2 0 :<br>D-PHY data lane 2 negative terminal. |
| CSI2 0 lane3 data | CSI0_DATAP3 | Input | CSI2 0 :<br>D-PHY data lane 3 positive terminal. |
| CSI2 0 lane3 data | CSI0_DATAN3 | Input | CSI2 0 :<br>D-PHY data lane 3 negative terminal. |
| CSI2 0 clock | CSI0_CLKP | Input | CSI2 0 :<br>D-PHY clock lane positive terminal. |
| CSI2 0 clock | CSI0_CLKN | Input | CSI2 0 :<br>D-PHY clock lane negative terminal. |
| CSI2 0 REXT | CSI0_REXT | Input | CSI2 0 : External precision resistor.<br>200 ohm precision resistor connected to ground. |
| CSI2 1 lane0 data | CSI1_DATAP0 | Input | CSI2 1 :<br>D-PHY data lane 0 positive terminal. |
| CSI2 1 lane0 data | CSI1_DATAN0 | Input | CSI2 1 :<br>D-PHY data lane 0 negative terminal. |
| CSI2 1 lane1 data | CSI1_DATAP1 | Input | CSI2 1 :<br>D-PHY data lane 1 positive terminal. |
| CSI2 1 lane1 data | CSI1_DATAN1 | Input | CSI2 1 :<br>D-PHY data lane 1 negative terminal. |
| CSI2 1 lane2 data | CSI1_DATAP2 | Input | CSI2 1 :<br>D-PHY data lane 2 positive terminal. |
| CSI2 1 lane2 data | CSI1_DATAN2 | Input | CSI2 1 :<br>D-PHY data lane 2 negative terminal. |
| CSI2 1 lane3 data | CSI1_DATAP3 | Input | CSI2 1 :<br>D-PHY data lane 3 positive terminal. |
| CSI2 1 lane3 data | CSI1_DATAN3 | Input | CSI2 1 :<br>D-PHY data lane 3 negative terminal. |
| CSI2 1 clock | CSI1_CLKP | Input | CSI2 1 :<br>D-PHY clock lane positive terminal. |
| CSI2 1 clock | CSI1_CLKN | Input | CSI2 1 :<br>D-PHY clock lane negative terminal. |

| Pin Name | Abbreviation | I/O | Function |
|---|---|---|---|
| CSI2 1 REXT | CSI1_REXT | Input | CSI2 1 : External precision resistor.<br>200 ohm precision resistor connected to ground. |

### 53.1.4 Register Configuration

The following lists the base address of each CSI2 module.

| Base Address Name | Base Address |
|---|---|
| CSI20 | H'FE50 0000 |
| CSI21 | H'FE54 0000 |

#### 53.1.4.1 Register Configuration

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| CSI2 | CSI2 [n] Core version(n=0-1) | CSI2[n]_VERSION | R | <CSI2[n]> + H'0 0000 | H'3135 312A | 32 |
| CSI2 | CSI2 [n] Number of lanes(n=0-1) | CSI2[n]_N_LANES | R/W | <CSI2[n]> + H'0 0004 | H'0000 0003 | 32 |
| CSI2 | CSI2 [n] Logic Reset(n=0-1) | CSI2[n]_CSI2_RESETN | R/W | <CSI2[n]> + H'0 0008 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Main interrupt status(n=0-1) | CSI2[n]_INT_ST_MAIN | RC | <CSI2[n]> + H'0 000C | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Data Type fields for Data ID Monitors 1(n=0-1) | CSI2[n]_DATA_IDS_1 | R/W | <CSI2[n]> + H'0 0010 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY Mode(n=0-1) | CSI2[n]_PHY_MODE | R/W | <CSI2[n]> + H'0 001C | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Virtual Channel fields for Data ID Monitors 1(n=0-1) | CSI2[n]_DATA_IDS_VC_1 | R/W | <CSI2[n]> + H'0 0030 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY Shutdown(n=0-1) | CSI2[n]_PHY_SHUTDOWNZ | R/W | <CSI2[n]> + H'0 0040 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] DPHY reset(n=0-1) | CSI2[n]_DPHY_RSTZ | R/W | <CSI2[n]> + H'0 0044 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] RX PHY status(n=0-1) | CSI2[n]_PHY_RX | R | <CSI2[n]> + H'0 0048 | H'0001 0000 | 32 |
| CSI2 | CSI2 [n] STOPSTATE PHY status(n=0-1) | CSI2[n]_PHY_STOPSTATE | R | <CSI2[n]> + H'0 004C | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY Calibration(n=0-1) | CSI2[n]_PHY_CAL | RC | <CSI2[n]> + H'0 00CC | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Fatal interruption caused by PHY(n=0-1) | CSI2[n]_INT_ST_PHY_FATAL | RC | <CSI2[n]> + H'0 00E0 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for fatal interruption caused by PHY(n=0-1) | CSI2[n]_INT_MSK_PHY_FATAL | R/W | <CSI2[n]> + H'0 00E4 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Fatal interruption caused during Packet Construction(n=0-1) | CSI2[n]_INT_ST_PKT_FATAL | RC | <CSI2[n]> + H'0 00F0 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for fatal interruption caused during Packet Construction(n=0-1) | CSI2[n]_INT_MSK_PKT_FATAL | R/W | <CSI2[n]> + H'0 00F4 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Interruption caused by PHY(n=0-1) | CSI2[n]_INT_ST_PHY | RC | <CSI2[n]> + H'0 0110 | H'0000 0000 | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| CSI2 | CSI2 [n] Mask for interruption caused by PHY(n=0-1) | CSI2[n]_INT_MSK_PHY | R/W | <CSI2[n]> + H'0 0114 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Interruption occured during Line construction(n=0-1) | CSI2[n]_INT_ST_LINE | RC | <CSI2[n]> + H'0 0130 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for interruption occured during Line construction(n=0-1) | CSI2[n]_INT_MSK_LINE | R/W | <CSI2[n]> + H'0 0134 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Fatal Interruption caused by Frame Boundaries(n=0-1) | CSI2[n]_INT_ST_BNDRY_FRAME_FATAL | RC | <CSI2[n]> + H'0 0280 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for fatal interruption caused by Frame Boundaries(n=0-1) | CSI2[n]_INT_MSK_BNDRY_FRAME_FATAL | R/W | <CSI2[n]> + H'0 0284 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Fatal Interruption caused by Frame Sequence(n=0-1) | CSI2[n]_INT_ST_SEQ_FRAME_FATAL | RC | <CSI2[n]> + H'0 0290 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for fatal interruption caused by Frame Sequence(n=0-1) | CSI2[n]_INT_MSK_SEQ_FRAME_FATAL | R/W | <CSI2[n]> + H'0 0294 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Fatal Interruption caused by Payload CRC(n=0-1) | CSI2[n]_INT_ST_PLD_CRC_FATAL | RC | <CSI2[n]> + H'0 02B0 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for fatal interruption caused by Payload CRC(n=0-1) | CSI2[n]_INT_MSK_PLD_CRC_FATAL | R/W | <CSI2[n]> + H'0 02B4 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Interruption caused by Data Type(n=0-1) | CSI2[n]_INT_ST_DATA_ID | RC | <CSI2[n]> + H'0 02C0 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for interruption caused by Data Type(n=0-1) | CSI2[n]_INT_MSK_DATA_ID | R/W | <CSI2[n]> + H'0 02C4 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Interruption caused by Header single bit errors(n=0-1) | CSI2[n]_INT_ST_ECC_CORRECTED | RC | <CSI2[n]> + H'0 02D0 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for interruption caused by Header single bit errors(n=0-1) | CSI2[n]_INT_MSK_ECC_CORRECTED | R/W | <CSI2[n]> + H'0 02D4 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Functional AP fatal interruption caused by PHY(n=0-1) | CSI2[n]_INT_ST_FAP_PHY_FATAL | RC | <CSI2[n]> + H'0 0360 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for functional AP fatal interruption caused by PHY(n=0-1) | CSI2[n]_INT_MSK_FAP_PHY_FATAL | R/W | <CSI2[n]> + H'0 0364 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Force for functional AP fatal interruption caused by PHY(n=0-1) | CSI2[n]_INT_FORCE_FAP_PHY_FATAL | R/W | <CSI2[n]> + H'0 0368 | H'0000 0000 | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| CSI2 | CSI2 [n] Functional AP fatal interruption caused during Packet Construction(n=0-1) | CSI2[n]_INT_ST_FAP_PKT_FATAL | R | <CSI2[n]> + H'0 0370 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for functional AP fatal interruption caused during Packet Construction(n=0-1) | CSI2[n]_INT_MSK_FAP_PKT_FATAL | R/W | <CSI2[n]> + H'0 0374 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Force for functional AP fatal interruption caused during Packet Construction(n=0-1) | CSI2[n]_INT_FORCE_FAP_PKT_FATAL | R/W | <CSI2[n]> + H'0 0378 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Functional AP interruption caused by PHY(n=0-1) | CSI2[n]_INT_ST_FAP_PHY | RC | <CSI2[n]> + H'0 0390 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for functional AP interruption caused by PHY(n=0-1) | CSI2[n]_INT_MSK_FAP_PHY | R/W | <CSI2[n]> + H'0 0394 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Force for functional AP interruption caused by PHY(n=0-1) | CSI2[n]_INT_FORCE_FAP_PHY | R/W | <CSI2[n]> + H'0 0398 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Functional AP fatal Interruption caused by Frame Boundaries(n=0-1) | CSI2[n]_INT_ST_FAP_BNDRY_FRAME_FATAL | RC | <CSI2[n]> + H'0 0420 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for functional AP fatal Interruption caused by Frame Boundaries(n=0-1) | CSI2[n]_INT_MSK_FAP_BNDRY_FRAME_FATAL | R/W | <CSI2[n]> + H'0 0424 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Force for functional AP fatal Interruption caused by Frame Boundaries(n=0-1) | CSI2[n]_INT_FORCE_FAP_BNDRY_FRAME_FATAL | R/W | <CSI2[n]> + H'0 0428 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Functional AP fatal interruption caused by Frame Sequence(n=0-1) | CSI2[n]_INT_ST_FAP_SEQ_FRAME_FATAL | RC | <CSI2[n]> + H'0 0430 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for functional AP fatal interruption caused by Frame Sequence(n=0-1) | CSI2[n]_INT_MSK_FAP_SEQ_FRAME_FATAL | R/W | <CSI2[n]> + H'0 0434 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Force for functional AP fatal interruption caused by Frame Sequence(n=0-1) | CSI2[n]_INT_FORCE_FAP_SEQ_FRAME_FATAL | R/W | <CSI2[n]> + H'0 0438 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Functional AP fatal interruption caused by Payload CRC(n=0-1) | CSI2[n]_INT_ST_FAP_PLD_CRC_FATAL | RC | <CSI2[n]> + H'0 0450 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for functional AP fatal interruption caused by Payload CRC(n=0-1) | CSI2[n]_INT_MSK_FAP_PLD_CRC_FATAL | R/W | <CSI2[n]> + H'0 0454 | H'0000 0000 | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| CSI2 | CSI2 [n] Force for functional AP fatal interruption caused by Payload CRC(n=0-1) | CSI2[n]_INT_FORCE_FAP_PLD_CRC_FATAL | R/W | <CSI2[n]> + H'0 0458 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Functional AP interruption caused by Header single bit errors(n=0-1) | CSI2[n]_INT_ST_FAP_ECC_CORRECTED | RC | <CSI2[n]> + H'0 0470 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Mask for functional AP interruption caused by Header single bit errors(n=0-1) | CSI2[n]_INT_MSK_FAP_ECC_CORRECTED | R/W | <CSI2[n]> + H'0 0474 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Force for functional AP interruption caused by Header single bit errors(n=0-1) | CSI2[n]_INT_FORCE_FAP_ECC_CORRECTED | R/W | <CSI2[n]> + H'0 0478 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Field Detect Control Register(n=0-1) | CSI2[n]_FLDC | R/W | <CSI2[n]> + H'0 0804 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Field Detect Data Register(n=0-1) | CSI2[n]_FLDD | R/W | <CSI2[n]> + H'0 0808 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] IDI Control Register(n=0-1) | CSI2[n]_IDIC | R/W | <CSI2[n]> + H'0 0810 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] Signal Override1 Register(n=0-1) | CSI2[n]_OVR1 | R/W | <CSI2[n]> + H'0 0848 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY ENABLE Register(n=0-1) | CSI2[n]_PHY_EN | R/W | <CSI2[n]> + H'0 2000 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY Force RX Mode Control Register(n=0-1) | CSI2[n]_FRXM | R/W | <CSI2[n]> + H'0 2004 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY Frequency Control Register(n=0-1) | CSI2[n]_PHYPLL | R/W | <CSI2[n]> + H'0 2050 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] CSI0CLK Frequency Configuration Preset Register(n=0-1) | CSI2[n]_CSI0CLKFCPR | R/W | <CSI2[n]> + H'0 2054 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY Test Interface Write Register(n=0-1) | CSI2[n]_PHTW | R/W | <CSI2[n]> + H'0 2060 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY Test Interface Read Register(n=0-1) | CSI2[n]_PHTR | R | <CSI2[n]> + H'0 2064 | H'0000 0000 | 32 |
| CSI2 | CSI2 [n] PHY Test Interface Clear Register(n=0-1) | CSI2[n]_PHTC | R/W | <CSI2[n]> + H'0 2068 | H'0000 0001 | 32 |
| CSI2 | CSI2 [n] PHY Status Register(n=0-1) | CSI2[n]_ST_PHYST | R | <CSI2[n]> + H'0 2814 | H'0001 0F00 | 32 |

### 53.1.5 Connected Module

Table 53.2 Connected Module

| Connected Modules | Function of the Connected Modules |
|---|---|
| CPG | Output and control of the clock signals, execution of software reset |

| Connected Modules | Function of the Connected Modules |
|---|---|
| INTC | Interrupt controller |
| ISP/CS | Image Signal Processor / Channel Selector |
| ECM | Module related to Functional Safety |

For details about interrupt channel, refer to chapter Interrupt Controller (see 15.INTC).

## 53.2 Register Description

[Legend]

Initial value: Register value after a reset

—: Undefined value

R/W: Readable/writable. The written value can be read.

R/WC1: Readable/writable. Writing 1 initializes the bit. Writing 0 is ignored.

R: Read-only. The write value should always be 0.

—/WB: Write-only. The read value is undefined.

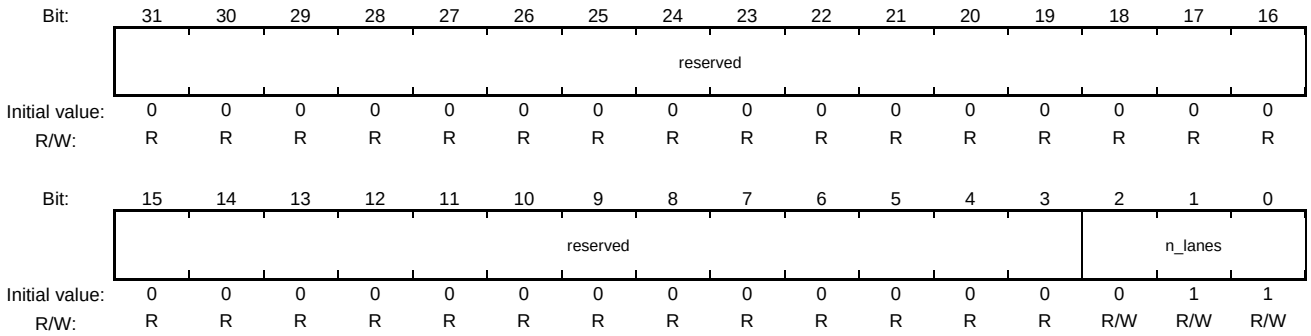
### 53.2.1 CSI2 [n] Core version(n=0-1)(CSI2[n]_VERSION)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | version | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | version | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

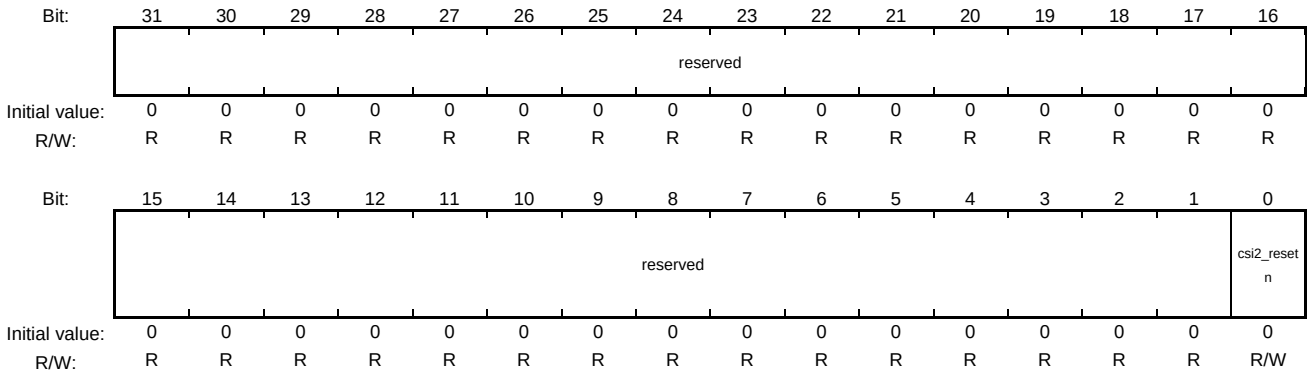
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | version | H'3135 312A | R | This field indicates the version of the LINK. |

### 53.2.2 CSI2 [n] Number of lanes(n=0-1)(CSI2[n]_N_LANES)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | reserved | All 0 | R | Reserved and read as zero. |
| 2 to 0 | n_lanes | B'011 | R/W | This can only be updated when the PHY lane is in stopstate.Number of active data lanes:<br>Values:<br>0x0 (DATALANE1): 1 Data Lane<br>0x1 (DATALANE2): 2 Data Lanes<br>0x2 (DATALANE3): 3 Data Lanes<br>0x3 (DATALANE4): 4 Data Lanes |

### 53.2.3 CSI2 [n] Logic Reset(n=0-1)(CSI2[n]_CSI2_RESETN)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | reserved | All 0 | R | Reserved and read as zero. |
| 0 | csi2_resetn | B'0 | R/W | LINK and PHY reset output. Active Low. |

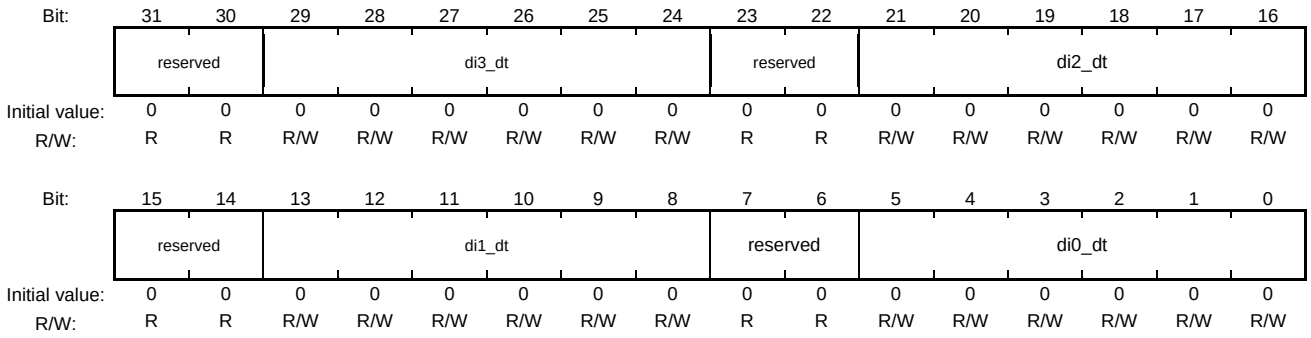
### 53.2.4 CSI2 [n] Main interrupt status(n=0-1)(CSI2[n]_INT_ST_MAIN)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | status_int_line | status_int_phy |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | RC | RC |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | status_int_ecc_corrected | status_int_data_id | status_int_pld_crc_fatal | status_int_crc_frame_fatal | status_int_seq_frame_fatal | status_int_bndry_frame_fatal | status_int_pkt_fatal | status_int_phy_fatal |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | RC | RC | RC | RC | RC | RC | RC | RC |

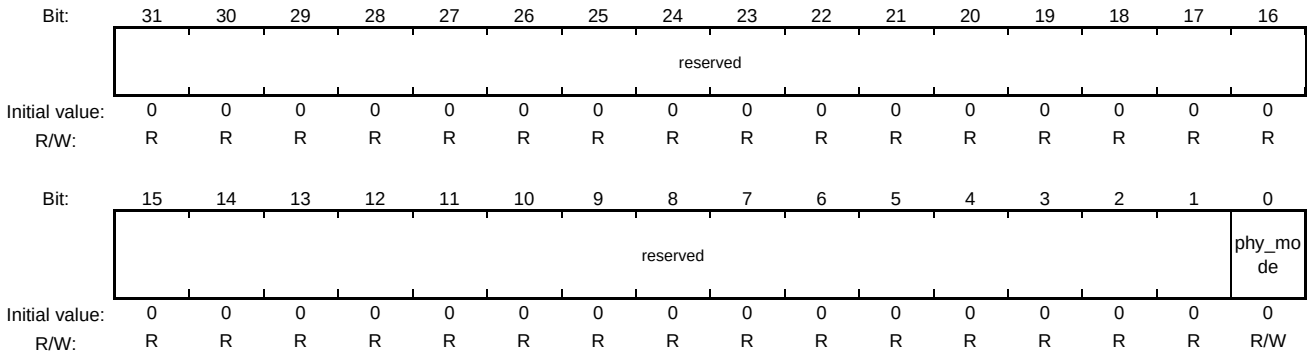
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | reserved | All 0 | R | Reserved and read as zero. |
| 25 | reserved | B'0 | R | Reserved and read as zero. |
| 24 | reserved | B'0 | R | Reserved and read as zero. |
| 23 | reserved | B'0 | R | Reserved and read as zero. |
| 22 | reserved | B'0 | R | Reserved and read as zero. |
| 21 | reserved | B'0 | R | Reserved and read as zero. |
| 20 | reserved | B'0 | R | Reserved and read as zero. |
| 19 | reserved | B'0 | R | Reserved and read as zero. |
| 18 | reserved | B'0 | R | Reserved and read as zero. |
| 17 | status_int_line | B'0 | RC | Status of int_st_line |
| 16 | status_int_phy | B'0 | RC | Status of int_st_phy. |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | status_int_ecc_corrected | B'0 | RC | Status of int_ecc_corrected. |
| 6 | status_int_data_id | B'0 | RC | Status of int_data_id. |
| 5 | status_int_pld_crc_fatal | B'0 | RC | Status of int_pld_crc_fatal. |
| 4 | status_int_crc_frame_fatal | B'0 | RC | Status of int_crc_frame_fatal. |
| 3 | status_int_seq_frame_fatal | B'0 | RC | Status of int_seq_frame_fatal. |
| 2 | status_int_bndry_frame_fatal | B'0 | RC | Status of int_st_bndry_frame_fatal. |
| 1 | status_int_pkt_fatal | B'0 | RC | Status of int_st_pkt_fatal. |
| 0 | status_int_phy_fatal | B'0 | RC | Status of int_st_phy_fatal. |

### 53.2.5 CSI2 [n] Data Type fields for Data ID Monitors 1(n=0-1)(CSI2[n]_DATA_IDS_1)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | reserved | All 0 | R | Reserved and read as zero. |
| 29 to 24 | di3_dt | All 0 | R/W | Data type for programmed data ID 3. |
| 23, 22 | reserved | All 0 | R | Reserved and read as zero. |
| 21 to 16 | di2_dt | All 0 | R/W | Data type for programmed Data ID 2. |
| 15, 14 | reserved | All 0 | R | Reserved and read as zero. |
| 13 to 8 | di1_dt | All 0 | R/W | Data type for programmed data ID 1. |
| 7, 6 | reserved | All 0 | R | Reserved and read as zero. |
| 5 to 0 | di0_dt | All 0 | R/W | Data type for programmed data ID 0. |

### 53.2.6 CSI2 [n] PHY Mode(n=0-1)(CSI2[n]_PHY_MODE)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | reserved | All 0 | R | Reserved and read as zero. |
| 0 | phy_mode | B'0 | R/W | Select the PHY interface:<br>Values:<br>0x0 (DPHY): Select D-PHY<br>0x1 : prohibit |

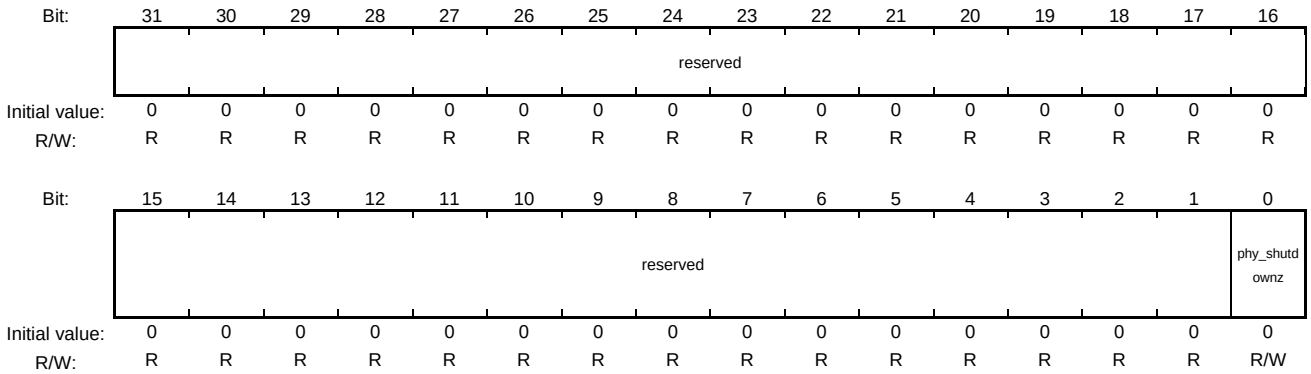
### 53.2.7 CSI2 [n] Virtual Channel fields for Data ID Monitors 1(n=0-1)(CSI2[n]_DATA_IDS_VC_1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | di3_vcx_2 | di3_vcx_0_1 | | di3_vc | | reserved | | | di2_vcx_2 | di2_vcx_0_1 | | di2_vc | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | di1_vcx_2 | di1_vcx_0_1 | | di1_vc | | reserved | | | di0_vcx_2 | di0_vcx_0_1 | | di0_vc | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

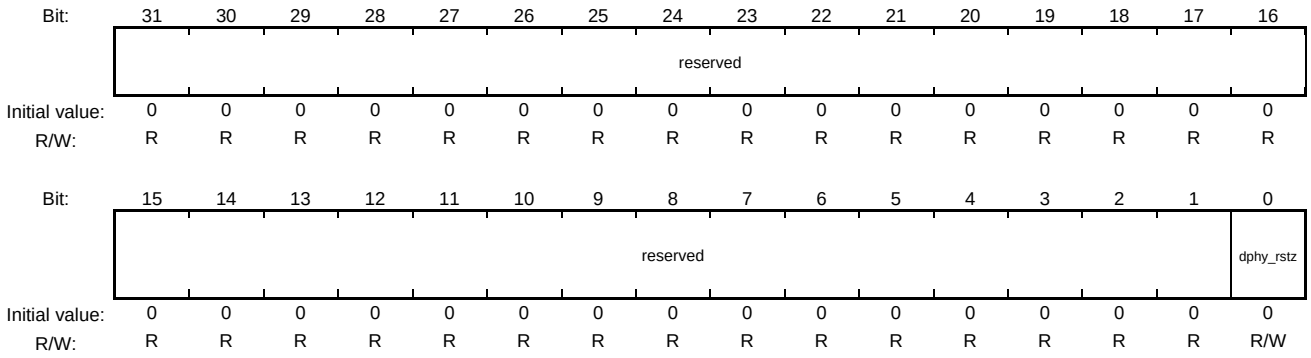
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | reserved | All 0 | R | Reserved and read as zero. |
| 28 | di3_vcx_2 | B'0 | R/W | Virtual channel extension extra bit for programmed data ID 3 |
| 27, 26 | di3_vcx_0_1 | All 0 | R/W | Virtual channel extension for programmed data ID 3. |
| 25, 24 | di3_vc | All 0 | R/W | Virtual channel for programmed data ID 3. |
| 23 to 21 | reserved | All 0 | R | Reserved and read as zero. |
| 20 | di2_vcx_2 | B'0 | R/W | Virtual channel extension extra bit for programmed data ID 2 |
| 19, 18 | di2_vcx_0_1 | All 0 | R/W | Virtual channel extension for programmed data ID 2. |
| 17, 16 | di2_vc | All 0 | R/W | Virtual channel for programmed data ID 2. |
| 15 to 13 | reserved | All 0 | R | Reserved and read as zero. |
| 12 | di1_vcx_2 | B'0 | R/W | Virtual channel extension extra bit for programmed data ID 1 |
| 11, 10 | di1_vcx_0_1 | All 0 | R/W | Virtual channel extension for programmed data ID 1. |
| 9, 8 | di1_vc | All 0 | R/W | Virtual channel for programmed data ID 1. |
| 7 to 5 | reserved | All 0 | R | Reserved and read as zero. |
| 4 | di0_vcx_2 | B'0 | R/W | Virtual channel extension extra bit for programmed data ID 0 |
| 3, 2 | di0_vcx_0_1 | All 0 | R/W | Virtual channel extension for programmed data ID 0. |
| 1, 0 | di0_vc | All 0 | R/W | Virtual channel for programmed data ID 0. |

### 53.2.8 CSI2 [n] PHY Shutdown(n=0-1)(CSI2[n]_PHY_SHUTDOWNZ)



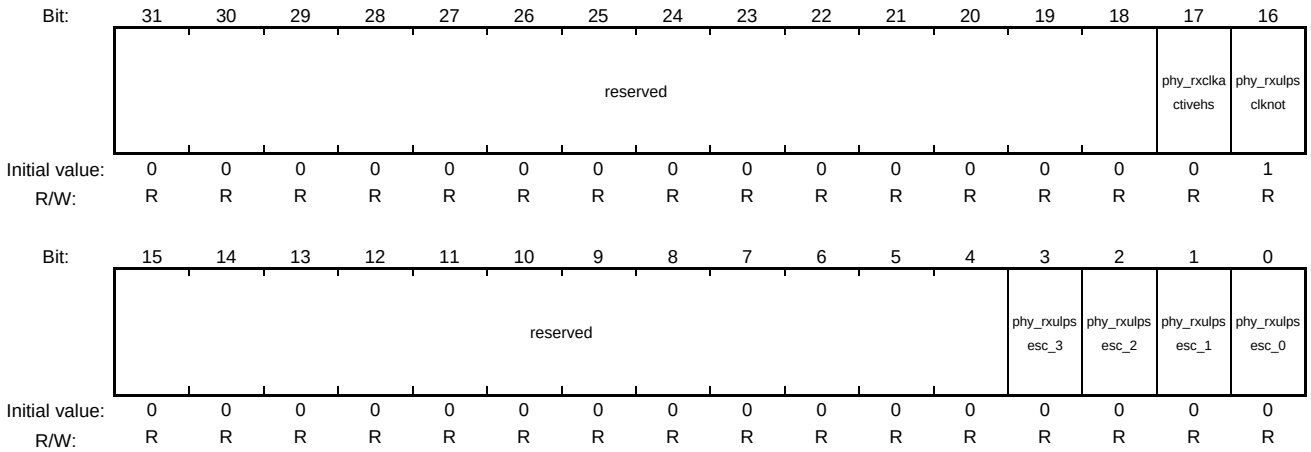

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | reserved | All 0 | R | Reserved and read as zero. |
| 0 | phy_shutdownz | B'0 | R/W | Shutdown input. This line is used to place the complete macro in power down. All analog blocks are in power down mode and digital logic is cleared. Active Low. |

### 53.2.9 CSI2 [n] DPHY reset(n=0-1)(CSI2[n]_DPHY_RSTZ)



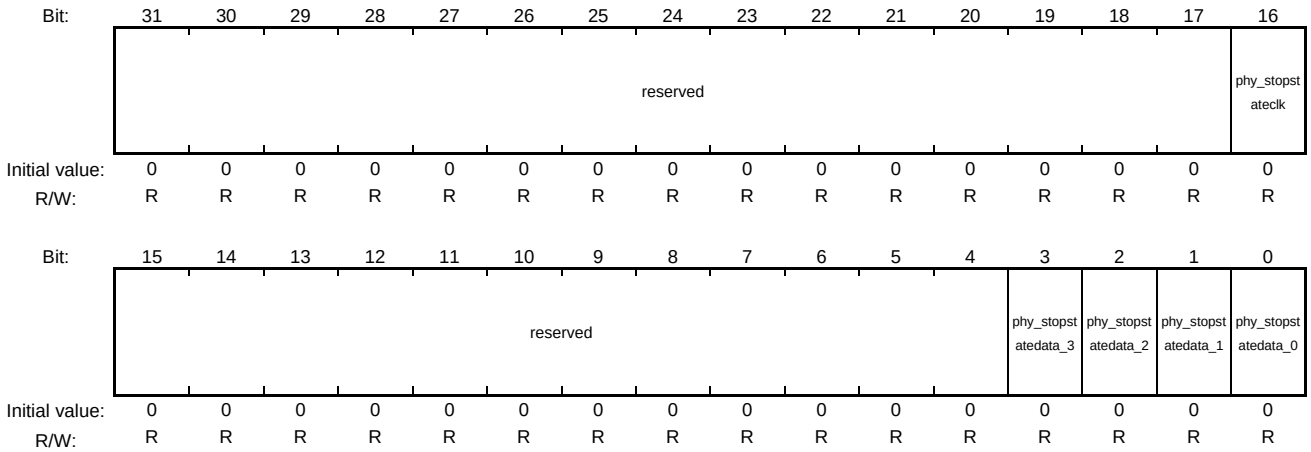

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | reserved | All 0 | R | Reserved and read as zero. |
| 0 | dphy_rstz | B'0 | R/W | PHY reset output. Active Low. |

### 53.2.10 CSI2 [n] RX PHY status(n=0-1)(CSI2[n]_PHY_RX)



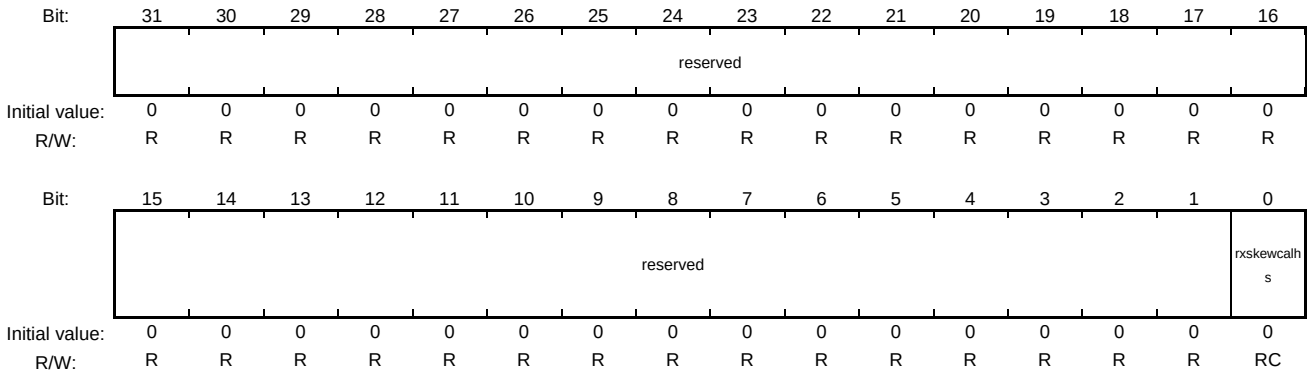

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | reserved | All 0 | R | Reserved and read as zero. |
| 17 | phy_rxclkactivehs | B'0 | R | Indicates that D-PHY clock lane is actively receiving a DDR clock |
| 16 | phy_rxulpsclknot | B'1 | R | Active Low. This signal indicates that D-PHY Clock Lane module has entered the Ultra Low Power state |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | phy_rxulpsesc_3 | B'0 | R | Lane module 3 has entered the Ultra Low Power mode |
| 2 | phy_rxulpsesc_2 | B'0 | R | Lane module 2 has entered the Ultra Low Power mode |
| 1 | phy_rxulpsesc_1 | B'0 | R | Lane module 1 has entered the Ultra Low Power mode |
| 0 | phy_rxulpsesc_0 | B'0 | R | Lane module 0 has entered the Ultra Low Power mode. |

### 53.2.11 CSI2 [n] STOPSTATE PHY status(n=0-1)(CSI2[n]_PHY_STOPSTATE)



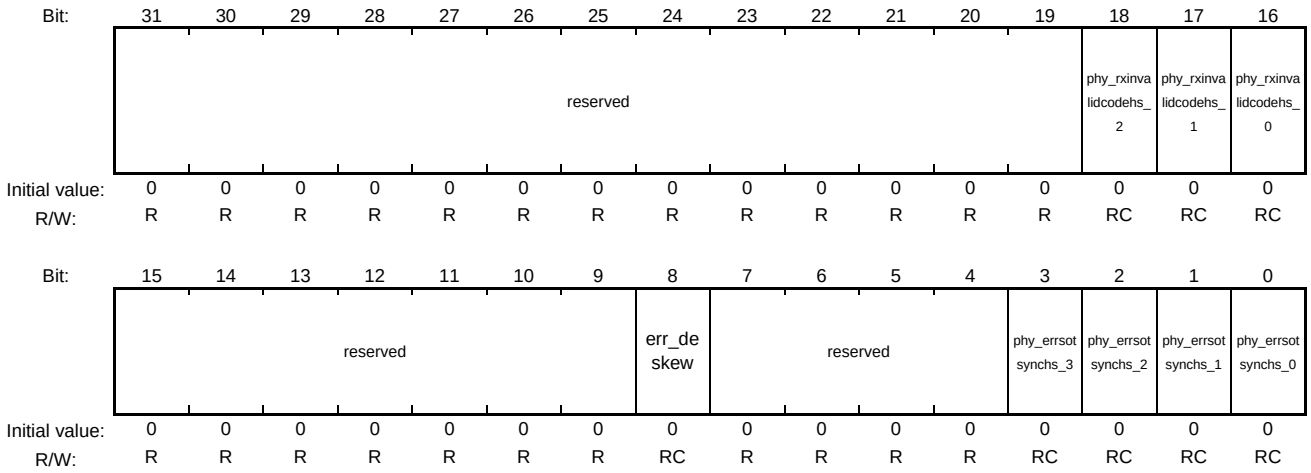

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | reserved | All 0 | R | Reserved and read as zero. |
| 16 | phy_stopstateclk | B'0 | R | D-PHY Clock lane in Stop state |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | phy_stopstatedata_3 | B'0 | R | Data lane 3 in Stop state |
| 2 | phy_stopstatedata_2 | B'0 | R | Data lane 2 in Stop state |
| 1 | phy_stopstatedata_1 | B'0 | R | Data lane 1 in Stop state |
| 0 | phy_stopstatedata_0 | B'0 | R | Data lane 0 in Stop state |

### 53.2.12 CSI2 [n] PHY Calibration(n=0-1)(CSI2[n]_PHY_CAL)



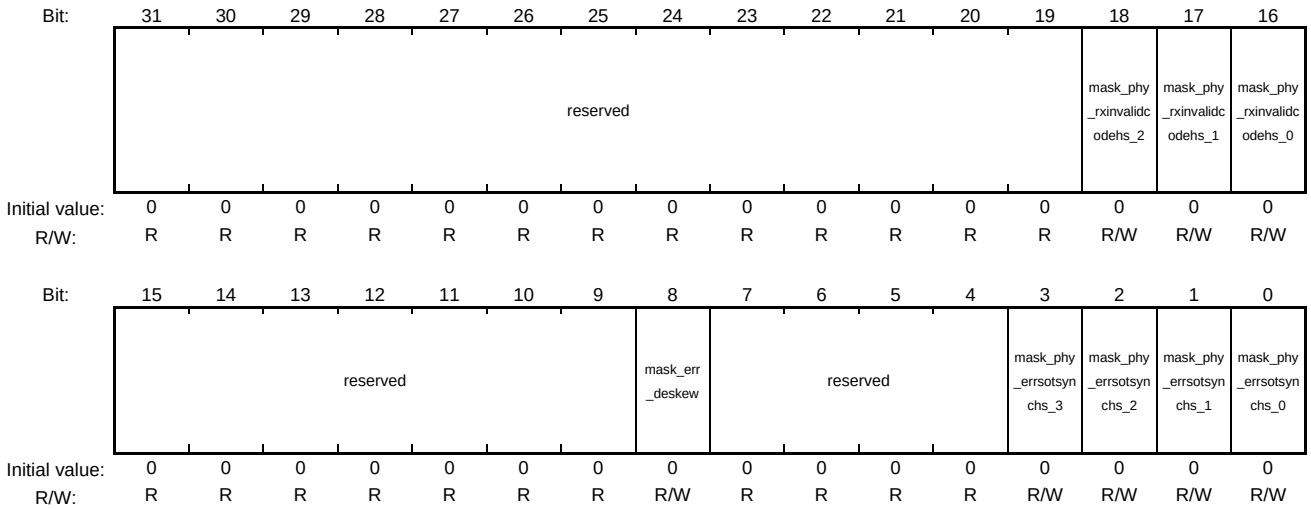

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | reserved | All 0 | R | Reserved and read as zero. |
| 0 | rxskewcalhs | B'0 | RC | A low-to-high transition on rxskewcalhs signal means that the PHY has initiated the de-skew calibration. |

### 53.2.13 CSI2 [n] Fatal interruption caused by PHY(n=0-1)(CSI2[n]_INT_ST_PHY_FATAL)



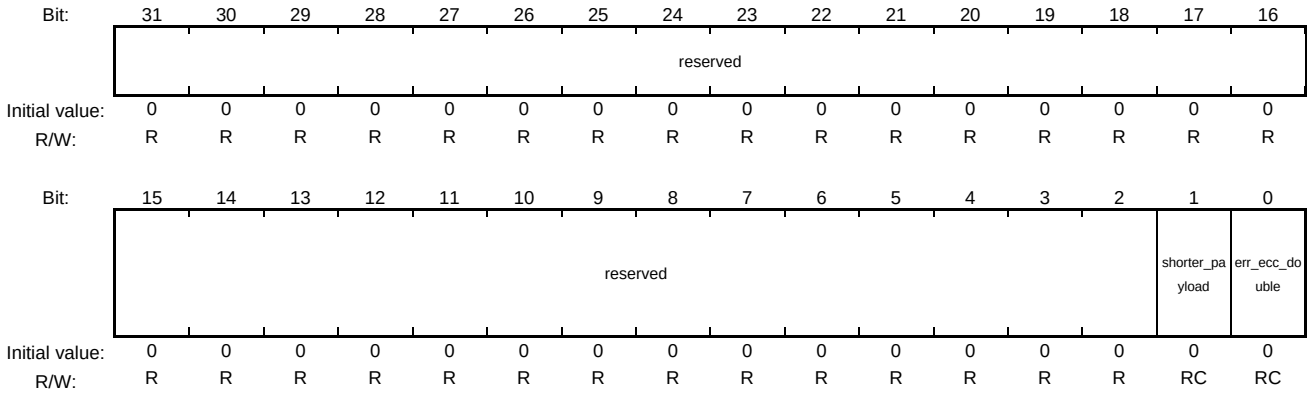

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | reserved | All 0 | R | Reserved and read as zero. |
| 18 | phy_rxinvalidcodehs_2 | B'0 | RC | High Speed Invalid Code Word Detection on lane 2 |
| 17 | phy_rxinvalidcodehs_1 | B'0 | RC | High Speed Invalid Code Word Detection on lane 1 |
| 16 | phy_rxinvalidcodehs_0 | B'0 | RC | High Speed Invalid Code Word Detection on lane 0 |
| 15 to 9 | reserved | All 0 | R | Reserved and read as zero. |
| 8 | err_deskew | B'0 | RC | Reports whenever data is lost due to an existent skew between lanes greater than 2 rxwordclkhs |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | phy_errsotsynchs_3 | B'0 | RC | Start of transmission error on data lane 3 (no synchronization achieved) |
| 2 | phy_errsotsynchs_2 | B'0 | RC | Start of transmission error on data lane 2 (no synchronization achieved) |
| 1 | phy_errsotsynchs_1 | B'0 | RC | Start of transmission error on data lane 1 (no synchronization achieved) |
| 0 | phy_errsotsynchs_0 | B'0 | RC | Start of transmission error on data lane 0 (no synchronization achieved). |

### 53.2.14 CSI2 [n] Mask for fatal interruption caused by PHY(n=0-1)(CSI2[n]_INT_MSK_PHY_FATAL)



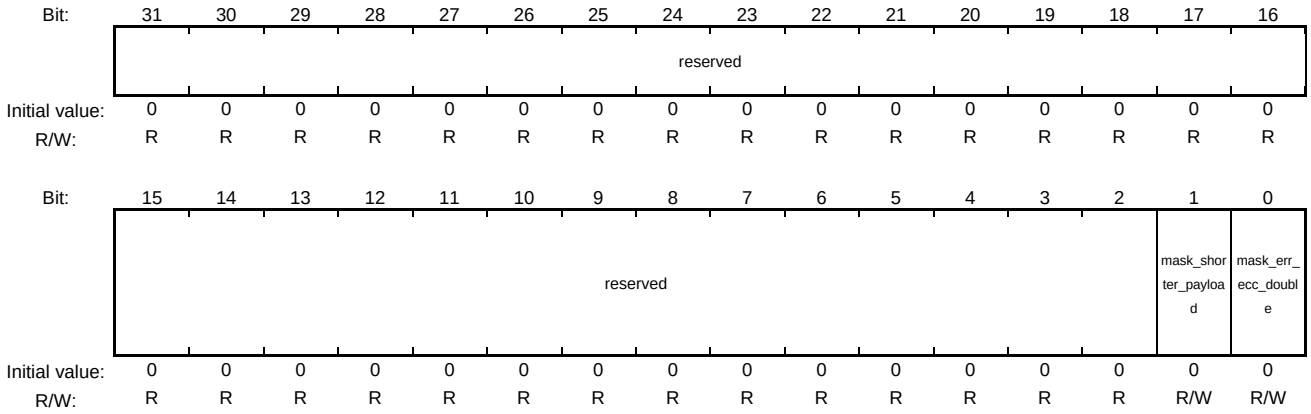

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | reserved | All 0 | R | Reserved and read as zero. |
| 18 | mask_phy_rxinvalidcodehs_2 | B'0 | R/W | Mask for phy_rxinvalidcodehs_2 |
| 17 | mask_phy_rxinvalidcodehs_1 | B'0 | R/W | Mask for phy_rxinvalidcodehs_1 |
| 16 | mask_phy_rxinvalidcodehs_0 | B'0 | R/W | Mask for phy_rxinvalidcodehs_0 |
| 15 to 9 | reserved | All 0 | R | Reserved and read as zero. |
| 8 | mask_err_deskew | B'0 | R/W | Mask for err_deskew |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | mask_phy_errsotsynchs_3 | B'0 | R/W | Mask for phy_errsotsynchs_3 |
| 2 | mask_phy_errsotsynchs_2 | B'0 | R/W | Mask for phy_errsotsynchs_2 |
| 1 | mask_phy_errsotsynchs_1 | B'0 | R/W | Mask for phy_errsotsynchs_1 |
| 0 | mask_phy_errsotsynchs_0 | B'0 | R/W | Mask for phy_errsotsynchs_0 |

### 53.2.15 CSI2 [n] Fatal interruption caused during Packet Construction(n=0-1)(CSI2[n]_INT_ST_PKT_FATAL)



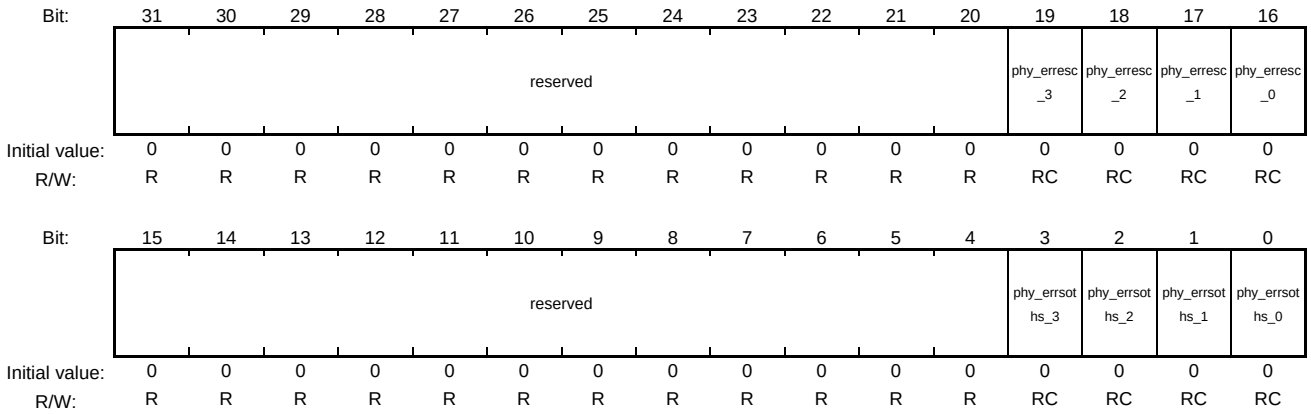

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | reserved | All 0 | R | Reserved and read as zero. |
| 1 | shorter_payload | B'0 | RC | D-PHY mode: Reported greater WC than received, unrecoverable. |
| 0 | err_ecc_double | B'0 | RC | D-PHY mode: Header ECC contains at least 2 errors, unrecoverable. |

### 53.2.16 CSI2 [n] Mask for fatal interruption caused during Packet Construction(n=0-1)(CSI2[n]_INT_MSK_PKT_FATAL)



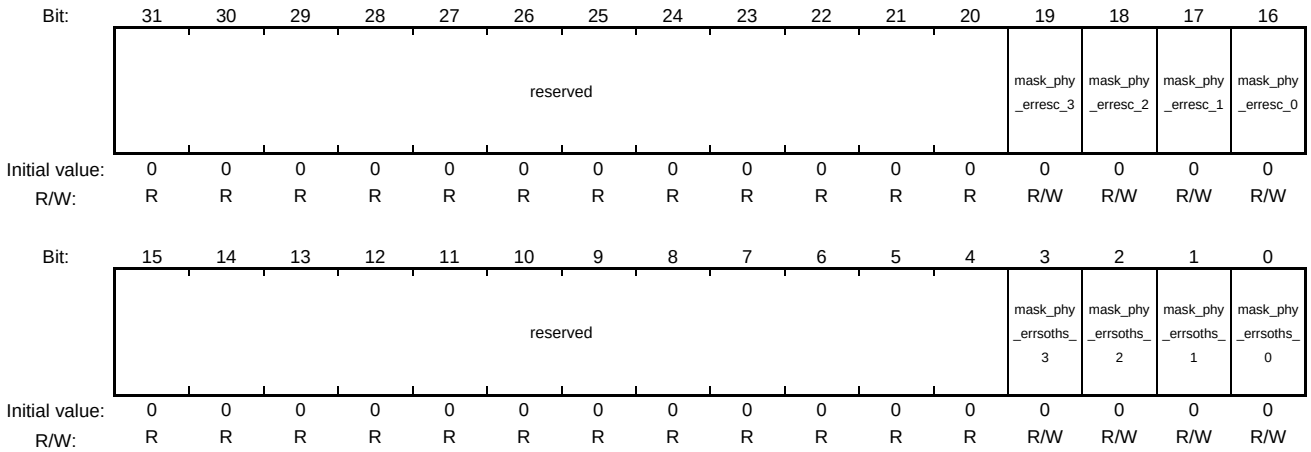

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | reserved | All 0 | R | Reserved and read as zero. |
| 1 | mask_shorter_payload | B'0 | R/W | Mask for shorter_payload. |
| 0 | mask_err_ecc_double | B'0 | R/W | Mask for err_ecc_double. |

### 53.2.17 CSI2 [n] Interruption caused by PHY(n=0-1)(CSI2[n]_INT_ST_PHY)



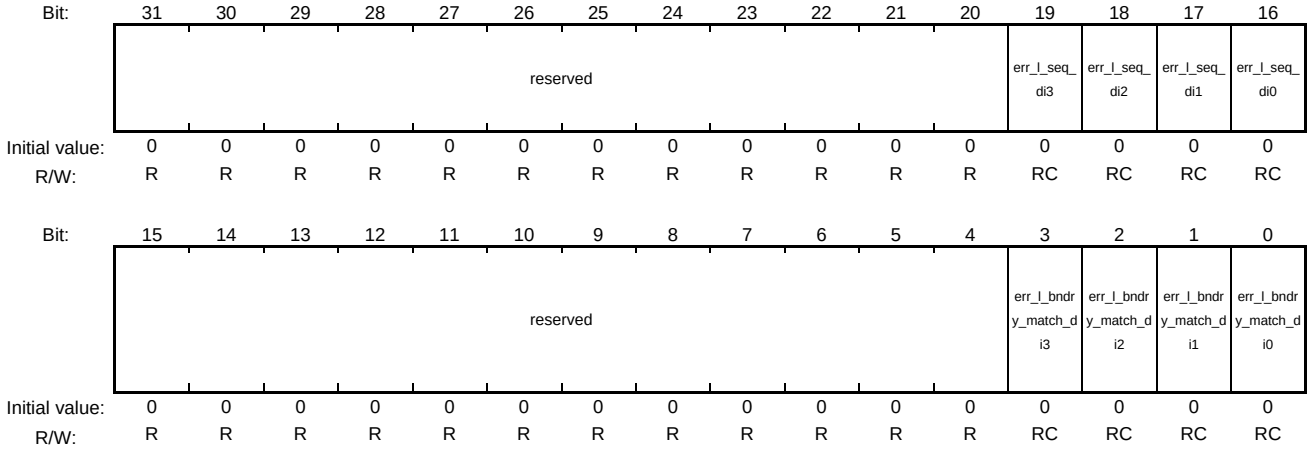

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R | Reserved and read as zero. |
| 23 | reserved | B'0 | R | Reserved and read as zero. |
| 22 | reserved | B'0 | R | Reserved and read as zero. |
| 21 | reserved | B'0 | R | Reserved and read as zero. |
| 20 | reserved | B'0 | R | Reserved and read as zero. |
| 19 | phy_erresc_3 | B'0 | RC | Escape Entry Error on data lane 3 |
| 18 | phy_erresc_2 | B'0 | RC | Escape Entry Error on data lane 2 |
| 17 | phy_erresc_1 | B'0 | RC | Escape Entry Error on data lane 1 |
| 16 | phy_erresc_0 | B'0 | RC | Escape Entry Error on data lane 0 |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | phy_errsoths_3 | B'0 | RC | Start of transmission error on data lane 3 (synchronization can still be achieved) |
| 2 | phy_errsoths_2 | B'0 | RC | Start of transmission error on data lane 2 (synchronization can still be achieved) |
| 1 | phy_errsoths_1 | B'0 | RC | Start of transmission error on data lane 1 (synchronization can still be achieved) |
| 0 | phy_errsoths_0 | B'0 | RC | Start of transmission error on data lane 0 (synchronization can still be achieved) |

### 53.2.18 CSI2 [n] Mask for interruption caused by PHY(n=0-1)(CSI2[n]_INT_MSK_PHY)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R | Reserved and read as zero. |
| 23 | reserved | B'0 | R | Reserved and read as zero. |
| 22 | reserved | B'0 | R | Reserved and read as zero. |
| 21 | reserved | B'0 | R | Reserved and read as zero. |
| 20 | reserved | B'0 | R | Reserved and read as zero. |
| 19 | mask_phy_erresc_3 | B'0 | R/W | Mask for phy_erresc_3 |
| 18 | mask_phy_erresc_2 | B'0 | R/W | Mask for phy_erresc_2 |
| 17 | mask_phy_erresc_1 | B'0 | R/W | Mask for phy_erresc_1 |
| 16 | mask_phy_erresc_0 | B'0 | R/W | Mask for phy_erresc_0 |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | mask_phy_errsoths_3 | B'0 | R/W | Mask for phy_errsoths_3 |
| 2 | mask_phy_errsoths_2 | B'0 | R/W | Mask for phy_errsoths_2 |
| 1 | mask_phy_errsoths_1 | B'0 | R/W | Mask for phy_errsoths_1 |
| 0 | mask_phy_errsoths_0 | B'0 | R/W | Mask for phy_errsoths_0 |

### 53.2.19 CSI2 [n] Interruption occured during Line construction(n=0-1)(CSI2[n]_INT_ST_LINE)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R | Reserved and read as zero. |
| 23 | reserved | B'0 | R | Reserved and read as zero. |
| 22 | reserved | B'0 | R | Reserved and read as zero. |
| 21 | reserved | B'0 | R | Reserved and read as zero. |
| 20 | reserved | B'0 | R | Reserved and read as zero. |
| 19 | err_l_seq_di3 | B'0 | RC | Error in the sequence of lines for vc3 and dt3 |
| 18 | err_l_seq_di2 | B'0 | RC | Error in the sequence of lines for vc2 and dt2 |
| 17 | err_l_seq_di1 | B'0 | RC | Error in the sequence of lines for vc1 and dt1 |
| 16 | err_l_seq_di0 | B'0 | RC | Error in the sequence of lines for vc0 and dt0 |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | err_l_bndry_match_di3 | B'0 | RC | Error matching line start with line end for vc3 and dt3 |
| 2 | err_l_bndry_match_di2 | B'0 | RC | Error matching line start with line end for vc2 and dt2 |
| 1 | err_l_bndry_match_di1 | B'0 | RC | Error matching line start with line end for vc1 and dt1 |
| 0 | err_l_bndry_match_di0 | B'0 | RC | Error matching line start with line end for vc0 and dt0 |

### 53.2.20 CSI2 [n] Mask for interruption occured during Line construction(n=0-1)(CSI2[n]_INT_MSK_LINE)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | mask_err_l_seq_di3 | mask_err_l_seq_di2 | mask_err_l_seq_di1 | mask_err_l_seq_di0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | mask_err_l_bndry_match_di3 | mask_err_l_bndry_match_di2 | mask_err_l_bndry_match_di1 | mask_err_l_bndry_match_di0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R | Reserved and read as zero. |
| 23 | reserved | B'0 | R | Reserved and read as zero. |
| 22 | reserved | B'0 | R | Reserved and read as zero. |
| 21 | reserved | B'0 | R | Reserved and read as zero. |
| 20 | reserved | B'0 | R | Reserved and read as zero. |
| 19 | mask_err_l_seq_di3 | B'0 | R/W | Mask for err_l_seq_di3 |
| 18 | mask_err_l_seq_di2 | B'0 | R/W | Mask for err_l_seq_di2 |
| 17 | mask_err_l_seq_di1 | B'0 | R/W | Mask for err_l_seq_di1 |
| 16 | mask_err_l_seq_di0 | B'0 | R/W | Mask for err_l_seq_di0 |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | mask_err_l_bndry_match_di3 | B'0 | R/W | Mask for err_l_bndry_match_di3 |
| 2 | mask_err_l_bndry_match_di2 | B'0 | R/W | Mask for err_l_bndry_match_di2 |
| 1 | mask_err_l_bndry_match_di1 | B'0 | R/W | Mask for err_l_bndry_match_di1 |
| 0 | mask_err_l_bndry_match_di0 | B'0 | R/W | Mask for err_l_bndry_match_di0 |

### 53.2.21 CSI2 [n] Fatal Interruption caused by Frame Boundaries(n=0-1)(CSI2[n]_INT_ST_BNDRY_FRAME_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_f_bndry_match_vc15 | err_f_bndry_match_vc14 | err_f_bndry_match_vc13 | err_f_bndry_match_vc12 | err_f_bndry_match_vc11 | err_f_bndry_match_vc10 | err_f_bndry_match_vc9 | err_f_bndry_match_vc8 | err_f_bndry_match_vc7 | err_f_bndry_match_vc6 | err_f_bndry_match_vc5 | err_f_bndry_match_vc4 | err_f_bndry_match_vc3 | err_f_bndry_match_vc2 | err_f_bndry_match_vc1 | err_f_bndry_match_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | reserved | All 0 | R | Reserved and read as zero. |
| 15 | err_f_bndry_match_vc15 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 15 |
| 14 | err_f_bndry_match_vc14 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 14 |
| 13 | err_f_bndry_match_vc13 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 13 |
| 12 | err_f_bndry_match_vc12 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 12 |
| 11 | err_f_bndry_match_vc11 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 11 |
| 10 | err_f_bndry_match_vc10 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 10 |
| 9 | err_f_bndry_match_vc9 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 9 |
| 8 | err_f_bndry_match_vc8 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 8 |
| 7 | err_f_bndry_match_vc7 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 7 |
| 6 | err_f_bndry_match_vc6 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 6 |
| 5 | err_f_bndry_match_vc5 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 5 |
| 4 | err_f_bndry_match_vc4 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 4 |
| 3 | err_f_bndry_match_vc3 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 3 |
| 2 | err_f_bndry_match_vc2 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 2 |
| 1 | err_f_bndry_match_vc1 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 1 |
| 0 | err_f_bndry_match_vc0 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 0 |

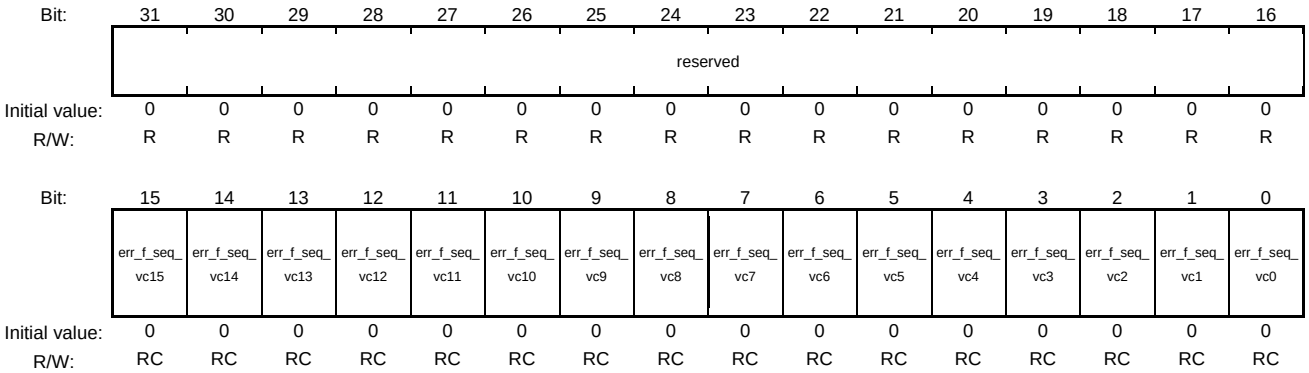
### 53.2.22 CSI2 [n] Mask for fatal interruption caused by Frame Boundaries(n=0-1)(CSI2[n]_INT_MSK_BNDRY_FRAME_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_f_bndry_match_vc15 | err_f_bndry_match_vc14 | err_f_bndry_match_vc13 | err_f_bndry_match_vc12 | err_f_bndry_match_vc11 | err_f_bndry_match_vc10 | err_f_bndry_match_vc9 | err_f_bndry_match_vc8 | err_f_bndry_match_vc7 | err_f_bndry_match_vc6 | err_f_bndry_match_vc5 | err_f_bndry_match_vc4 | err_f_bndry_match_vc3 | err_f_bndry_match_vc2 | err_f_bndry_match_vc1 | err_f_bndry_match_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

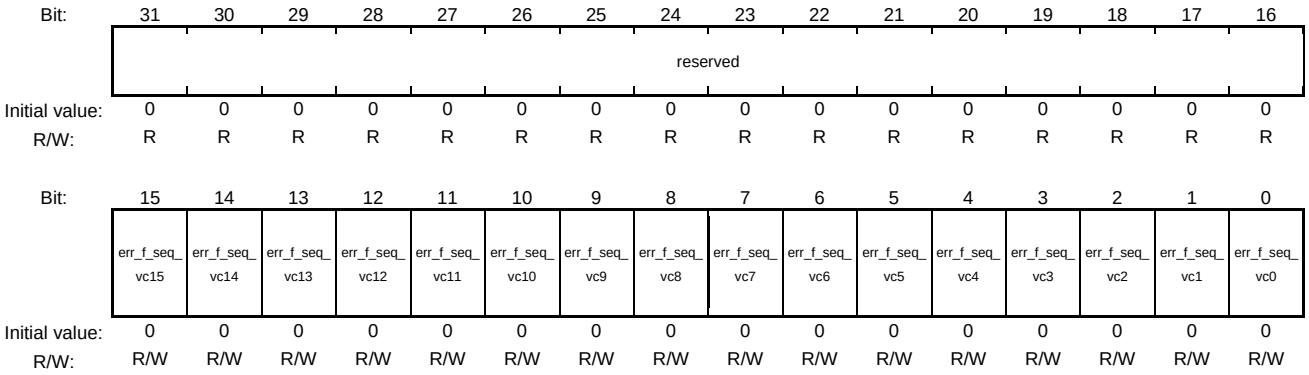
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | reserved | All 0 | R/W | Reserved and read as zero. |
| 15 | err_f_bndry_ match_vc15 | B'0 | R/W | Mask for err_f_bndry_match_vc15 |
| 14 | err_f_bndry_ match_vc14 | B'0 | R/W | Mask for err_f_bndry_match_vc14 |
| 13 | err_f_bndry_ match_vc13 | B'0 | R/W | Mask for err_f_bndry_match_vc13 |
| 12 | err_f_bndry_ match_vc12 | B'0 | R/W | Mask for err_f_bndry_match_vc12 |
| 11 | err_f_bndry_ match_vc11 | B'0 | R/W | Mask for err_f_bndry_match_vc11 |
| 10 | err_f_bndry_ match_vc10 | B'0 | R/W | Mask for err_f_bndry_match_vc10 |
| 9 | err_f_bndry_ match_vc9 | B'0 | R/W | Mask for err_f_bndry_match_vc9 |
| 8 | err_f_bndry_ match_vc8 | B'0 | R/W | Mask for err_f_bndry_match_vc8 |
| 7 | err_f_bndry_ match_vc7 | B'0 | R/W | Mask for err_f_bndry_match_vc7 |
| 6 | err_f_bndry_ match_vc6 | B'0 | R/W | Mask for err_f_bndry_match_vc6 |
| 5 | err_f_bndry_ match_vc5 | B'0 | R/W | Mask for err_f_bndry_match_vc5 |
| 4 | err_f_bndry_ match_vc4 | B'0 | R/W | Mask for err_f_bndry_match_vc4 |
| 3 | err_f_bndry_ match_vc3 | B'0 | R/W | Mask for err_f_bndry_match_vc3 |
| 2 | err_f_bndry_ match_vc2 | B'0 | R/W | Mask for err_f_bndry_match_vc2 |
| 1 | err_f_bndry_ match_vc1 | B'0 | R/W | Mask for err_f_bndry_match_vc1 |
| 0 | err_f_bndry_ match_vc0 | B'0 | R/W | Mask for err_f_bndry_match_vc0 |

### 53.2.23 CSI2 [n] Fatal Interruption caused by Frame Sequence(n=0-1)(CSI2[n]_INT_ST_SEQ_FRAME_FATAL)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | reserved | All 0 | R | Reserved and read as zero. |
| 15 | err_f_seq_vc15 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 15 |
| 14 | err_f_seq_vc14 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 14 |
| 13 | err_f_seq_vc13 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 13 |
| 12 | err_f_seq_vc12 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 12 |
| 11 | err_f_seq_vc11 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 11 |
| 10 | err_f_seq_vc10 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 10 |
| 9 | err_f_seq_vc9 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 9 |
| 8 | err_f_seq_vc8 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 8 |
| 7 | err_f_seq_vc7 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 7 |
| 6 | err_f_seq_vc6 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 6 |
| 5 | err_f_seq_vc5 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 5 |
| 4 | err_f_seq_vc4 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 4 |
| 3 | err_f_seq_vc3 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 3 |
| 2 | err_f_seq_vc2 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 2 |
| 1 | err_f_seq_vc1 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 1 |
| 0 | err_f_seq_vc0 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 0 |

### 53.2.24 CSI2 [n] Mask for fatal interruption caused by Frame Sequence(n=0-1)(CSI2[n]_INT_MSK_SEQ_FRAME_FATAL)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | reserved | All 0 | R | Reserved and read as zero. |
| 15 | err_f_seq_vc15 | B'0 | R/W | Mask for err_f_seq_vc15 |
| 14 | err_f_seq_vc14 | B'0 | R/W | Mask for err_f_seq_vc14 |
| 13 | err_f_seq_vc13 | B'0 | R/W | Mask for err_f_seq_vc13 |
| 12 | err_f_seq_vc12 | B'0 | R/W | Mask for err_f_seq_vc12 |
| 11 | err_f_seq_vc11 | B'0 | R/W | Mask for err_f_seq_vc11 |
| 10 | err_f_seq_vc10 | B'0 | R/W | Mask for err_f_seq_vc10 |
| 9 | err_f_seq_vc9 | B'0 | R/W | Mask for err_f_seq_vc9 |
| 8 | err_f_seq_vc8 | B'0 | R/W | Mask for err_f_seq_vc8 |
| 7 | err_f_seq_vc7 | B'0 | R/W | Mask for err_f_seq_vc7 |
| 6 | err_f_seq_vc6 | B'0 | R/W | Mask for err_f_seq_vc6 |
| 5 | err_f_seq_vc5 | B'0 | R/W | Mask for err_f_seq_vc5 |
| 4 | err_f_seq_vc4 | B'0 | R/W | Mask for err_f_seq_vc4 |
| 3 | err_f_seq_vc3 | B'0 | R/W | Mask for err_f_seq_vc3 |
| 2 | err_f_seq_vc2 | B'0 | R/W | Mask for err_f_seq_vc2 |
| 1 | err_f_seq_vc1 | B'0 | R/W | Mask for err_f_seq_vc1 |
| 0 | err_f_seq_vc0 | B'0 | R/W | Mask for err_f_seq_vc0 |

### 53.2.25 CSI2 [n] Fatal Interruption caused by Payload CRC(n=0-1)(CSI2[n]_INT_ST_PLD_CRC_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_crc_vc 15 | err_crc_vc 14 | err_crc_vc 13 | err_crc_vc 12 | err_crc_vc 11 | err_crc_vc 10 | err_crc_vc 9 | err_crc_vc 8 | err_crc_vc 7 | err_crc_vc 6 | err_crc_vc 5 | err_crc_vc 4 | err_crc_vc 3 | err_crc_vc 2 | err_crc_vc 1 | err_crc_vc 0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved and read as zero. |
| 15 | err_crc_vc15 | B'0 | RC | Payload Checksum error detected on virtual channel 15 |
| 14 | err_crc_vc14 | B'0 | RC | Payload Checksum error detected on virtual channel 14 |
| 13 | err_crc_vc13 | B'0 | RC | Payload Checksum error detected on virtual channel 13 |
| 12 | err_crc_vc12 | B'0 | RC | Payload Checksum error detected on virtual channel 12 |
| 11 | err_crc_vc11 | B'0 | RC | Payload Checksum error detected on virtual channel 11 |
| 10 | err_crc_vc10 | B'0 | RC | Payload Checksum error detected on virtual channel 10 |
| 9 | err_crc_vc9 | B'0 | RC | Payload Checksum error detected on virtual channel 9 |
| 8 | err_crc_vc8 | B'0 | RC | Payload Checksum error detected on virtual channel 8 |
| 7 | err_crc_vc7 | B'0 | RC | Payload Checksum error detected on virtual channel 7 |
| 6 | err_crc_vc6 | B'0 | RC | Payload Checksum error detected on virtual channel 6 |
| 5 | err_crc_vc5 | B'0 | RC | Payload Checksum error detected on virtual channel 5 |
| 4 | err_crc_vc4 | B'0 | RC | Payload Checksum error detected on virtual channel 4 |
| 3 | err_crc_vc3 | B'0 | RC | Payload Checksum error detected on virtual channel 3 |
| 2 | err_crc_vc2 | B'0 | RC | Payload Checksum error detected on virtual channel 2 |
| 1 | err_crc_vc1 | B'0 | RC | Payload Checksum error detected on virtual channel 1 |
| 0 | err_crc_vc0 | B'0 | RC | Payload Checksum error detected on virtual channel 0 |

### 53.2.26 CSI2 [n] Mask for fatal interruption caused by Payload CRC(n=0-1)(CSI2[n]_INT_MSK_PLD_CRC_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_crc_vc 15 | err_crc_vc 14 | err_crc_vc 13 | err_crc_vc 12 | err_crc_vc 11 | err_crc_vc 10 | err_crc_vc 9 | err_crc_vc 8 | err_crc_vc 7 | err_crc_vc 6 | err_crc_vc 5 | err_crc_vc 4 | err_crc_vc 3 | err_crc_vc 2 | err_crc_vc 1 | err_crc_vc 0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved and read as zero. |
| 15 | err_crc_vc15 | B'0 | R/W | Mask for err_crc_vc15 |
| 14 | err_crc_vc14 | B'0 | R/W | Mask for err_crc_vc14 |
| 13 | err_crc_vc13 | B'0 | R/W | Mask for err_crc_vc13 |
| 12 | err_crc_vc12 | B'0 | R/W | Mask for err_crc_vc12 |
| 11 | err_crc_vc11 | B'0 | R/W | Mask for err_crc_vc11 |
| 10 | err_crc_vc10 | B'0 | R/W | Mask for err_crc_vc10 |
| 9 | err_crc_vc9 | B'0 | R/W | Mask for err_crc_vc9 |
| 8 | err_crc_vc8 | B'0 | R/W | Mask for err_crc_vc8 |
| 7 | err_crc_vc7 | B'0 | R/W | Mask for err_crc_vc7 |
| 6 | err_crc_vc6 | B'0 | R/W | Mask for err_crc_vc6 |
| 5 | err_crc_vc5 | B'0 | R/W | Mask for err_crc_vc5 |
| 4 | err_crc_vc4 | B'0 | R/W | Mask for err_crc_vc4 |
| 3 | err_crc_vc3 | B'0 | R/W | Mask for err_crc_vc3 |
| 2 | err_crc_vc2 | B'0 | R/W | Mask for err_crc_vc2 |
| 1 | err_crc_vc1 | B'0 | R/W | Mask for err_crc_vc1 |
| 0 | err_crc_vc0 | B'0 | R/W | Mask for err_crc_vc0 |

### 53.2.27 CSI2 [n] Interruption caused by Data Type(n=0-1)(CSI2[n]_INT_ST_DATA_ID)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_id_vc15 | err_id_vc14 | err_id_vc13 | err_id_vc12 | err_id_vc11 | err_id_vc10 | err_id_vc9 | err_id_vc8 | err_id_vc7 | err_id_vc6 | err_id_vc5 | err_id_vc4 | err_id_vc3 | err_id_vc2 | err_id_vc1 | err_id_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved and read as zero. |
| 15 | err_id_vc15 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 15 |
| 14 | err_id_vc14 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 14 |
| 13 | err_id_vc13 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 13 |
| 12 | err_id_vc12 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 12 |
| 11 | err_id_vc11 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 11 |
| 10 | err_id_vc10 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 10 |
| 9 | err_id_vc9 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 9 |
| 8 | err_id_vc8 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 8 |
| 7 | err_id_vc7 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 7 |
| 6 | err_id_vc6 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 6 |
| 5 | err_id_vc5 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 5 |
| 4 | err_id_vc4 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 4 |
| 3 | err_id_vc3 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 3 |
| 2 | err_id_vc2 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 2 |
| 1 | err_id_vc1 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 1 |
| 0 | err_id_vc0 | B'0 | RC | Unrecognized or unimplemented data type detected in virtual channel 0 |

### 53.2.28 CSI2 [n] Mask for interruption caused by Data Type(n=0-1)(CSI2[n]_INT_MSK_DATA_ID)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_id_vc15 | err_id_vc14 | err_id_vc13 | err_id_vc12 | err_id_vc11 | err_id_vc10 | err_id_vc9 | err_id_vc8 | err_id_vc7 | err_id_vc6 | err_id_vc5 | err_id_vc4 | err_id_vc3 | err_id_vc2 | err_id_vc1 | err_id_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved and read as zero. |
| 15 | err_id_vc15 | B'0 | R/W | Mask for err_id_vc15 |
| 14 | err_id_vc14 | B'0 | R/W | Mask for err_id_vc14 |
| 13 | err_id_vc13 | B'0 | R/W | Mask for err_id_vc13 |
| 12 | err_id_vc12 | B'0 | R/W | Mask for err_id_vc12 |
| 11 | err_id_vc11 | B'0 | R/W | Mask for err_id_vc11 |
| 10 | err_id_vc10 | B'0 | R/W | Mask for err_id_vc10 |
| 9 | err_id_vc9 | B'0 | R/W | Mask for err_id_vc9 |
| 8 | err_id_vc8 | B'0 | R/W | Mask for err_id_vc8 |
| 7 | err_id_vc7 | B'0 | R/W | Mask for err_id_vc7 |
| 6 | err_id_vc6 | B'0 | R/W | Mask for err_id_vc6 |
| 5 | err_id_vc5 | B'0 | R/W | Mask for err_id_vc5 |
| 4 | err_id_vc4 | B'0 | R/W | Mask for err_id_vc4 |
| 3 | err_id_vc3 | B'0 | R/W | Mask for err_id_vc3 |
| 2 | err_id_vc2 | B'0 | R/W | Mask for err_id_vc2 |
| 1 | err_id_vc1 | B'0 | R/W | Mask for err_id_vc1 |
| 0 | err_id_vc0 | B'0 | R/W | Mask for err_id_vc0 |

### 53.2.29 CSI2 [n] Interruption caused by Header single bit errors(n=0-1)(CSI2[n]_INT_ST_ECC_CORRECTED)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_ecc_corrected15 | err_ecc_corrected14 | err_ecc_corrected13 | err_ecc_corrected12 | err_ecc_corrected11 | err_ecc_corrected10 | err_ecc_corrected9 | err_ecc_corrected8 | err_ecc_corrected7 | err_ecc_corrected6 | err_ecc_corrected5 | err_ecc_corrected4 | err_ecc_corrected3 | err_ecc_corrected2 | err_ecc_corrected1 | err_ecc_corrected0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved and read as zero. |
| 15 | err_ecc_corrected15 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 15. |
| 14 | err_ecc_corrected14 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 14. |
| 13 | err_ecc_corrected13 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 13. |
| 12 | err_ecc_corrected12 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 12. |
| 11 | err_ecc_corrected11 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 11. |
| 10 | err_ecc_corrected10 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 10. |
| 9 | err_ecc_corrected9 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 9. |
| 8 | err_ecc_corrected8 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 8. |
| 7 | err_ecc_corrected7 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 7. |
| 6 | err_ecc_corrected6 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 6. |
| 5 | err_ecc_corrected5 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 5. |
| 4 | err_ecc_corrected4 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 4. |
| 3 | err_ecc_corrected3 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 3. |
| 2 | err_ecc_corrected2 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 2. |
| 1 | err_ecc_corrected1 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 1. |
| 0 | err_ecc_corrected0 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 0. |

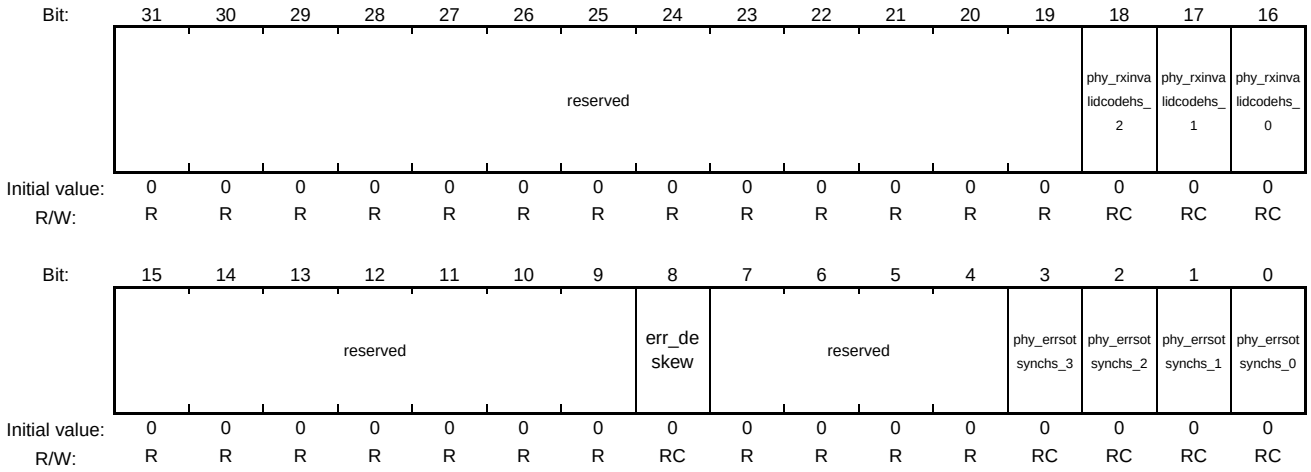
### 53.2.30 CSI2 [n] Mask for interruption caused by Header single bit errors(n=0-1)(CSI2[n]_INT_MSK_ECC_CORRECTED)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_ecc_corrected15 | err_ecc_corrected14 | err_ecc_corrected13 | err_ecc_corrected12 | err_ecc_corrected11 | err_ecc_corrected10 | err_ecc_corrected9 | err_ecc_corrected8 | err_ecc_corrected7 | err_ecc_corrected6 | err_ecc_corrected5 | err_ecc_corrected4 | err_ecc_corrected3 | err_ecc_corrected2 | err_ecc_corrected1 | err_ecc_corrected0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

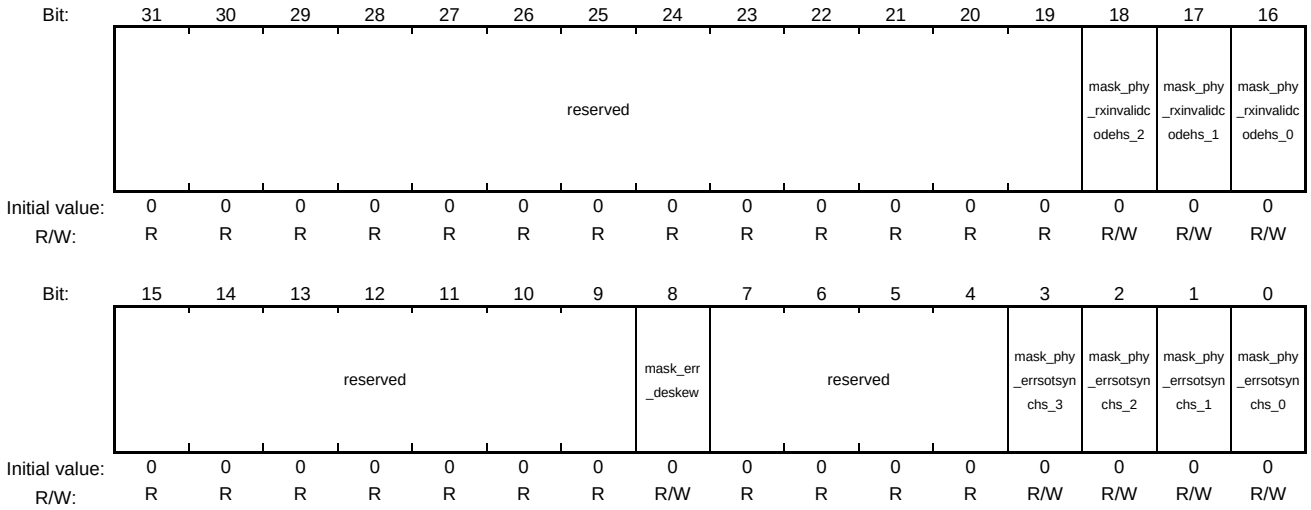
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved and read as zero. |
| 15 | err_ecc_corrected15 | B'0 | R/W | Mask for err_ecc_corrected_vc15 |
| 14 | err_ecc_corrected14 | B'0 | R/W | Mask for err_ecc_corrected_vc14 |
| 13 | err_ecc_corrected13 | B'0 | R/W | Mask for err_ecc_corrected_vc13 |
| 12 | err_ecc_corrected12 | B'0 | R/W | Mask for err_ecc_corrected_vc12 |
| 11 | err_ecc_corrected11 | B'0 | R/W | Mask for err_ecc_corrected_vc11 |
| 10 | err_ecc_corrected10 | B'0 | R/W | Mask for err_ecc_corrected_vc10 |
| 9 | err_ecc_corrected9 | B'0 | R/W | Mask for err_ecc_corrected_vc9 |
| 8 | err_ecc_corrected8 | B'0 | R/W | Mask for err_ecc_corrected_vc8 |
| 7 | err_ecc_corrected7 | B'0 | R/W | Mask for err_ecc_corrected_vc7 |
| 6 | err_ecc_corrected6 | B'0 | R/W | Mask for err_ecc_corrected_vc6 |
| 5 | err_ecc_corrected5 | B'0 | R/W | Mask for err_ecc_corrected_vc5 |
| 4 | err_ecc_corrected4 | B'0 | R/W | Mask for err_ecc_corrected_vc4 |
| 3 | err_ecc_corrected3 | B'0 | R/W | Mask for err_ecc_corrected_vc3 |
| 2 | err_ecc_corrected2 | B'0 | R/W | Mask for err_ecc_corrected_vc2 |
| 1 | err_ecc_corrected1 | B'0 | R/W | Mask for err_ecc_corrected_vc1 |
| 0 | err_ecc_corrected0 | B'0 | R/W | Mask for err_ecc_corrected_vc0 |

### 53.2.31 CSI2 [n] Functional AP fatal interruption caused by PHY(n=0-1)(CSI2[n]_INT_ST_FAP_PHY_FATAL)



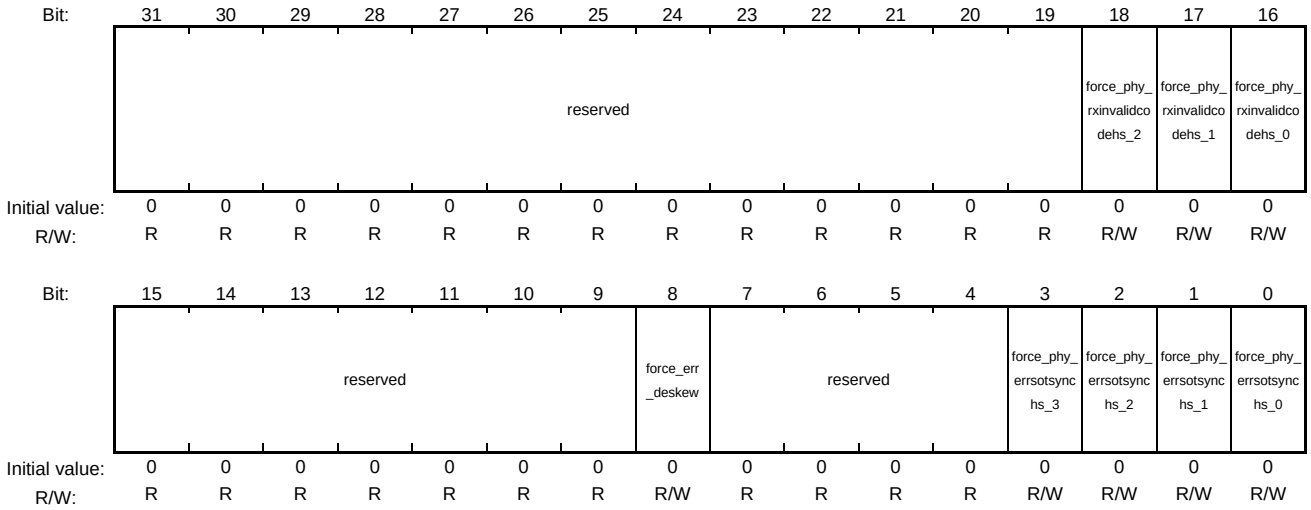

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | reserved | All 0 | R | Reserved and read as zero. |
| 18 | phy_rxinvalidcodehs_2 | B'0 | RC | High Speed Invalid Code Word Detection on lane 2 |
| 17 | phy_rxinvalidcodehs_1 | B'0 | RC | High Speed Invalid Code Word Detection on lane 1 |
| 16 | phy_rxinvalidcodehs_0 | B'0 | RC | High Speed Invalid Code Word Detection on lane 0 |
| 15 to 9 | reserved | All 0 | R | Reserved and read as zero. |
| 8 | err_deskew | B'0 | RC | Reports whenever data is lost due to an existent skew between lanes greater than 2 rxwordclkhs |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | phy_errsotsynchs_3 | B'0 | RC | Start of transmission error on data lane 3 (no synchronization achieved) |
| 2 | phy_errsotsynchs_2 | B'0 | RC | Start of transmission error on data lane 2 (no synchronization achieved) |
| 1 | phy_errsotsynchs_1 | B'0 | RC | Start of transmission error on data lane 1 (no synchronization achieved) |
| 0 | phy_errsotsynchs_0 | B'0 | RC | Start of transmission error on data lane 0 (no synchronization achieved) |

### 53.2.32 CSI2 [n] Mask for functional AP fatal interruption caused by PHY(n=0-1)(CSI2[n]_INT_MSK_FAP_PHY_FATAL)



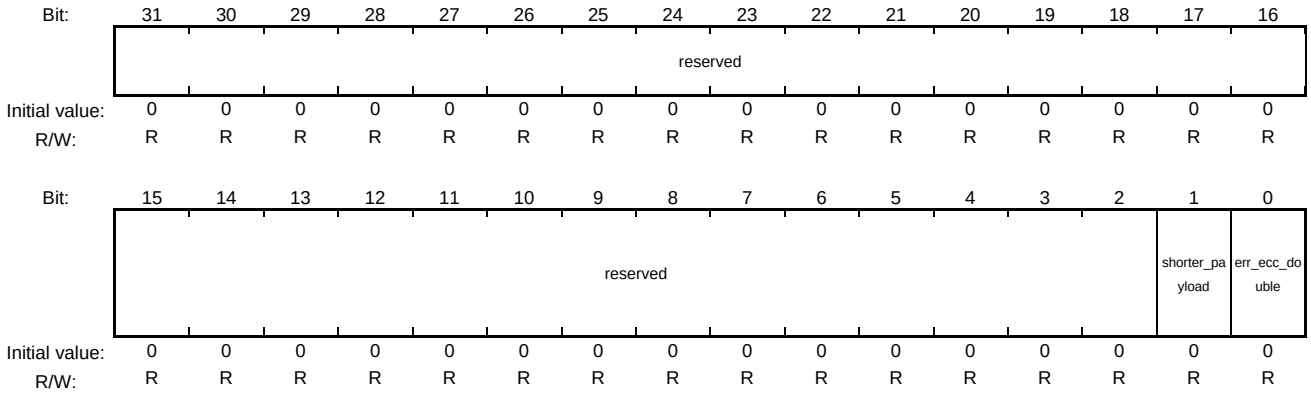

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | reserved | All 0 | R | Reserved and read as zero. |
| 18 | mask_phy_rxinvalidcodehs_2 | B'0 | R/W | Mask for phy_rxinvalidcodehs_2 |
| 17 | mask_phy_rxinvalidcodehs_1 | B'0 | R/W | Mask for phy_rxinvalidcodehs_1 |
| 16 | mask_phy_rxinvalidcodehs_0 | B'0 | R/W | Mask for phy_rxinvalidcodehs_0 |
| 15 to 9 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 8 | mask_err_deskew | B'0 | R/W | Mask for err_deskew |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | mask_phy_errsotsynchs_3 | B'0 | R/W | Mask for phy_errsotsynchs_3 |
| 2 | mask_phy_errsotsynchs_2 | B'0 | R/W | Mask for phy_errsotsynchs_2 |
| 1 | mask_phy_errsotsynchs_1 | B'0 | R/W | Mask for phy_errsotsynchs_1 |
| 0 | mask_phy_errsotsynchs_0 | B'0 | R/W | Mask for phy_errsotsynchs_0 |

### 53.2.33 CSI2 [n] Force for functional AP fatal interruption caused by PHY(n=0-1)(CSI2[n]_INT_FORCE_FAP_PHY_FATAL)



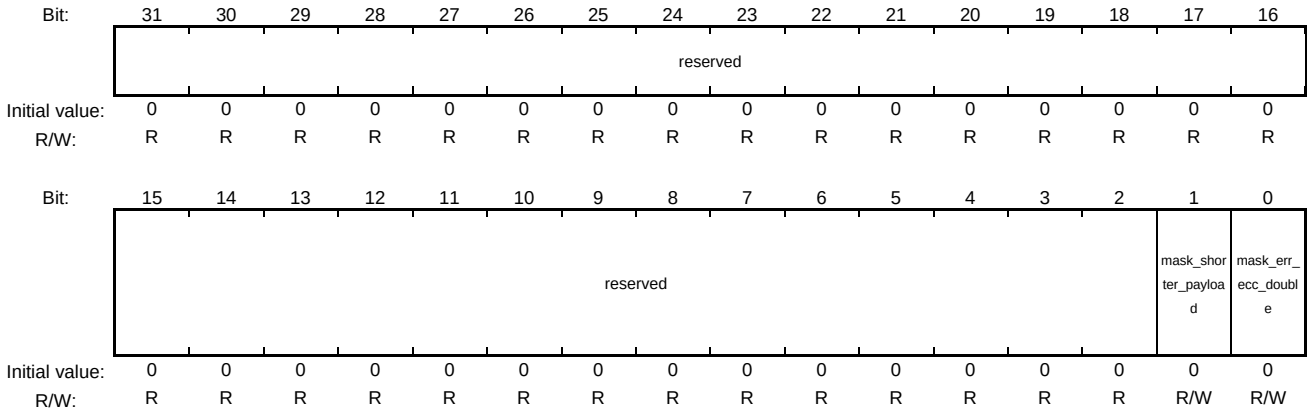

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | reserved | All 0 | R | Reserved and read as zero. |
| 18 | force_phy_rxinvalidcodehs_2 | B'0 | R/W | Force phy_rxinvalidcodehs_2 |
| 17 | force_phy_rxinvalidcodehs_1 | B'0 | R/W | Force phy_rxinvalidcodehs_1 |
| 16 | force_phy_rxinvalidcodehs_0 | B'0 | R/W | Force phy_rxinvalidcodehs_0 |
| 15 to 9 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 8 | force_err_deskew | B'0 | R/W | Force err_deskew |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | force_phy_errsotsynchs_3 | B'0 | R/W | Force phy_errsotsynchs_3 |
| 2 | force_phy_errsotsynchs_2 | B'0 | R/W | Force phy_errsotsynchs_2 |
| 1 | force_phy_errsotsynchs_1 | B'0 | R/W | Force phy_errsotsynchs_1 |
| 0 | force_phy_errsotsynchs_0 | B'0 | R/W | Force phy_errsotsynchs_0 |

### 53.2.34 CSI2 [n] Functional AP fatal interruption caused during Packet Construction(n=0-1)(CSI2[n]_INT_ST_FAP_PKT_FATAL)



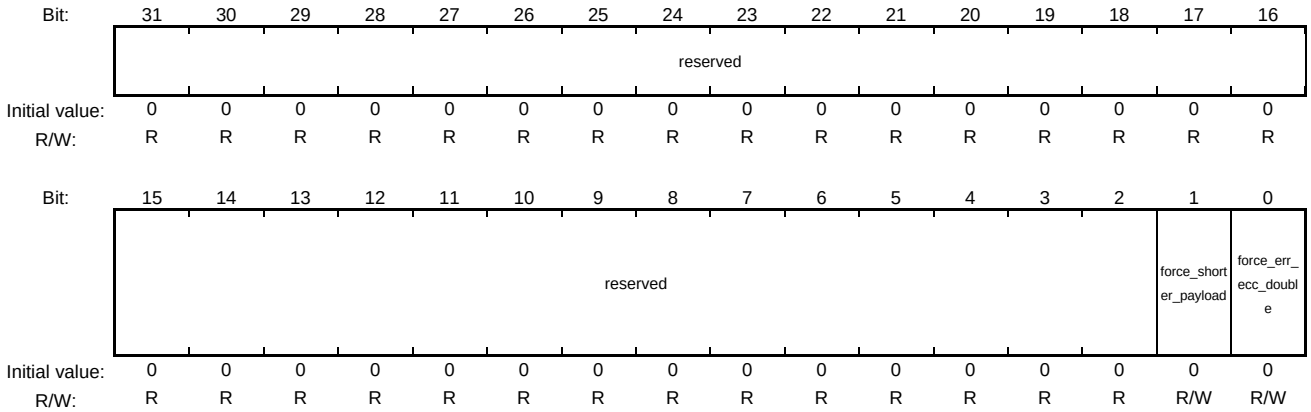

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | reserved | All 0 | R | Reserved and read as zero. |
| 1 | shorter_payload | B'0 | R | D-PHY mode: Reported greater WC than received, unrecoverable. |
| 0 | err_ecc_double | B'0 | R | D-PHY mode: Header ECC contains at least 2 errors, unrecoverable. |

### 53.2.35 CSI2 [n] Mask for functional AP fatal interruption caused during Packet Construction(n=0-1)(CSI2[n]_INT_MSK_FAP_PKT_FATAL)



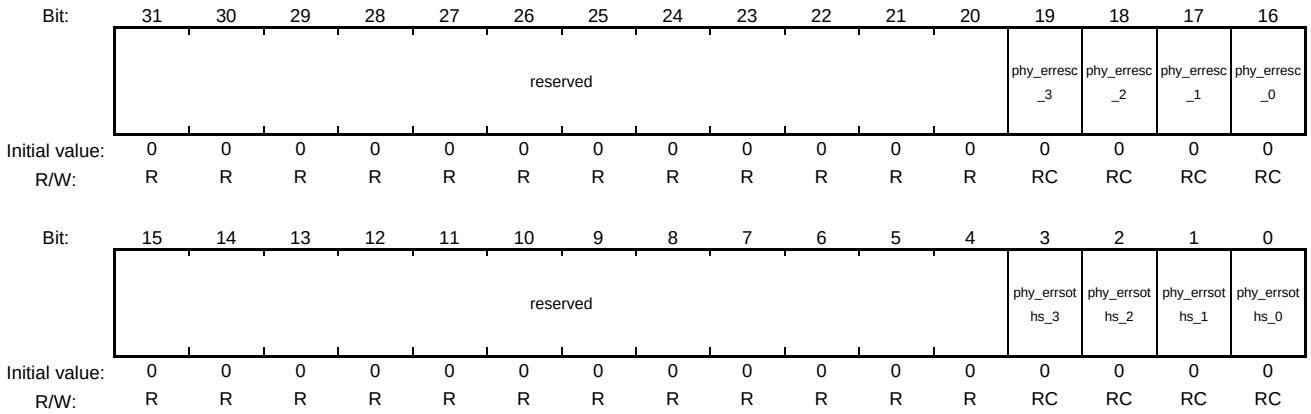

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | reserved | All 0 | R | Reserved and read as zero. |
| 1 | mask_shorter_payload | B'0 | R/W | Mask for shorter_payload. |
| 0 | mask_err_ecc_double | B'0 | R/W | Mask for err_ecc_double. |

### 53.2.36 CSI2 [n] Force for functional AP fatal interruption caused during Packet Construction(n=0-1)(CSI2[n]_INT_FORCE_FAP_PKT_FATAL)



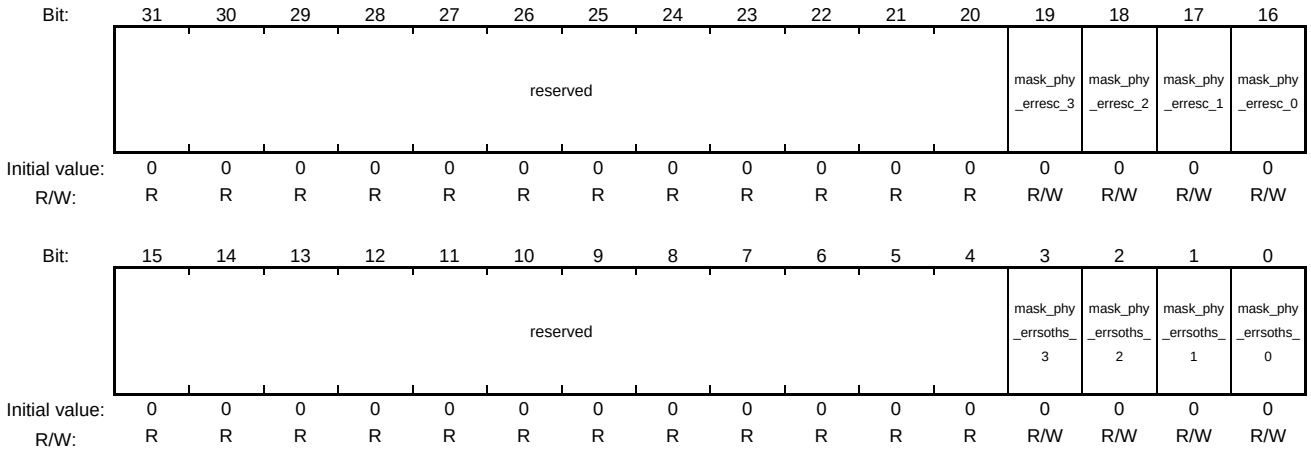

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | reserved | All 0 | R | Reserved and read as zero. |
| 1 | force_shorter_payload | B'0 | R/W | Force shorter_payload. |
| 0 | force_err_ecc_double | B'0 | R/W | Force err_ecc_double. |

### 53.2.37 CSI2 [n] Functional AP interruption caused by PHY(n=0-1)(CSI2[n]_INT_ST_FAP_PHY)



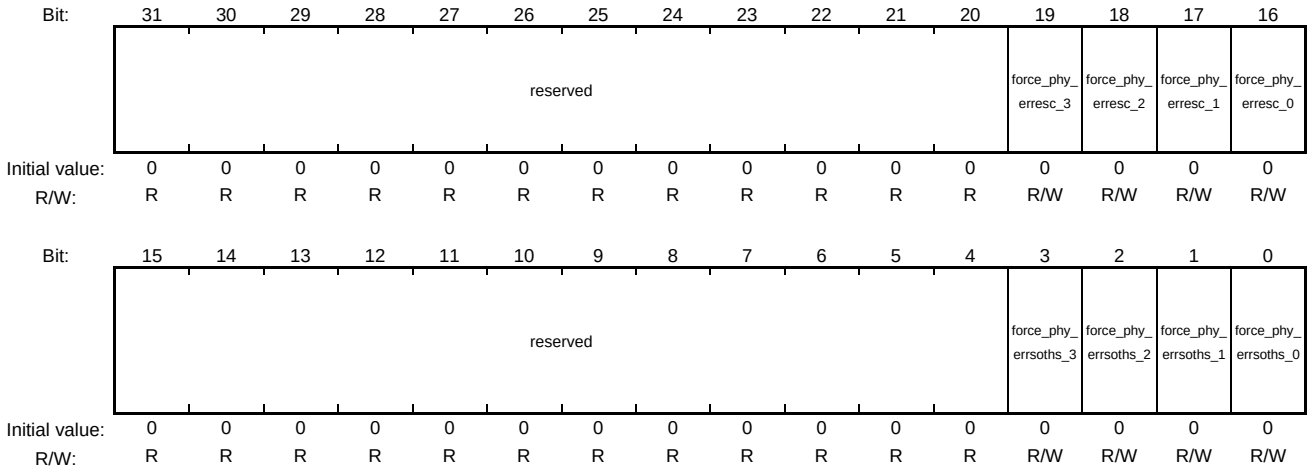

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R | Reserved and read as zero. |
| 23 | reserved | B'0 | R | Reserved and read as zero. |
| 22 | reserved | B'0 | R | Reserved and read as zero. |
| 21 | reserved | B'0 | R | Reserved and read as zero. |
| 20 | reserved | B'0 | R | Reserved and read as zero. |
| 19 | phy_erresc_3 | B'0 | RC | Escape Entry Error on data lane 3 |
| 18 | phy_erresc_2 | B'0 | RC | Escape Entry Error on data lane 2 |
| 17 | phy_erresc_1 | B'0 | RC | Escape Entry Error on data lane 1 |
| 16 | phy_erresc_0 | B'0 | RC | Escape Entry Error on data lane 0 |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | phy_errsoths_3 | B'0 | RC | Start of transmission error on data lane 3 (synchronization can still be achieved) |
| 2 | phy_errsoths_2 | B'0 | RC | Start of transmission error on data lane 2 (synchronization can still be achieved) |
| 1 | phy_errsoths_1 | B'0 | RC | Start of transmission error on data lane 1 (synchronization can still be achieved) |
| 0 | phy_errsoths_0 | B'0 | RC | Start of transmission error on data lane 0 (synchronization can still be achieved) |

### 53.2.38 CSI2 [n] Mask for functional AP interruption caused by PHY(n=0-1)(CSI2[n]_INT_MSK_FAP_PHY)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R | Reserved and read as zero. |
| 23 | reserved | B'0 | R | Reserved and read as zero. |
| 22 | reserved | B'0 | R | Reserved and read as zero. |
| 21 | reserved | B'0 | R | Reserved and read as zero. |
| 20 | reserved | B'0 | R | Reserved and read as zero. |
| 19 | mask_phy_erresc_3 | B'0 | R/W | Mask for phy_erresc_3 |
| 18 | mask_phy_erresc_2 | B'0 | R/W | Mask for phy_erresc_2 |
| 17 | mask_phy_erresc_1 | B'0 | R/W | Mask for phy_erresc_1 |
| 16 | mask_phy_erresc_0 | B'0 | R/W | Mask for phy_erresc_0 |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | mask_phy_errsoths_3 | B'0 | R/W | Mask for phy_errsoths_3 |
| 2 | mask_phy_errsoths_2 | B'0 | R/W | Mask for phy_errsoths_2 |
| 1 | mask_phy_errsoths_1 | B'0 | R/W | Mask for phy_errsoths_1 |
| 0 | mask_phy_errsoths_0 | B'0 | R/W | Mask for phy_errsoths_0 |

### 53.2.39 CSI2 [n] Force for functional AP interruption caused by PHY(n=0-1)(CSI2[n]_INT_FORCE_FAP_PHY)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R | Reserved and read as zero. |
| 23 | reserved | B'0 | R | Reserved and read as zero. |
| 22 | reserved | B'0 | R | Reserved and read as zero. |
| 21 | reserved | B'0 | R | Reserved and read as zero. |
| 20 | reserved | B'0 | R | Reserved and read as zero. |
| 19 | force_phy_erresc_3 | B'0 | R/W | Force phy_erresc_3 |
| 18 | force_phy_erresc_2 | B'0 | R/W | Force phy_erresc_2 |
| 17 | force_phy_erresc_1 | B'0 | R/W | Force phy_erresc_1 |
| 16 | force_phy_erresc_0 | B'0 | R/W | Force phy_erresc_0 |
| 15 to 8 | reserved | All 0 | R | Reserved and read as zero. |
| 7 | reserved | B'0 | R | Reserved and read as zero. |
| 6 | reserved | B'0 | R | Reserved and read as zero. |
| 5 | reserved | B'0 | R | Reserved and read as zero. |
| 4 | reserved | B'0 | R | Reserved and read as zero. |
| 3 | force_phy_errsoths_3 | B'0 | R/W | Force phy_errsoths_3 |
| 2 | force_phy_errsoths_2 | B'0 | R/W | Force phy_errsoths_2 |
| 1 | force_phy_errsoths_1 | B'0 | R/W | Force phy_errsoths_1 |
| 0 | force_phy_errsoths_0 | B'0 | R/W | Force phy_errsoths_0 |

### 53.2.40 CSI2 [n] Functional AP fatal Interruption caused by Frame Boundaries(n=0-1)(CSI2[n]_INT_ST_FAP_BNDRY_FRAME_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_f_bndry_match_vc15 | err_f_bndry_match_vc14 | err_f_bndry_match_vc13 | err_f_bndry_match_vc12 | err_f_bndry_match_vc11 | err_f_bndry_match_vc10 | err_f_bndry_match_vc9 | err_f_bndry_match_vc8 | err_f_bndry_match_vc7 | err_f_bndry_match_vc6 | err_f_bndry_match_vc5 | err_f_bndry_match_vc4 | err_f_bndry_match_vc3 | err_f_bndry_match_vc2 | err_f_bndry_match_vc1 | err_f_bndry_match_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 | err_f_bndry_match_vc15 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 15 |
| 14 | err_f_bndry_match_vc14 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 14 |
| 13 | err_f_bndry_match_vc13 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 13 |
| 12 | err_f_bndry_match_vc12 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 12 |
| 11 | err_f_bndry_match_vc11 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 11 |
| 10 | err_f_bndry_match_vc10 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 10 |
| 9 | err_f_bndry_match_vc9 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 9 |
| 8 | err_f_bndry_match_vc8 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 8 |
| 7 | err_f_bndry_match_vc7 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 7 |
| 6 | err_f_bndry_match_vc6 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 6 |
| 5 | err_f_bndry_match_vc5 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 5 |
| 4 | err_f_bndry_match_vc4 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 4 |
| 3 | err_f_bndry_match_vc3 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 3 |
| 2 | err_f_bndry_match_vc2 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 2 |
| 1 | err_f_bndry_match_vc1 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 1 |
| 0 | err_f_bndry_match_vc0 | B'0 | RC | Error matching Frame Start with Frame End for virtual channel 0 |

### 53.2.41 CSI2 [n] Mask for functional AP fatal Interruption caused by Frame Boundaries(n=0-1)(CSI2[n]_INT_MSK_FAP_BNDRY_FRAME_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_f_bndry_match_vc15 | err_f_bndry_match_vc14 | err_f_bndry_match_vc13 | err_f_bndry_match_vc12 | err_f_bndry_match_vc11 | err_f_bndry_match_vc10 | err_f_bndry_match_vc9 | err_f_bndry_match_vc8 | err_f_bndry_match_vc7 | err_f_bndry_match_vc6 | err_f_bndry_match_vc5 | err_f_bndry_match_vc4 | err_f_bndry_match_vc3 | err_f_bndry_match_vc2 | err_f_bndry_match_vc1 | err_f_bndry_match_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 | err_f_bndry_match_vc15 | B'0 | R/W | Mask for err_f_bndry_match_vc15 |
| 14 | err_f_bndry_match_vc14 | B'0 | R/W | Mask for err_f_bndry_match_vc14 |
| 13 | err_f_bndry_match_vc13 | B'0 | R/W | Mask for err_f_bndry_match_vc13 |
| 12 | err_f_bndry_match_vc12 | B'0 | R/W | Mask for err_f_bndry_match_vc12 |
| 11 | err_f_bndry_match_vc11 | B'0 | R/W | Mask for err_f_bndry_match_vc11 |
| 10 | err_f_bndry_match_vc10 | B'0 | R/W | Mask for err_f_bndry_match_vc10 |
| 9 | err_f_bndry_match_vc9 | B'0 | R/W | Mask for err_f_bndry_match_vc9 |
| 8 | err_f_bndry_match_vc8 | B'0 | R/W | Mask for err_f_bndry_match_vc8 |
| 7 | err_f_bndry_match_vc7 | B'0 | R/W | Mask for err_f_bndry_match_vc7 |
| 6 | err_f_bndry_match_vc6 | B'0 | R/W | Mask for err_f_bndry_match_vc6 |
| 5 | err_f_bndry_match_vc5 | B'0 | R/W | Mask for err_f_bndry_match_vc5 |
| 4 | err_f_bndry_match_vc4 | B'0 | R/W | Mask for err_f_bndry_match_vc4 |
| 3 | err_f_bndry_match_vc3 | B'0 | R/W | Mask for err_f_bndry_match_vc3 |
| 2 | err_f_bndry_match_vc2 | B'0 | R/W | Mask for err_f_bndry_match_vc2 |
| 1 | err_f_bndry_match_vc1 | B'0 | R/W | Mask for err_f_bndry_match_vc1 |
| 0 | err_f_bndry_match_vc0 | B'0 | R/W | Mask for err_f_bndry_match_vc0 |

### 53.2.42 CSI2 [n] Force for functional AP fatal Interruption caused by Frame Boundaries(n=0-1)(CSI2[n]_INT_FORCE_FAP_BNDRY_FRAME_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_f_bndry_match_vc15 | err_f_bndry_match_vc14 | err_f_bndry_match_vc13 | err_f_bndry_match_vc12 | err_f_bndry_match_vc11 | err_f_bndry_match_vc10 | err_f_bndry_match_vc9 | err_f_bndry_match_vc8 | err_f_bndry_match_vc7 | err_f_bndry_match_vc6 | err_f_bndry_match_vc5 | err_f_bndry_match_vc4 | err_f_bndry_match_vc3 | err_f_bndry_match_vc2 | err_f_bndry_match_vc1 | err_f_bndry_match_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 | err_f_bndry_match_vc15 | B'0 | R/W | Force err_f_bndry_match_vc15 |
| 14 | err_f_bndry_match_vc14 | B'0 | R/W | Force err_f_bndry_match_vc14 |
| 13 | err_f_bndry_match_vc13 | B'0 | R/W | Force err_f_bndry_match_vc13 |
| 12 | err_f_bndry_match_vc12 | B'0 | R/W | Force err_f_bndry_match_vc12 |
| 11 | err_f_bndry_match_vc11 | B'0 | R/W | Force err_f_bndry_match_vc11 |
| 10 | err_f_bndry_match_vc10 | B'0 | R/W | Force err_f_bndry_match_vc10 |
| 9 | err_f_bndry_match_vc9 | B'0 | R/W | Force err_f_bndry_match_vc9 |
| 8 | err_f_bndry_match_vc8 | B'0 | R/W | Force err_f_bndry_match_vc8 |
| 7 | err_f_bndry_match_vc7 | B'0 | R/W | Force err_f_bndry_match_vc7 |
| 6 | err_f_bndry_match_vc6 | B'0 | R/W | Force err_f_bndry_match_vc6 |
| 5 | err_f_bndry_match_vc5 | B'0 | R/W | Force err_f_bndry_match_vc5 |
| 4 | err_f_bndry_match_vc4 | B'0 | R/W | Force err_f_bndry_match_vc4 |
| 3 | err_f_bndry_match_vc3 | B'0 | R/W | Force err_f_bndry_match_vc3 |
| 2 | err_f_bndry_match_vc2 | B'0 | R/W | Force err_f_bndry_match_vc2 |
| 1 | err_f_bndry_match_vc1 | B'0 | R/W | Force err_f_bndry_match_vc1 |
| 0 | err_f_bndry_match_vc0 | B'0 | R/W | Force err_f_bndry_match_vc0 |

### 53.2.43 CSI2 [n] Functional AP fatal interruption caused by Frame Sequence(n=0-1)(CSI2[n]_INT_ST_FAP_SEQ_FRAME_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_f_seq_vc15 | err_f_seq_vc14 | err_f_seq_vc13 | err_f_seq_vc12 | err_f_seq_vc11 | err_f_seq_vc10 | err_f_seq_vc9 | err_f_seq_vc8 | err_f_seq_vc7 | err_f_seq_vc6 | err_f_seq_vc5 | err_f_seq_vc4 | err_f_seq_vc3 | err_f_seq_vc2 | err_f_seq_vc1 | err_f_seq_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 | err_f_seq_vc15 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 15 |
| 14 | err_f_seq_vc14 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 14 |
| 13 | err_f_seq_vc13 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 13 |
| 12 | err_f_seq_vc12 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 12 |
| 11 | err_f_seq_vc11 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 11 |
| 10 | err_f_seq_vc10 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 10 |
| 9 | err_f_seq_vc9 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 9 |
| 8 | err_f_seq_vc8 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 8 |
| 7 | err_f_seq_vc7 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 7 |
| 6 | err_f_seq_vc6 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 6 |
| 5 | err_f_seq_vc5 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 5 |
| 4 | err_f_seq_vc4 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 4 |
| 3 | err_f_seq_vc3 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 3 |
| 2 | err_f_seq_vc2 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 2 |
| 1 | err_f_seq_vc1 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 1 |
| 0 | err_f_seq_vc0 | B'0 | RC | Incorrect Frame sequence detected in virtual channel 0 |

### 53.2.44 CSI2 [n] Mask for functional AP fatal interruption caused by Frame Sequence(n=0-1)(CSI2[n]_INT_MSK_FAP_SEQ_FRAME_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_f_seq_vc15 | err_f_seq_vc14 | err_f_seq_vc13 | err_f_seq_vc12 | err_f_seq_vc11 | err_f_seq_vc10 | err_f_seq_vc9 | err_f_seq_vc8 | err_f_seq_vc7 | err_f_seq_vc6 | err_f_seq_vc5 | err_f_seq_vc4 | err_f_seq_vc3 | err_f_seq_vc2 | err_f_seq_vc1 | err_f_seq_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 | err_f_seq_vc15 | B'0 | R/W | Mask for err_f_seq_vc15 |
| 14 | err_f_seq_vc14 | B'0 | R/W | Mask for err_f_seq_vc14 |
| 13 | err_f_seq_vc13 | B'0 | R/W | Mask for err_f_seq_vc13 |
| 12 | err_f_seq_vc12 | B'0 | R/W | Mask for err_f_seq_vc12 |
| 11 | err_f_seq_vc11 | B'0 | R/W | Mask for err_f_seq_vc11 |
| 10 | err_f_seq_vc10 | B'0 | R/W | Mask for err_f_seq_vc10 |
| 9 | err_f_seq_vc9 | B'0 | R/W | Mask for err_f_seq_vc9 |
| 8 | err_f_seq_vc8 | B'0 | R/W | Mask for err_f_seq_vc8 |
| 7 | err_f_seq_vc7 | B'0 | R/W | Mask for err_f_seq_vc7 |
| 6 | err_f_seq_vc6 | B'0 | R/W | Mask for err_f_seq_vc6 |
| 5 | err_f_seq_vc5 | B'0 | R/W | Mask for err_f_seq_vc5 |
| 4 | err_f_seq_vc4 | B'0 | R/W | Mask for err_f_seq_vc4 |
| 3 | err_f_seq_vc3 | B'0 | R/W | Mask for err_f_seq_vc3 |
| 2 | err_f_seq_vc2 | B'0 | R/W | Mask for err_f_seq_vc2 |
| 1 | err_f_seq_vc1 | B'0 | R/W | Mask for err_f_seq_vc1 |
| 0 | err_f_seq_vc0 | B'0 | R/W | Mask for err_f_seq_vc0 |

### 53.2.45 CSI2 [n] Force for functional AP fatal interruption caused by Frame Sequence(n=0-1)(CSI2[n]_INT_FORCE_FAP_SEQ_FRAME_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_f_seq_vc15 | err_f_seq_vc14 | err_f_seq_vc13 | err_f_seq_vc12 | err_f_seq_vc11 | err_f_seq_vc10 | err_f_seq_vc9 | err_f_seq_vc8 | err_f_seq_vc7 | err_f_seq_vc6 | err_f_seq_vc5 | err_f_seq_vc4 | err_f_seq_vc3 | err_f_seq_vc2 | err_f_seq_vc1 | err_f_seq_vc0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 | err_f_seq_vc15 | B'0 | R/W | Force err_f_seq_vc15 |
| 14 | err_f_seq_vc14 | B'0 | R/W | Force err_f_seq_vc14 |
| 13 | err_f_seq_vc13 | B'0 | R/W | Force err_f_seq_vc13 |
| 12 | err_f_seq_vc12 | B'0 | R/W | Force err_f_seq_vc12 |
| 11 | err_f_seq_vc11 | B'0 | R/W | Force err_f_seq_vc11 |
| 10 | err_f_seq_vc10 | B'0 | R/W | Force err_f_seq_vc10 |
| 9 | err_f_seq_vc9 | B'0 | R/W | Force err_f_seq_vc9 |
| 8 | err_f_seq_vc8 | B'0 | R/W | Force err_f_seq_vc8 |
| 7 | err_f_seq_vc7 | B'0 | R/W | Force err_f_seq_vc7 |
| 6 | err_f_seq_vc6 | B'0 | R/W | Force err_f_seq_vc6 |
| 5 | err_f_seq_vc5 | B'0 | R/W | Force err_f_seq_vc5 |
| 4 | err_f_seq_vc4 | B'0 | R/W | Force err_f_seq_vc4 |
| 3 | err_f_seq_vc3 | B'0 | R/W | Force err_f_seq_vc3 |
| 2 | err_f_seq_vc2 | B'0 | R/W | Force err_f_seq_vc2 |
| 1 | err_f_seq_vc1 | B'0 | R/W | Force err_f_seq_vc1 |
| 0 | err_f_seq_vc0 | B'0 | R/W | Force err_f_seq_vc0 |

### 53.2.46 CSI2 [n] Functional AP fatal interruption caused by Payload CRC(n=0-1)(CSI2[n]_INT_ST_FAP_PLD_CRC_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_crc_vc 15 | err_crc_vc 14 | err_crc_vc 13 | err_crc_vc 12 | err_crc_vc 11 | err_crc_vc 10 | err_crc_vc 9 | err_crc_vc 8 | err_crc_vc 7 | err_crc_vc 6 | err_crc_vc 5 | err_crc_vc 4 | err_crc_vc 3 | err_crc_vc 2 | err_crc_vc 1 | err_crc_vc 0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 | err_crc_vc15 | B'0 | RC | Payload Checksum error detected on virtual channel 15 |
| 14 | err_crc_vc14 | B'0 | RC | Payload Checksum error detected on virtual channel 14 |
| 13 | err_crc_vc13 | B'0 | RC | Payload Checksum error detected on virtual channel 13 |
| 12 | err_crc_vc12 | B'0 | RC | Payload Checksum error detected on virtual channel 12 |
| 11 | err_crc_vc11 | B'0 | RC | Payload Checksum error detected on virtual channel 11 |
| 10 | err_crc_vc10 | B'0 | RC | Payload Checksum error detected on virtual channel 10 |
| 9 | err_crc_vc9 | B'0 | RC | Payload Checksum error detected on virtual channel 9 |
| 8 | err_crc_vc8 | B'0 | RC | Payload Checksum error detected on virtual channel 8 |
| 7 | err_crc_vc7 | B'0 | RC | Payload Checksum error detected on virtual channel 7 |
| 6 | err_crc_vc6 | B'0 | RC | Payload Checksum error detected on virtual channel 6 |
| 5 | err_crc_vc5 | B'0 | RC | Payload Checksum error detected on virtual channel 5 |
| 4 | err_crc_vc4 | B'0 | RC | Payload Checksum error detected on virtual channel 4 |
| 3 | err_crc_vc3 | B'0 | RC | Payload Checksum error detected on virtual channel 3 |
| 2 | err_crc_vc2 | B'0 | RC | Payload Checksum error detected on virtual channel 2 |
| 1 | err_crc_vc1 | B'0 | RC | Payload Checksum error detected on virtual channel 1 |
| 0 | err_crc_vc0 | B'0 | RC | Payload Checksum error detected on virtual channel 0 |

### 53.2.47 CSI2 [n] Mask for functional AP fatal interruption caused by Payload CRC(n=0-1)(CSI2[n]_INT_MSK_FAP_PLD_CRC_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_crc_vc 15 | err_crc_vc 14 | err_crc_vc 13 | err_crc_vc 12 | err_crc_vc 11 | err_crc_vc 10 | err_crc_vc 9 | err_crc_vc 8 | err_crc_vc 7 | err_crc_vc 6 | err_crc_vc 5 | err_crc_vc 4 | err_crc_vc 3 | err_crc_vc 2 | err_crc_vc 1 | err_crc_vc 0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 | err_crc_vc15 | B'0 | R/W | Mask for err_crc_vc15 |
| 14 | err_crc_vc14 | B'0 | R/W | Mask for err_crc_vc14 |
| 13 | err_crc_vc13 | B'0 | R/W | Mask for err_crc_vc13 |
| 12 | err_crc_vc12 | B'0 | R/W | Mask for err_crc_vc12 |
| 11 | err_crc_vc11 | B'0 | R/W | Mask for err_crc_vc11 |
| 10 | err_crc_vc10 | B'0 | R/W | Mask for err_crc_vc10 |
| 9 | err_crc_vc9 | B'0 | R/W | Mask for err_crc_vc9 |
| 8 | err_crc_vc8 | B'0 | R/W | Mask for err_crc_vc8 |
| 7 | err_crc_vc7 | B'0 | R/W | Mask for err_crc_vc7 |
| 6 | err_crc_vc6 | B'0 | R/W | Mask for err_crc_vc6 |
| 5 | err_crc_vc5 | B'0 | R/W | Mask for err_crc_vc5 |
| 4 | err_crc_vc4 | B'0 | R/W | Mask for err_crc_vc4 |
| 3 | err_crc_vc3 | B'0 | R/W | Mask for err_crc_vc3 |
| 2 | err_crc_vc2 | B'0 | R/W | Mask for err_crc_vc2 |
| 1 | err_crc_vc1 | B'0 | R/W | Mask for err_crc_vc1 |
| 0 | err_crc_vc0 | B'0 | R/W | Mask for err_crc_vc0 |

### 53.2.48 CSI2 [n] Force for functional AP fatal interruption caused by Payload CRC(n=0-1)(CSI2[n]_INT_FORCE_FAP_PLD_CRC_FATAL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_crc_vc 15 | err_crc_vc 14 | err_crc_vc 13 | err_crc_vc 12 | err_crc_vc 11 | err_crc_vc 10 | err_crc_vc 9 | err_crc_vc 8 | err_crc_vc 7 | err_crc_vc 6 | err_crc_vc 5 | err_crc_vc 4 | err_crc_vc 3 | err_crc_vc 2 | err_crc_vc 1 | err_crc_vc 0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 | err_crc_vc15 | B'0 | R/W | Force err_crc_vc15 |
| 14 | err_crc_vc14 | B'0 | R/W | Force err_crc_vc14 |
| 13 | err_crc_vc13 | B'0 | R/W | Force err_crc_vc13 |
| 12 | err_crc_vc12 | B'0 | R/W | Force err_crc_vc12 |
| 11 | err_crc_vc11 | B'0 | R/W | Force err_crc_vc11 |
| 10 | err_crc_vc10 | B'0 | R/W | Force err_crc_vc10 |
| 9 | err_crc_vc9 | B'0 | R/W | Force err_crc_vc9 |
| 8 | err_crc_vc8 | B'0 | R/W | Force err_crc_vc8 |
| 7 | err_crc_vc7 | B'0 | R/W | Force err_crc_vc7 |
| 6 | err_crc_vc6 | B'0 | R/W | Force err_crc_vc6 |
| 5 | err_crc_vc5 | B'0 | R/W | Force err_crc_vc5 |
| 4 | err_crc_vc4 | B'0 | R/W | Force err_crc_vc4 |
| 3 | err_crc_vc3 | B'0 | R/W | Force err_crc_vc3 |
| 2 | err_crc_vc2 | B'0 | R/W | Force err_crc_vc2 |
| 1 | err_crc_vc1 | B'0 | R/W | Force err_crc_vc1 |
| 0 | err_crc_vc0 | B'0 | R/W | Force err_crc_vc0 |

### 53.2.49 CSI2 [n] Functional AP interruption caused by Header single bit errors(n=0-1)(CSI2[n]_INT_ST_FAP_ECC_CORRECTED)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_ecc_corrected15 | err_ecc_corrected14 | err_ecc_corrected13 | err_ecc_corrected12 | err_ecc_corrected11 | err_ecc_corrected10 | err_ecc_corrected9 | err_ecc_corrected8 | err_ecc_corrected7 | err_ecc_corrected6 | err_ecc_corrected5 | err_ecc_corrected4 | err_ecc_corrected3 | err_ecc_corrected2 | err_ecc_corrected1 | err_ecc_corrected0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved and read as zero. |
| 15 | err_ecc_corrected15 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 15. |
| 14 | err_ecc_corrected14 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 14. |
| 13 | err_ecc_corrected13 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 13. |
| 12 | err_ecc_corrected12 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 12. |
| 11 | err_ecc_corrected11 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 11. |
| 10 | err_ecc_corrected10 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 10. |
| 9 | err_ecc_corrected9 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 9. |
| 8 | err_ecc_corrected8 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 8. |
| 7 | err_ecc_corrected7 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 7. |
| 6 | err_ecc_corrected6 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 6. |
| 5 | err_ecc_corrected5 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 5. |
| 4 | err_ecc_corrected4 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 4. |
| 3 | err_ecc_corrected3 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 3. |
| 2 | err_ecc_corrected2 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 2. |
| 1 | err_ecc_corrected1 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 1. |
| 0 | err_ecc_corrected0 | B'0 | RC | D-PHY mode: Header error detected and corrected on virtual channel 0. |

### 53.2.50 CSI2 [n] Mask for functional AP interruption caused by Header single bit errors(n=0-1)(CSI2[n]_INT_MSK_FAP_ECC_CORRECTED)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_ecc_corrected15 | err_ecc_corrected14 | err_ecc_corrected13 | err_ecc_corrected12 | err_ecc_corrected11 | err_ecc_corrected10 | err_ecc_corrected9 | err_ecc_corrected8 | err_ecc_corrected7 | err_ecc_corrected6 | err_ecc_corrected5 | err_ecc_corrected4 | err_ecc_corrected3 | err_ecc_corrected2 | err_ecc_corrected1 | err_ecc_corrected0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 | err_ecc_corrected15 | B'0 | R/W | Mask for err_ecc_corrected_vc15 |
| 14 | err_ecc_corrected14 | B'0 | R/W | Mask for err_ecc_corrected_vc14 |
| 13 | err_ecc_corrected13 | B'0 | R/W | Mask for err_ecc_corrected_vc13 |
| 12 | err_ecc_corrected12 | B'0 | R/W | Mask for err_ecc_corrected_vc12 |
| 11 | err_ecc_corrected11 | B'0 | R/W | Mask for err_ecc_corrected_vc11 |
| 10 | err_ecc_corrected10 | B'0 | R/W | Mask for err_ecc_corrected_vc10 |
| 9 | err_ecc_corrected9 | B'0 | R/W | Mask for err_ecc_corrected_vc9 |
| 8 | err_ecc_corrected8 | B'0 | R/W | Mask for err_ecc_corrected_vc8 |
| 7 | err_ecc_corrected7 | B'0 | R/W | Mask for err_ecc_corrected_vc7 |
| 6 | err_ecc_corrected6 | B'0 | R/W | Mask for err_ecc_corrected_vc6 |
| 5 | err_ecc_corrected5 | B'0 | R/W | Mask for err_ecc_corrected_vc5 |
| 4 | err_ecc_corrected4 | B'0 | R/W | Mask for err_ecc_corrected_vc4 |
| 3 | err_ecc_corrected3 | B'0 | R/W | Mask for err_ecc_corrected_vc3 |
| 2 | err_ecc_corrected2 | B'0 | R/W | Mask for err_ecc_corrected_vc2 |
| 1 | err_ecc_corrected1 | B'0 | R/W | Mask for err_ecc_corrected_vc1 |
| 0 | err_ecc_corrected0 | B'0 | R/W | Mask for err_ecc_corrected_vc0 |

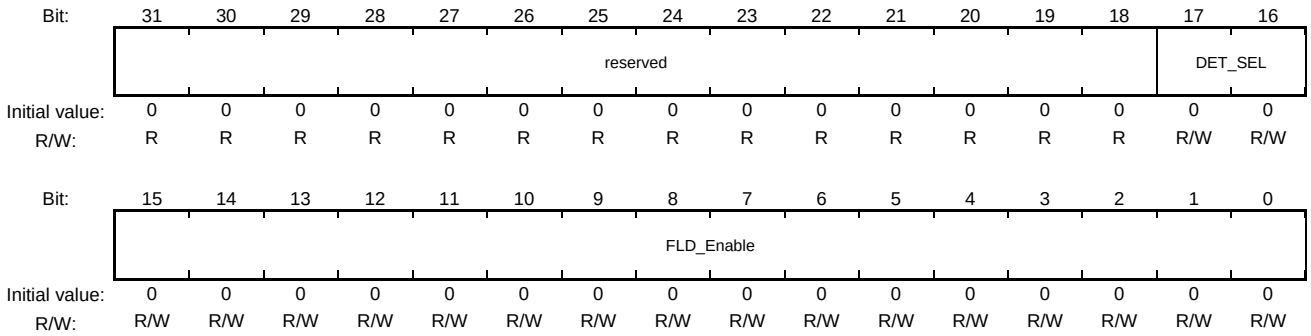
### 53.2.51 CSI2 [n] Force for functional AP interruption caused by Header single bit errors(n=0-1)(CSI2[n]_INT_FORCE_FAP_ECC_CORRECTED)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_ecc_corrected15 | err_ecc_corrected14 | err_ecc_corrected13 | err_ecc_corrected12 | err_ecc_corrected11 | err_ecc_corrected10 | err_ecc_corrected9 | err_ecc_corrected8 | err_ecc_corrected7 | err_ecc_corrected6 | err_ecc_corrected5 | err_ecc_corrected4 | err_ecc_corrected3 | err_ecc_corrected2 | err_ecc_corrected1 | err_ecc_corrected0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

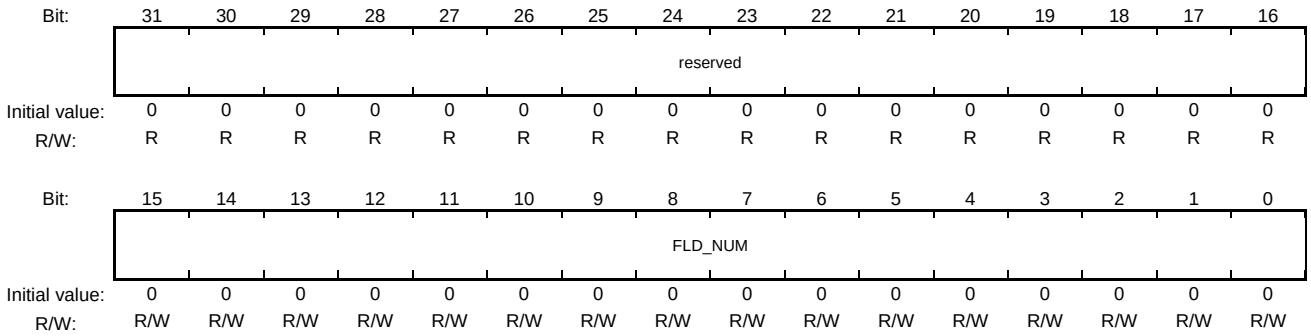
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 | err_ecc_corrected15 | B'0 | R/W | Force err_ecc_corrected_vc15 |
| 14 | err_ecc_corrected14 | B'0 | R/W | Force err_ecc_corrected_vc14 |
| 13 | err_ecc_corrected13 | B'0 | R/W | Force err_ecc_corrected_vc13 |
| 12 | err_ecc_corrected12 | B'0 | R/W | Force err_ecc_corrected_vc12 |
| 11 | err_ecc_corrected11 | B'0 | R/W | Force err_ecc_corrected_vc11 |
| 10 | err_ecc_corrected10 | B'0 | R/W | Force err_ecc_corrected_vc10 |
| 9 | err_ecc_corrected9 | B'0 | R/W | Force err_ecc_corrected_vc9 |
| 8 | err_ecc_corrected8 | B'0 | R/W | Force err_ecc_corrected_vc8 |
| 7 | err_ecc_corrected7 | B'0 | R/W | Force err_ecc_corrected_vc7 |
| 6 | err_ecc_corrected6 | B'0 | R/W | Force err_ecc_corrected_vc6 |
| 5 | err_ecc_corrected5 | B'0 | R/W | Force err_ecc_corrected_vc5 |
| 4 | err_ecc_corrected4 | B'0 | R/W | Force err_ecc_corrected_vc4 |
| 3 | err_ecc_corrected3 | B'0 | R/W | Force err_ecc_corrected_vc3 |
| 2 | err_ecc_corrected2 | B'0 | R/W | Force err_ecc_corrected_vc2 |
| 1 | err_ecc_corrected1 | B'0 | R/W | Force err_ecc_corrected_vc1 |
| 0 | err_ecc_corrected0 | B'0 | R/W | Force err_ecc_corrected_vc0 |

### 53.2.52 CSI2 [n] Field Detect Control Register(n=0-1)(CSI2[n]_FLDC)



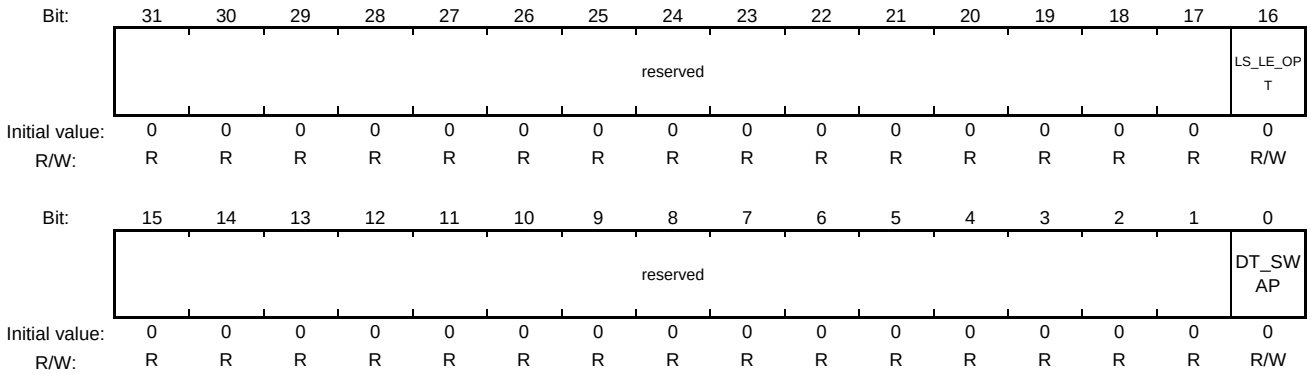

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 17, 16 | DET_SEL | B'00 | R/W | Even field detection condition select<br>00 : The field is detected as the even field when FLD_NUM matches WC<br>01 : The field is detected as the even field when FLD_NUM[0] matches WC[0].<br>10 : The field is detected as the even field when FLD_NUM[0] matches WC[8].<br>11: Setting is prohibited. |
| 15 to 0 | FLD_Enable | H'0000 | R/W | Virtual Channel even field detection control<br>This bit turns on or off even field detection of Virtual Channel.<br>H'0000 : FLD OFF<br>H’0001 : Channel 0 even field detection control<br>H’0002 : Channel 1 even field detection control<br>H’0004 : Channel 2 even field detection control<br>H’0008 : Channel 3 even field detection control<br>H’0010 : Channel 4 even field detection control<br>H’0020 : Channel 5 even field detection control<br>H’0040 : Channel 6 even field detection control<br>H’0080 : Channel 7 even field detection control<br>H’0100 : Channel 8 even field detection control<br>H’0200 : Channel 9 even field detection control<br>H’0400 : Channel 10 even field detection control<br>H’0800 : Channel 11 even field detection control<br>H’1000 : Channel 12 even field detection control<br>H’2000 : Channel 13 even field detection control<br>H’4000 : Channel 14 even field detection control<br>H’8000 : Channel 15 even field detection control |

### 53.2.53 CSI2 [n] Field Detect Data Register(n=0-1)(CSI2[n]_FLDD)



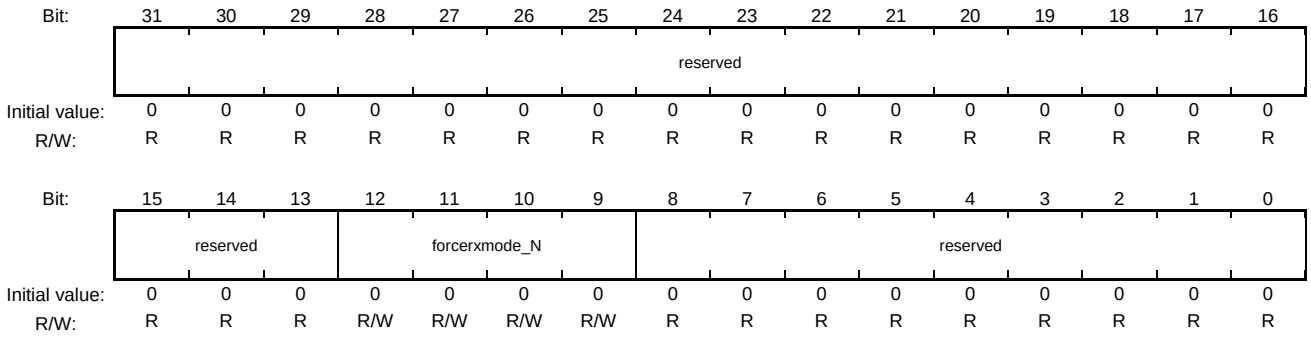

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 15 to 0 | FLD_NUM | All 0 | R/W | Even field number setting |

### 53.2.54 CSI2 [n] IDI Control Register(n=0-1)(CSI2[n]_IDIC)



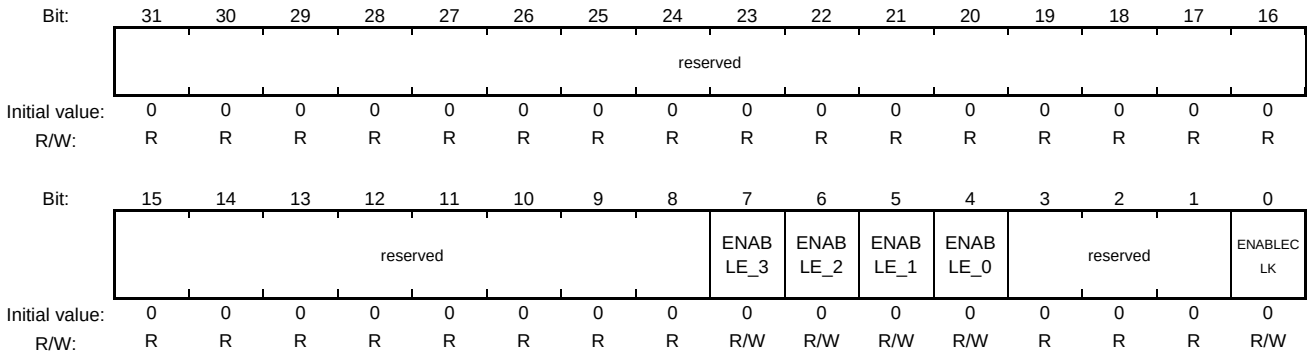

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 16 | LS_LE_OPT | B'0 | R/W | ISP Output HD Control<br>0 : Output HD when DT = long Paket (Cannot output when DT=0x10(Null),DT=0x11(BrunkingData))<br>1 : Set DT = LS (0x2) / DT = LE(0x3) |
| 15 to 1 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 0 | DT_SWAP | B'0 | R/W | ISP Output Data Swap Control<br>0 : SWAP OFF<br>1 : SWAP ON<br>Effective for YUV420 8-bit legacy & YUV422-8bit. |

### 53.2.55 CSI2 [n] Signal Override1 Register(n=0-1)(CSI2[n]_OVR1)



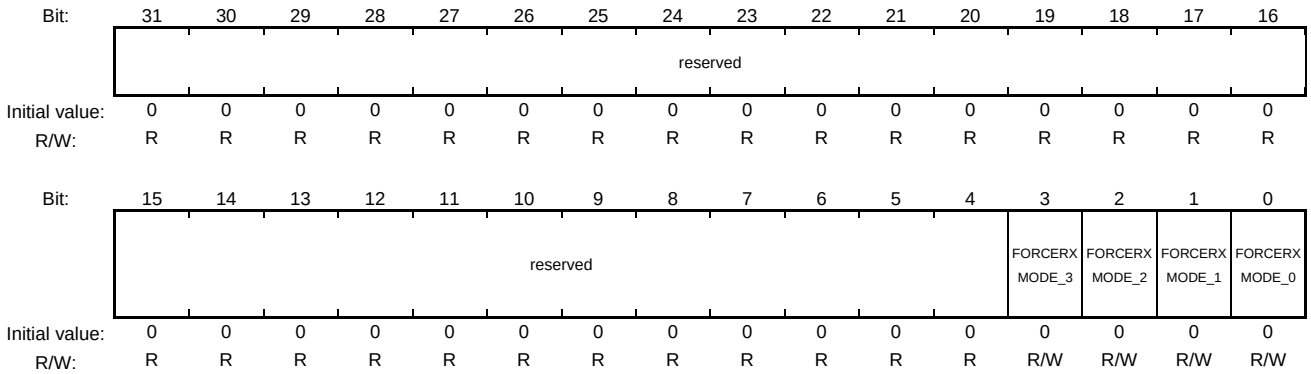

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 12 to 9 | forcerxmode_N | All 0 | R/W | Override enable for FRXM/forcerxmode_N |
| 8 to 0 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |

### 53.2.56 CSI2 [n] PHY ENABLE Register(n=0-1)(CSI2[n]_PHY_EN)



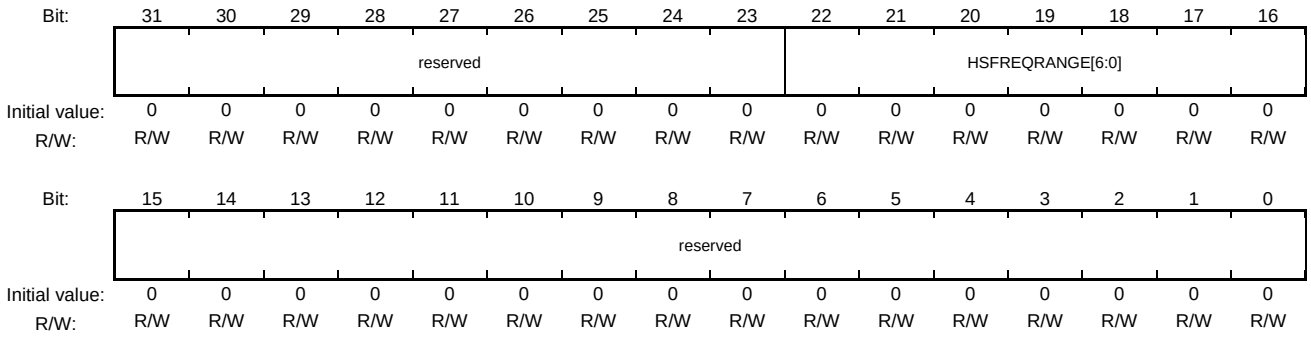

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | reserved | B'0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 7 | ENABLE_3 | B'0 | R/W | ENABLE_3 |
| 6 | ENABLE_2 | B'0 | R/W | ENABLE_2 |
| 5 | ENABLE_1 | B'0 | R/W | ENABLE_1 |
| 4 | ENABLE_0 | B'0 | R/W | ENABLE_0 |
| 3 to 1 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 0 | ENABLECLK | B'0 | R/W | ENABLECLK |

### 53.2.57 CSI2 [n] PHY Force RX Mode Control Register(n=0-1)(CSI2[n]_FRXM)



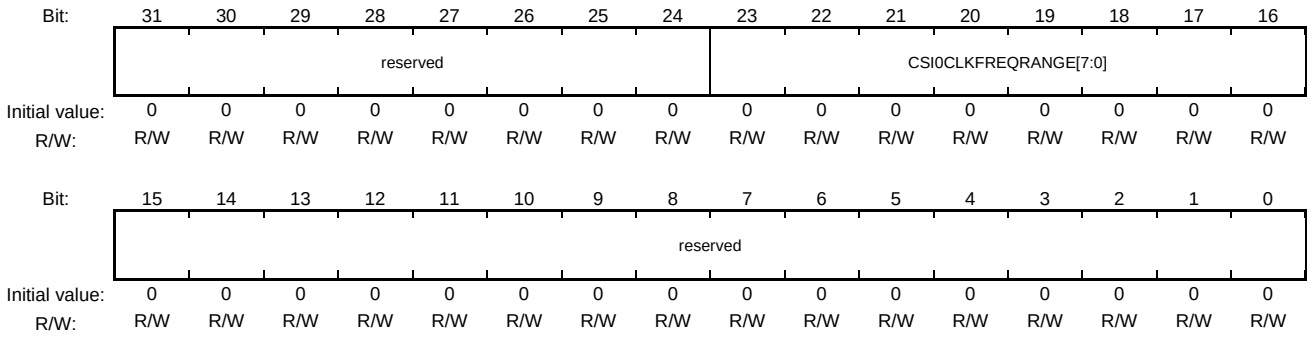

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 3 | FORCERXMODE_3 | B'0 | R/W | FORCERXMODE_3 |
| 2 | FORCERXMODE_2 | B'0 | R/W | FORCERXMODE_2 |
| 1 | FORCERXMODE_1 | B'0 | R/W | FORCERXMODE_1 |
| 0 | FORCERXMODE_0 | B'0 | R/W | FORCERXMODE_0 |

### 53.2.58 CSI2 [n] PHY Frequency Control Register(n=0-1)(CSI2[n]_PHYPLL)



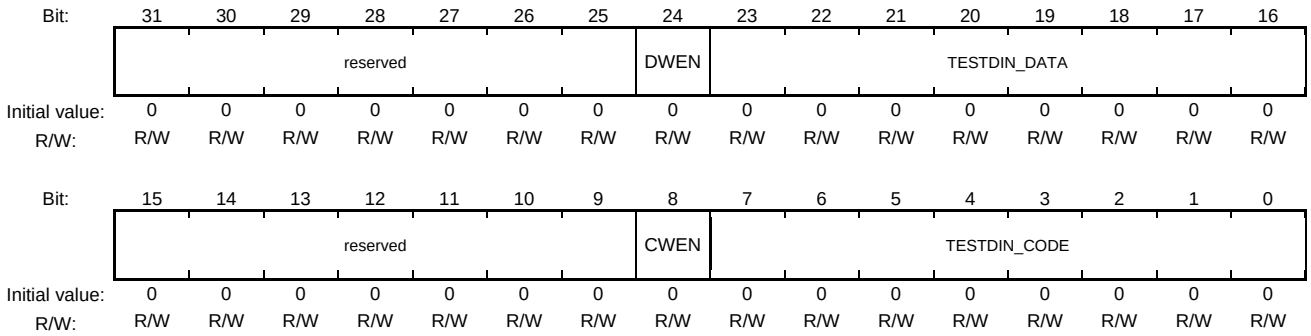

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | reserved | All 0 | R/W | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 22 to 16 | HSFREQRANGE[6:0] | All 0 | R/W | PHY high speed range select bits<br>These bits set the HSFREQRANGE pin of the PHY. These bits should be set according to the received frequency. |
| 15 to 0 | reserved | All 0 | R/W | Reserved<br>These bits are read as D.C. The write value should always be 0. |

### 53.2.59 CSI2 [n] CSI0CLK Frequency Configuration Preset Register(n=0-1)(CSI2[n]_CSI0CLKFCPR)



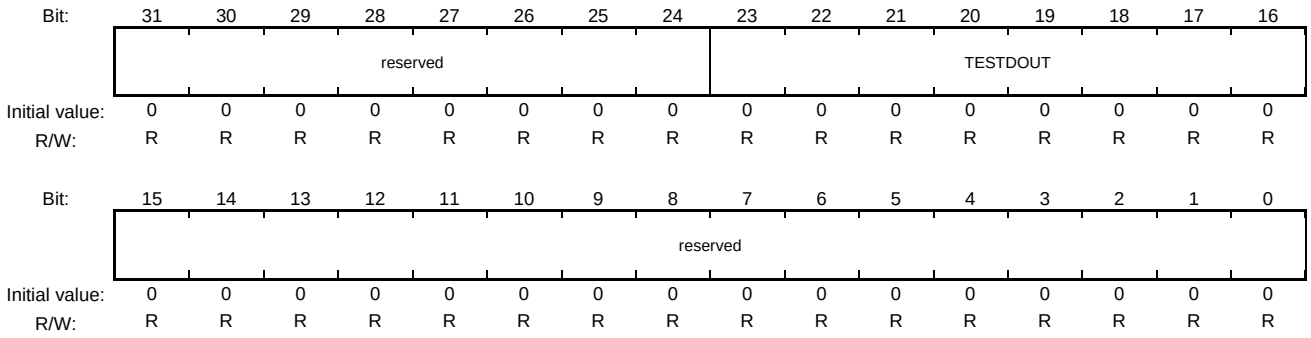

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R/W | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 23 to 16 | CSI0CLKFREQRANGE[7:0] | All 0 | R/W | CSI0ϕ frequency configuration<br>It should be set to H'0C. |
| 15 to 0 | reserved | All 0 | R/W | Reserved<br>These bits are read as D.C. The write value should always be 0. |

### 53.2.60 CSI2 [n] PHY Test Interface Write Register(n=0-1)(CSI2[n]_PHTW)



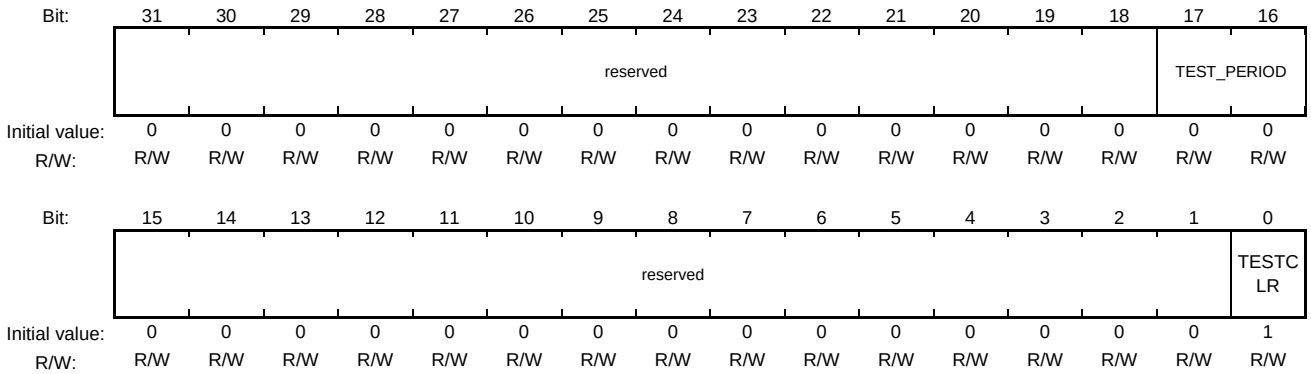

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | reserved | All 0 | R/W | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 24 | DWEN | B'0 | R/W | TESTDIN_DATA enable<br>0: Disable<br>1: Enable<br>This bit is auto cleared when value of TESTDIN_DATA sets to PHY.<br>If DWEN is set to 1'b1, it should be read PHTW to confirm that DWEN is auto cleared to 1'b0. |
| 23 to 16 | TESTDIN_DATA | All 0 | R/W | TESTDATE Input for DATA |
| 15 to 9 | reserved | All 0 | R/W | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 8 | CWEN | B'0 | R/W | TESTDIN_CODE enable<br>0: Disable<br>1: Enable<br>This bit is auto cleared when value of TESTDIN_CODE sets to PHY.<br>If CWEN is set to 1'b1, it should be read PHTW to confirm that CWEN is auto cleared to 1'b0. |
| 7 to 0 | TESTDIN_CODE | All 0 | R/W | TESTDATE Input for CODE |

### 53.2.61 CSI2 [n] PHY Test Interface Read Register(n=0-1)(CSI2[n]_PHTR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. |
| 23 to 16 | TESTDOUT | All 0 | R | Those bits are TESTDIN_DATA loopback value from CSI2 PHY.<br>If those bits are read, it is necessary to confirm DWEN and CWEN in PHTW is 1'b0. |
| 15 to 0 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. |

### 53.2.62 CSI2 [n] PHY Test Interface Clear Register(n=0-1)(CSI2[n]_PHTC)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | reserved | All 0 | R/W | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 17, 16 | TEST_PERIOD | B'00 | R/W | Control TESTCLK divider<br>00 : divide by 1<br>01 : divide by 2<br>10 : divide by 4<br>11 : divide by 8 |
| 15 to 1 | reserved | All 0 | R/W | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 0 | TESTCLR | B'1 | R/W | This bit sets TESTCLR of the PHY. |

### 53.2.63 CSI2 [n] PHY Status Register(n=0-1)(CSI2[n]_ST_PHYST)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST_PHY_READY | reserved | | | | | | | | | | | | | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ST_STOPSTATE_DCK | reserved | | | ST_STOPSTATE_3 | ST_STOPSTATE_2 | ST_STOPSTATE_1 | ST_STOPSTATE_0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ST_PHY_READY | B'0 | R | ST_PHY_READY (fixed to 0) |
| 30 to 18 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 17 to 8 | — | All 0 | R | Reserved |
| 7 | ST_STOPSTATE_DCK | B'0 | R | ST_STOPSTATE_DCK (stopstateclk) |
| 6 to 4 | reserved | All 0 | R | Reserved<br>These bits are read as D.C. The write value should always be 0. |
| 3 | ST_STOPSTATE_3 | B'0 | R | ST_STOPSTATE_3 (stopstatedata_3) |
| 2 | ST_STOPSTATE_2 | B'0 | R | ST_STOPSTATE_2 (stopstatedata_2) |
| 1 | ST_STOPSTATE_1 | B'0 | R | ST_STOPSTATE_1 (stopstatedata_1) |
| 0 | ST_STOPSTATE_0 | B'0 | R | ST_STOPSTATE_0 (stopstatedata_0) |

## 53.3 Operation

### 53.3.1 System

A software reset can be set by controlling the software reset register 3,4 (SRCR3,4) and software reset clear register 3,4 (SRSTCLR3,4) of the clock pulse generator (CPG) module., CSI2 0 correspond to bits 31 in the SRCR3 and SRSTCLR3 registers, respectively. CSI2 1 correspond to bits 0 in the SRCR4 and SRSTCLR4 registers, respectively. To apply a software reset to any of the CSI2 channels, set the corresponding bit in the SRCR3,4 register. Enabling the bit initializes all the registers of the corresponding CSI2 channel. To cancel a software reset of any of the CSI2 channels, set the corresponding bit in the SRSTCLR3,4 register. Starts CFGCLK after canceling the software reset.( Set CSIφ(CFGCLK) to PLL5VCO/2/2/40. see CPG CSICKCR. CSICKCR/DIV[5:0]=H'27. CSIφ frequency [MHz] = PLL5VCO x 1/2 x 1/2 x 1 / (setting + 1)).

To apply a software reset to any of the CSI2 channels, execute the following procedure using the register corresponding to the channel.

1. Set the corresponding bit in the SRCR3,4 register of the Clock Pulse Generator (CPG) module.

To cancel a software reset of any of the CSI2 channels and start the channel, execute the following procedure using the register corresponding to the channel.

2. Set the corresponding bit in the SRSTCLR3,4 register of the Clock Pulse Generator (CPG) module.

### 53.3.2 LINK

This chapter explains the basics of the controller programming, how to send and receive image or video in high-speed mode, and how to enter and exit ULPS. It has the following topics:

To program LINK, you must be familiar with the MIPI CSI-2, MIPI D-PHY specifications. Every software programmable register of LINK is accessible through the APB interface. For details about the LINK register map and all the LINK registers used for programming, see the "Register Descriptions" chapter 53.2.

#### 53.3.2.1 LINK Functions

Table 53.3 LINK Functions explains some of the main functions of LINK to operate in the CSI2 environment. This table explains when these functions are used and how they are implemented by accessing the APB registers. From this, you can build your own testbench functions or software code.

Table 53.3 LINK Functions

| Function Name | Function Usage | Procedure |
|---|---|---|
| Set or clear PHY Reset | This function is used during the PHY initialization, to set up the analog and digital part of the PHY in the default state.<br>This function is also used to turn off the PHY for power saving purposes. | 1. Clear PHY_SHUTDOWNZ, bit[0] and DPHY_RSTZ, bit[0] to put the PHY in the reset state.<br>2. Set PHY_SHUTDOWNZ, bit[0] and DPHY_RSTZ, bit[0] to remove the PHY from reset state. |

| Function Name | Function Usage | Procedure |
|---|---|---|
| Configure the number of active lanes | This function configures a static register, which contains the number of lanes to be used by LINK when receiving the data. This function is used during initialization and it should not be used during operational mode. | 1. Configure N_LANES, bit[2:0], to define the number of active lanes in the system:<br>- 000: 1 lane<br>- 001: 2 lanes<br>- 010: 3 lanes<br>- 011: 4 lanes |
| Configure the data ID values | This function is used during initialization to set up the pair that identifies the data type and virtual channel of the lines that are expected to be received from the camera device. | 1. Configure DATA_IDS_1 and DATA_IDS_VC_1 to define the first four pairs of data types and virtual channels.<br>2. The unused monitors must be set with zeros. |
| Release LINK from reset | This function is mainly used when initializing LINK. It sets the controller's internal logic to the default state.<br>It is also used when changing the configuration. In this case, while changing the static configuration register values, the LINK should be in the reset state. | 1. Clear CSI2_RESETN, bit[0] to put the CSI2 internal logic in the reset state.<br>2. Set CSI2_RESETN, bit[0] to remove the CSI2 internal logic from the reset state. |
| Define errors to be masked | This function configures errors that are supposed to trigger an interrupt. | 1. Assert bit corresponding to the error that is supposed to trigger an interrupt in INT_MSK_<group> |
| Check that clock lane is in Stop state<br>(For D-PHY only) | This function checks if the clock lane is in the Stop state or not. When the clock lane is correctly initialized it is said to be in a Stop state. This function is used during initialization and also during the enter and exit procedures of ULPS. | 1. Read PHY_STOPSTATE, bit[16].<br>If bit 16 is set, clock lane is detected in Stop state. |
| Check that data lanes are in Stop state | This function checks if the data lanes are in the Stop state or not. When a data lane is correctly initialized and ready to receive high-speed data, it is said to be in a Stop state. This function is used during initialization and also during the enter and exit procedures of ULPS. | 1. Read PHY_STOPSTATE, bit[3:0].<br>If bits 0 to 3 are set, data lanes 0 to 3 are detected in Stop state, respectively. |
| Check that clock lane is in High Speed mode<br>(For D-PHY only) | This function checks the status of the clock lane reported by LINK. The clock lane either receives a high-speed clock signal or not. This function is used during initialization, to check if the clock is being correctly received. This function is also used during the enter and exit procedures of ULPS. | 1. Read PHY_RX, bit[17].<br>If set, high-speed clock signal is received in clock lane. |
| Check ULPS | This function is used during the enter and exit procedures of ULPS. This function checks if the PHY status bits report that the respective lanes are detected in ULPS. In a CSI2 Host system, the enter and exit state of the ULPS is triggered by the camera device, so LINK only observes the status of the lanes. | 1. Read PHY_RX, bit[3:0]. If bits 0 to 3 are set, data lanes 0 to 3 are detected in ULPS, respectively.<br>2. Read PHY_RX, bit[16]. If the bit is clear, clock lane is detected in ULPS.<br>(For D-PHY only). |

#### 53.3.2.2 LINK Programming Flow

The following are the main tasks performed when a CSI2 system captures a still image or an image video from a camera device:

Start up LINK: Setting LINK and PHY with the default values.

Initialize LINK: Configuring LINK for the proper operation mode and configuring PHY for the required frequency range.

Start the High Speed Reception Mode: Setting up LINK and PHY to be ready to receive high-speed information.

Detect Errors: Detecting where an error occurs.

Stop the High Speed Reception Mode: Disabling LINK and PHY.

Enter and Exit ULPS: Entering and exiting Ultra Low Power Mode (ULPS). This sequence is optional.

(1) Start up LINK contain the programming sequences for these tasks. For the register configuration details of each step, see Table 53.3.

**(1) Start up LINK**

Follow the programming sequence shown in **Figure 53.2** to start up LINK.

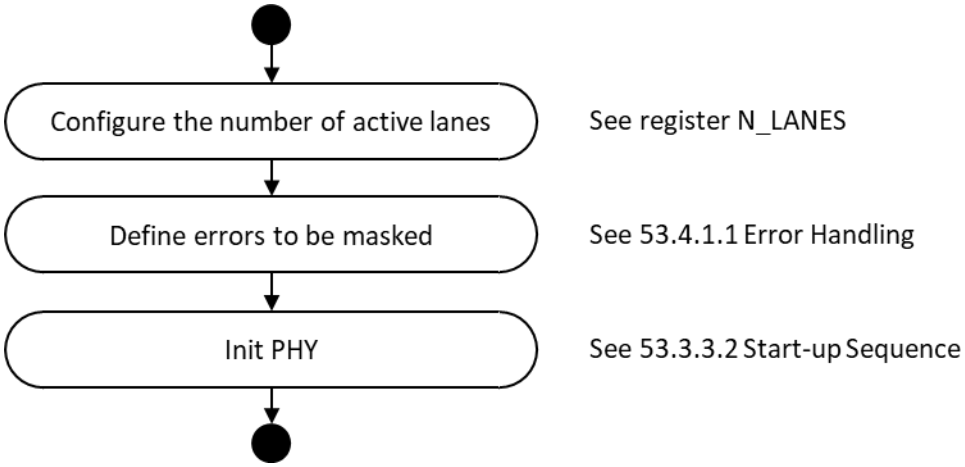



**Figure 53.2 Start up LINK**

### (2) Start the High Speed Reception Mode

Follow the programming sequence shown in Figure 53.3 to start the LINK high-speed reception mode. Set up the camera device to start high-speed clock transmission. At the end of this sequence, LINK is ready to receive high-speed data.

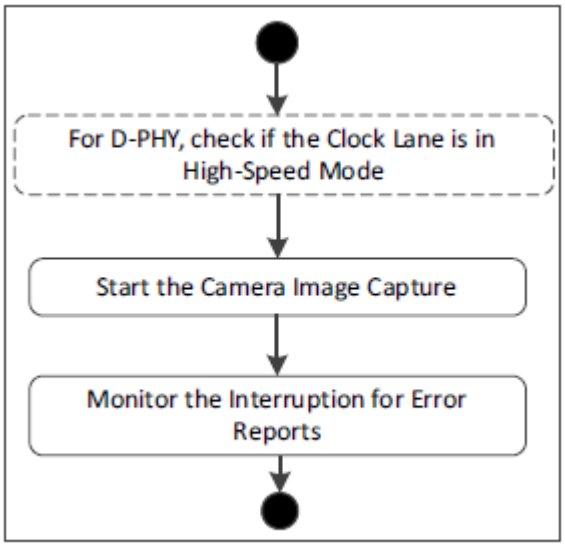



Figure 53.3 Start the High Speed Reception Mode

### (3) Detect Errors

Follow the programming sequences shown in Figure 53.4 to detect why the interrupt pin rises.

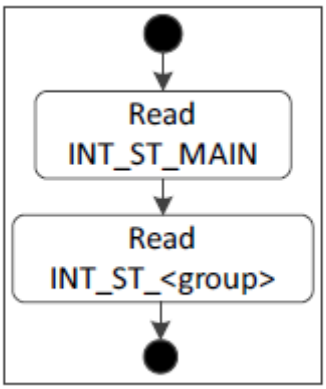



Figure 53.4 Detect Errors

An unmasked error triggers the interrupt pin.

Read INT_ST_MAIN:

Clear on read register. Read this register to know from which group the interruption comes from Reading this register clears the interrupt pin.

Read INT_ST_<group>:

Clear on read register. Read this register in order to know which error occurred. Reading this register does NOT clear the interrupt pin.

For descriptions of all errors that LINK can identify, see 53.4.1.1 “Error Handling” .

### (4) Stop the High Speed Reception Mode

Follow the programming sequence shown in Figure 53.5 to stop the LINK high-speed reception mode. After the image or video has been completely received, the CSI2 system is turned off.

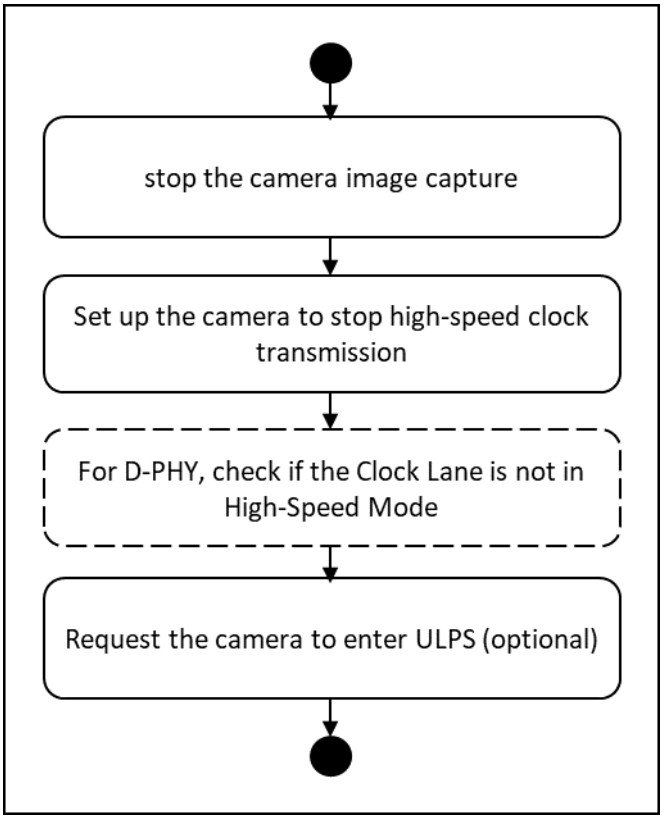



Figure 53.5 Stop the High Speed Reception Mode

**(5) Enter and Exit ULPS**

Follow the programming sequences shown in Figure 53.6 and Figure 53.7 to enter and exit ULPS respectively. After the image or video has been completely received, the CSI-2 system is turned off.

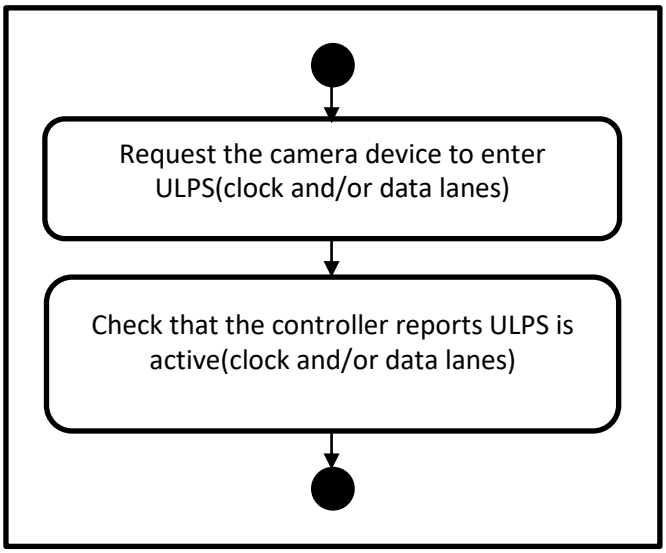



Figure 53.6 Enter ULPS

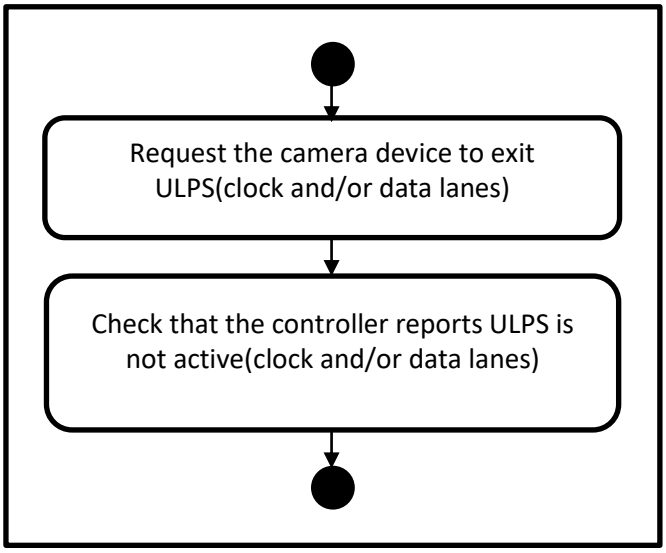



Figure 53.7 Exit ULPS

**Note**

- The PHY requires over 5ms as the wakeup time by means of ULPS exit.

**(6) Shutdown**

Stop the camera image and shut down and reset with the PHY register.

CSI2_RESETN/csi2_resetn = 1'b0; DPHY_RSTZ/dphy_rstz = 1'b0; SHUTDOWNZ/phy_shutdownz = 1'b0.

(see register CSI2_RESETN/csi2_resetn ,PHY_SHUTDOWNZ/phy_shutdownz, DPHY_RSTZ/dphy_rstz)

#### 53.3.2.3 FLD (field) Signals

When a frame start short packet is received, if the field value of the short packet data satisfies the condition specified with the field detection control register (FLD), FLD is driven high. When a frame end short packet is received, FLD is driven low. (This applies to the channel specified with the VC value in the packet header.)

#### 53.3.2.4 Interrupt

Both the interrupt pin as well as the applicable Interrupt Status registers are set at the same time. The default value of the INT_MASK_<group> (Interrupt Mask) registers is 0, thus masking the interrupts. By default, the interrupt pin is disabled.

### 53.3.3 D-PHY

#### 53.3.3.1 Shutdown Mode

This mode is the lowest power consumption mode, where all analog blocks are disabled, and digital logic is reset. The current consumption is given by the analog standby current and the digital logic leakage current. It is entered asynchronously when DPHY_RSTZ/dphy_rstz and PHY_SHUTDOWNZ/phy_shutdownz are in low state. It should be ensured that PHTC/testclr is in high state by default, as it acts as an active high reset to the control block responsible for the configuration values preset. In this mode, analog pins are in high-impedance (Hi-Z).

All necessary MIPI D-PHY configurations to set the desired operation should be made with the MIPI DPHY on this mode, before Start-up.

#### 53.3.3.2 Start-up Sequence

Before D-PHY Start-up, the D-PHY needs to be properly configured for the desired operation. This needs to be done with the PHY in shutdown mode and it is assumed that such configurations are static or at least are stable prior to leaving the Shutdown mode.

The D-PHY operating frequency, input reference clock frequency and DDL target oscillation frequency need to be configured.

■ D-PHY Operating frequency: HSFREQRANGE[6:0]/hsfreqrange[6:0]

To configure the bit rate operation you should directly configure HSFREQRANGE[6:0] for the desired frequency range . The D-PHY Operating frequency needs to be configured for the desired operating frequency according to Table 53.13 “Frequency Ranges and Defaults”. You can only configure HSFREQRANGE[6:0] before the D-PHY Start-up. You can not change HSFREQRANGE[6:0] setting during active operation.

■ Input reference clock frequency: CSI0CLKFREQRANGE[7:0]/csi0clkfreqrange[7:0]Regarding input system clock frequency, CSI0CLKFREQRANGE[7:0] needs to be H’0C.

■ DDL target oscillation frequency:

This DDL target oscillation frequency(12bit data: osc_freq_target[11:0]) needs to be configured for the desired operating frequency according to Table 53.13 “Frequency Ranges and Defaults”.

The desired operating frequency configuration is required to meet the D-PHY specification global timing parameters and intervals. The D-PHY specification has several global timing parameters and intervals which have to be followed to ensure proper operation.

Table 53.13 Frequency Rages and Defaults

| Range (Mbps) | Default Bit Rate (Mbps) | HSFREQRANGE[6:0] (Binary notation) | Osc_freq_target[11:0] (Decimal notation) |
|---|---|---|---|
| 80 - 97.125 | 80 | 0000000 | 425 |
| 80 - 107.625 | 90 | 0010000 | 425 |
| 83.125 - 118.125 | 100 | 0100000 | 425 |
| 92.625 - 128.625 | 110 | 0110000 | 425 |
| 102.125 - 139.125 | 120 | 0000001 | 425 |

| Range (Mbps) | Default Bit Rate (Mbps) | HSFREQRANGE[6:0] (Binary notation) | Osc_freq_target[11:0] (Decimal notation) |
|---|---|---|---|
| 111.625 - 149.625 | 130 | 0010001 | 425 |
| 121.125 - 160.125 | 140 | 0100001 | 425 |
| 130.625 - 170.625 | 150 | 0110001 | 425 |
| 140.125 - 181.125 | 160 | 0000010 | 425 |
| 149.625 - 191.625 | 170 | 0010010 | 425 |
| 159.125 - 202.125 | 180 | 0100010 | 425 |
| 168.625 - 212.625 | 190 | 0110010 | 425 |
| 182.875 - 228.375 | 205 | 0000011 | 425 |
| 197.125 - 244.125 | 220 | 0010011 | 425 |
| 211.375 - 259.875 | 235 | 0100011 | 425 |
| 225.625 - 275.625 | 250 | 0110011 | 425 |
| 249.375 - 301.875 | 275 | 0000100 | 425 |
| 273.125 - 328.125 | 300 | 0010100 | 425 |
| 296.875 - 354.375 | 325 | 0100101 | 425 |
| 320.625 - 380.625 | 350 | 0110101 | 425 |
| 368.125 - 433.125 | 400 | 0000101 | 425 |
| 415.625 - 485.625 | 450 | 0010110 | 425 |
| 463.125 - 538.125 | 500 | 0100110 | 425 |
| 510.625 - 590.625 | 550 | 0110111 | 425 |
| 558.125 - 643.125 | 600 | 0000111 | 425 |
| 605.625 - 695.625 | 650 | 0011000 | 425 |
| 653.125 - 748.125 | 700 | 0101000 | 425 |
| 700.625 - 800.625 | 750 | 0111001 | 425 |
| 748.125 - 853.125 | 800 | 0001001 | 425 |
| 795.625 - 905.625 | 850 | 0011001 | 425 |
| 843.125 - 958.125 | 900 | 0101001 | 425 |
| 890.625 - 1010.625 | 950 | 0111010 | 425 |
| 938.125 - 1063.125 | 1000 | 0001010 | 425 |
| 985.625 - 1115.625 | 1050 | 0011010 | 425 |
| 1033.125 - 1168.125 | 1100 | 0101010 | 425 |
| 1080.625 - 1220.625 | 1150 | 0111011 | 425 |
| 1128.125 - 1273.125 | 1200 | 0001011 | 425 |
| 1175.625 - 1325.625 | 1250 | 0011011 | 425 |
| 1223.125 - 1378.125 | 1300 | 0101011 | 425 |
| 1270.625 - 1430.625 | 1350 | 0111100 | 425 |

| Range (Mbps) | Default Bit Rate (Mbps) | HSFREQRANGE[6:0] (Binary notation) | Osc_freq_target[11:0] (Decimal notation) |
|---|---|---|---|
| 1318.125 - 1483.125 | 1400 | 0001100 | 425 |
| 1365.625 - 1535.625 | 1450 | 0011100 | 425 |
| 1413.125 - 1588.125 | 1500 | 0101100 | 425 |
| 1460.625 - 1640.625 | 1550 | 0111101 | 229 |
| 1508.125 - 1693.125 | 1600 | 0001101 | 237 |
| 1555.625 - 1745.625 | 1650 | 0011101 | 244 |
| 1603.125 - 1798.125 | 1700 | 0101110 | 252 |
| 1650.625 - 1850.625 | 1750 | 0111110 | 259 |
| 1698.125 - 1903.125 | 1800 | 0001110 | 266 |
| 1745.625 - 1955.625 | 1850 | 0011110 | 274 |
| 1793.125 - 2008.125 | 1900 | 0101111 | 281 |
| 1840.625 - 2060.625 | 1950 | 0111111 | 289 |
| 1888.125 - 2113.125 | 2000 | 0001111 | 296 |
| 1935.625 - 2165.625 | 2050 | 1000000 | 303 |
| 1983.125 - 2218.125 | 2100 | 1000001 | 311 |
| 2030.625 - 2270.625 | 2150 | 1000010 | 318 |
| 2078.125 - 2323.125 | 2200 | 1000011 | 325 |
| 2125.625 - 2375.625 | 2250 | 1000100 | 333 |
| 2173.125 - 2428.125 | 2300 | 1000101 | 340 |
| 2220.625 - 2480.625 | 2350 | 1000110 | 348 |
| 2268.125 - 2500 | 2400 | 1000111 | 355 |
| 2315.625 - 2500 | 2450 | 1001000 | 362 |
| 2363.125 - 2500 | 2500 | 1001001 | 370 |

By default, the D-PHY Start-up sequence performs termination calibration considering the presence of an external termination resistor.

The PHY consists of four data lanes, but applications can use four or lesser data lanes. In such cases, you are granted access to individual enabling signals, ENABLE_N(N=0,1,2,3 corresponding to each lanes), that control which lanes should be used and evolve through all the necessary initialization steps. The ENABLE_N must not be changed during operation. They should be set in power-down.

If only a single lane is to be enabled, that lane must be lane 0.

After the reset — DPHY_RSTZ/dphy_rstz and PHY_SHUTDOWNZ/phy_shutdownz—are released, the PHY begins an initialization sequence that allows its correct operation. PHY_SHUTDOWNZ/phy_shutdownz should precede DPHY_RSTZ/dphy_rstz; it is also assumed that the Configuration Clock is available and stable by that time. The initialization sequence determines that the bandgap and biasing blocks are enabled first. After the current references get settled, a second step is triggered where the internal calibrations are performed.

The deskew algorithm selection is done automatically based on HSFREQRANGE/hsfreqrange[6:0] configuration but can be overridden through PHTW, as well as all other Start-up calibration algorithms. When this second step is complete, the control is passed to the lanes, which handle the power management for LP/HS requests, enabling or disabling the corresponding receivers. All initialization steps are complete once the PHY_STOPSSTATE/phy_stopstatedata_N(N=0, 1, 2, 3) and PHY_STOPSSTATE/phy_stopstateclk get asserted (if D-PHY Transmitter is driving LP11).

For operating frequencies > 1.5 Gbps, a deskew compensation calibration algorithm can be executed (enabled by default for operating frequencies > 1.5 Gbps), if the RX D-PHY is triggered through a defined pattern sent by the TX after the D-PHY RX is already powered-up and in STOPSTATE.

Start-up configuration are provided below. For D-PHY configuration must be exercised after setting PHTC/testclr and before D-PHY leaves shutdown mode (both PHY_SHUTDOWNZ/phy_shutdownz and DPHY_RSTZ/dphy_rstz are cleared) and start D-PHY Start-up sequence.

T0: Set CSI2_RESETN/csi2_resetn = 1’b0； DPHY_RSTZ/dphy_rstz = 1'b0; PHY_SHUTDOWNZ/phy_shutdownz = 1'b0; PHTC/TESTCLR = 1'b1.

T1: Set FRXM/FORCERXMODE_N(N=0, 1, 2, 3) = 1'b1, OVR1/forcerxmode_N (N=0, 1, 2, 3) = 1’b1 to force lane into Wait for Stop State.

Set FLDC, FLDD, IDIC.

T2: Set PHTC/TESTCLR = 1'b0; CSI2_RESETN/csi2_resetn = 1’b1.

T3: PHY register is programmed/read with PHY in reset.

T3-1: Set PHYPLL/HSFREQRANGE[6:0] to the target operating frequencies according to Table 53.13 “Frequency Ranges and Defaults”.

T3-2: Refer to Table 53.13 “Frequency Ranges and Defaults” and configure through PHTW with the appropriate DDL target oscillation frequency (defined as osc_freq_target[11:0]) as follows. (*2)

PHTW = H’0100_0100

PHTW = H’01_**pp**_01E2 (where **pp** is osc_freq_target[7:0])

PHTW = H’010_**q**_01E3 (where **q** is osc_freq_target[11:8])

PHTW = H’0101_01E4

T3-3: (Only used in case the speed is less than or equal to 1.5 Gbps.) (*2)

PHTW = H’0100_0100

PHTW = H’013C_0108

T3-4: Set CSI0CLKFCPR/csi0clkfreqrange[7:0] = 8'b00001100.

T3-5: Set PHY_EN/ENABLE_N(N=0,1,2,3 corresponding to each lane) and PHY_EN/ENABLECLK = 1'b1.

T3-6: (Only used in case the speed is higher than 1.5 Gbps.) (*2)

PHTW = H’0100_0100

PHTW = H’0180_01E0

PHTW = H’0131_01E1

PHTW = H’0106_0100

PHTW = H’0111_0111

PHTW = H’0108_0100

PHTW = H’0111_0111

PHTW = H’010A_0100

PHTW = H’0111_0111

PHTW = H’010C_0100

PHTW = H’0111_0111

PHTW = H’0101_0100

PHTW = H’0131_01AA

PHTW = H’0105_0100

PHTW = H’0105_0109

PHTW = H’0107_0100

PHTW = H’0105_0109

PHTW = H’0109_0100

PHTW = H’0105_0109

PHTW = H’010B_0100

PHTW = H’0105_0109

T3-7:

PHTW = H’0101_0100

PHTW = H’0106_01ab

T4: Set PHY_SHUTDOWNZ/phy_shutdownz = 1'b1; DPHY_RSTZ/dphy_rstz = 1'b1.

T5: Internal calibrations ongoing. No configurations are accepted during power-on-reset (POR). Wait for PHTR[19:16] that POR is complete. (*2)

PHTW = H’0100_0100

PHTW = H’0000_011E

read PHTR and wait PHTR[19:16] = H’7

T6: (Skip)

T7: Wait until ST_PHYST/ST_STOPSTATE_N (N=0, 1, 2, 3) and ST_PHYST/ST_STOPSTATE_DCK are asserted indicating PHY is receiving LP11 in enabled datalanes and clocklane. (*1)

T8: Set FRXM/FORCERXMODE_N(N=0, 1, 2, 3) = 1'b0.

**Note**

- CPG/CSI2 0,1 MSTOP is off and CPG/CSI2 Clock is start before T0. (see 53.3.1 System)
- (*1) It is possible to skip ST_PHYST/ST_STOPSTATE_N (N=0, 1, 2, 3) and ST_PHYST/ST_STOPSTATE_DCK registers confirmation by software at T7. It is necessary to input D-PHY LP-11 to CSI2 and PHY must go through the stop state after T5 PHTR[19:16]=H'7 assertion as hardware. To skip T7 does not affect functional safety.
- (*2) Each step must be issued after both DWEN=0 and CWEN=0 are confirmed.

## 53.4 Usage Notes

### 53.4.1 LINK

#### 53.4.1.1 Error Handling

This chapter describes all the errors the LINK can identify. For each identified error, if applicable, a reset is recommended.

Table 53.14 LINK Error List

| Register | Field | Description | Error Frequency | Action/Consequence | Recommended reset |
|---|---|---|---|---|---|
| INT_ST_PHY_FATAL | phy_rxinvalidcodehs_* | High Speed Invalid Code Word Detection on lane *. | Single | Follow PHY recommendation | - |
| | | | Multiple | Follow PHY recommendation | csi2_resetn<br>dphy_rstz |
| | err_deskew | Reports whenever data is lost due to an existent skew between lanes greater than 2 rxwordclkhs. | Single | No action required. Autonomous Recovery. | - |
| | | | Multiple | Reset is recommended. | csi2_resetn |
| | phy_errsotsynchs_* | Start of transmission error on data lane * (no synchronization achieved). | Single | Follow PHY recommendation | - |
| | | | Multiple | Reset is recommended. | csi2_resetn<br>dphy_rstz |
| INT_ST_PKT_FATAL | err_ecc_double | D-PHY:<br>Header ECC contains at least 2 errors,<br>Unrecoverable | Single | No action required. Packets with this error are not delivered in IDI interface. | - |
| | | | Multiple | Reset is recommended. May indicate ecosystem issue. | csi2_resetn<br>dphy_rstz |
| INT_ST_BNDRY_FRAME_FATAL | err_f_bndry_match_vc* | Error matching Frame Start with Frame End for virtual channel *. | Single | No action required.<br>No consequence. | - |
| | | | Multiple | Reset is recommended.<br>May indicate ecosystem issue. | csi2_resetn<br>dphy_rstz |
| INT_ST_SEQ_FRAME_FATAL | err_f_seq_vc* | Incorrect Frame sequence detected in virtual channel *. | Single | No action required.<br>No consequence. | - |
| | | | Multiple | Reset is recommended.<br>May indicate ecosystem issue. | csi2_resetn<br>dphy_rstz |
| INT_ST_CRC_FRAME_FATAL | err_frame_data_vc* | Last received Frame in virtual channel *, had at least one CRC error. | Single | No action required.<br>May indicate ecosystem issue. | - |
| | | | Multiple | Reset is recommended.<br>May indicate ecosystem issue. | csi2_resetn<br>dphy_rstz |
| INT_ST_PLD_CRC_FATAL | err_crc_vc* | Payload Checksum error detected | Single | No action required. May indicate ecosystem issue. | - |

| Register | Field | Description | Error Frequency | Action/Consequence | Recommended reset |
|---|---|---|---|---|---|
| | | on virtual channel *. | Multiple | Reset is recommended. May indicate ecosystem issue. | csi2_resetn dphy_rstz |
| INT_ST_PHY | phy_erresc* | Escape Entry Error on data lane *. | Single | Follow PHY recommendation. | - |
| | | | Multiple | Reset is recommended. | csi2_resetn dphy_rstz |
| | phy_errsoths_* | Start of transmission error on data lane * (synchronization can still be achieved). | Single | Follow PHY recommendation | - |
| | | | Multiple | Reset is recommended. | csi2_resetn dphy_rstz |
| INT_ST_LINE | err_l_seq_di* | Error in the sequence of lines for vc* and dt*. | Single | No impact on the controller, If no other error is flagged may indicate issues at TX side. | - |
| | | | Multiple | No impact on the controller, If no other error is flagged may indicate issues at TX side. | - |
| | err_l_bndry_match_di* | Error matching line start with line end for vc* and dt*. | Single | No impact on the controller, If no other error is flagged may indicate issues at TX side. | - |
| | | | Multiple | No impact on the controller, If no other error is flagged may indicate issues at TX side. | - |
| INT_ST_DATA_ ID | err_id_vc* | Unrecognized or unimplemented data type detected in virtual channel *. | Single | No impact. | - |
| | | | Multiple | Check configurations is recommended. May indicate ecosystem issue. | - |
| INT_ST_ECC_CORRECTED | err_ecc_corrected* | D-PHY: Header error detected and corrected on virtual channel *. | Single | No impact on the controller. | - |
| | | | Multiple | No impact on the controller. May indicate link degradation. | - |

### 53.4.2 PHY

#### 53.4.2.1 D-PHY High-Speed Deskew Calibration (>1.5Gbps)

In order to achieve data rates in the High-Speed Data mode, a calibration mode was introduced in D-PHY 1.2 spec. The implemented algorithm will be exercised while D-PHY is in idle mode.

The initial deskew calibration is mandatory and the periodic deskew calibration is recommended. Over $2^{\wedge 14}$ UI is recommended for $T_{SKEWCAL}$ of the periodic deskew calibration. When "$2^{\wedge 14}$ UI" > 10us, over $2^{\wedge 13}$ UI is recommended.

## 53.5 Safety Mechanisms

### 53.5.1 Packet ECC and payload CRC of CSI2

**(a) Overview**

CSI2 packet ECC/CSI payload checker is to check consistency of incoming MIPI CSI-2 packet header data and packet payload data (pixel stream in one line). When using D-PHY mode, ECC (Error Check Code) code is used to check the packet header. For packet payload checking, CRC is used.

Supports fatal interruption related with Payload Checksum(CRC) error.

Supports interruption related with header error detected and corrected for a specific virtual channel.

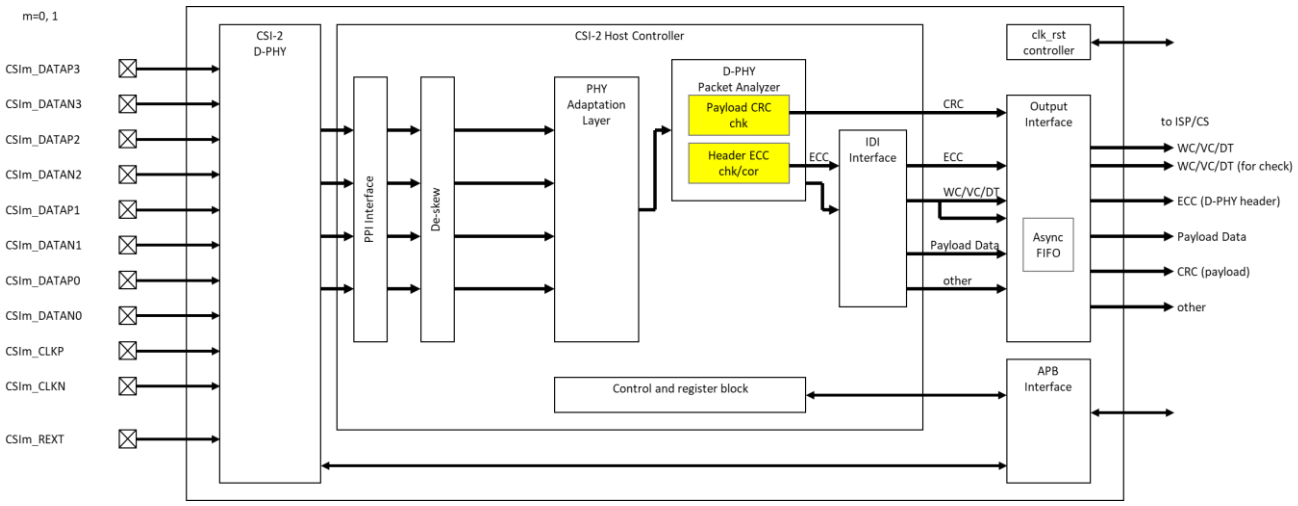



**Figure 53.8 Block Diagram for FuSa**

**(b) Related Registers**

**Table 53.17 Registers (Functional Safety Registers)**

| Register Name | Abbreviation | R/W | Offset Address | Initial Value | Access Size*3 |
|---|---|---|---|---|---|
| Functional AP fatal interruption caused by PHY | INT_ST_FAP_PHY_FATAL | RC | H'0360 | H'0000 0000 | 32 |
| Functional AP fatal interruption caused by PHY | INT_MSK_FAP_PHY_FATAL | R/W | H'0364 | H'0000 0000 | 32 |
| Mask for functional AP fatal interruption caused by PHY | INT_FORCE_FAP_PHY_FATAL | R/W | H'0368 | H'0000 0000 | 32 |
| Force for functional AP fatal interruption caused by PHY | INT_ST_FAP_PKT_FATAL | RC | H'0370 | H'0000 0000 | 32 |
| Functional AP fatal interruption caused during Packet Construction | INT_MSK_FAP_PKT_FATAL | R/W | H'0374 | H'0000 0000 | 32 |
| Mask for functional AP fatal interruption caused during Packet Construction | INT_FORCE_FAP_PKT_FATAL | R/W | H'0378 | H'0000 0000 | 32 |
| Force for functional AP fatal interruption caused during Packet Construction | INT_ST_FAP_PHY | RC | H'0390 | H'0000 0000 | 32 |
| Functional AP interruption caused by PHY | INT_MSK_FAP_PHY | R/W | H'0394 | H'0000 0000 | 32 |
| Mask for functional AP interruption caused by PHY | INT_FORCE_FAP_PHY | R/W | H'0398 | H'0000 0000 | 32 |
| Functional AP fatal Interruption caused by Frame Boundaries | INT_ST_FAP_BNDRY_FRAME_FATAL | RC | H'0420 | H'0000 0000 | 32 |
| Mask for functional AP fatal Interruption caused by Frame Boundaries | INT_MSK_FAP_BNDRY_FRAME_FATAL | R/W | H'0424 | H'0000 0000 | 32 |
| Force for functional AP fatal Interruption caused by Frame Boundaries | INT_FORCE_FAP_BNDRY_FRAME_FATAL | R/W | H'0428 | H'0000 0000 | 32 |
| Functional AP fatal interruption caused by Frame Sequence | INT_ST_FAP_SEQ_FRAME_FATAL | RC | H'0430 | H'0000 0000 | 32 |
| Mask for functional AP fatal interruption caused by Frame Sequence | INT_MSK_FAP_SEQ_FRAME_FATAL | R/W | H'0434 | H'0000 0000 | 32 |
| Force for functional AP fatal interruption caused by Frame Sequence | INT_FORCE_FAP_SEQ_FRAME_FATAL | R/W | H'0438 | H'0000 0000 | 32 |

| Register Name | Abbreviation | R/W | Offset Address | Initial Value | Access Size*3 |
|---|---|---|---|---|---|
| Functional AP fatal interruption caused by Payload CRC | INT_ST_FAP_PLD_CRC_FATAL | RC | H'0450 | H'0000 0000 | 32 |
| Mask for functional AP fatal interruption caused by Payload CRC. | INT_MSK_FAP_PLD_CRC_FATAL | R/W | H'0454 | H'0000 0000 | 32 |
| Force for functional AP fatal interruption caused by Payload CRC | INT_FORCE_FAP_PLD_CRC_FATAL | R/W | H'0458 | H'0000 0000 | 32 |
| Functional AP interruption caused by Header single bit errors. | INT_ST_FAP_ECC_CORRECTED | RC | H'0470 | H'0000 0000 | 32 |
| Mask for functional AP interruption caused by Header single bit errors. | INT_MSK_FAP_ECC_CORRECTED | R/W | H'0474 | H'0000 0000 | 32 |
| Force for functional AP interruption caused by Header single bit errors | INT_FORCE_FAP_ECC_CORRECTED | R/W | H'0478 | H'0000 0000 | 32 |

# 54. Image Signal Processor (ISP)/Channel Selector (CS)

## 54.1 Overview

The image signal processor (ISP) is an image pixel correction engine to correct input image signals from video input interfaces. This module supports one MIPI CSI-2 and two parallel digital video interfaces as the video input interfaces. As image processing algorithms, the ISP supports demosaicing (RGB pixel interpolation from a RAW image), white balance control, color space conversion from RGB to YUV, and the HDR image processing to support the HDR image sensor connection.

### 54.1.1 Features

- Video Input Interface
  - MIPI CSI-2
    - MIPI CSI-2 Interface (2.5 Gbps x 4 lane@D-PHY, 16 Virtual Channel (VC) supports ) x 2 ports
  - Parallel Digital Video Port (DVP)
    - Not supported
- Input format
  - Multi exposure High Dynamic Range (HDR) supports up to 2:1 multi-exposure, or one sensor combined HDR.
  - Maximum resolution:
    - 5120(w) x 4096 (h)
    - Only even numbers are available regarding image width (horizontal picture size).
  - Maximum pixel throughput :
    - 1 pixel/clock de-rated by 2-5% blanking requirement depending on configuration.
    - MIPI CSI-2 input
      - Maximum pixel rate via MIPI CSI-2 input per 1 port : 720 [Mpixel/s]
    - DVP Input
      - DVP input is not supported
  - Sensor data format :
    - MIPI CSI-2 input :
      - RAW8, RAW10, RAW12, RAW14, RAW16, RAW20, RAW24, YUV420 8-bit, YUV420 10-bit, YUV422 8-bit, YUV422 10-bit, YUV422 12-bit (*1), YUV422 Y 12-bit UV 8-bit (*1), RGB565, RGB888

      *1: This format is not supported in the MIPI CSI-2 standard.
    - DVP input :
      - DVP input is not supported
  - CFA patterns :
    - RGGB,
- Output format

- 2 independent image output channels (Human Vision Image and Machine Vision Image) are written out as the DMA via the AXI master interface
- Pixel format for human vision image : RGB, YUV422, YUV420
- Pixel format for machine vision image : RGB, YUV422, YUV420, LUV444, LUV422, LUV420

- Processing Functions
  - Defect Pixel Correction
  - Noise Filter
  - Local Tone Mapping
  - Histogram Generation (4color x 256bin x 16bit)
  - Demosaicing
  - Color Correction
  - Dual Video Output (Human Vision and Machine Vision)
- Color processing
  - : Demosaic RGGB, RCCG, RCCB, RCCC
- Interfaces
  - AXI master interface
    - Number of ports
      - Write only, AXI3, 256bit width x 2 ports for DMA of image output and histogram output
      - Write only, AXI3, 64bit width x 2 ports for DMA of defect pixel information output
    - Clock :
      - S0D1_VIO $\phi$
  - APB slave interface
    - Number of ports : 2
    - Bit width : 32 bits
    - Clock
      - S0D2_VIO $\phi$
  - Interrupts : 2 ports
  - Error request :
    - 2 ports
  - WUP(Wake UP) interface :
    - 4 ports
- Channel Selector
  - Routing of CSI packet based on VCs, data type, etc.

- Pixel reconstructor
- Supporting 16 VCs . 1 VC can be passed per one channel at VC based filtering.

- Other features
  - Meta data (embedded data) processing
    - Meta data can be transferred to the DRAM by the VIN
  - Image data Capturing
    - Incoming image data can be transferred to the DRAM by the VIN in parallel with ISP processing. This function is achieved by using the channels connected to VIN (Channel 4~11). When using this function, the VIN setting is additionally required.
  - LUT Based Pixel Reconstructor
    - Pixel Reconstructor in the Channel Selector supports non-standard pixel packing for the incoming MIPI CSI-2 pixel stream
    - The unpacking method can be selected by using the LUT. The LUT specifies the bit position in the pixel buffer (32 [bit] x N)
    - The following formats are supported.
      - YUV444 : Y16, U8, V8
      - YUV444: Y16, U10, V10
      - YUV444: Y16, U12, V12
      - YUV444: Y12, U8, V8
      - YUV422: Y12, UV8
  - De-interleaving Filter
    - The LUT based pixel sub-sampling at the last stage of filtering chain of Channel Selector
    - By setting pass/drop for each pixel in one window using the LUT, any sub-samplings can be realized
    - The window size (the LUT size) varies from 1 pixel (minimum) to 512 pixels (maximum)
  - Streaming Bridge (Memory-to-Memory operation with VSPX)
    - Memory-to-Memory operation is supported by using the “Streaming Bridge”
    - The ISP can receive input stream from the VSPX via the “Streaming Bridge”
    - The VSPX reads input images for the ISP from DRAM and provides the ISP with them
    - Segmented image is supported.
    - Static gear down of ISP core clock frequency for the purpose of reduction of power consumption.
  - Pre-processing of RAW Image
    - Supported processing
      - RAW domain image scaling: Scaling horizontal size and vertical size of input image to 1/2.
      - CFA conversion: 4x4 RGB-IR to 2x2 Bayer (RGGB)

- ✧ Pixel format
  - Input pixel format : RAW8, RAW10, RAW12, RAW14, RAW16 , RAW20, RAW24
- ✧ Input image size :
  - Image width : Minimum 128 [pixel/line]. Maximum 5120 [pixel/line].
  - Image height : Minimum 128 [line]. Maximum 4096 [line].
- ✧ Restrictions of image width and image height
  - Input image width (number of pixels per one line) must be multiple of 2 (even value only).
  - Input image height (number of lines per one frame) must be multiple of 2 (even value only).
  - Scaling ratio is only 1/2 and aspect ratio shall be equal to the input image.
    - When 1/2 scaling, both horizontal and vertical shall be scaled. Scaling horizontal only or scaling vertical only is forbidden.
  - When 1/2 scaling, simultaneous use with either de-interleaving filter or line count filter is prohibited

### 54.1.2 Base Address

| Base Address Name | Base Address |
|---|---|
| ISP_CS0 | H'FED0 0000 |
| ISP_CS1 | H'FED2 0000 |
| ISP_CORE0 | H'FEC0 0000 |

### 54.1.3 Block Diagram

#### 54.1.3.1 ISP Channel Selector 0

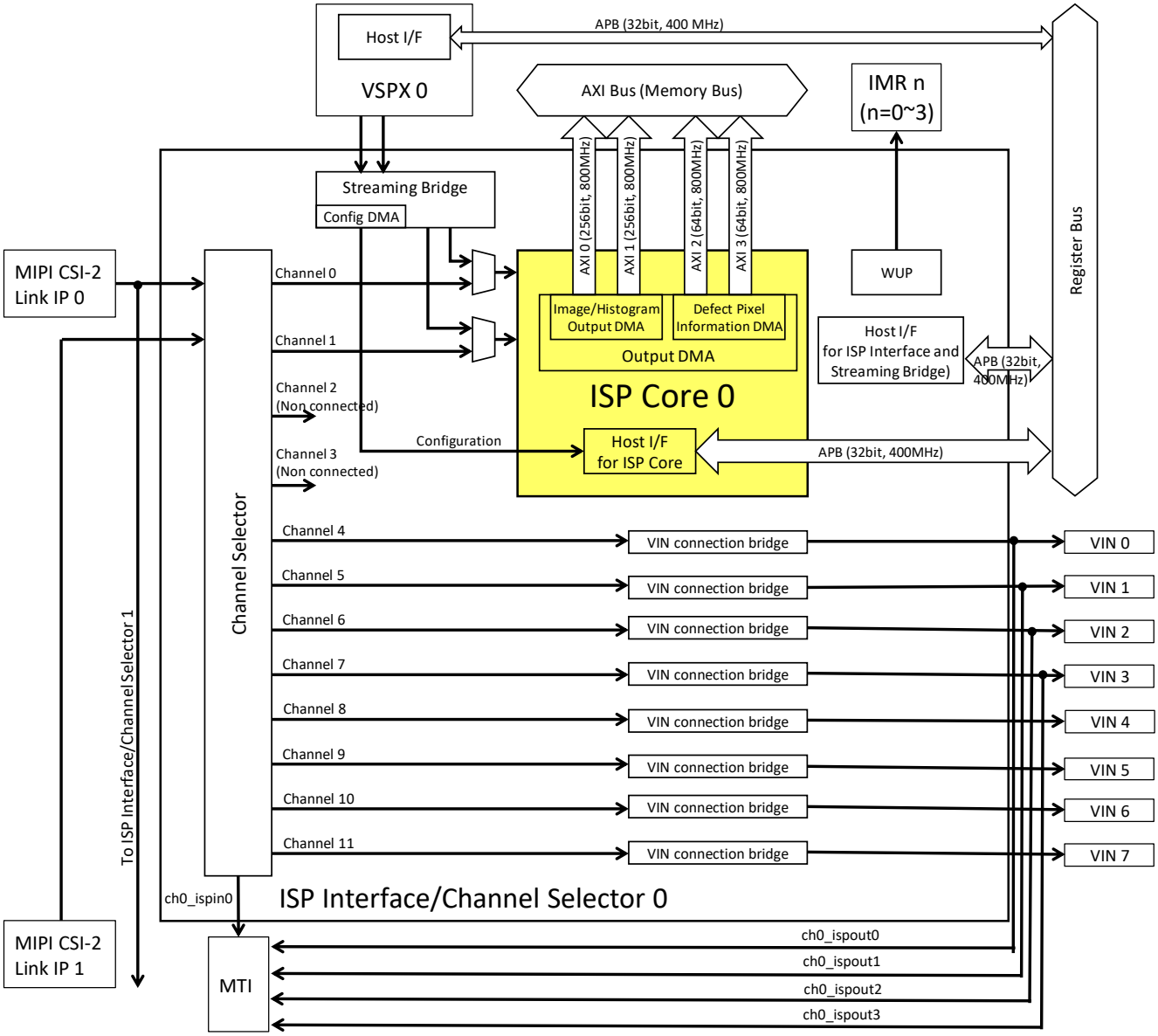



Figure 54.3 Block Diagram of ISP/Channel Selector 0

#### 54.1.3.2 ISP/Channel Selector 1

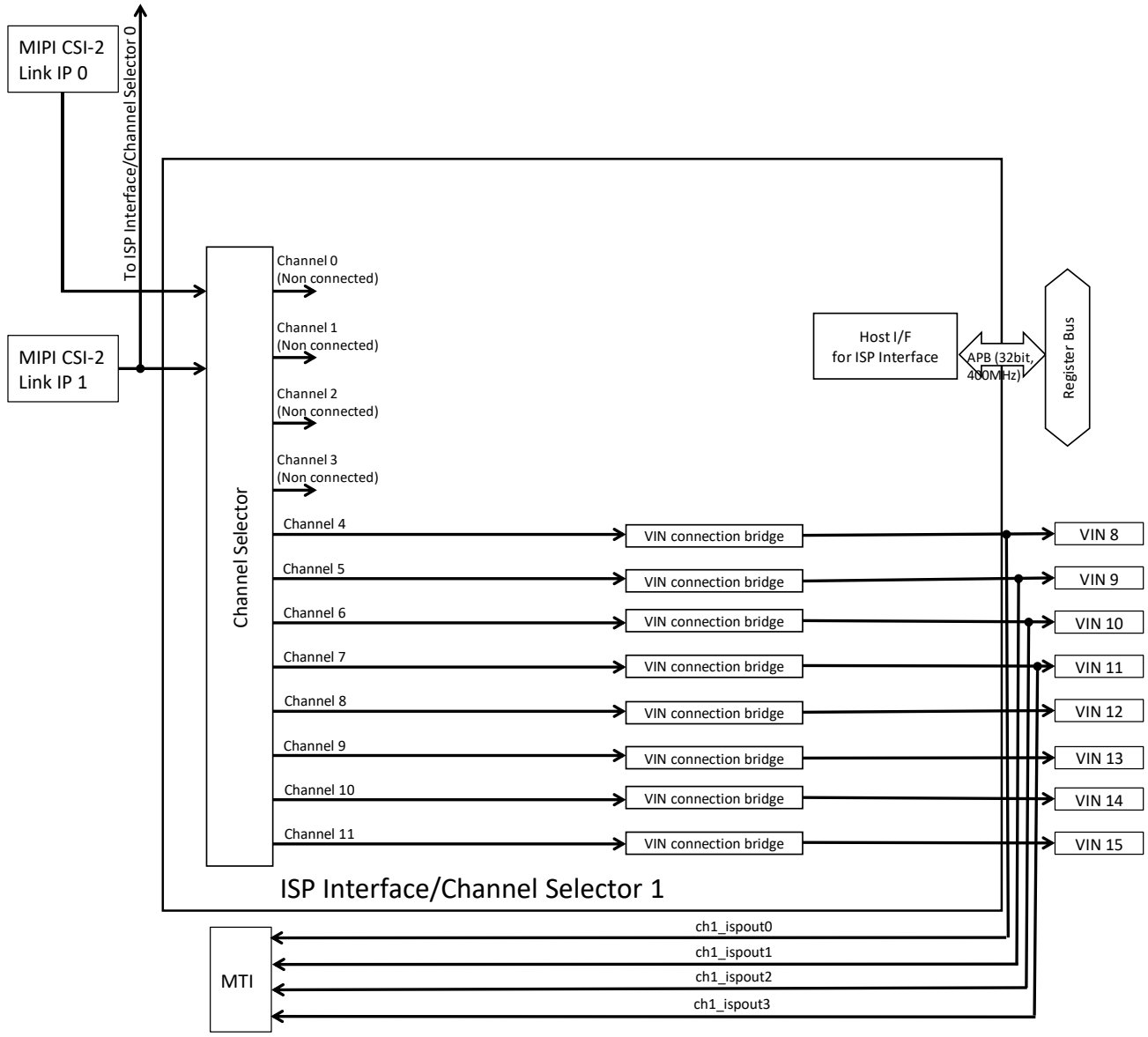



Figure 54.4 Block Diagram of ISP/Channel Selector 1

## 54.2 Register List

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ISP_CS | ISP Version Register [c](c=0-1) | ISPVCR_[c] | R | <ISP_CS[c]>+H'0000 | Undefined | 32 |
| ISP_CS | ISP Input FIFO Control Register [c](c=0-1) | ISPFIFOCTRL_[c] | R/W | <ISP_CS[c]>+H'0004 | H'00003207 | 32 |
| ISP_CS | ISP Input Select Control Register [c](c=0-1) | ISPINPUTSEL0_[c] | R/W | <ISP_CS[c]>+H'0008 | H'40121000 | 32 |
| ISP_CS | ISP Start Register [c](c=0-1) | ISPSTART_[c] | R/W | <ISP_CS[c]>+H'0014 | H'00000000 | 32 |
| ISP_CS | ISP Interrupt Status Register [c](c=0-1) | ISPINT_STATUS_[c] | R | <ISP_CS[c]>+H'0040 | H'00000000 | 32 |
| ISP_CS | ISP Error Request0 Status Register [c](c=0-1) | ISPERR0_STATUS_[c] | R | <ISP_CS[c]>+H'0044 | H'00000000 | 32 |
| ISP_CS | ISP Error Request1 Status Register [c](c=0-1) | ISPERR1_STATUS_[c] | R | <ISP_CS[c]>+H'0048 | H'00000000 | 32 |
| ISP_CS | ISP Error Request2 Status Register [c](c=0-1) | ISPERR2_STATUS_[c] | R | <ISP_CS[c]>+H'004C | H'00000000 | 32 |
| ISP_CS | ISP Error Request3 Status Register [c](c=0-1) | ISPERR3_STATUS_[c] | R | <ISP_CS[c]>+H'0080 | H'00000000 | 32 |
| ISP_CS | ISP Error Request4 Status Register [c](c=0-1) | ISPERR4_STATUS_[c] | R | <ISP_CS[c]>+H'0084 | H'00000000 | 32 |
| ISP_CS | ISP Interrupt Clear Register [c](c=0-1) | ISPINT_CLEAR_[c] | W | <ISP_CS[c]>+H'0050 | H'00000000 | 32 |
| ISP_CS | ISP Error Request0 Clear Register [c](c=0-1) | ISPERR0_CLEAR_[c] | W | <ISP_CS[c]>+H'0054 | H'00000000 | 32 |
| ISP_CS | ISP Error Request1 Clear Register [c](c=0-1) | ISPERR1_CLEAR_[c] | W | <ISP_CS[c]>+H'0058 | H'00000000 | 32 |
| ISP_CS | ISP Error Request2 Clear Register [c](c=0-1) | ISPERR2_CLEAR_[c] | W | <ISP_CS[c]>+H'005C | H'00000000 | 32 |
| ISP_CS | ISP Error Request3 Clear Register [c](c=0-1) | ISPERR3_CLEAR_[c] | W | <ISP_CS[c]>+H'0090 | H'00000000 | 32 |
| ISP_CS | ISP Error Request4 Clear Register [c](c=0-1) | ISPERR4_CLEAR_[c] | W | <ISP_CS[c]>+H'0094 | H'00000000 | 32 |
| ISP_CS | ISP Interrupt Enable Register [c](c=0-1) | ISPINT_ENABLE_[c] | R/W | <ISP_CS[c]>+H'0060 | H'00000000 | 32 |
| ISP_CS | ISP Error Request0 Enable Register [c](c=0-1) | ISPERR0_ENABLE_[c] | R/W | <ISP_CS[c]>+H'0064 | H'00000000 | 32 |
| ISP_CS | ISP Error Request1 Enable Register [c](c=0-1) | ISPERR1_ENABLE_[c] | R/W | <ISP_CS[c]>+H'0068 | H'00000000 | 32 |
| ISP_CS | ISP Error Request2 Enable Register [c](c=0-1) | ISPERR2_ENABLE_[c] | R/W | <ISP_CS[c]>+H'006C | H'00000000 | 32 |
| ISP_CS | ISP Error Request3 Enable Register [c](c=0-1) | ISPERR3_ENABLE_[c] | R/W | <ISP_CS[c]>+H'00A0 | H'00000000 | 32 |
| ISP_CS | ISP Error Request4 Enable Register [c](c=0-1) | ISPERR4_ENABLE_[c] | R/W | <ISP_CS[c]>+H'00A4 | H'00000000 | 32 |
| ISP_CS | ISP Padding Control for CSI FIFO [c](c=0-1) | ISP_PADDING_CTRL_[c] | R/W | <ISP_CS[c]>+H'00C0 | H'70080020 | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ISP_CS | Pixel Reconstructor Custom Format Register[n] [c](n=0-7)(c=0-1) | ISPPROC_CUSTOM_FORMAT[n]_CTRL_[c] | R/W | <ISP_CS[c]>+H'1010+4*[n] | H'00000000 | 32 |
| ISP_CS | Pixel Reconstructor Processing Mode Register[k] [c](k=0-63)(c=0-1) | ISPPROCMODE_DT[k]_[c] | R/W | <ISP_CS[c]>+H'1100+4*[k] | H'00000000 | 32 |
| ISP_CS | Pixel Reconstructor Custom Format LUT Register[m]_[s] [c](m=0-63)(s=0-7)(c=0-1) | ISPPROC_CUSTOM_FORMAT_LUT0_W[m]_B[s]_[c] | R/W | <ISP_CS[c]>+H'8000+H'20*[m]+4*[s] | H'00000000 | 32 |
| ISP_CS | Pixel Count Checker MAX Pixel Register CH[d] [c](d=0-11)(c=0-1) | ISP_PIXEL_COUNT_MAX_CH[d]_[c] | R/W | <ISP_CS[c]>+H'4000+H'10*[d] | H'03FFFFF | 32 |
| ISP_CS | Pixel Count Checker MIN Pixel Register CH[d] [c](d=0-11)(c=0-1) | ISP_PIXEL_COUNT_MIN_CH[d]_[c] | R/W | <ISP_CS[c]>+H'4004+H'10*[d] | H'00000000 | 32 |
| ISP_CS | Pixel Count Checker Monitor Pixel Register CH[d] [c](d=0-11)(c=0-1) | ISP_PIXEL_COUNT_MONITOR_CH[d]_[c] | R/W | <ISP_CS[c]>+H'4008+H'10*[d] | H'00000000 | 32 |
| ISP_CS | Channel Selector VC and ID Filter Register CH[d] [c](d=0-11)(c=0-1) | ISPCS_FILER_ID_CH[d]_[c] | R/W | <ISP_CS[c]>+H'3000+H'100*[d] | H'00000001 | 32 |
| ISP_CS | Channel Selector Line Count Modulo Register VC0-7 CH[d] [c](d=0-11)(c=0-1) | ISPCS_LC_MODULO_CH[d]_[c] | R/W | <ISP_CS[c]>+H'3004+H'100*[d] | H'00000000 | 32 |
| ISP_CS | Channel Selector Line Count Modulo Register VC8-15 CH[d] [c](d=0-11)(c=0-1) | ISPCS_LC_MODULO1_CH[d]_[c] | R/W | <ISP_CS[c]>+H'3010+H'100*[d] | H'00000000 | 32 |
| ISP_CS | Channel Selector DT Code03 Register CH[d] [c](d=0-11)(c=0-1) | ISPCS_DT_CODE03_CH[d]_[c] | R/W | <ISP_CS[c]>+H'3008+H'100*[d] | H'00000000 | 32 |
| ISP_CS | Channel Selector DT Code47 Register CH[d] [c](d=0-11)(c=0-1) | ISPCS_DT_CODE47_CH[d]_[c] | R/W | <ISP_CS[c]>+H'300C+H'100*[d] | H'00000000 | 32 |
| ISP_CS | Channel Selector Horizontal Clipping Register DT[h] CH[d] [c](d=0-11)(h=0-3)(c=0-1) | ISPCS_H_CLIP_DT_CODE[h]_CH[d]_[c] | R/W | <ISP_CS[c]>+H'3020+H'100*[d]+4*[h] | H'3FFF0000 | 32 |
| ISP_CS | Channel Selector Vertical Clipping Register DT[v] CH[d] [c](d=0-11)(v=0-3)(c=0-1) | ISPCS_V_CLIP_DT_CODE[v]_CH[d]_[c] | R/W | <ISP_CS[c]>+H'3030+H'100*[d]+4*[v] | H'3FFF0000 | 32 |
| ISP_CS | Channel Selector De-Interleaving Filter Control CH[d] [c](d=0-11)(c=0-1) | ISPCS_DI_FILTER_CTRL_CH[d]_[c] | R/W | <ISP_CS[c]>+H'3040+H'100*[d] | H'000001FF | 32 |
| ISP_CS | Channel Selector De-Interleaving Filter LUT[p] CH[d] [c](d=0-11)(p=0-15)(c=0-1) | ISPCS_DI_FILTER_LUT[p]_CH[d]_[c] | R/W | <ISP_CS[c]>+H'3080+H'100*[d]+4*[p] | H'FFFFFFFF | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ISP_CS | Write Protect Control Register [c](c=0-1) | ISPWP_CTRL_[c] | R/W | <ISP_CS[c]>+H'0100 | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control FIELD MAX VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_FIELD_MAX_VC[b]_[c] | R/W | <ISP_CS[c]>+H'4800+H'100*[b] | H'FFFFFFFF | 32 |
| ISP_CS | ISP Sync Checker Control FIELD MIN VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_FIELD_MIN_VC[b]_[c] | R/W | <ISP_CS[c]>+H'4804+H'100*[b] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control VSYNC MAX VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_VSYNC_MAX_VC[b]_[c] | R/W | <ISP_CS[c]>+H'4808+H'100*[b] | H'FFFFFFFF | 32 |
| ISP_CS | ISP Sync Checker Control VSYNC MIN VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_VSYNC_MIN_VC[b]_[c] | R/W | <ISP_CS[c]>+H'480C+H'100*[b] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control HSYNC Rise MAX VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_HSYNC_RISE_MAX_VC[b]_[c] | R/W | <ISP_CS[c]>+H'4810+H'100*[b] | H'0000FFFF | 32 |
| ISP_CS | ISP Sync Checker Control HSYNC Rise MIN VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_HSYNC_RISE_MIN_VC[b]_[c] | R/W | <ISP_CS[c]>+H'4814+H'100*[b] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control HSYNC Toggle MAX VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_HSYNC_TOGGLE_MAX_VC[b]_[c] | R/W | <ISP_CS[c]>+H'4818+H'100*[b] | H'FFFFFFFF | 32 |
| ISP_CS | ISP Sync Checker Control HSYNC Toggle MIN VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_HSYNC_TOGGLE_MIN_VC[b]_[c] | R/W | <ISP_CS[c]>+H'481C+H'100*[b] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control DE MAX VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_DE_MAX_VC[b]_[c] | R/W | <ISP_CS[c]>+H'4820+H'100*[b] | H'0000FFFF | 32 |
| ISP_CS | ISP Sync Checker Control DE MIN VC[b] [c](b=0-7)(c=0-1) | ISP_SYNC_CHECK_DE_MIN_VC[b]_[c] | R/W | <ISP_CS[c]>+H'4824+H'100*[b] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control FIELD MAX VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_FIELD_MAX_VC[g]_[c] | R/W | <ISP_CS[c]>+H'6000+H'100*[g] | H'FFFFFFFF | 32 |
| ISP_CS | ISP Sync Checker Control FIELD MIN VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_FIELD_MIN_VC[g]_[c] | R/W | <ISP_CS[c]>+H'6004+H'100*[g] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control VSYNC MAX VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_VSYNC_MAX_VC[g]_[c] | R/W | <ISP_CS[c]>+H'6008+H'100*[g] | H'FFFFFFFF | 32 |
| ISP_CS | ISP Sync Checker Control VSYNC MIN VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_VSYNC_MIN_VC[g]_[c] | R/W | <ISP_CS[c]>+H'600C+H'100*[g] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control HSYNC Rise MAX VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_HSYNC_RISE_MAX_VC[g]_[c] | R/W | <ISP_CS[c]>+H'6010+H'100*[g] | H'0000FFFF | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ISP_CS | ISP Sync Checker Control HSYNC Rise MIN VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_HSYNC_RISE_MIN_VC[g]_[c] | R/W | <ISP_CS[c]>+H'6014+H'100*[g] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control HSYNC Toggle MAX VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_HSYNC_TOGGLE_MAX_VC[g]_[c] | R/W | <ISP_CS[c]>+H'6018+H'100*[g] | H'FFFFFFFF | 32 |
| ISP_CS | ISP Sync Checker Control HSYNC Toggle MIN VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_HSYNC_TOGGLE_MIN_VC[g]_[c] | R/W | <ISP_CS[c]>+H'601C+H'100*[g] | H'00000000 | 32 |
| ISP_CS | ISP Sync Checker Control DE MAX VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_DE_MAX_VC[g]_[c] | R/W | <ISP_CS[c]>+H'6020+H'100*[g] | H'0000FFFF | 32 |
| ISP_CS | ISP Sync Checker Control DE MIN VC[g] [c](g=8-15)(c=0-1) | ISP_SYNC_CHECK_DE_MIN_VC[g]_[c] | R/W | <ISP_CS[c]>+H'6024+H'100*[g] | H'00000000 | 32 |
| ISP_CS | ISP Streamer Mode Register [c](c=0-1) | ISP_STREAMER_MODE_[c] | R/W | <ISP_CS[c]>+H'7000 | H'00000000 | 32 |
| ISP_CS | ISP Streamer V Blank Register [c](c=0-1) | ISP_STREAMER_V_BLANK_[c] | R/W | <ISP_CS[c]>+H'7004 | H'00000800 | 32 |
| ISP_CS | ISP Streamer H Blank Register [c](c=0-1) | ISP_STREAMER_H_BLANK_[c] | R/W | <ISP_CS[c]>+H'7008 | H'00000040 | 32 |
| ISP_CS | ISP Streamer Config DMA Control Register0 [c](c=0-1) | ISP_STREAMER_CONFIG_DMA_CONTROL_[c] | R/W | <ISP_CS[c]>+H'7100 | H'FE880300 | 32 |
| ISP_CS | ISP Streamer Config DMA Status [c](c=0-1) | ISP_STREAMER_CONFIG_DMA_STATUS_[c] | R | <ISP_CS[c]>+H'7804 | H'00000000 | 32 |
| ISP_CS | ISP Streamer Config DMA Control Register1 [c](c=0-1) | ISP_STREAMER_CONFIG_DMA_CONTROL1_[c] | R/W | <ISP_CS[c]>+H'2100 | H'00000000 | 32 |
| ISP_CS | ISP Streamer Config DMA Control Register2 [c](c=0-1) | ISP_STREAMER_CONFIG_DMA_CONTROL2_[c] | R/W | <ISP_CS[c]>+H'2104 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Mode Register [c](c=0-1) | RAW_SCALER_MODE_[c] | R/W | <ISP_CS[c]>+H'9800 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Output Vblank Register [c](c=0-1) | RAW_SCALER_OUT_VBLANK_[c] | R/W | <ISP_CS[c]>+H'9804 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Output Hblank Register [c](c=0-1) | RAW_SCALER_OUT_HBLANK_[c] | R/W | <ISP_CS[c]>+H'9808 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Output Sync Interval Register [c](c=0-1) | RAW_SCALER_OUT_VS_HS_[c] | R/W | <ISP_CS[c]>+H'980C | H'00000000 | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ISP_CS | ISP RAW Scaler Input Left Shift Register [c](c=0-1) | RAW_SCALER_INPUT_LSHIFT_[c] | R/W | <ISP_CS[c]>+H'9810 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Image Size Register [c](c=0-1) | RAW_SCALER_IMAGE_SIZE_[c] | R/W | <ISP_CS[c]>+H'9814 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Right Shift Register [c](c=0-1) | RAW_SCALER_CONV_RSHIFT_[c] | R/W | <ISP_CS[c]>+H'9818 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Coefficient Select ROW0 Register [c](c=0-1) | RAW_SCALER_COEFF_SEL_ROW0_[c] | R/W | <ISP_CS[c]>+H'9820 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Coefficient Select ROW1 Register [c](c=0-1) | RAW_SCALER_COEFF_SEL_ROW1_[c] | R/W | <ISP_CS[c]>+H'9824 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Coefficient Select ROW2 Register [c](c=0-1) | RAW_SCALER_COEFF_SEL_ROW2_[c] | R/W | <ISP_CS[c]>+H'9828 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Coefficient Select ROW3 Register [c](c=0-1) | RAW_SCALER_COEFF_SEL_ROW3_[c] | R/W | <ISP_CS[c]>+H'982C | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Rounding0 Register [c](c=0-1) | RAW_SCALER_ROUNDING0_[c] | R/W | <ISP_CS[c]>+H'9830 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Rounding1 Register [c](c=0-1) | RAW_SCALER_ROUNDING1_[c] | R/W | <ISP_CS[c]>+H'9834 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Rounding2 Register [c](c=0-1) | RAW_SCALER_ROUNDING2_[c] | R/W | <ISP_CS[c]>+H'9838 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Rounding3 Register [c](c=0-1) | RAW_SCALER_ROUNDING3_[c] | R/W | <ISP_CS[c]>+H'983C | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Minimum Saturation Register [c](c=0-1) | RAW_SCALER_SAT_MIN_[c] | R/W | <ISP_CS[c]>+H'9840 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Maximum Saturation Register [c](c=0-1) | RAW_SCALER_SAT_MAX_[c] | R/W | <ISP_CS[c]>+H'9844 | H'00FFFFFF | 32 |
| ISP_CS | ISP RAW Scaler Final Left Shift Register [c](c=0-1) | RAW_SCALER_SAT_FINAL_LSHIFT_[c] | R/W | <ISP_CS[c]>+H'9848 | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M2_M2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M2_M2_[c] | R/W | <ISP_CS[c]>+H'9000+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M2_M1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M2_M1_[c] | R/W | <ISP_CS[c]>+H'9004+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] | RAW_SCALER_COEFF[e]_M2_00_[c] | R/W | <ISP_CS[c]>+H'9008+H'100*[e] | H'00000000 | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| | Register_M2_00 [c](e=0-3)(c=0-1) | | | | | |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M2_P1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M2_P1_[c] | R/W | <ISP_CS[c]>+H'900C+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M2_P2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M2_P2_[c] | R/W | <ISP_CS[c]>+H'9010+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M1_M2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M1_M2_[c] | R/W | <ISP_CS[c]>+H'9014+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M1_M1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M1_M1_[c] | R/W | <ISP_CS[c]>+H'9018+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M1_00 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M1_00_[c] | R/W | <ISP_CS[c]>+H'901C+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M1_P1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M1_P1_[c] | R/W | <ISP_CS[c]>+H'9020+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_M1_P2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_M1_P2_[c] | R/W | <ISP_CS[c]>+H'9024+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_00_M2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_00_M2_[c] | R/W | <ISP_CS[c]>+H'9028+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_00_M1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_00_M1_[c] | R/W | <ISP_CS[c]>+H'902C+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_00_00 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_00_00_[c] | R/W | <ISP_CS[c]>+H'9030+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_00_P1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_00_P1_[c] | R/W | <ISP_CS[c]>+H'9034+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_00_P2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_00_P2_[c] | R/W | <ISP_CS[c]>+H'9038+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P1_M2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P1_M2_[c] | R/W | <ISP_CS[c]>+H'903C+H'100*[e] | H'00000000 | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P1_M1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P1_M1_[c] | R/W | <ISP_CS[c]>+H'9040+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P1_00 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P1_00_[c] | R/W | <ISP_CS[c]>+H'9044+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P1_P1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P1_P1_[c] | R/W | <ISP_CS[c]>+H'9048+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P1_P2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P1_P2_[c] | R/W | <ISP_CS[c]>+H'904C+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P2_M2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P2_M2_[c] | R/W | <ISP_CS[c]>+H'9050+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P2_M1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P2_M1_[c] | R/W | <ISP_CS[c]>+H'9054+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P2_00 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P2_00_[c] | R/W | <ISP_CS[c]>+H'9058+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P2_P1 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P2_P1_[c] | R/W | <ISP_CS[c]>+H'905C+H'100*[e] | H'00000000 | 32 |
| ISP_CS | ISP RAW Scaler Convolution Coefficient[e] Register_P2_P2 [c](e=0-3)(c=0-1) | RAW_SCALER_COEFF[e]_P2_P2_[c] | R/W | <ISP_CS[c]>+H'9060+H'100*[e] | H'00000000 | 32 |
| ISP_CORE | ISP Core Interrupt Status [c](c=0-0) | ISPCORE_INT_STATUS_[c] | R | <ISP_CORE[c]>+H'80000 | H'00000002 | 32 |
| ISP_CORE | ISP Core Interrupt Enable [c](c=0-0) | ISPCORE_INT_ENABLE_[c] | R/W | <ISP_CORE[c]>+H'80004 | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Mode Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_MODE_[c] | R/W | <ISP_CORE[c]>+H'84000+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Pixel LSB Position Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_PIXEL_POSITION_[c] | R/W | <ISP_CORE[c]>+H'84004+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Pixel Bitwidth Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_PIXEL_BITWIDTH_MINUS1_[c] | R/W | <ISP_CORE[c]>+H'84008+H'1000*[i]+H'100*[f] | H'00000000 | 32 |

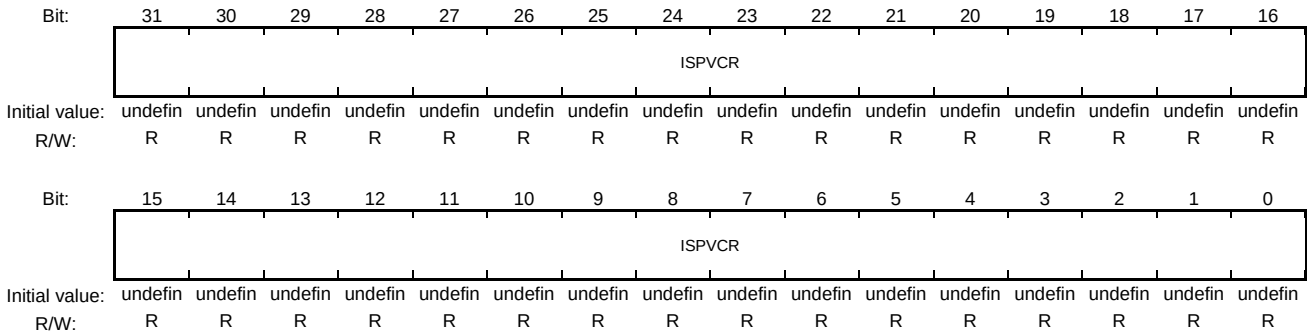
| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Pixel BPP register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_PIXEL_BPP_[c] | R/W | <ISP_CORE[c]>+H'8400C+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Base Address component0 Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_BASE_ADDRESS_COMP0_[c] | R/W | <ISP_CORE[c]>+H'84010+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Base Address component1 Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_BASE_ADDRESS_COMP1_[c] | R/W | <ISP_CORE[c]>+H'84014+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Base Address component2 Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_BASE_ADDRESS_COMP2_[c] | R/W | <ISP_CORE[c]>+H'84018+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Base Address component3 Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_BASE_ADDRESS_COMP3_[c] | R/W | <ISP_CORE[c]>+H'8401C+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Stride component0 Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_STRIDE_COMP0_[c] | R/W | <ISP_CORE[c]>+H'84020+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Stride component1 Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_STRIDE_COMP1_[c] | R/W | <ISP_CORE[c]>+H'84024+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Stride component2 Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_STRIDE_COMP2_[c] | R/W | <ISP_CORE[c]>+H'84028+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA Stride component3 Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_STRIDE_COMP3_[c] | R/W | <ISP_CORE[c]>+H'8402C+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame[f] Output DMA AXI ID Register [c](i=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME[f]_AXI_ID_[c] | R/W | <ISP_CORE[c]>+H'84030+H'1000*[i]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Histogram Output DMA Base Address Frame0 Register [c](c=0-0) | ISPCORE_DMA_HISTOGRAM_BASE_ADDRESS_FRAME0_[c] | R/W | <ISP_CORE[c]>+H'84500 | H'00000000 | 32 |
| ISP_CORE | ISP Core Histogram Output DMA Base | ISPCORE_DMA_HISTOGRAM_BASE_ADDRE | R/W | <ISP_CORE[c]>+H'84504 | H'00000000 | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| | Address Frame1 Register [c](c=0-0) | SS_FRAME1_[c ] | | | | |
| ISP_CORE | ISP Core Histogram Output DMA Base Address Frame2 Register [c](c=0-0) | ISPCORE_DMA _HISTOGRAM_ BASE_ADDRE SS_FRAME2_[c ] | R/W | <ISP_CORE[c]>+H'845 08 | H'00000000 | 32 |
| ISP_CORE | ISP Core Histogram Output DMA Base Address Frame3 Register [c](c=0-0) | ISPCORE_DMA _HISTOGRAM_ BASE_ADDRE SS_FRAME3_[c ] | R/W | <ISP_CORE[c]>+H'845 0C | H'00000000 | 32 |
| ISP_CORE | ISP Core Histogram Output DMA AXI ID Register [c](c=0-0) | ISPCORE_DMA _HISTOGAM_A XI_ID_[c] | R/W | <ISP_CORE[c]>+H'845 10 | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Output DMA AXI Config Register [c](i=0-1)(c=0-0) | ISPCORE_DMA _IMAGE[i]_AXI_ CONFIG_[c] | R/W | <ISP_CORE[c]>+H'848 00+H'1000*[i] | H'00000000 | 32 |
| ISP_CORE | ISP Wakeup Config Register [c](c=0-0) | ISPCORE_WU P_CONFIG_[c] | R/W | <ISP_CORE[c]>+H'848 04 | H'00000000 | 32 |
| ISP_CORE | ISP Wakeup Position0 Register [c](c=0-0) | ISPCORE_WU P_POS0_[c] | R/W | <ISP_CORE[c]>+H'848 10 | H'00000000 | 32 |
| ISP_CORE | ISP Wakeup Position1 Register [c](c=0-0) | ISPCORE_WU P_POS1_[c] | R/W | <ISP_CORE[c]>+H'848 14 | H'00000000 | 32 |
| ISP_CORE | ISP Wakeup Position2 Register [c](c=0-0) | ISPCORE_WU P_POS2_[c] | R/W | <ISP_CORE[c]>+H'848 18 | H'00000000 | 32 |
| ISP_CORE | ISP Wakeup Position3 Register [c](c=0-0) | ISPCORE_WU P_POS3_[c] | R/W | <ISP_CORE[c]>+H'848 1C | H'00000000 | 32 |
| ISP_CS | ISP Wakeup Enable Register [c](c=0-0) | ISPCORE_WU P_ENABLE_CH ANNELS_[c] | R/W | <ISP_CS[c]>+H'2000 | H'00000000 | 32 |
| ISP_CORE | ISP Core Defect Pixel Information[j] Frame[f] DMA Base Address Register [c](j=0-1)(f=0-3)(c=0-0) | ISPCORE_DMA _DPCC[j]_FRA ME[f]_BASE_A DDRESS_[c] | R/W | <ISP_CORE[c]>+H'860 10+H'1000*[j]+H'100*[f] | H'00000000 | 32 |
| ISP_CORE | ISP Core Defect Pixel Information[j] DMA AXI Config Register [c](j=0-1)(c=0-0) | ISPCORE_DMA _DPCC[j]_AXI_ CONFIG_[c] | R/W | <ISP_CORE[c]>+H'868 00+H'1000*[j] | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Output DMA Flushing Out Register [c](i=0-1)(c=0-0) | ISPCORE_DMA _IMAGE[i]_FLU SH_OUT_[c] | R/W | <ISP_CORE[c]>+H'844 00+H'1000*[i] | H'00000000 | 32 |
| ISP_CORE | ISP Core Defect Pixel Information[j]. Monitor of the number of detected defect pixel count in previous frame [c](j=0-1)(c=0-0) | ISPCORE_DMA _DPCC[j]_PIXE L_COUNT_[c] | R | <ISP_CORE[c]>+H'860 00+H'1000*[j]+H'918 | H'00000000 | 32 |
| ISP_CORE | ISP Core Defect Pixel Information[j]. Upper limit of number of defect pixel count [c](j=0-1)(c=0-0) | ISPCORE_DMA _DPCC[j]_PIXE L_COUNT_MA X_[c] | R/W | <ISP_CORE[c]>+H'860 00+H'1000*[j]+H'91C | H'00FFFFF F | 32 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ISP_CORE | ISP Core Back Pressure Setting [c](c=0-0) | ISPCORE_BACK_PRESSURE_[c] | R/W | <ISP_CORE[c]>+H'80008 | H'00000101 | 32 |
| ISP_CORE | Status of Defect Pixel Information[j] DMA [c](j=0-1)(c=0-0) | ISPCORE_DMA_DPCC[j]_STATUS_[c] | R/W | <ISP_CORE[c]>+H'86000+H'1000*[j]+H'A80 | H'00000000 | 32 |
| ISP_CORE | ISP Core Image[i] Frame Monitor Register [c](i=0-1)(c=0-0) | ISPCORE_DMA_IMAGE[i]_FRAME_MONITOR_[c] | R | <ISP_CORE[c]>+H'8490C+H'1000*[i] | H'00000000 | 32 |
| ISP_CORE | ISP Core Defect Pixel Information[j] DMA Flushing Out Register [c](j=0-1)(c=0-0) | ISPCORE_DMA_DPCC[j]_FLUSH_OUT_[c] | R/W | <ISP_CORE[c]>+H'86400+H'1000*[j] | H'00000000 | 32 |

## 54.3 Register Description

### 54.3.1 ISP Version Register [c](c=0-1)(ISPVCR_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ISPVCR | Undefined | R | ISP Version Control Register<br>H'0056344D |

Note 1: The value to be read varies depending on the product

### 54.3.2 ISP Input FIFO Control Register [c](c=0-1)(ISPFIFOCTRL_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | FIFO_PUSH_CSI | — | — |
| Initial value: | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | H'0 | R/W | Reserved |
| 15 to 12 | — | H'3 | R/W | Reserved |
| 11 to 8 | — | H'2 | R/W | Reserved |
| 7 to 3 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 2 | FIFO_PUSH_CSI | H'1 | R/W | FIFO ENABLE for CSI<br>0 : CSI FIFO disable<br>1 : CSI FIFO enable [default] |
| 1, 0 | — | All 0 | R | Reserved. (These bits are always read as 0.) |

### 54.3.3 ISP Input Select Control Register [c](c=0-1)(ISPINPUTSEL0_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEL_CSI0 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TRACE_SEL0 | | | | — | — | — | — | CSI_SEL | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | SEL_CSI0 | H'0 | R/W | Select CSI input for Pixel Reconstructor<br>0 : CSI input 0 [default]<br>1 : CSI input 1 |
| 30 to 16 | — | H'4012 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 15 to 12 | — | H'1 | R/W | Reserved |
| 11 to 8 | TRACE_SEL0 | H'0 | R/W | Selection signal of channels to output to trace port 0<br>0~11 : Channel n<br>Others : Forbidden |
| 7 to 4 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 3 to 0 | CSI_SEL | H'0 | R/W | Select of ISP core<br>0 : Select MIPI CSI-2 input<br>Others : Forbidden |

### 54.3.4 ISP Start Register [c](c=0-1)(ISPSTART_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | START_ISP | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | START_ISP | H'0 | W | Starting and stopping control of ISP<br>- H’FFFF : Start ISP<br>- H’0000 : Stop ISP [default]<br>This register is write only.<br>By writing H’FFFF ISP processing is started.<br>By writing H’0000 ISP processing is stopped.<br>To start ISP processing some additional procedure in addition to write this register is needed.<br><br>All of configuration should be completed before starting ISP processing. Any register writing operations are forbidden during ISP processing is started. |

### 54.3.5 ISP Interrupt Status Register [c](c=0-1)(ISPINT_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST31 | ST30 | ST29 | ST28 | ST27 | ST26 | ST25 | ST24 | ST23 | ST22 | ST21 | ST20 | ST19 | ST18 | ST17 | ST16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST15 | ST14 | ST13 | ST12 | ST11 | ST10 | ST9 | ST8 | ST7 | ST6 | ST5 | ST4 | ST3 | ST2 | ST1 | ST0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ST31 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 30 | ST30 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 29 | ST29 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 28 | ST28 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 27 | ST27 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 26 | ST26 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 25 | ST25 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 24 | ST24 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 23 | ST23 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 22 | ST22 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 21 | ST21 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 20 | ST20 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 19 | ST19 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 18 | ST18 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | ST17 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 16 | ST16 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 15 | ST15 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 14 | ST14 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 13 | ST13 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 12 | ST12 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 11 | ST11 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | ST10 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 9 | ST9 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 8 | ST8 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 7 | ST7 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 6 | ST6 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 5 | ST5 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 4 | ST4 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | ST3 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 2 | ST2 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 1 | ST1 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |
| 0 | ST0 | H'0 | R | Interrupt status.<br>- 0: Interrupt is not happened [default]<br>- 1: Interrupt is happened<br>Bit assignment is shown in Section 54.4.6.1<br>This reister is read only (not writable).<br>It can be cleared to 0 by writing 1 to the corresponding clear register CLR[n] (n=0~31) in ISPINT_CLEAR. |

### 54.3.6 ISP Error Request0 Status Register [c](c=0-1)(ISPERR0_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST31 | ST30 | ST29 | ST28 | ST27 | ST26 | ST25 | ST24 | ST23 | ST22 | ST21 | ST20 | ST19 | ST18 | ST17 | ST16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST15 | ST14 | ST13 | ST12 | ST11 | ST10 | ST9 | ST8 | ST7 | ST6 | ST5 | ST4 | ST3 | ST2 | ST1 | ST0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ST31 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 30 | ST30 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 29 | ST29 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 28 | ST28 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 27 | ST27 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 26 | ST26 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 25 | ST25 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 24 | ST24 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 23 | ST23 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 22 | ST22 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 21 | ST21 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 20 | ST20 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 19 | ST19 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 18 | ST18 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | ST17 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 16 | ST16 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 15 | ST15 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 14 | ST14 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 13 | ST13 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 12 | ST12 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 11 | ST11 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | ST10 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 9 | ST9 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 8 | ST8 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 7 | ST7 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 6 | ST6 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 5 | ST5 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 4 | ST4 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | ST3 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 2 | ST2 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 1 | ST1 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |
| 0 | ST0 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request0 Clear Register register |

### 54.3.7 ISP Error Request1 Status Register [c](c=0-1)(ISPERR1_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST31 | ST30 | ST29 | ST28 | ST27 | ST26 | ST25 | ST24 | ST23 | ST22 | ST21 | ST20 | ST19 | ST18 | ST17 | ST16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST15 | ST14 | ST13 | ST12 | ST11 | ST10 | ST9 | ST8 | ST7 | ST6 | ST5 | ST4 | ST3 | ST2 | ST1 | ST0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ST31 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 30 | ST30 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 29 | ST29 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 28 | ST28 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 27 | ST27 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 26 | ST26 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 25 | ST25 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 24 | ST24 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 23 | ST23 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 22 | ST22 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 21 | ST21 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 20 | ST20 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 19 | ST19 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 18 | ST18 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | ST17 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 16 | ST16 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 15 | ST15 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 14 | ST14 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 13 | ST13 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 12 | ST12 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 11 | ST11 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | ST10 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 9 | ST9 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 8 | ST8 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 7 | ST7 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 6 | ST6 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 5 | ST5 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 4 | ST4 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | ST3 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 2 | ST2 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 1 | ST1 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |
| 0 | ST0 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request1 Clear Register register |

### 54.3.8 ISP Error Request2 Status Register [c](c=0-1)(ISPERR2_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST31 | ST30 | ST29 | ST28 | ST27 | ST26 | ST25 | ST24 | ST23 | ST22 | ST21 | ST20 | ST19 | ST18 | ST17 | ST16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST15 | ST14 | ST13 | ST12 | ST11 | ST10 | ST9 | ST8 | ST7 | ST6 | ST5 | ST4 | ST3 | ST2 | ST1 | ST0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ST31 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 30 | ST30 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 29 | ST29 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 28 | ST28 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 27 | ST27 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 26 | ST26 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 25 | ST25 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 24 | ST24 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 23 | ST23 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 22 | ST22 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 21 | ST21 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 20 | ST20 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 19 | ST19 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 18 | ST18 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | ST17 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 16 | ST16 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 15 | ST15 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 14 | ST14 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 13 | ST13 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 12 | ST12 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 11 | ST11 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | ST10 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 9 | ST9 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 8 | ST8 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 7 | ST7 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 6 | ST6 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 5 | ST5 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 4 | ST4 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | ST3 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 2 | ST2 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 1 | ST1 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |
| 0 | ST0 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request2 Clear Register register |

### 54.3.9 ISP Interrupt Clear Register [c](c=0-1)(ISPINT_CLEAR_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR31 | CLR30 | CLR29 | CLR28 | CLR27 | CLR26 | CLR25 | CLR24 | CLR23 | CLR22 | CLR21 | CLR20 | CLR19 | CLR18 | CLR17 | CLR16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR15 | CLR14 | CLR13 | CLR12 | CLR11 | CLR10 | CLR9 | CLR8 | CLR7 | CLR6 | CLR5 | CLR4 | CLR3 | CLR2 | CLR1 | CLR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CLR31 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 30 | CLR30 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 29 | CLR29 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 28 | CLR28 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 27 | CLR27 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 26 | CLR26 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 25 | CLR25 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 24 | CLR24 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 23 | CLR23 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 22 | CLR22 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 21 | CLR21 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 20 | CLR20 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 19 | CLR19 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | CLR18 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 17 | CLR17 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 16 | CLR16 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 15 | CLR15 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 14 | CLR14 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 13 | CLR13 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 12 | CLR12 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 11 | CLR11 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 10 | CLR10 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 9 | CLR9 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 8 | CLR8 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 7 | CLR7 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 6 | CLR6 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | CLR5 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 4 | CLR4 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 3 | CLR3 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 2 | CLR2 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 1 | CLR1 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |
| 0 | CLR0 | H'0 | W | Clear register for interrupt status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Interrupt Status Register register is cleared to 0 |

### 54.3.10 ISP Error Request0 Clear Register [c](c=0-1)(ISPERR0_CLEAR_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR31 | CLR30 | CLR29 | CLR28 | CLR27 | CLR26 | CLR25 | CLR24 | CLR23 | CLR22 | CLR21 | CLR20 | CLR19 | CLR18 | CLR17 | CLR16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR15 | CLR14 | CLR13 | CLR12 | CLR11 | CLR10 | CLR9 | CLR8 | CLR7 | CLR6 | CLR5 | CLR4 | CLR3 | CLR2 | CLR1 | CLR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CLR31 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 30 | CLR30 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 29 | CLR29 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 28 | CLR28 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 27 | CLR27 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 26 | CLR26 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 25 | CLR25 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 24 | CLR24 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 23 | CLR23 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 22 | CLR22 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 21 | CLR21 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 20 | CLR20 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 19 | CLR19 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | CLR18 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 17 | CLR17 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 16 | CLR16 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 15 | CLR15 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 14 | CLR14 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 13 | CLR13 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 12 | CLR12 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 11 | CLR11 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 10 | CLR10 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 9 | CLR9 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 8 | CLR8 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 7 | CLR7 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 6 | CLR6 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | CLR5 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 4 | CLR4 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 3 | CLR3 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 2 | CLR2 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 1 | CLR1 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |
| 0 | CLR0 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request0 Status Register register is cleared to 0 |

### 54.3.11 ISP Error Request1 Clear Register [c](c=0-1)(ISPERR1_CLEAR_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR31 | CLR30 | CLR29 | CLR28 | CLR27 | CLR26 | CLR25 | CLR24 | CLR23 | CLR22 | CLR21 | CLR20 | CLR19 | CLR18 | CLR17 | CLR16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR15 | CLR14 | CLR13 | CLR12 | CLR11 | CLR10 | CLR9 | CLR8 | CLR7 | CLR6 | CLR5 | CLR4 | CLR3 | CLR2 | CLR1 | CLR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CLR31 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 30 | CLR30 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 29 | CLR29 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 28 | CLR28 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 27 | CLR27 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 26 | CLR26 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 25 | CLR25 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 24 | CLR24 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 23 | CLR23 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 22 | CLR22 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 21 | CLR21 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 20 | CLR20 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 19 | CLR19 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | CLR18 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 17 | CLR17 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 16 | CLR16 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 15 | CLR15 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 14 | CLR14 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 13 | CLR13 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 12 | CLR12 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 11 | CLR11 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 10 | CLR10 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 9 | CLR9 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 8 | CLR8 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 7 | CLR7 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 6 | CLR6 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | CLR5 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 4 | CLR4 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 3 | CLR3 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 2 | CLR2 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 1 | CLR1 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |
| 0 | CLR0 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request1 Status Register register is cleared to 0 |

### 54.3.12 ISP Error Request2 Clear Register [c](c=0-1)(ISPERR2_CLEAR_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR31 | CLR30 | CLR29 | CLR28 | CLR27 | CLR26 | CLR25 | CLR24 | CLR23 | CLR22 | CLR21 | CLR20 | CLR19 | CLR18 | CLR17 | CLR16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR15 | CLR14 | CLR13 | CLR12 | CLR11 | CLR10 | CLR9 | CLR8 | CLR7 | CLR6 | CLR5 | CLR4 | CLR3 | CLR2 | CLR1 | CLR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CLR31 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 30 | CLR30 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 29 | CLR29 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 28 | CLR28 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 27 | CLR27 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 26 | CLR26 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 25 | CLR25 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 24 | CLR24 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 23 | CLR23 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 22 | CLR22 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 21 | CLR21 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 20 | CLR20 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 19 | CLR19 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | CLR18 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 17 | CLR17 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 16 | CLR16 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 15 | CLR15 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 14 | CLR14 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 13 | CLR13 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 12 | CLR12 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 11 | CLR11 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 10 | CLR10 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 9 | CLR9 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 8 | CLR8 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 7 | CLR7 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 6 | CLR6 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | CLR5 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 4 | CLR4 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 3 | CLR3 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 2 | CLR2 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 1 | CLR1 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |
| 0 | CLR0 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request2 Status Register register is cleared to 0 |

### 54.3.13 ISP Interrupt Enable Register [c](c=0-1)(ISPINT_ENABLE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN31 | EN30 | EN29 | EN28 | EN27 | EN26 | EN25 | EN24 | EN23 | EN22 | EN21 | EN20 | EN19 | EN18 | EN17 | EN16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN15 | EN14 | EN13 | EN12 | EN11 | EN10 | EN9 | EN8 | EN7 | EN6 | EN5 | EN4 | EN3 | EN2 | EN1 | EN0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EN31 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 30 | EN30 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 29 | EN29 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 28 | EN28 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 27 | EN27 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 26 | EN26 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 25 | EN25 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 24 | EN24 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 23 | EN23 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 22 | EN22 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 21 | EN21 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 20 | EN20 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 19 | EN19 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | EN18 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 17 | EN17 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 16 | EN16 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 15 | EN15 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 14 | EN14 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 13 | EN13 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 12 | EN12 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 11 | EN11 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 10 | EN10 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 9 | EN9 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 8 | EN8 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 7 | EN7 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 6 | EN6 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | EN5 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 4 | EN4 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 3 | EN3 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 2 | EN2 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 1 | EN1 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |
| 0 | EN0 | H'0 | R/W | Enable register of interrupt request notification to INTC<br>- 0 : Interrupt request is not notified [default]<br>- 1 : Interrupt request is notified to INTC |

### 54.3.14 ISP Error Request0 Enable Register [c](c=0-1)(ISPERR0_ENABLE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN31 | EN30 | EN29 | EN28 | EN27 | EN26 | EN25 | EN24 | EN23 | EN22 | EN21 | EN20 | EN19 | EN18 | EN17 | EN16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN15 | EN14 | EN13 | EN12 | EN11 | EN10 | EN9 | EN8 | EN7 | EN6 | EN5 | EN4 | EN3 | EN2 | EN1 | EN0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EN31 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 30 | EN30 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 29 | EN29 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 28 | EN28 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 27 | EN27 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 26 | EN26 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 25 | EN25 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 24 | EN24 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 23 | EN23 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 22 | EN22 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 21 | EN21 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 20 | EN20 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 19 | EN19 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | EN18 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 17 | EN17 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 16 | EN16 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 15 | EN15 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 14 | EN14 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 13 | EN13 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 12 | EN12 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 11 | EN11 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 10 | EN10 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 9 | EN9 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 8 | EN8 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 7 | EN7 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 6 | EN6 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | EN5 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 4 | EN4 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 3 | EN3 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 2 | EN2 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 1 | EN1 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 0 | EN0 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

### 54.3.15 ISP Error Request1 Enable Register [c](c=0-1)(ISPERR1_ENABLE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN31 | EN30 | EN29 | EN28 | EN27 | EN26 | EN25 | EN24 | EN23 | EN22 | EN21 | EN20 | EN19 | EN18 | EN17 | EN16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN15 | EN14 | EN13 | EN12 | EN11 | EN10 | EN9 | EN8 | EN7 | EN6 | EN5 | EN4 | EN3 | EN2 | EN1 | EN0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EN31 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 30 | EN30 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 29 | EN29 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 28 | EN28 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 27 | EN27 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 26 | EN26 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 25 | EN25 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 24 | EN24 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 23 | EN23 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 22 | EN22 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 21 | EN21 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 20 | EN20 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 19 | EN19 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | EN18 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 17 | EN17 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 16 | EN16 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 15 | EN15 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 14 | EN14 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 13 | EN13 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 12 | EN12 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 11 | EN11 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 10 | EN10 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 9 | EN9 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 8 | EN8 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 7 | EN7 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 6 | EN6 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | EN5 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 4 | EN4 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 3 | EN3 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 2 | EN2 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 1 | EN1 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 0 | EN0 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

### 54.3.16 ISP Error Request2 Enable Register [c](c=0-1)(ISPERR2_ENABLE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN31 | EN30 | EN29 | EN28 | EN27 | EN26 | EN25 | EN24 | EN23 | EN22 | EN21 | EN20 | EN19 | EN18 | EN17 | EN16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN15 | EN14 | EN13 | EN12 | EN11 | EN10 | EN9 | EN8 | EN7 | EN6 | EN5 | EN4 | EN3 | EN2 | EN1 | EN0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EN31 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 30 | EN30 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 29 | EN29 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 28 | EN28 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 27 | EN27 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 26 | EN26 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 25 | EN25 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 24 | EN24 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 23 | EN23 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 22 | EN22 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 21 | EN21 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 20 | EN20 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 19 | EN19 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | EN18 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 17 | EN17 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 16 | EN16 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 15 | EN15 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 14 | EN14 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 13 | EN13 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 12 | EN12 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 11 | EN11 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 10 | EN10 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 9 | EN9 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 8 | EN8 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 7 | EN7 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 6 | EN6 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | EN5 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 4 | EN4 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 3 | EN3 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 2 | EN2 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 1 | EN1 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 0 | EN0 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

### 54.3.17 ISP Error Request3 Status Register [c](c=0-1)(ISPERR3_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST31 | ST30 | ST29 | ST28 | ST27 | ST26 | ST25 | ST24 | ST23 | ST22 | ST21 | ST20 | ST19 | ST18 | ST17 | ST16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST15 | ST14 | ST13 | ST12 | ST11 | ST10 | ST9 | ST8 | ST7 | ST6 | ST5 | ST4 | ST3 | ST2 | ST1 | ST0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ST31 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 30 | ST30 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 29 | ST29 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 28 | ST28 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 27 | ST27 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 26 | ST26 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 25 | ST25 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 24 | ST24 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 23 | ST23 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 22 | ST22 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 21 | ST21 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 20 | ST20 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 19 | ST19 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 18 | ST18 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | ST17 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 16 | ST16 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 15 | ST15 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 14 | ST14 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 13 | ST13 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 12 | ST12 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 11 | ST11 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | ST10 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 9 | ST9 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 8 | ST8 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 7 | ST7 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 6 | ST6 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 5 | ST5 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 4 | ST4 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | ST3 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 2 | ST2 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 1 | ST1 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |
| 0 | ST0 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request3 Clear Register register |

### 54.3.18 ISP Error Request4 Status Register [c](c=0-1)(ISPERR4_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST31 | ST30 | ST29 | ST28 | ST27 | ST26 | ST25 | ST24 | ST23 | ST22 | ST21 | ST20 | ST19 | ST18 | ST17 | ST16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ST15 | ST14 | ST13 | ST12 | ST11 | ST10 | ST9 | ST8 | ST7 | ST6 | ST5 | ST4 | ST3 | ST2 | ST1 | ST0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ST31 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 30 | ST30 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 29 | ST29 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 28 | ST28 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 27 | ST27 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 26 | ST26 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 25 | ST25 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 24 | ST24 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 23 | ST23 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 22 | ST22 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 21 | ST21 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 20 | ST20 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 19 | ST19 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 18 | ST18 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | ST17 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 16 | ST16 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 15 | ST15 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 14 | ST14 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 13 | ST13 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 12 | ST12 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 11 | ST11 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | ST10 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 9 | ST9 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 8 | ST8 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 7 | ST7 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 6 | ST6 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 5 | ST5 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 4 | ST4 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | ST3 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 2 | ST2 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 1 | ST1 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |
| 0 | ST0 | H'0 | R | Error status.<br>- 0: Error is not happened [default]<br>- 1: Error is happened<br>Bit assignment is shown in Section 54.4.7.3<br>This reister is read only (not writable).<br>ST[n] (n=0~31) can be cleared to 0 by writing 1 to the corresponding bit in CLR[n] (n=0~31) in ISP Error Request4 Clear Register register |

### 54.3.19 ISP Error Request3 Clear Register [c](c=0-1)(ISPERR3_CLEAR_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR31 | CLR30 | CLR29 | CLR28 | CLR27 | CLR26 | CLR25 | CLR24 | CLR23 | CLR22 | CLR21 | CLR20 | CLR19 | CLR18 | CLR17 | CLR16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR15 | CLR14 | CLR13 | CLR12 | CLR11 | CLR10 | CLR9 | CLR8 | CLR7 | CLR6 | CLR5 | CLR4 | CLR3 | CLR2 | CLR1 | CLR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CLR31 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 30 | CLR30 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 29 | CLR29 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 28 | CLR28 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 27 | CLR27 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 26 | CLR26 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 25 | CLR25 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 24 | CLR24 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 23 | CLR23 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 22 | CLR22 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 21 | CLR21 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 20 | CLR20 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 19 | CLR19 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | CLR18 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 17 | CLR17 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 16 | CLR16 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 15 | CLR15 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 14 | CLR14 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 13 | CLR13 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 12 | CLR12 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 11 | CLR11 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 10 | CLR10 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 9 | CLR9 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 8 | CLR8 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 7 | CLR7 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 6 | CLR6 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | CLR5 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 4 | CLR4 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 3 | CLR3 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 2 | CLR2 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 1 | CLR1 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |
| 0 | CLR0 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request3 Status Register register is cleared to 0 |

### 54.3.20 ISP Error Request4 Clear Register [c](c=0-1)(ISPERR4_CLEAR_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR31 | CLR30 | CLR29 | CLR28 | CLR27 | CLR26 | CLR25 | CLR24 | CLR23 | CLR22 | CLR21 | CLR20 | CLR19 | CLR18 | CLR17 | CLR16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLR15 | CLR14 | CLR13 | CLR12 | CLR11 | CLR10 | CLR9 | CLR8 | CLR7 | CLR6 | CLR5 | CLR4 | CLR3 | CLR2 | CLR1 | CLR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CLR31 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 30 | CLR30 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 29 | CLR29 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 28 | CLR28 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 27 | CLR27 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 26 | CLR26 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 25 | CLR25 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 24 | CLR24 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 23 | CLR23 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 22 | CLR22 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 21 | CLR21 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 20 | CLR20 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 19 | CLR19 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | CLR18 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 17 | CLR17 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 16 | CLR16 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 15 | CLR15 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 14 | CLR14 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 13 | CLR13 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 12 | CLR12 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 11 | CLR11 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 10 | CLR10 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 9 | CLR9 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 8 | CLR8 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 7 | CLR7 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 6 | CLR6 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | CLR5 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 4 | CLR4 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 3 | CLR3 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 2 | CLR2 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 1 | CLR1 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |
| 0 | CLR0 | H'0 | W | Clear register for error status ST[n] (n=0~31).<br>By writing 1 to CLK[n], ST[n] in ISP Error Request4 Status Register register is cleared to 0 |

### 54.3.21 ISP Error Request3 Enable Register [c](c=0-1)(ISPERR3_ENABLE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN31 | EN30 | EN29 | EN28 | EN27 | EN26 | EN25 | EN24 | EN23 | EN22 | EN21 | EN20 | EN19 | EN18 | EN17 | EN16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN15 | EN14 | EN13 | EN12 | EN11 | EN10 | EN9 | EN8 | EN7 | EN6 | EN5 | EN4 | EN3 | EN2 | EN1 | EN0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EN31 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 30 | EN30 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 29 | EN29 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 28 | EN28 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 27 | EN27 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 26 | EN26 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 25 | EN25 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 24 | EN24 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 23 | EN23 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 22 | EN22 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 21 | EN21 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 20 | EN20 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 19 | EN19 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | EN18 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 17 | EN17 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 16 | EN16 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 15 | EN15 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 14 | EN14 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 13 | EN13 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 12 | EN12 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 11 | EN11 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 10 | EN10 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 9 | EN9 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 8 | EN8 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 7 | EN7 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 6 | EN6 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | EN5 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 4 | EN4 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 3 | EN3 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 2 | EN2 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 1 | EN1 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 0 | EN0 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

### 54.3.22 ISP Error Request4 Enable Register [c](c=0-1)(ISPERR4_ENABLE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN31 | EN30 | EN29 | EN28 | EN27 | EN26 | EN25 | EN24 | EN23 | EN22 | EN21 | EN20 | EN19 | EN18 | EN17 | EN16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EN15 | EN14 | EN13 | EN12 | EN11 | EN10 | EN9 | EN8 | EN7 | EN6 | EN5 | EN4 | EN3 | EN2 | EN1 | EN0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EN31 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 30 | EN30 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 29 | EN29 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 28 | EN28 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 27 | EN27 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 26 | EN26 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 25 | EN25 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 24 | EN24 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 23 | EN23 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 22 | EN22 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 21 | EN21 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 20 | EN20 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 19 | EN19 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | EN18 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 17 | EN17 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 16 | EN16 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 15 | EN15 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 14 | EN14 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 13 | EN13 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 12 | EN12 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 11 | EN11 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 10 | EN10 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 9 | EN9 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 8 | EN8 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 7 | EN7 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 6 | EN6 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | EN5 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 4 | EN4 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 3 | EN3 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 2 | EN2 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 1 | EN1 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |
| 0 | EN0 | H'0 | R/W | Enable register of error request notification to ECM<br>- 0 : Error request is not notified [default]<br>- 1 : Error request is notified to ECM |

### 54.3.23 ISP Padding Control for CSI FIFO [c](c=0-1)(ISP_PADDING_CTRL_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PADDING_NUM | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | H'70080 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 11 to 0 | PADDING_NUM | H'20 | R/W | Number of cycles for padding to push into CSI FIFO at the last cycle of each long packet<br>Write value should be greater than 32. Writing of values less than 32 is prohibited.<br><br>Notes: This register is write protected. Before writhing this register, write H'FB69 to ISPWP_CTRL.UNLOCK_CODE_L Register. |

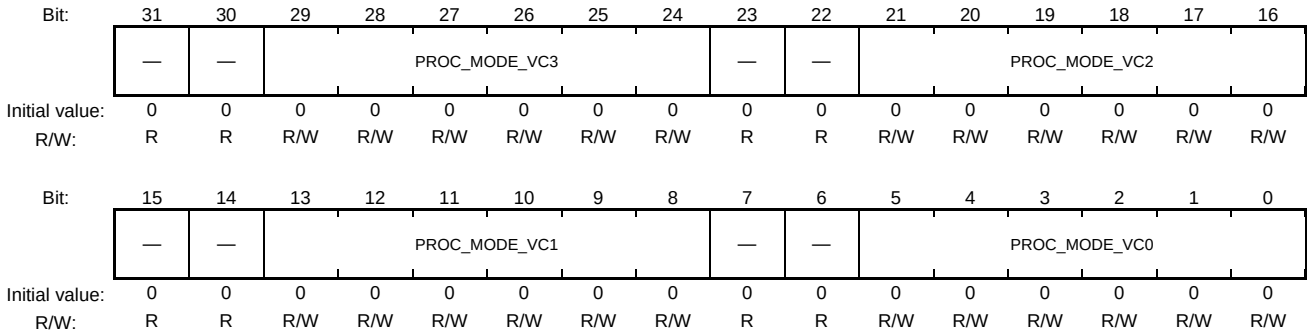
### 54.3.24 Pixel Reconstructor Custom Format Register[n] [c](n=0-7)(c=0-1)(ISPPROC_CUSTOM_FORMAT[n]_CTRL_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | LUT_LENGTH_MINUS1 | | | | | | — | — | LUT_START_ADDRESS | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | POPNUM_MINUS1 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

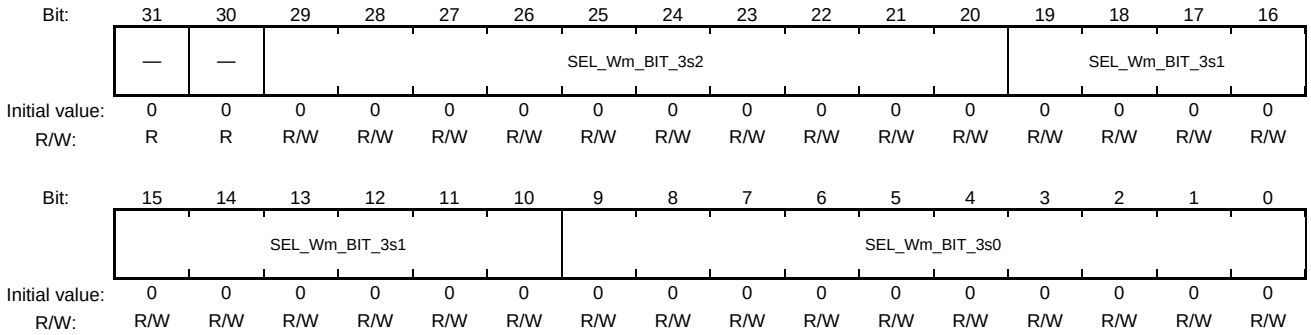
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 29 to 24 | LUT_LENGTH_MINUS1 | H'0 | R/W | Configuration register for LUT based pixel reconstructor. See the 54.4.2.3 for detail |
| 23, 22 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 21 to 16 | LUT_START_ADDRESS | H'0 | R/W | Configuration register for LUT based pixel reconstructor. See the 54.4.2.3 for detail |
| 15 to 4 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 3 to 0 | POPNUM_MINUS1 | H'0 | R/W | Configuration register for LUT based pixel reconstructor. See the 54.4.2.3 for detail |

### 54.3.25 Pixel Reconstructor Processing Mode Register[k] [c](k=0-63)(c=0-1)(ISPPROCMODE_DT[k]_[c])



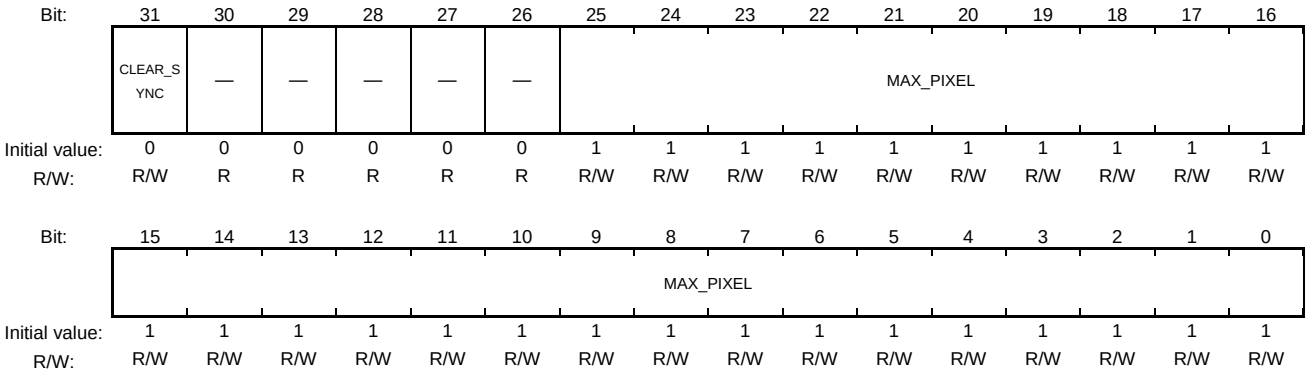

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 29 to 24 | PROC_MODE_VC3 | H'0 | R/W | Select the processing method of CSI packet with “DT_code = n” sent from VC3, VC7, VC11 and VC15<br>Meaning of each setting value is shown in the Table 54.1 |
| 23, 22 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 21 to 16 | PROC_MODE_VC2 | H'0 | R/W | Select the processing method of CSI packet with “DT_code = n” sent from VC2, VC6, VC10 and VC14<br>Meaning of each setting value is shown in the Table 54.1 |
| 15, 14 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 13 to 8 | PROC_MODE_VC1 | H'0 | R/W | Select the processing method of CSI packet with “DT_code = n” sent from VC1, VC5, VC9 and VC13<br>Meaning of each setting value is shown in the Table 54.1 |
| 7, 6 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 5 to 0 | PROC_MODE_VC0 | H'0 | R/W | Select the processing method of CSI packet with “DT_code = n” sent from VC0,VC4,VC8 and VC12.<br>Meaning of each setting value is shown in the Table 54.1 |

### 54.3.26 Pixel Reconstructor Custom Format LUT Register[m]_[s] [c](m=0-63)(s=0-7)(c=0-1)(ISPPROC_CUSTOM_FORMAT_LUT0_W[m]_B[s]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 29 to 20 | SEL_Wm_BIT_3s2 | H'0 | R/W | Configuration register for LUT based pixel reconstructor. See the 54.4.2.3 for detail |
| 19 to 10 | SEL_Wm_BIT_3s1 | H'0 | R/W | Configuration register for LUT based pixel reconstructor. See the 54.4.2.3 for detail |
| 9 to 0 | SEL_Wm_BIT_3s0 | H'0 | R/W | Configuration register for LUT based pixel reconstructor. See the 54.4.2.3 for detail |

### 54.3.27 Pixel Count Checker MAX Pixel Register CH[d] [c](d=0-11)(c=0-1)(ISP_PIXEL_COUNT_MAX_CH[d]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CLEAR_SYNC | H'0 | R/W | Selection of period to count pixels<br>0 : Number of pixels per line [default]<br>1 : Number of pixels per frame |
| 30 to 26 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 25 to 0 | MAX_PIXEL | H'3FFFFFF | R/W | Maximum number of pixels to asset the error status.<br>When number of pixels in the current line or frame is bigger than the value of MAX_PIXEL register, then error status is asserted |

### 54.3.28 Pixel Count Checker MIN Pixel Register CH[d] [c](d=0-11)(c=0-1)(ISP_PIXEL_COUNT_MIN_CH[d]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MIN_PIXEL | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MIN_PIXEL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 25 to 0 | MIN_PIXEL | H'0 | R/W | Minimum number of pixels to assert the error status.<br>When number of pixels in the current line or frame is less than the value of MIN_PIXEL register, then error status is asserted |

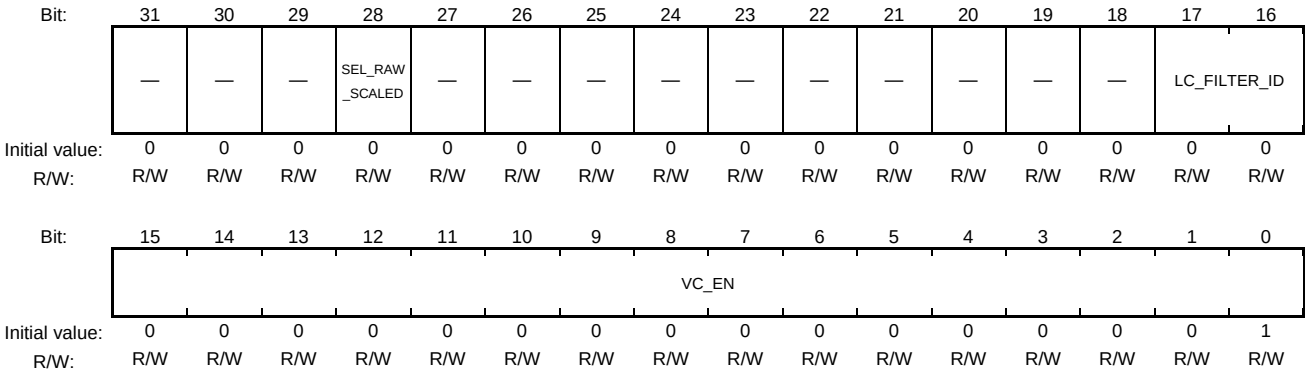
### 54.3.29 Pixel Count Checker Monitor Pixel Register CH[d] [c](d=0-11)(c=0-1)(ISP_PIXEL_COUNT_MONITOR_CH[d]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PIXEL_COUNT | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIXEL_COUNT | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 25 to 0 | PIXEL_COUNT | H'0 | R/W | Number of counted pixels |

### 54.3.30 Channel Selector VC and ID Filter Register CH[d] [c](d=0-11)(c=0-1)(ISPCS_FILER_ID_CH[d]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 28 | SEL_RAW_SCALED | H'0 | R/W | A bit to specify the input pixel stream for filter chain to select RAW scaled image<br>In order to use RAW scaler this bit should be set to 1.<br>0: Not select the RAW scaled image [default]<br>1: Selects RAW scaled image. |
| 27 to 18 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 17, 16 | LC_FILTER_ID | H'0 | R/W | Number of ID to pass at line count filter |
| 15 to 0 | VC_EN | H'1 | R/W | Virtual channel (VC) number to pass is specified at VC_Filter.<br>When VC_EN[n] == 1, VCn is passed.<br>VC_EN shall be either one hot value or two hot value.<br>Up to 2 VCs can be passed per one VC_Filter.<br>Neither one hot value nor two hot value are forbidden |

### 54.3.31 Channel Selector Line Count Modulo Register VC0-7 CH[d] [c](d=0-11)(c=0-1)(ISPCS_LC_MODULO_CH[d]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | MODULO_MINUS1_VC7 | | | — | MODULO_MINUS1_VC3 | | | — | MODULO_MINUS1_VC6 | | | — | MODULO_MINUS1_VC2 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | MODULO_MINUS1_VC5 | | | — | MODULO_MINUS1_VC1 | | | — | MODULO_MINUS1_VC4 | | | — | MODULO_MINUS1_VC0 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 30 to 28 | MODULO_MINUS1_VC7 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC7) for calculating the line count ID |
| 27 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 26 to 24 | MODULO_MINUS1_VC3 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC3) for calculating the line count ID |
| 23 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 22 to 20 | MODULO_MINUS1_VC6 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC6) for calculating the line count ID |
| 19 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 18 to 16 | MODULO_MINUS1_VC2 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC2) for calculating the line count ID |
| 15 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 14 to 12 | MODULO_MINUS1_VC5 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC5) for calculating the line count ID |
| 11 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 10 to 8 | MODULO_MINUS1_VC1 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC1) for calculating the line count ID |
| 7 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 6 to 4 | MODULO_MINUS1_VC4 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC4) for calculating the line count ID |
| 3 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 2 to 0 | MODULO_MINUS1_VC0 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC0) for calculating the line count ID |

### 54.3.32 Channel Selector Line Count Modulo Register VC8-15 CH[d] [c](d=0-11)(c=0-1)(ISPCS_LC_MODULO1_CH[d]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | MODULO_MINUS1_VC15 | | | — | MODULO_MINUS1_VC11 | | | — | MODULO_MINUS1_VC14 | | | — | MODULO_MINUS1_VC10 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | MODULO_MINUS1_VC13 | | | — | MODULO_MINUS1_VC9 | | | — | MODULO_MINUS1_VC12 | | | — | MODULO_MINUS1_VC8 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 30 to 28 | MODULO_MINUS1_VC15 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC15) for calculating the line count ID |
| 27 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 26 to 24 | MODULO_MINUS1_VC11 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC11) for calculating the line count ID |
| 23 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 22 to 20 | MODULO_MINUS1_VC14 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC14) for calculating the line count ID |
| 19 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 18 to 16 | MODULO_MINUS1_VC10 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC10) for calculating the line count ID |
| 15 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 14 to 12 | MODULO_MINUS1_VC13 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC13) for calculating the line count ID |
| 11 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 10 to 8 | MODULO_MINUS1_VC9 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC9) for calculating the line count ID |
| 7 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 6 to 4 | MODULO_MINUS1_VC12 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC12) for calculating the line count ID |
| 3 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 2 to 0 | MODULO_MINUS1_VC8 | H'0 | R/W | Divisor minus 1 for modulo calculation on the line number of the input frame (VC8) for calculating the line count ID |

### 54.3.33 Channel Selector DT Code03 Register CH[d] [c](d=0-11)(c=0-1)(ISPCS_DT_CODE03_CH[d]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENABLE3 | HOLD3 | DT_CODE3 | | | | | | ENABLE2 | HOLD2 | DT_CODE2 | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENABLE1 | HOLD1 | DT_CODE1 | | | | | | ENABLE0 | HOLD0 | DT_CODE0 | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ENABLE3 | H'0 | R/W | Enable bit for passing pixel to LC filter . |
| 30 | HOLD3 | H'0 | R/W | Enable bit for increment counter at Line Count ID.<br>0 : Line count is incremented [default]<br>1 : Line count is not incremented |
| 29 to 24 | DT_CODE3 | H'0 | R/W | Number of DT_CODE at MD Filter<br>When DT_CODE match setting value by MD Filter and ENABLE is 1, then data be passed |
| 23 | ENABLE2 | H'0 | R/W | Enable bit for passing pixel to LC filter . |
| 22 | HOLD2 | H'0 | R/W | Enable bit for increment counter at Line Count ID.<br>0 : Line count is incremented [default]<br>1 : Line count is not incremented |
| 21 to 16 | DT_CODE2 | H'0 | R/W | Number of DT_CODE at MD Filter<br>When DT_CODE match setting value by MD Filter and ENABLE is 1, then data be passed |
| 15 | ENABLE1 | H'0 | R/W | Enable bit for passing pixel to LC filter . |
| 14 | HOLD1 | H'0 | R/W | Enable bit for increment counter at Line Count ID.<br>0 : Line count is incremented [default]<br>1 : Line count is not incremented |
| 13 to 8 | DT_CODE1 | H'0 | R/W | Number of DT_CODE at MD Filter<br>When DT_CODE match setting value by MD Filter and ENABLE is 1, then data be passed |
| 7 | ENABLE0 | H'0 | R/W | Enable bit for passing pixel to LC filter . |
| 6 | HOLD0 | H'0 | R/W | Enable bit for increment counter at Line Count ID.<br>0 : Line count is incremented [default]<br>1 : Line count is not incremented |
| 5 to 0 | DT_CODE0 | H'0 | R/W | Number of DT_CODE at MD Filter<br>When DT_CODE match setting value by MD Filter and ENABLE is 1, then data be passed |

### 54.3.34 Channel Selector DT Code47 Register CH[d] [c](d=0-11)(c=0-1)(ISPCS_DT_CODE47_CH[d]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENABLE7 | HOLD7 | DT_CODE7 | | | | | | ENABLE6 | HOLD6 | DT_CODE6 | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ENABLE5 | HOLD5 | DT_CODE5 | | | | | | ENABLE4 | HOLD4 | DT_CODE4 | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ENABLE7 | H'0 | R/W | Enable bit for passing pixel to LC filter . |
| 30 | HOLD7 | H'0 | R/W | Enable bit for increment counter at Line Count ID.<br>0 : Line count is incremented [default]<br>1 : Line count is not incremented |
| 29 to 24 | DT_CODE7 | H'0 | R/W | Number of DT_CODE at MD Filter<br>When DT_CODE match setting value by MD Filter and ENABLE is 1, then data be passed |
| 23 | ENABLE6 | H'0 | R/W | Enable bit for passing pixel to LC filter . |
| 22 | HOLD6 | H'0 | R/W | Enable bit for increment counter at Line Count ID.<br>0 : Line count is incremented [default]<br>1 : Line count is not incremented |
| 21 to 16 | DT_CODE6 | H'0 | R/W | Number of DT_CODE at MD Filter<br>When DT_CODE match setting value by MD Filter and ENABLE is 1, then data be passed |
| 15 | ENABLE5 | H'0 | R/W | Enable bit for passing pixel to LC filter . |
| 14 | HOLD5 | H'0 | R/W | Enable bit for increment counter at Line Count ID.<br>0 : Line count is incremented [default]<br>1 : Line count is not incremented |
| 13 to 8 | DT_CODE5 | H'0 | R/W | Number of DT_CODE at MD Filter<br>When DT_CODE match setting value by MD Filter and ENABLE is 1, then data be passed |
| 7 | ENABLE4 | H'0 | R/W | Enable bit for passing pixel to LC filter . |
| 6 | HOLD4 | H'0 | R/W | Enable bit for increment counter at Line Count ID.<br>0 : Line count is incremented [default]<br>1 : Line count is not incremented |
| 5 to 0 | DT_CODE4 | H'0 | R/W | Number of DT_CODE at MD Filter<br>When DT_CODE match setting value by MD Filter and ENABLE is 1, then data be passed |

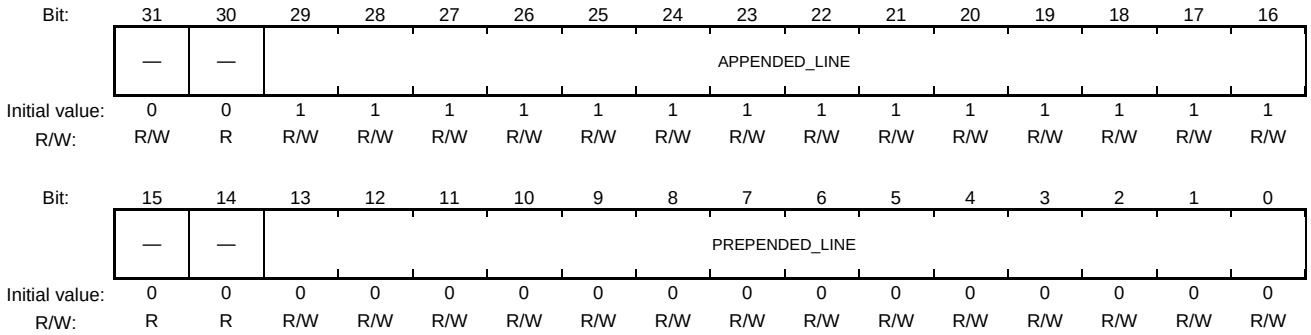
### 54.3.35 Channel Selector Horizontal Clipping Register DT[h] CH[d] [c](d=0-11)(h=0-3)(c=0-1)(ISPCS_H_CLIP_DT_CODE[h]_CH[d]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | APPENDED_PIXEL | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | PREPENDED_PIXEL | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 30 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 29 to 16 | APPENDED_PIXEL | H'3FFF | R/W | This bit specifies the number of removing pixels at the right edge of one line. |
| 15, 14 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 13 to 0 | PREPENDED_PIXEL | H'0 | R/W | This register specifies the number of removing pixels at the left edge of one line. |

### 54.3.36 Channel Selector Vertical Clipping Register DT[v] CH[d] [c](d=0-11)(v=0-3)(c=0-1)(ISPCS_V_CLIP_DT_CODE[v]_CH[d]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 30 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 29 to 16 | APPENDED_LINE | H'3FFF | R/W | This bit specifies the number of removed lines at the bottom edge of one frame |
| 15, 14 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 13 to 0 | PREPENDED_LINE | H'0 | R/W | This bit specifies the number of removed lines at the top edge |

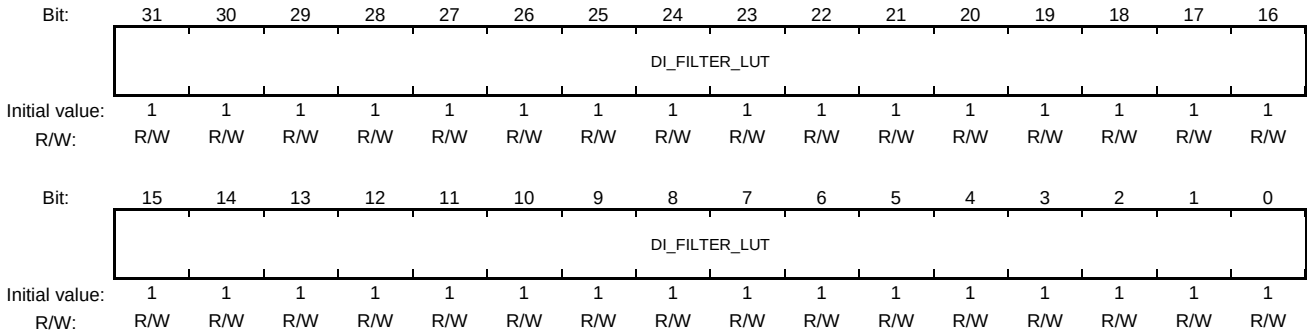
### 54.3.37 Channel Selector De-Interleaving Filter Control CH[d] [c](d=0-11)(c=0-1)(ISPCS_DI_FILTER_CTRL_CH[d]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CPLX | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LUT_LENGTH_MINUS1 | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CPLX | H'0 | R/W | Enabling control of divisor of pixel counter in one line<br>- 0 : Pixel counter divisor is disabled [default]<br>- 1 : Pixel counter is divided by 2 |
| 30 to 9 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 8 to 0 | LUT_LENGTH_MINUS1 | H'1FF | R/W | Value of minus 1 of the number of available bit in de-interleaving filter LUT.<br>For example, if LUT_LENGTH_MINUS1=21, then LUT size is 22 (0 to 21 is available) |

### 54.3.38 Channel Selector De-Interleaving Filter LUT[p] CH[d] [c](d=0-11)(p=0-15)(c=0-1)(ISPCS_DI_FILTER_LUT[p]_CH[d]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DI_FILTER_LUT | H'FFFFFFFF | R/W | Specify pass/drop control for each pixel in one packet. Each bit correspond to one pixel<br>Meaning the k-th bit of DI_FILTER_LUT[p]_CH[n] is as below<br>- 0 : Drop<br>- 1 : Pass [default] |

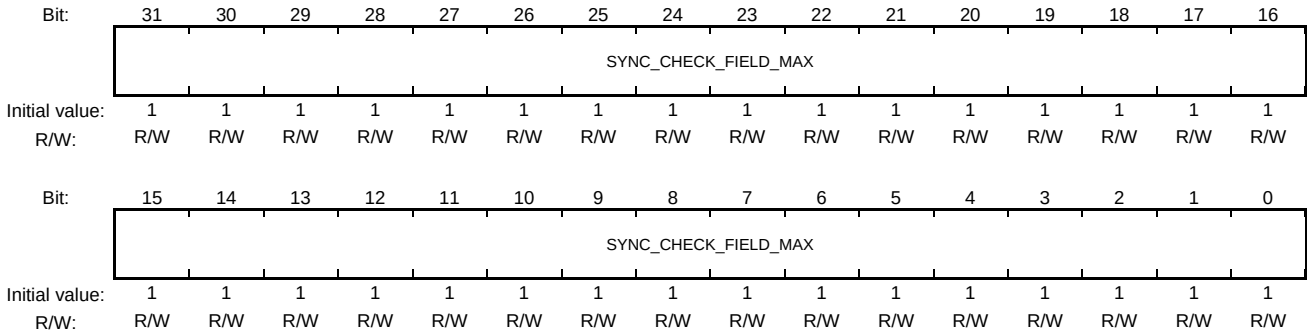
### 54.3.39 Write Protect Control Register [c](c=0-1)(ISPWP_CTRL_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UNLOCK_CODE_L | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | UNLOCK_CODE_L | H'0 | R/W | Write protection control of ISP_PADDING_CTRL register<br>- H'FB69 : Write protection is disabled. Target register is writable.<br>- H'0000 : Write protection is enabled. Target register cannot be written [default]<br>- Otherwise : Forbidden |

### 54.3.40 ISP Sync Checker Control FIELD MAX VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_FIELD_MAX_VC[b]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_FIELD_MAX | H'FFFFFFFF | R/W | Permitted maximum cycles which keeps the same value of the incoming FIELD signal.<br>If the cycle which keeps the high value or low value exceeds the value specified by this register, then SYNC_ERROR is asserted. |

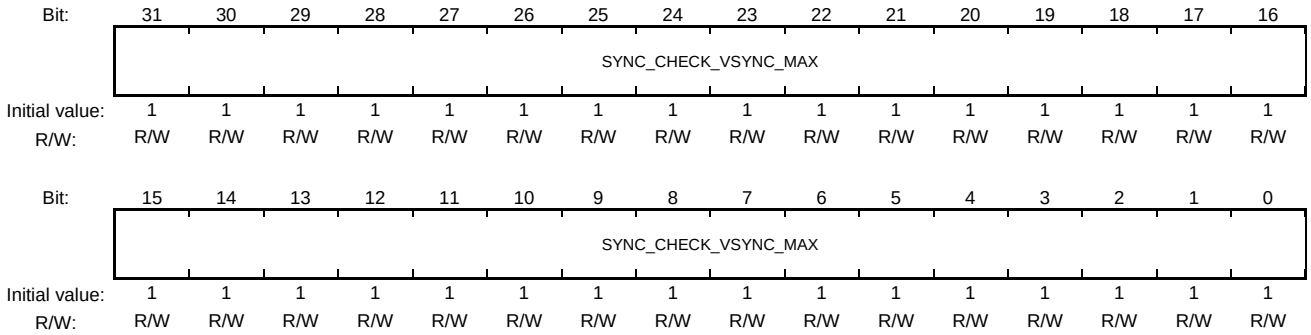
### 54.3.41 ISP Sync Checker Control FIELD MIN VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_FIELD_MIN_VC[b]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_FIELD_MIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_FIELD_MIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

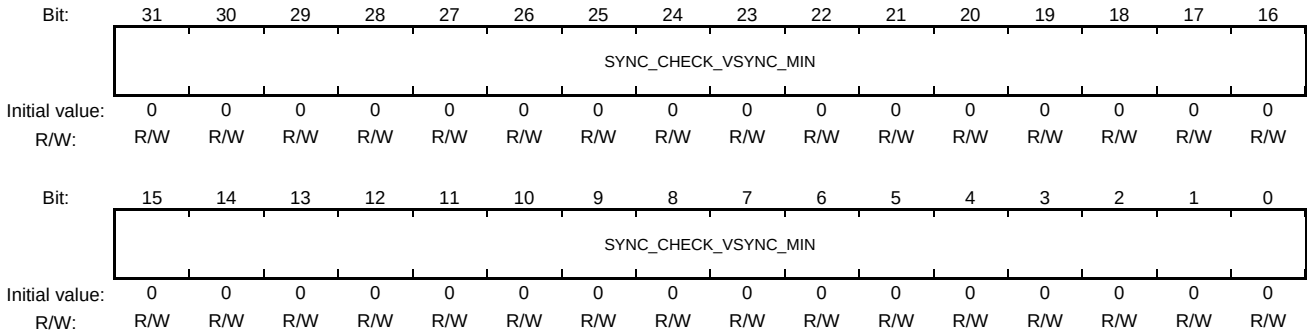
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_FIELD_MIN | H'0 | R/W | Permitted minimum cycles which keeps the same value of the incoming FIELD signal.<br>If the cycle which keeps the high value or low value is less than the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.42 ISP Sync Checker Control VSYNC MAX VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_VSYNC_MAX_VC[b]_[c])



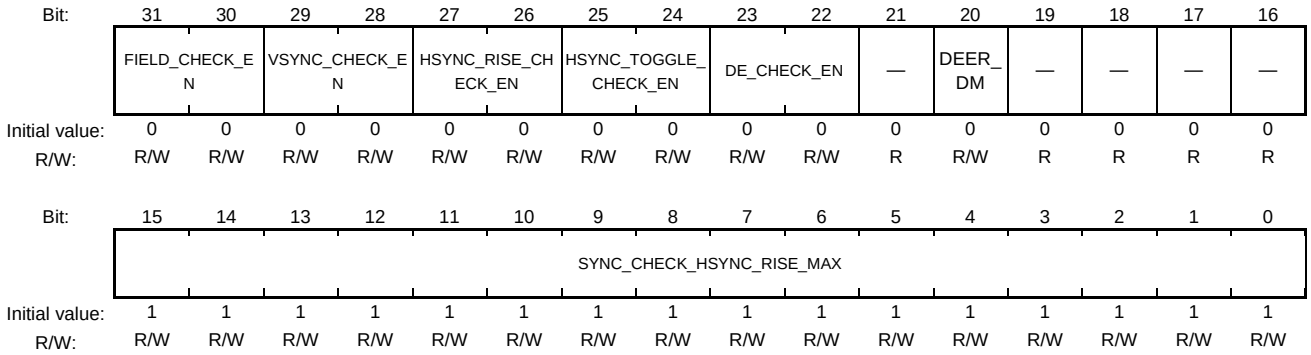

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_VSYNC_MAX | H'FFFFFFFF | R/W | Permitted maximum cycles which keeps the same value of the incoming VSYNC signal.<br>If the cycle which keeps the high value or low value exceeds the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.43 ISP Sync Checker Control VSYNC MIN VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_VSYNC_MIN_VC[b]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_VSYNC_MIN | H'0 | R/W | Permitted minimum cycles which keeps the same value of the incoming VSYNC signal.<br>If the cycle which keeps the high value or low value is less than the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.44 ISP Sync Checker Control HSYNC Rise MAX VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_HSYNC_RISE_MAX_VC[b]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | FIELD_CHECK_EN | H'0 | R/W | FIELD_CHECK_EN[1] : FIELD MAX Check Enable<br>FIELD_CHECK_EN[0] : FIELD MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 29, 28 | VSYNC_CHECK_EN | H'0 | R/W | VSYNC_CHECK_EN[1] : VSYNC MAX Check Enable<br>VSYNC_CHECK_EN[0] : VSYNC MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 27, 26 | HSYNC_RISE_CHECK_EN | H'0 | R/W | HSYNC_RISE_CHECK_EN[1] : HSYNC Rise MAX Check Enable<br>HSYNC_RISE_CHECK_EN[0] : HSYNC Rise MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 25, 24 | HSYNC_TOGGLE_CHECK_EN | H'0 | R/W | HSYNC_TOGGLE_CHECK_EN[1] : HSYNC Toggle MAX Check Enable<br>HSYNC_TOGGLE_CHECK_EN[0] : HSYNC Toggle MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 23, 22 | DE_CHECK_EN | H'0 | R/W | DE_CHECK_EN[1] : DE MAX Check Enable<br>DE_CHECK_EN[0] : DE MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 21 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 20 | DEER_DM | H'0 | R/W | 0 : Error detection is disabled when counted value is 0 [default]<br>1 : Error detection is enabled when counted value is 0<br><br>Note: This setting is valid when DE_CHECK_EN[0] is 1. |
| 19 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | SYNC_CHECK_HSYNC_RISE_MAX | H'FFFF | R/W | Permitted maximum number of rising edge of incoming HSYNC signal in one frame.<br>If the number of rising edge of HSYNC exceeds the value specified by this register, then SYNC_ERROR is asserted. |

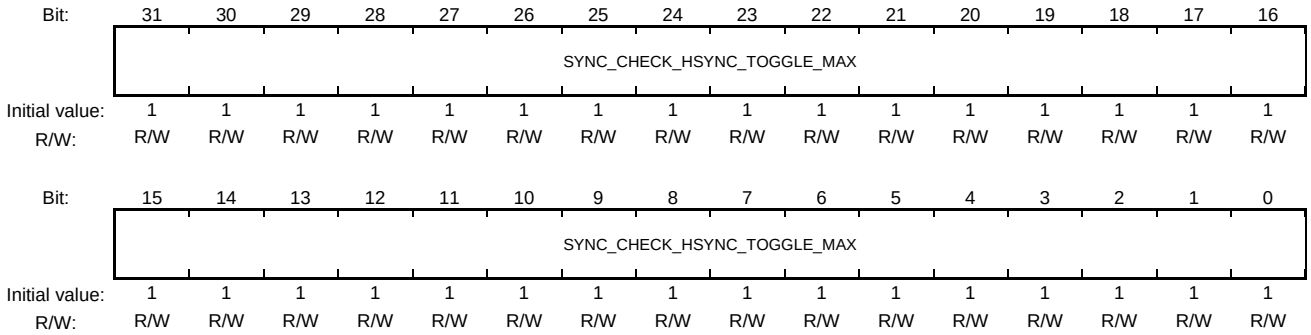
### 54.3.45 ISP Sync Checker Control HSYNC Rise MIN VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_HSYNC_RISE_MIN_VC[b]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_HSYNC_RISE_MIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

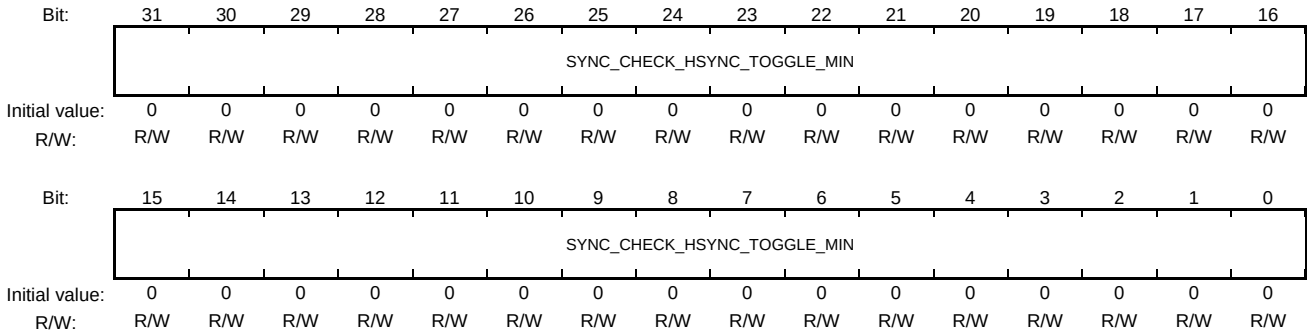
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | SYNC_CHECK_HSYNC_RISE_MIN | H'0 | R/W | Permitted minimum number of rising edge of incoming HSYNC signal in one frame.<br>If the number of rising edge of HSYNC is less than the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.46 ISP Sync Checker Control HSYNC Toggle MAX VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_HSYNC_TOGGLE_MAX_VC[b]_[c])



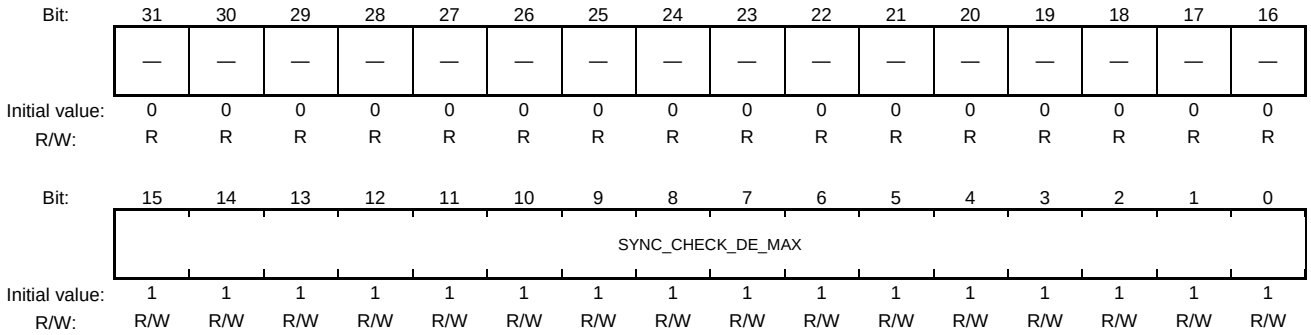

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_HSYNC_TOGGLE_MAX | H'FFFFFFFF | R/W | Permitted maximum cycles which keeps the same value of the incoming HSYNC signal.<br>If the cycle which keeps the high value or low value exceeds the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.47 ISP Sync Checker Control HSYNC Toggle MIN VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_HSYNC_TOGGLE_MIN_VC[b]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_HSYNC_TOGGLE_MIN | H'0 | R/W | Permitted minimum cycles which keeps the same value of the incoming HSYNC signal.<br>If the cycle which keeps the high value or low value is less than the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.48 ISP Sync Checker Control DE MAX VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_DE_MAX_VC[b]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | SYNC_CHECK_DE_MAX | H'FFFF | R/W | Permitted maximum cycles which keeps the same value of the incoming DE signal.<br>If the cycle which keeps the high value or low value exceeds the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.49 ISP Sync Checker Control DE MIN VC[b] [c](b=0-7)(c=0-1)(ISP_SYNC_CHECK_DE_MIN_VC[b]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_DE_MIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | SYNC_CHECK_DE_MIN | H'0 | R/W | Permitted minimum cycles which keeps the same value of the incoming DE signal.<br>If the cycle which keeps the high value or low value is less than the value specified by this register, then SYNC_ERROR is asserted. |

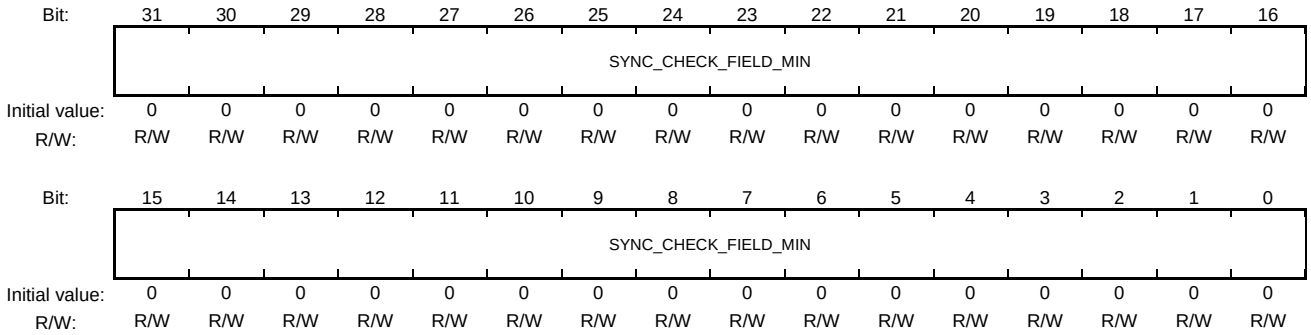
### 54.3.50 ISP Sync Checker Control FIELD MAX VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_FIELD_MAX_VC[g]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_FIELD_MAX | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_FIELD_MAX | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

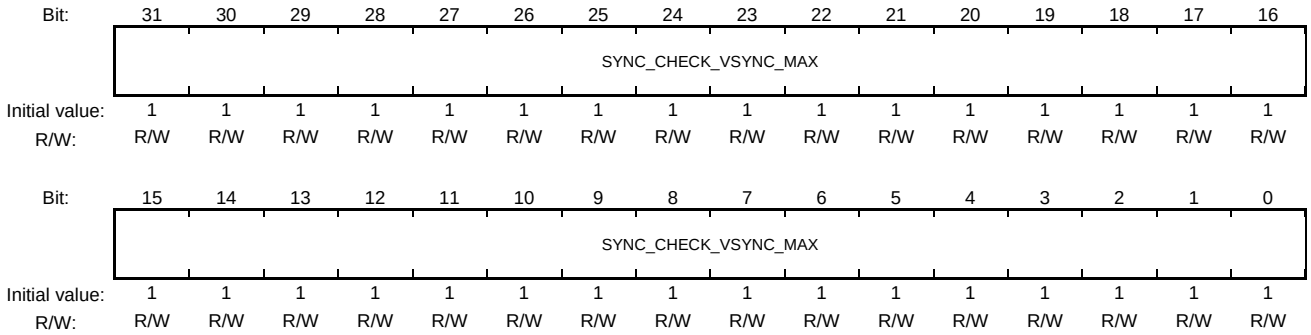
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_FIELD_MAX | H'FFFFFFFF | R/W | Permitted maximum cycles which keeps the same value of the incoming FIELD signal.<br>If the cycle which keeps the high value or low value exceeds the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.51 ISP Sync Checker Control FIELD MIN VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_FIELD_MIN_VC[g]_[c])



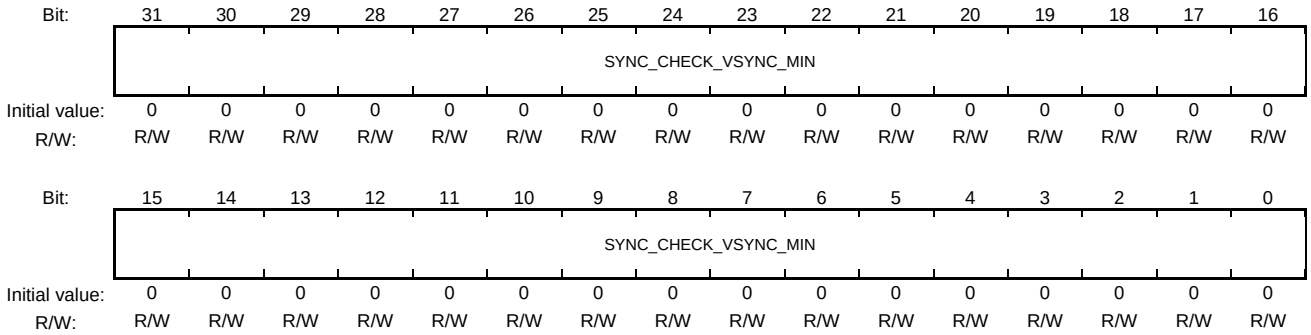

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_FIELD_MIN | H'0 | R/W | Permitted minimum cycles which keeps the same value of the incoming FIELD signal.<br>If the cycle which keeps the high value or low value is less than the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.52 ISP Sync Checker Control VSYNC MAX VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_VSYNC_MAX_VC[g]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_VSYNC_MAX | H'FFFFFFFF | R/W | Permitted maximum cycles which keeps the same value of the incoming VSYNC signal.<br>If the cycle which keeps the high value or low value exceeds the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.53 ISP Sync Checker Control VSYNC MIN VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_VSYNC_MIN_VC[g]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_VSYNC_MIN | H'0 | R/W | Permitted minimum cycles which keeps the same value of the incoming VSYNC signal.<br>If the cycle which keeps the high value or low value is less than the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.54 ISP Sync Checker Control HSYNC Rise MAX VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_HSYNC_RISE_MAX_VC[g]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FIELD_CHECK_EN | | VSYNC_CHECK_EN | | HSYNC_RISE_CHECK_EN | | HSYNC_TOGGLE_CHECK_EN | | DE_CHECK_EN | | — | DEER_DM | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_HSYNC_RISE_MAX | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | FIELD_CHECK_EN | H'0 | R/W | FIELD_CHECK_EN[1] : FIELD MAX Check Enable<br>FIELD_CHECK_EN[0] : FIELD MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 29, 28 | VSYNC_CHECK_EN | H'0 | R/W | VSYNC_CHECK_EN[1] : VSYNC MAX Check Enable<br>VSYNC_CHECK_EN[0] : VSYNC MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 27, 26 | HSYNC_RISE_CHECK_EN | H'0 | R/W | HSYNC_RISE_CHECK_EN[1] : HSYNC Rise MAX Check Enable<br>HSYNC_RISE_CHECK_EN[0] : HSYNC Rise MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 25, 24 | HSYNC_TOGGLE_CHECK_EN | H'0 | R/W | HSYNC_TOGGLE_CHECK_EN[1] : HSYNC Toggle MAX Check Enable<br>HSYNC_TOGGLE_CHECK_EN[0] : HSYNC Toggle MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 23, 22 | DE_CHECK_EN | H'0 | R/W | DE_CHECK_EN[1] : DE MAX Check Enable<br>DE_CHECK_EN[0] : DE MIN Check Enable<br>0 : disable[default]<br>1 : enable |
| 21 | — | B'0 | R | Reserved. (This bit is always read as 0.) |
| 20 | DEER_DM | H'0 | R/W | 0 : Error detection is disabled when counted value is 0 [default]<br>1 : Error detection is enabled when counted value is 0<br><br>Note: This setting is valid when DE_CHECK_EN[0] is 1. |
| 19 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | SYNC_CHECK_HSYNC_RISE_MAX | H'FFFF | R/W | Permitted maximum number of rising edge of incoming HSYNC signal in one frame.<br>If the number of rising edge of HSYNC exceeds the value specified by this register, then SYNC_ERROR is asserted. |

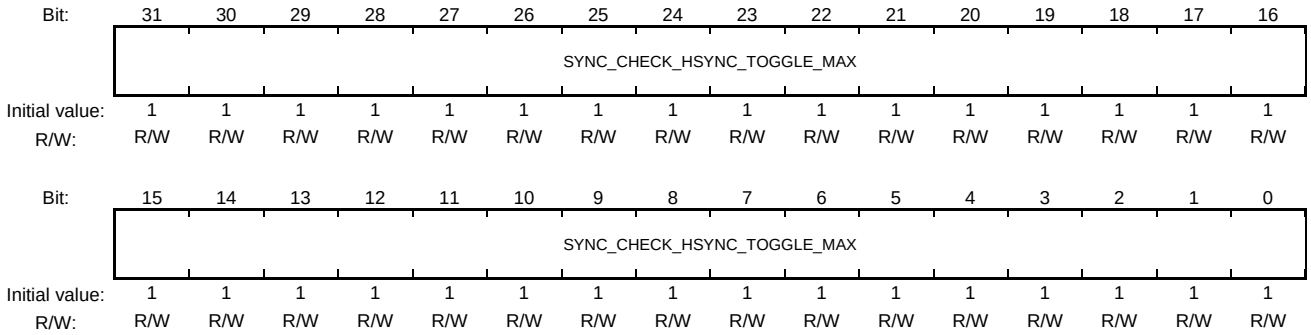
### 54.3.55 ISP Sync Checker Control HSYNC Rise MIN VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_HSYNC_RISE_MIN_VC[g]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_HSYNC_RISE_MIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

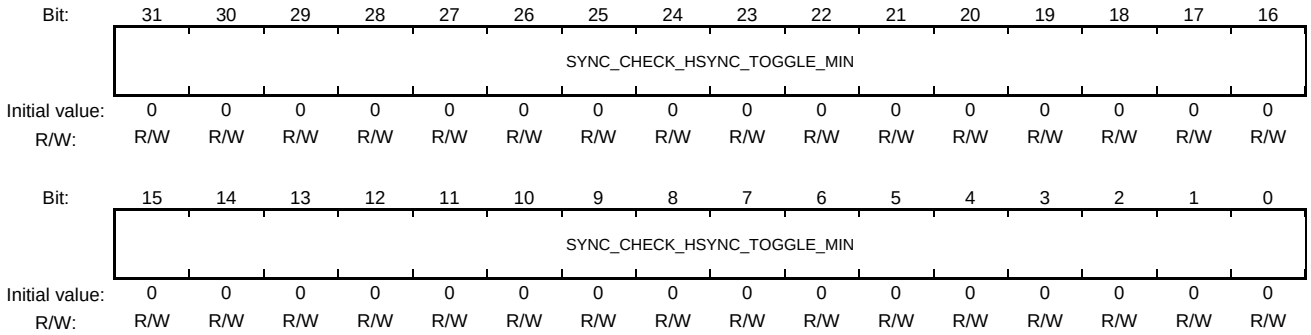
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | SYNC_CHECK_HSYNC_RISE_MIN | H'0 | R/W | Permitted minimum number of rising edge of incoming HSYNC signal in one frame.<br>If the number of rising edge of HSYNC is less than the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.56 ISP Sync Checker Control HSYNC Toggle MAX VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_HSYNC_TOGGLE_MAX_VC[g]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_HSYNC_TOGGLE_MAX | H'FFFFFFFF | R/W | Permitted maximum cycles which keeps the same value of the incoming HSYNC signal.<br>If the cycle which keeps the high value or low value exceeds the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.57 ISP Sync Checker Control HSYNC Toggle MIN VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_HSYNC_TOGGLE_MIN_VC[g]_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SYNC_CHECK_HSYNC_TOGGLE_MIN | H'0 | R/W | Permitted minimum cycles which keeps the same value of the incoming HSYNC signal.<br>If the cycle which keeps the high value or low value is less than the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.58 ISP Sync Checker Control DE MAX VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_DE_MAX_VC[g]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_DE_MAX | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | SYNC_CHECK_DE_MAX | H'FFFF | R/W | Permitted maximum cycles which keeps the same value of the incoming DE signal.<br>If the cycle which keeps the high value or low value exceeds the value specified by this register, then SYNC_ERROR is asserted. |

### 54.3.59 ISP Sync Checker Control DE MIN VC[g] [c](g=8-15)(c=0-1)(ISP_SYNC_CHECK_DE_MIN_VC[g]_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNC_CHECK_DE_MIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | SYNC_CHECK_DE_MIN | H'0 | R/W | Permitted minimum cycles which keeps the same value of the incoming DE signal.<br>If the cycle which keeps the high value or low value is less than the value specified by this register, then SYNC_ERROR is asserted. |

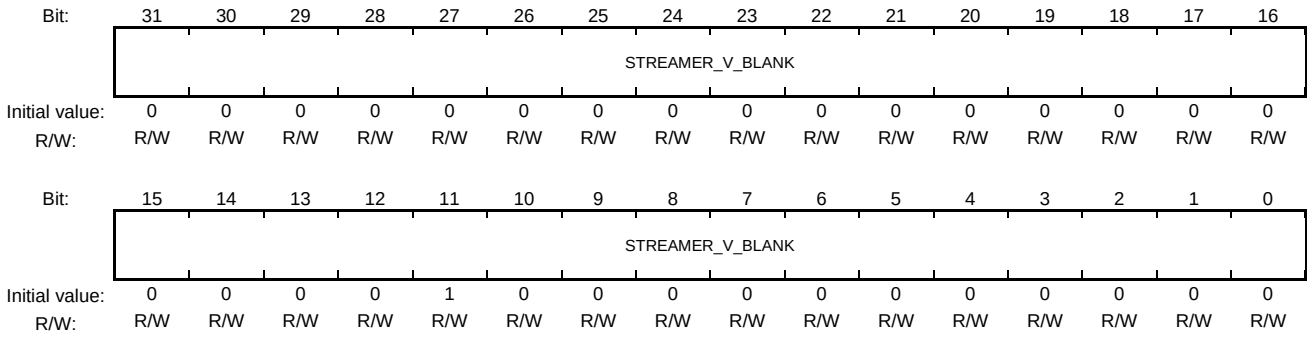
### 54.3.60 ISP Streamer Mode Register [c](c=0-1)(ISP_STREAMER_MODE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | STREAMER_ENABLE | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 3 to 0 | STREAMER_ENABLE | H'0 | R/W | On/OFF control of steaming from VSPX.<br>- B’0000 : Streaming mode from VSPX is disabled [default]<br>- B’1111 : Streaming mode from VSPX is enabled. In this case, input stream to ISP core is provided from VSPX instead of CSI via channel selector.<br>- Otherwise : Forbdden |

### 54.3.61 ISP Streamer V Blank Register [c](c=0-1)(ISP_STREAMER_V_BLANK_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | STREAMER_V_BLANK | H'800 | R/W | Number of cycles of vertical blanking period per one frame for input pixel stream which is provided to ISP core from in streaming mode from VSPX. Unit is cycle. |

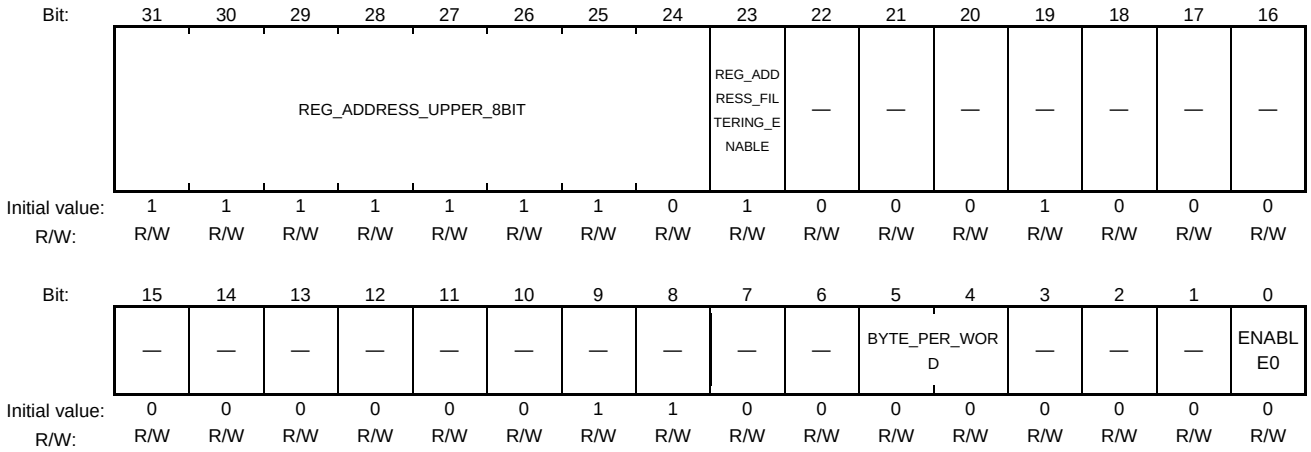
### 54.3.62 ISP Streamer H Blank Register [c](c=0-1)(ISP_STREAMER_H_BLANK_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STREAMER_H_BLANK | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | STREAMER_H_BLANK | H'40 | R/W | Number of cycles of horizontal blanking period per line for input pixel stream which is provided to ISP core from in streaming mode from VSPX. Unit is cycle. |

### 54.3.63 ISP Streamer Config DMA Control Register0 [c](c=0-1)(ISP_STREAMER_CONFIG_DMA_CONTROL_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | REG_ADDRESS_UPPER_8BIT | H'FE | R/W | Specifies the upper 8-bit value of the register address to be written to ISP Core. This register is meaningful only when REG_ADDRESS_FILTERING_ENABLE=1.<br>Has no meaning when REG_ADDRESS_FILTERING_ENABLE=0. |
| 23 | REG_ADDRESS_FILTERING_ENABLE | H'1 | R/W | ・Setting whether or nor REG_ADDRESS_UPPER_8BIT is matched with the upper 8bits of the register address.<br>・When REG_ADDRESS_FILTERING_ENABLE=1 [default].<br>When the upper 8 bits of the register address transferred from VSPX match this register, a register write is performed to ISP Core.<br>・When REG_ADDRESS_FILTERING_ENABLE=0<br>Always perform a register write to ISP Core (no matching of the upper 8 bits of the register address). |
| 22 to 6 | — | H'200C | R/W | Reserved bits. Do not write any different value from its initial value. |
| 5, 4 | BYTE_PER_WORD | H'0 | R/W | Data format of config data<br>B'00 : 4 byte/word [default]<br>Others: Forbidden |
| 3 to 1 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 0 | ENABLE0 | H'0 | R/W | Enabling config DMA<br>0 : Config DMA is disabled [default]<br>1 : Config DMA is enabled |

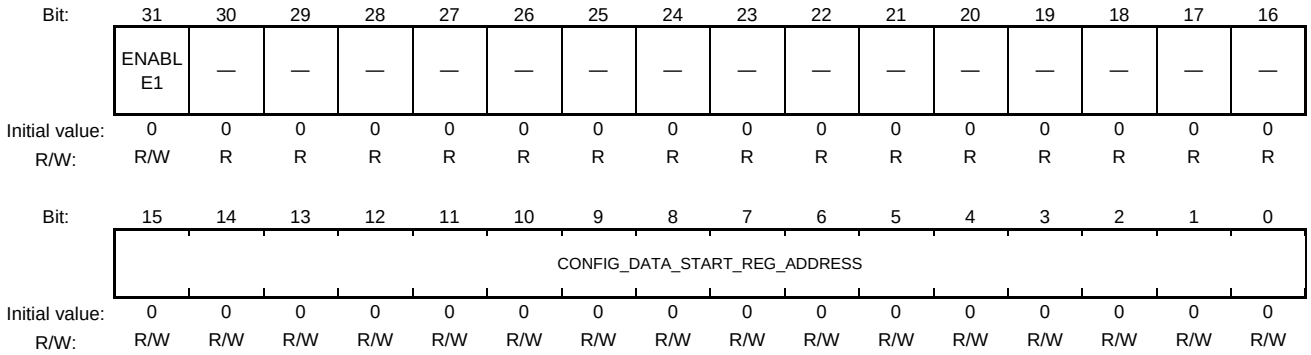
### 54.3.64 ISP Streamer Config DMA Status [c](c=0-1)(ISP_STREAMER_CONFIG_DMA_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CONFIG_DMA_READY_BUSY |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

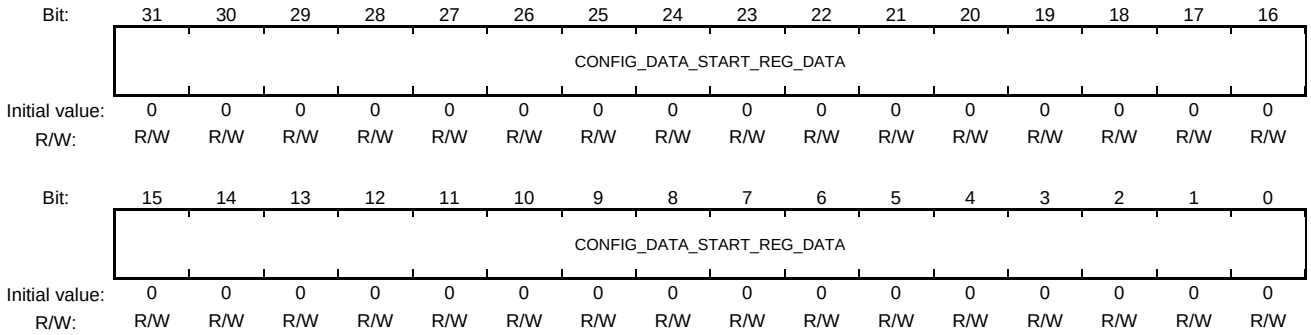
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 0 | CONFIG_DMA_READY_BUSY | H'0 | R | Transfer status of Config DMA<br>0 : Busy (Config dma is running)<br>1 : Ready (Config dma is not running) |

### 54.3.65 ISP Streamer Config DMA Control Register1 [c](c=0-1)(ISP_STREAMER_CONFIG_DMA_CONTROL1_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | ENABLE1 | H'0 | R/W | Enabling config DMA. In addition to set 1 to ENABLE0, this register should be set to 1 in order to enable Config DMA.<br>0 : Config DMA is disabled [default]<br>1 : Config DMA is enabled |
| 30 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | CONFIG_DATA_START_REG_ADDRESS | H'0 | R/W | The lower 16 bit value of the address of the first (address, write data) pair in the Config Data |

### 54.3.66 ISP Streamer Config DMA Control Register2 [c](c=0-1)(ISP_STREAMER_CONFIG_DMA_CONTROL2_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CONFIG_DATA_START_REG_DATA | H'0 | R/W | The write data value of the first (address, write data) pair in the Config Data |

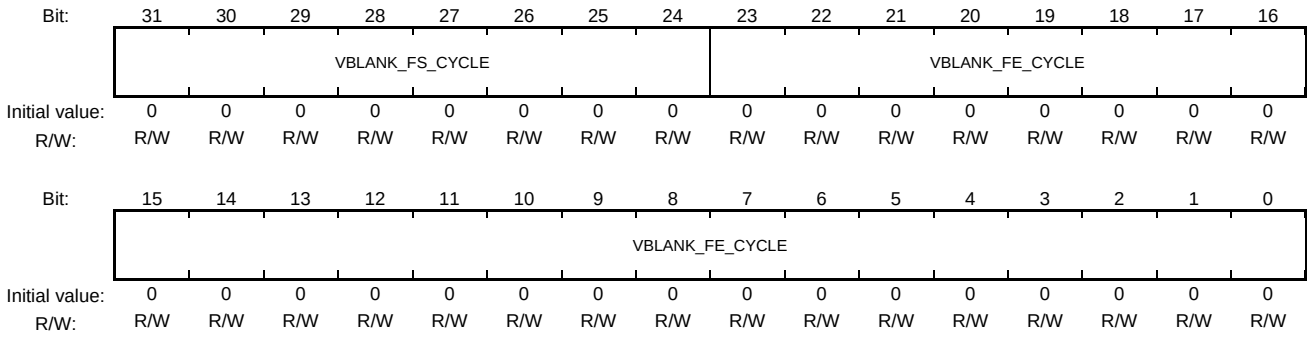
### 54.3.67 ISP RAW Scaler Mode Register [c](c=0-1)(RAW_SCALER_MODE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TARGET_VC | | | | — | — | TARGET_DT_CODE | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ENABLE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 27 to 24 | TARGET_VC | H'0 | R/W | Virtual channel (VC) number to input RAW scaler |
| 23, 22 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 21 to 16 | TARGET_DT_CODE | H'0 | R/W | Data type (DT) code to input RAW scaler |
| 15 to 1 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 0 | ENABLE | H'0 | R/W | Enable RAW scaler<br>0 : RAW scaler is disabled [default]<br>1 : RAW sacaler is enabled |

### 54.3.68 ISP RAW Scaler Output Vblank Register [c](c=0-1)(RAW_SCALER_OUT_VBLANK_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | VBLANK_FS_CYCLE | H'0 | R/W | Number of cycles of vertical blanking period at frame start timing of RAW scaler output image |
| 23 to 0 | VBLANK_FE_CYCLE | H'0 | R/W | Number of cycles of vertical blanking period at frame end timing of RAW scaler output image |

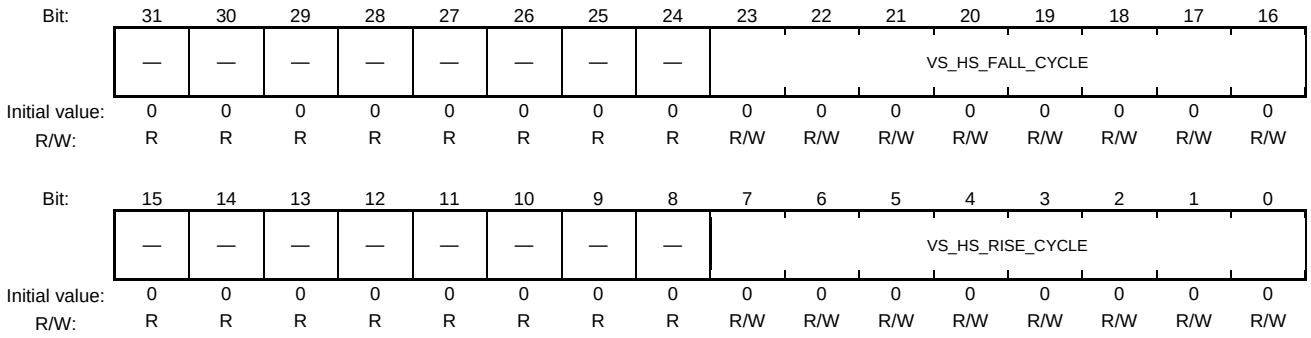
### 54.3.69 ISP RAW Scaler Output Hblank Register [c](c=0-1)(RAW_SCALER_OUT_HBLANK_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | BACK_PORCH_CYCLE | | | | — | — | — | — | FRONT_PORCH_CYCLE | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | HBLANK_CYCLE | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 27 to 24 | BACK_PORCH_CYCLE | H'0 | R/W | Number of cycles of back porch for each line of RAW scaler output image |
| 23 to 20 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 19 to 16 | FRONT_PORCH_CYCLE | H'0 | R/W | Number of cycles of front porch for each line of RAW scaler output image |
| 15 to 12 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 11 to 0 | HBLANK_CYCLE | H'0 | R/W | Number of cycles of horitontal blanking period for each line of RAW scaled image |

### 54.3.70 ISP RAW Scaler Output Sync Interval Register [c](c=0-1)(RAW_SCALER_OUT_VS_HS_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 23 to 16 | VS_HS_FALL_CYCLE | H'0 | R/W | Number of cycles from VSYNC fall timing to HSYNC fall timing of RAW scaler output image |
| 15 to 8 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 7 to 0 | VS_HS_RISE_CYCLE | H'0 | R/W | Number of cycles from VSYNC rise timing to HSYNC rise timing of RAW scaler output image |

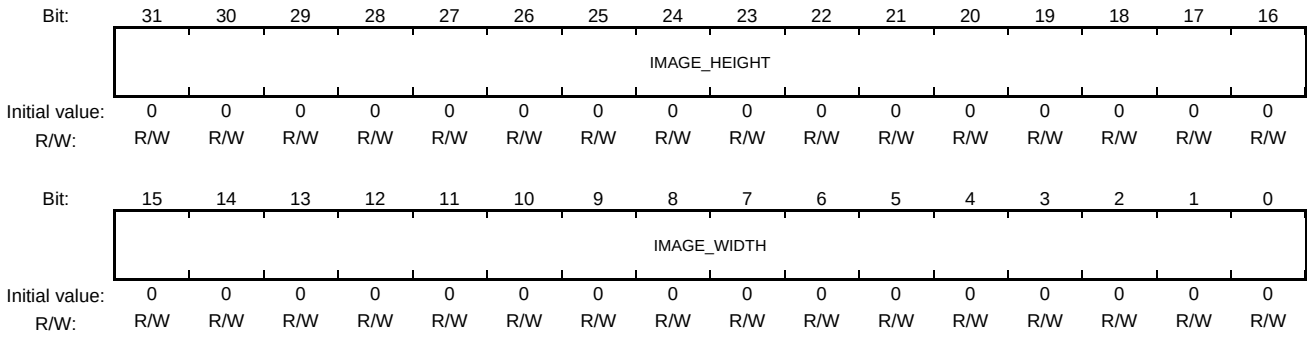
### 54.3.71 ISP RAW Scaler Input Left Shift Register [c](c=0-1)(RAW_SCALER_INPUT_LSHIFT_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | INPUT_LSHIFT | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

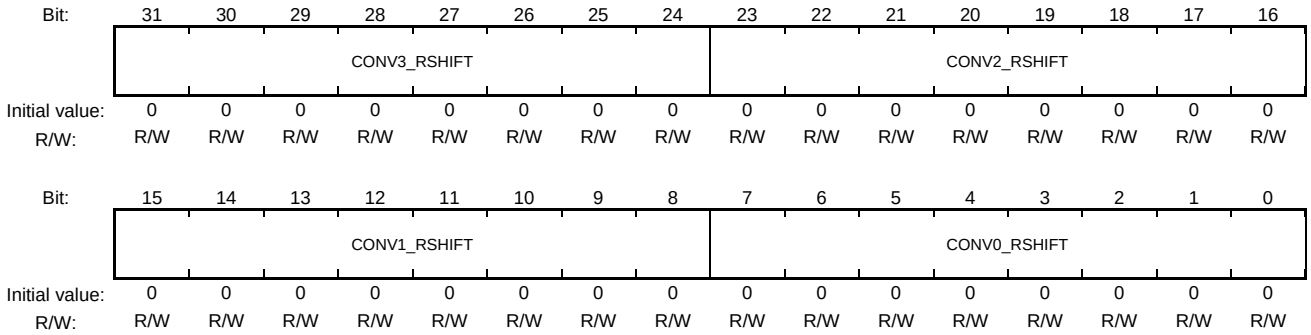
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 4 to 0 | INPUT_LSHIFT | H'0 | R/W | Left shift amount for RAW scaler input pixel before multiplying convolution coefficient |

### 54.3.72 ISP RAW Scaler Image Size Register [c](c=0-1)(RAW_SCALER_IMAGE_SIZE_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | IMAGE_HEIGHT | H'0 | R/W | Image height (number of lines in one frame) of RAW scaler input image |
| 15 to 0 | IMAGE_WIDTH | H'0 | R/W | Image width (number of pixels per line) of RAW scaler input image |

### 54.3.73 ISP RAW Scaler Convolution Right Shift Register [c](c=0-1)(RAW_SCALER_CONV_RSHIFT_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | CONV3_RSHIFT | H'0 | R/W | Right shift amount after convolution operation when coeff_select_value(x%4,y%4) is 3. Setting value of conv_rshift(3) |
| 23 to 16 | CONV2_RSHIFT | H'0 | R/W | Right shift amount after convolution operation when coeff_select_value(x%4,y%4) is 2. Setting value fo conv_rshift(2) |
| 15 to 8 | CONV1_RSHIFT | H'0 | R/W | Right shift amount after convolution operation when coeff_select_value(x%4,y%4) is 1. Setting value of conv_rshift(1) |
| 7 to 0 | CONV0_RSHIFT | H'0 | R/W | Right shift amount after convolution operation when coeff_select_value(x%4,y%4) is 0. Setting value of conv_rshift(0) |

### 54.3.74 ISP RAW Scaler Coefficient Select ROW0 Register [c](c=0-1)(RAW_SCALER_COEFF_SEL_ROW0_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COEFF_SEL_0_3 | | | | COEFF_SEL_0_2 | | | | COEFF_SEL_0_1 | | | | COEFF_SEL_0_0 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 12 | COEFF_SEL_0_3 | H'0 | R/W | Setting value of coeff_select_table(0,3). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(0,3)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 11 to 8 | COEFF_SEL_0_2 | H'0 | R/W | Setting value of coeff_select_table(0,2). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(0,2)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 7 to 4 | COEFF_SEL_0_1 | H'0 | R/W | Setting value of coeff_select_table(0,1). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(0,1)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 3 to 0 | COEFF_SEL_0_0 | H'0 | R/W | Setting value of coeff_select_table(0,0). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(0,0)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |

### 54.3.75 ISP RAW Scaler Coefficient Select ROW1 Register [c](c=0-1)(RAW_SCALER_COEFF_SEL_ROW1_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COEFF_SEL_1_3 | | | | COEFF_SEL_1_2 | | | | COEFF_SEL_1_1 | | | | COEFF_SEL_1_0 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 12 | COEFF_SEL_1_3 | H'0 | R/W | Setting value of coeff_select_table(1,3). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(1,3)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 11 to 8 | COEFF_SEL_1_2 | H'0 | R/W | Setting value of coeff_select_table(1,2). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(1,2)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 7 to 4 | COEFF_SEL_1_1 | H'0 | R/W | Setting value of coeff_select_table(1,1). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(1,1)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 3 to 0 | COEFF_SEL_1_0 | H'0 | R/W | Setting value of coeff_select_table(1,0). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(1,0)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |

### 54.3.76 ISP RAW Scaler Coefficient Select ROW2 Register [c](c=0-1)(RAW_SCALER_COEFF_SEL_ROW2_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COEFF_SEL_2_3 | | | | COEFF_SEL_2_2 | | | | COEFF_SEL_2_1 | | | | COEFF_SEL_2_0 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 12 | COEFF_SEL_2_3 | H'0 | R/W | Setting value of coeff_select_table(2,3). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(2,3)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 11 to 8 | COEFF_SEL_2_2 | H'0 | R/W | Setting value of coeff_select_table(2,2). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(2,2)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 7 to 4 | COEFF_SEL_2_1 | H'0 | R/W | Setting value of coeff_select_table(2,1). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(2,1)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 3 to 0 | COEFF_SEL_2_0 | H'0 | R/W | Setting value of coeff_select_table(2,0). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(2,0)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |

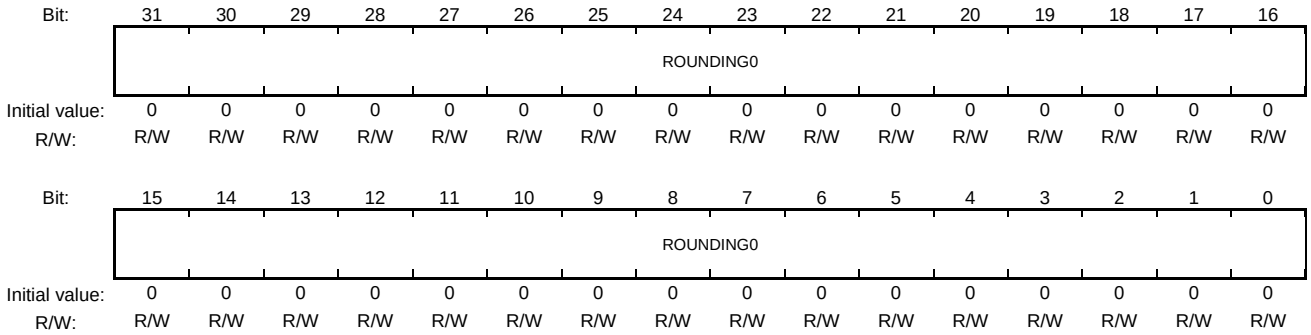
### 54.3.77 ISP RAW Scaler Coefficient Select ROW3 Register [c](c=0-1)(RAW_SCALER_COEFF_SEL_ROW3_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COEFF_SEL_3_3 | | | | COEFF_SEL_3_2 | | | | COEFF_SEL_3_1 | | | | COEFF_SEL_3_0 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

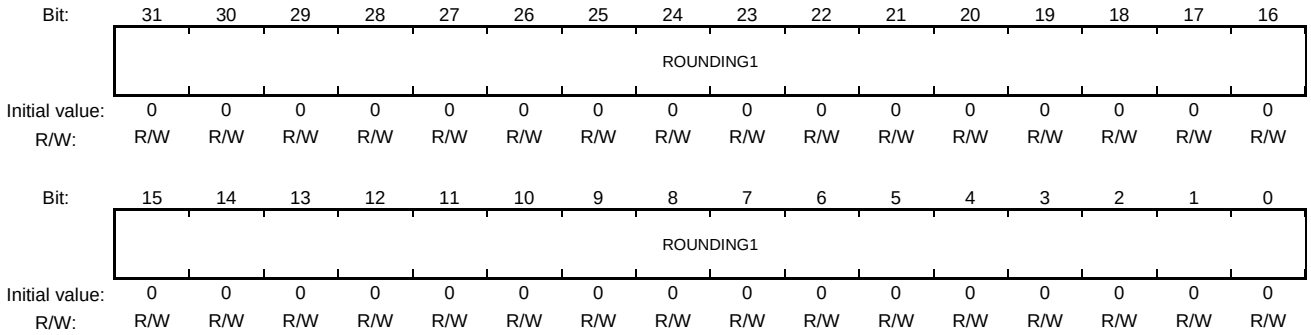
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 12 | COEFF_SEL_3_3 | H'0 | R/W | Setting value of coeff_select_table(3,3). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(3,3)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 11 to 8 | COEFF_SEL_3_2 | H'0 | R/W | Setting value of coeff_select_table(3,2). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(3,2)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 7 to 4 | COEFF_SEL_3_1 | H'0 | R/W | Setting value of coeff_select_table(3,1). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(3,1)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |
| 3 to 0 | COEFF_SEL_3_0 | H'0 | R/W | Setting value of coeff_select_table(3,0). Index of convolution coefficient when pixel coordinate is (x%4,y%4)=(3,0)<br>0,1,2,3: Uses "Coeff[s]_*_* (s=0,1,2,3)" resigter for convolution coefficient<br>15: No pixel will be output |

### 54.3.78 ISP RAW Scaler Convolution Rounding0 Register [c](c=0-1)(RAW_SCALER_ROUNDING0_[c])



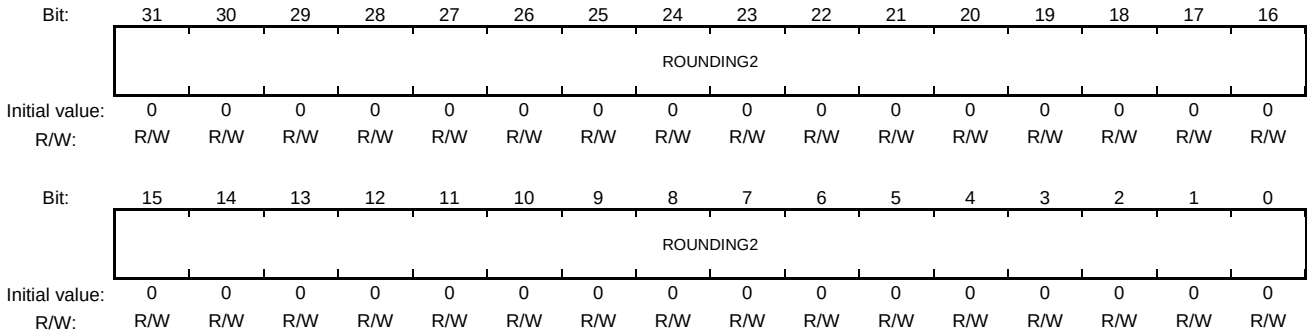

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ROUNDING0 | H'0 | R/W | Setting value of rounding(0) |

### 54.3.79 ISP RAW Scaler Convolution Rounding1 Register [c](c=0-1)(RAW_SCALER_ROUNDING1_[c])



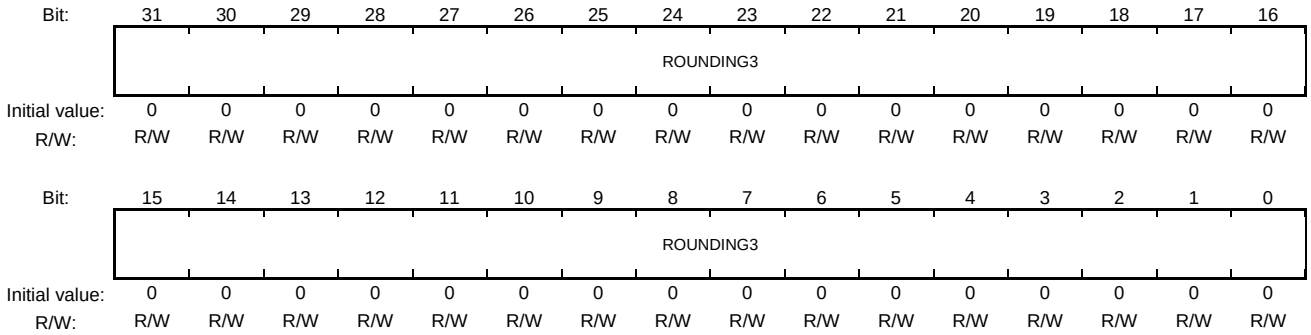

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ROUNDING1 | H'0 | R/W | Setting value of rounding(1) |

### 54.3.80 ISP RAW Scaler Convolution Rounding2 Register [c](c=0-1)(RAW_SCALER_ROUNDING2_[c])



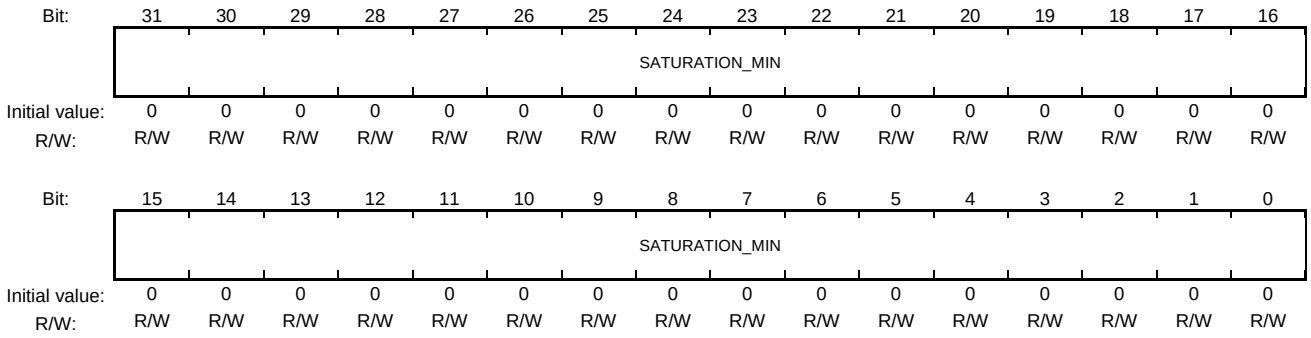

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ROUNDING2 | H'0 | R/W | Setting value of rounding(2) |

### 54.3.81 ISP RAW Scaler Convolution Rounding3 Register [c](c=0-1)(RAW_SCALER_ROUNDING3_[c])



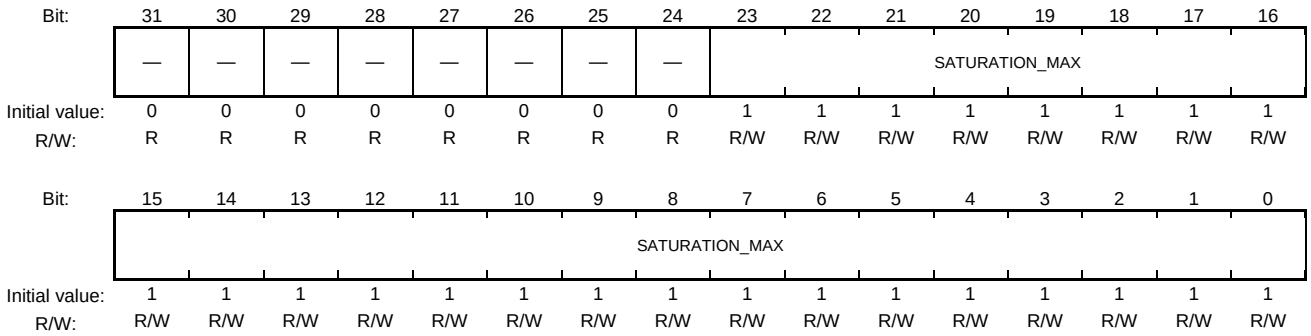

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ROUNDING3 | H'0 | R/W | Setting value of rounding(3) |

### 54.3.82 ISP RAW Scaler Minimum Saturation Register [c](c=0-1)(RAW_SCALER_SAT_MIN_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SATURATION_MIN | H'0 | R/W | Minimum value of saturation operation after convolution |

### 54.3.83 ISP RAW Scaler Maximum Saturation Register [c](c=0-1)(RAW_SCALER_SAT_MAX_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 23 to 0 | SATURATION_MAX | H'FFFFFF | R/W | Maximum value of saturation operation after convolution |

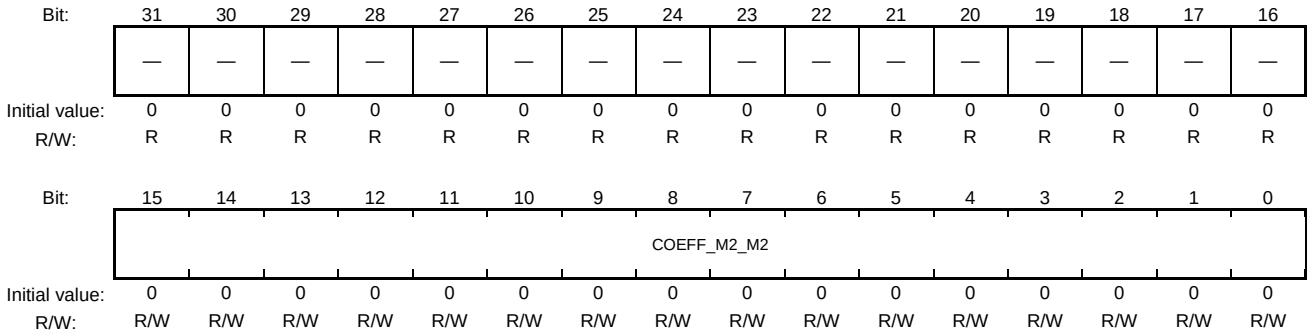
### 54.3.84 ISP RAW Scaler Final Left Shift Register [c](c=0-1)(RAW_SCALER_SAT_FINAL_LSHIFT_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | FINAL_LSHIFT | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

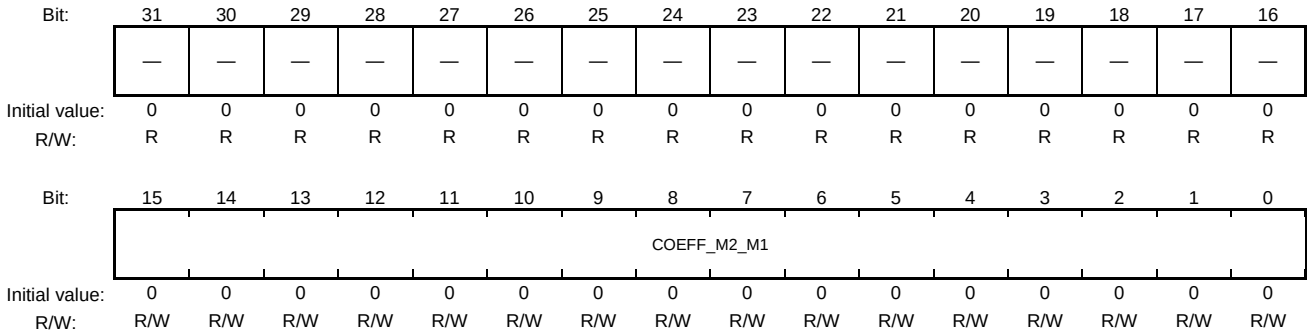
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 4 to 0 | FINAL_LSHIFT | H'0 | R/W | Left shift amount after convoution |

### 54.3.85 ISP RAW Scaler Convolution Coefficient[e] Register_M2_M2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M2_M2_[c])



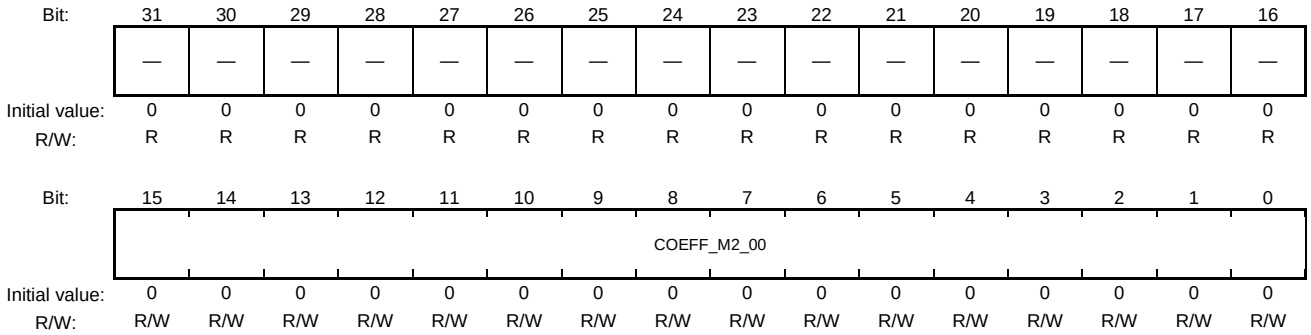

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M2_M2 | H'0 | R/W | Convoution coefficient |

### 54.3.86 ISP RAW Scaler Convolution Coefficient[e] Register_M2_M1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M2_M1_[c])



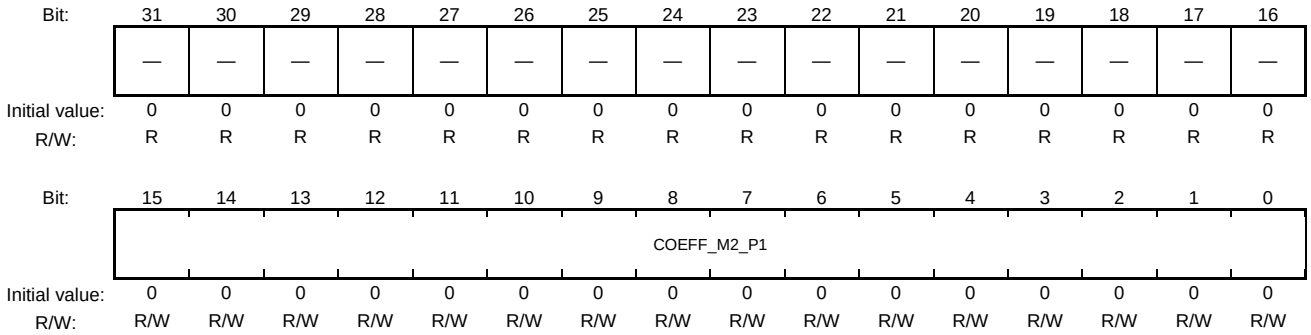

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M2_M1 | H'0 | R/W | Convoution coefficient |

### 54.3.87 ISP RAW Scaler Convolution Coefficient[e] Register_M2_00 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M2_00_[c])



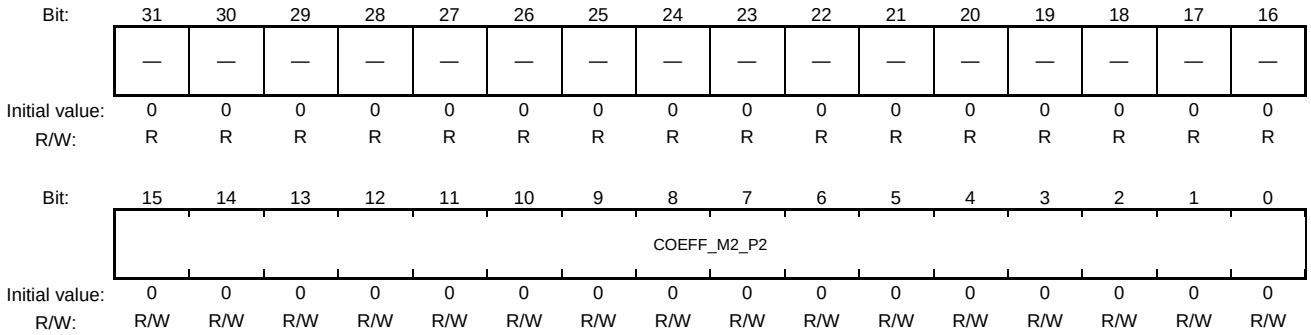

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M2_00 | H'0 | R/W | Convoution coefficient |

### 54.3.88 ISP RAW Scaler Convolution Coefficient[e] Register_M2_P1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M2_P1_[c])



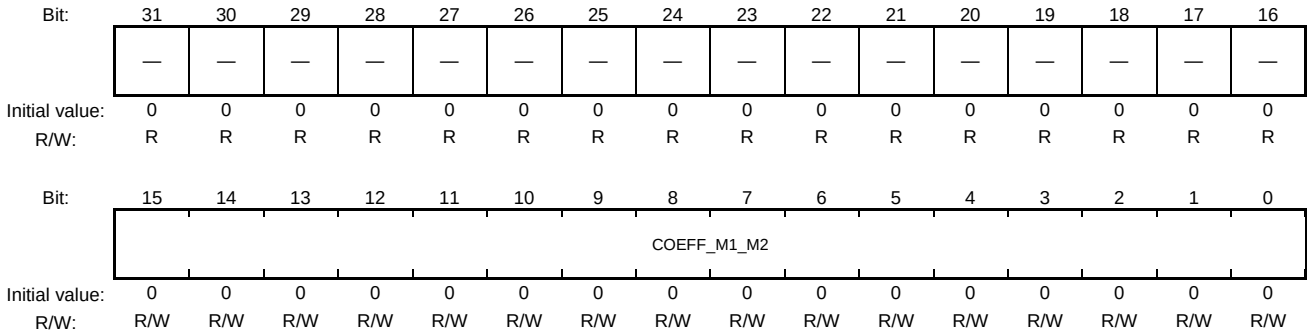

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M2_P1 | H'0 | R/W | Convoution coefficient |

### 54.3.89 ISP RAW Scaler Convolution Coefficient[e] Register_M2_P2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M2_P2_[c])



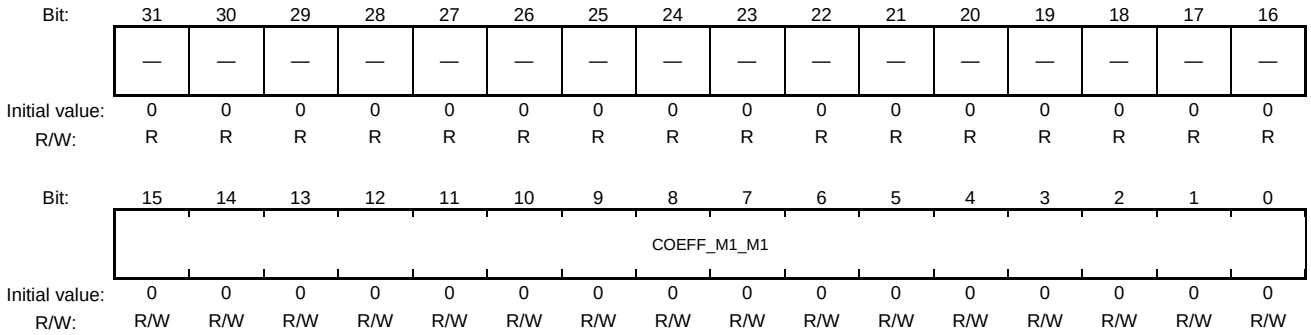

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M2_P2 | H'0 | R/W | Convoution coefficient |

### 54.3.90 ISP RAW Scaler Convolution Coefficient[e] Register_M1_M2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M1_M2_[c])



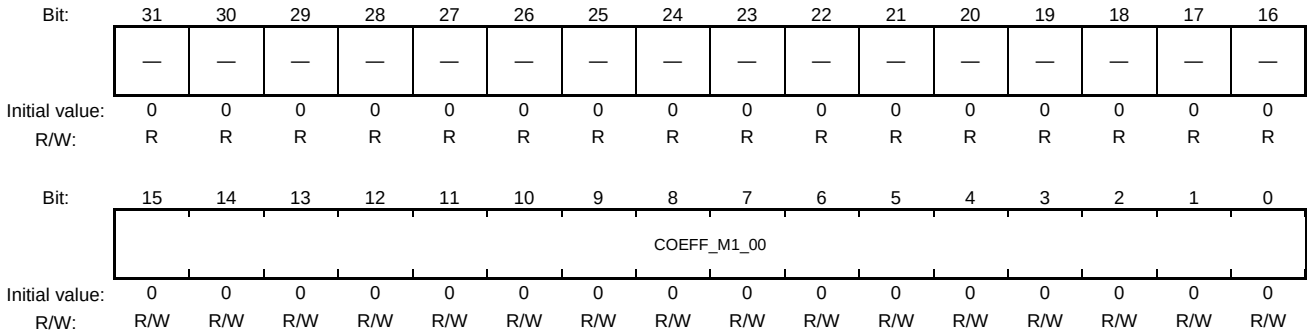

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M1_M2 | H'0 | R/W | Convoution coefficient |

### 54.3.91 ISP RAW Scaler Convolution Coefficient[e] Register_M1_M1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M1_M1_[c])



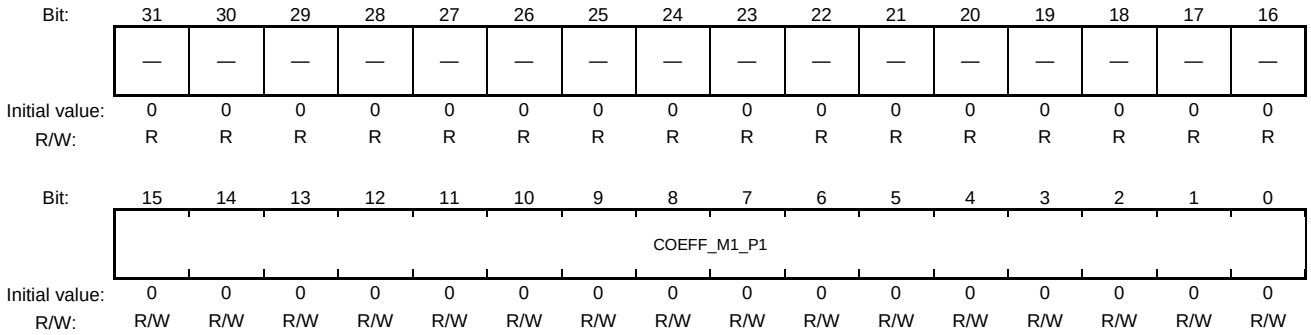

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M1_M1 | H'0 | R/W | Convoution coefficient |

### 54.3.92 ISP RAW Scaler Convolution Coefficient[e] Register_M1_00 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M1_00_[c])



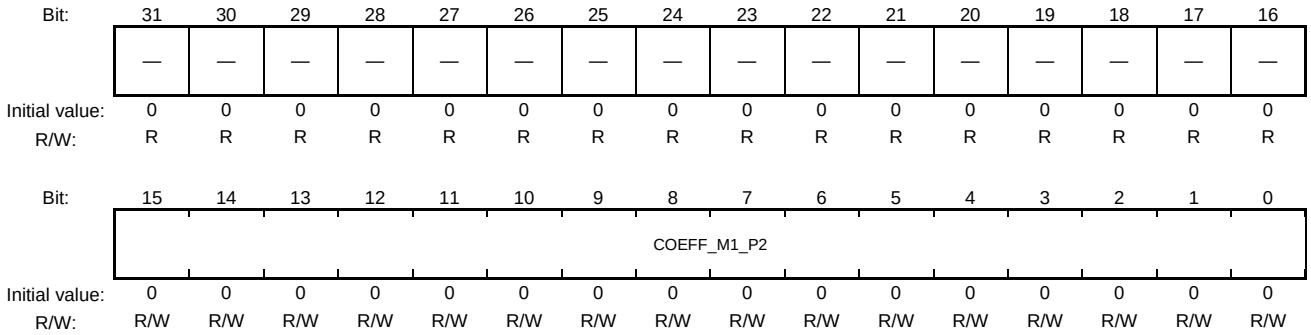

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M1_00 | H'0 | R/W | Convoution coefficient |

### 54.3.93 ISP RAW Scaler Convolution Coefficient[e] Register_M1_P1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M1_P1_[c])



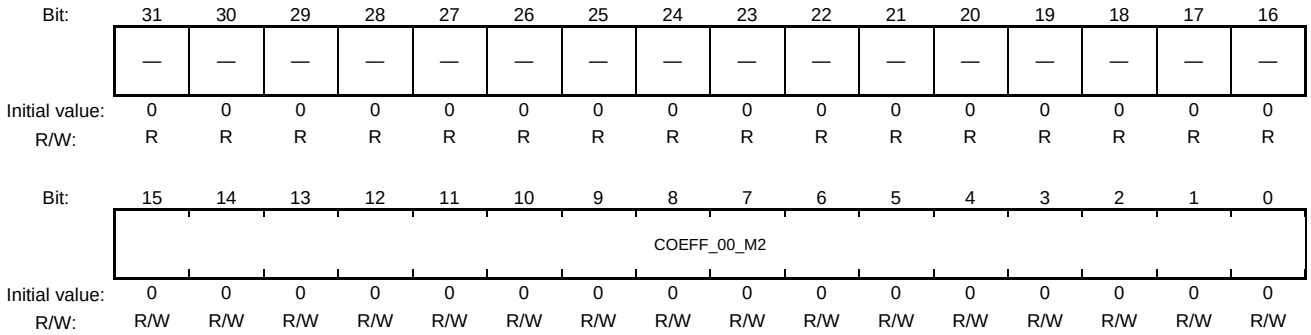

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M1_P1 | H'0 | R/W | Convoution coefficient |

### 54.3.94 ISP RAW Scaler Convolution Coefficient[e] Register_M1_P2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_M1_P2_[c])



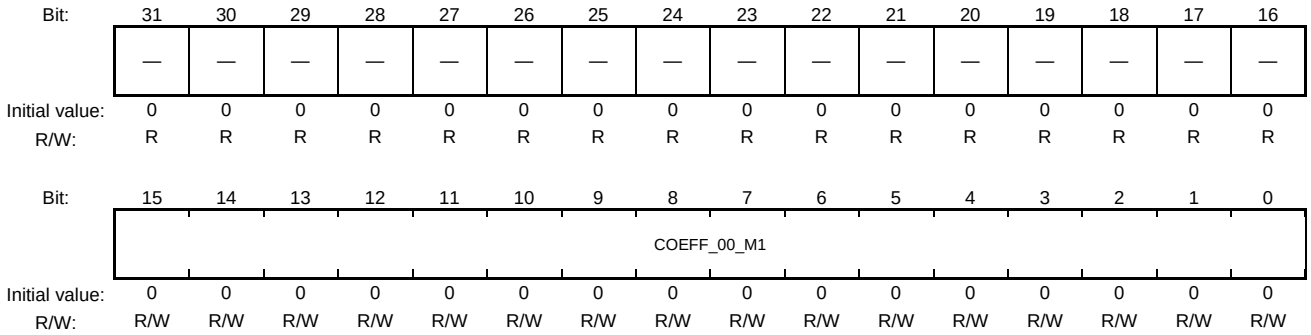

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_M1_P2 | H'0 | R/W | Convoution coefficient |

### 54.3.95 ISP RAW Scaler Convolution Coefficient[e] Register_00_M2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_00_M2_[c])



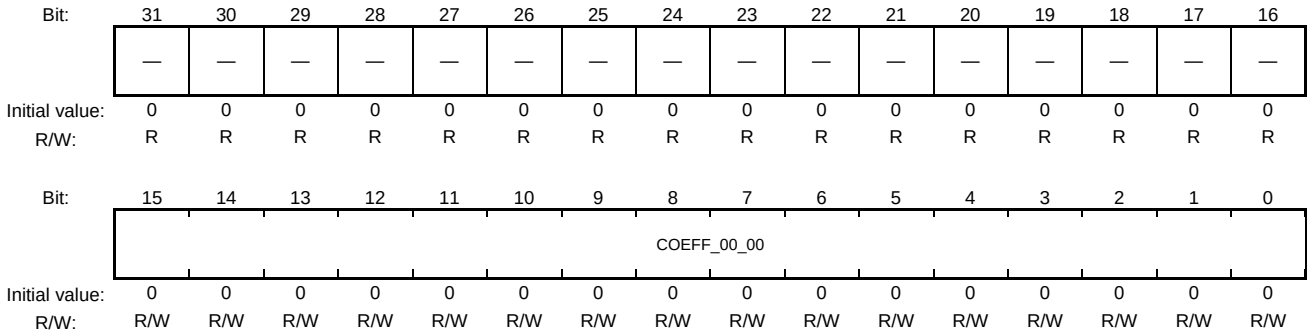

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_00_M2 | H'0 | R/W | Convoution coefficient |

### 54.3.96 ISP RAW Scaler Convolution Coefficient[e] Register_00_M1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_00_M1_[c])



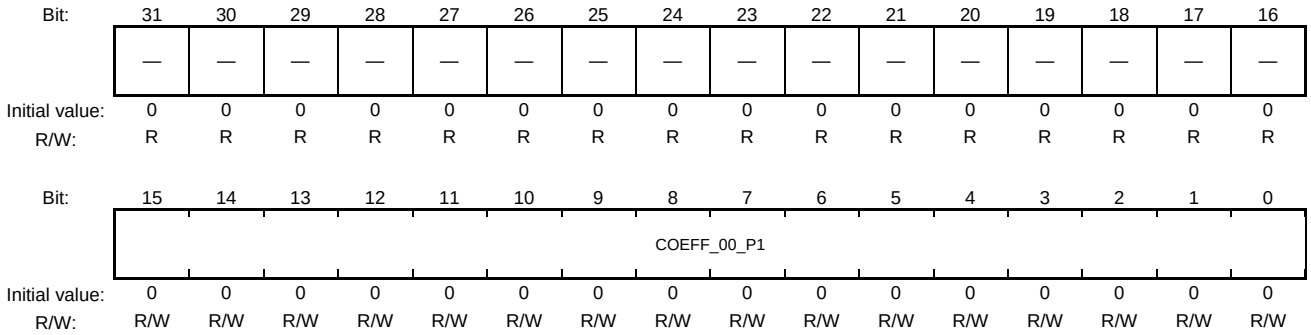

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_00_M1 | H'0 | R/W | Convoution coefficient |

### 54.3.97 ISP RAW Scaler Convolution Coefficient[e] Register_00_00 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_00_00_[c])



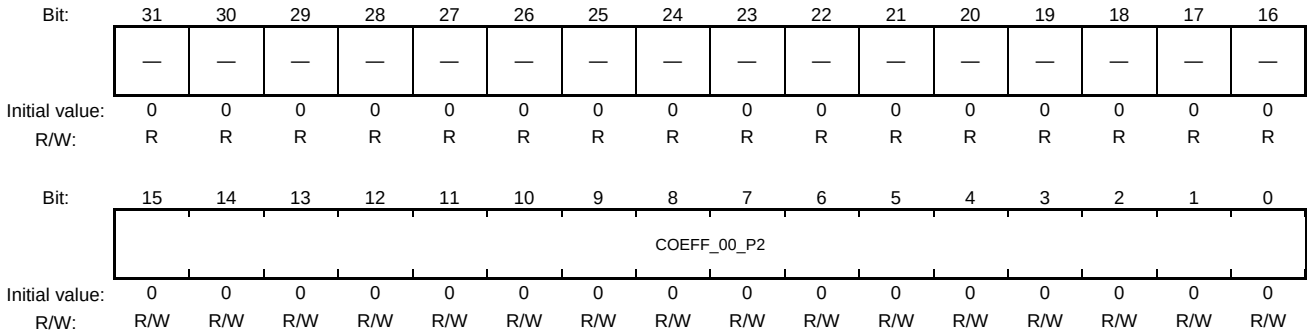

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_00_00 | H'0 | R/W | Convoution coefficient |

### 54.3.98 ISP RAW Scaler Convolution Coefficient[e] Register_00_P1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_00_P1_[c])



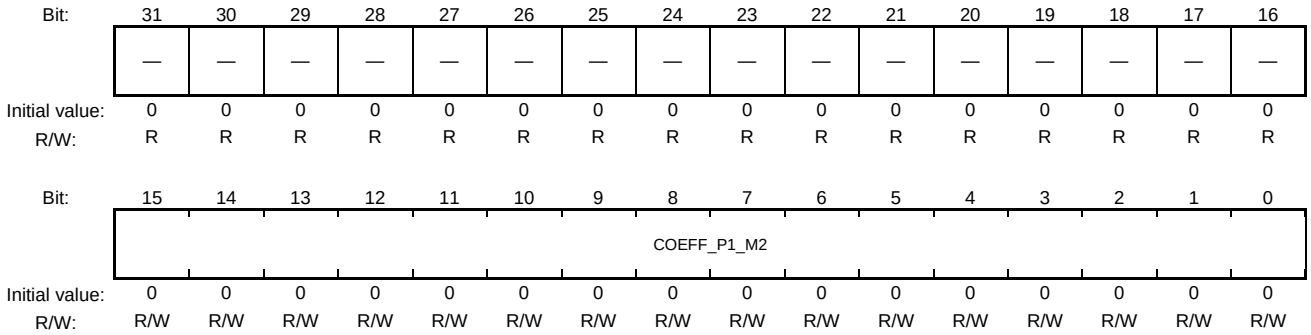

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_00_P1 | H'0 | R/W | Convoution coefficient |

### 54.3.99 ISP RAW Scaler Convolution Coefficient[e] Register_00_P2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_00_P2_[c])



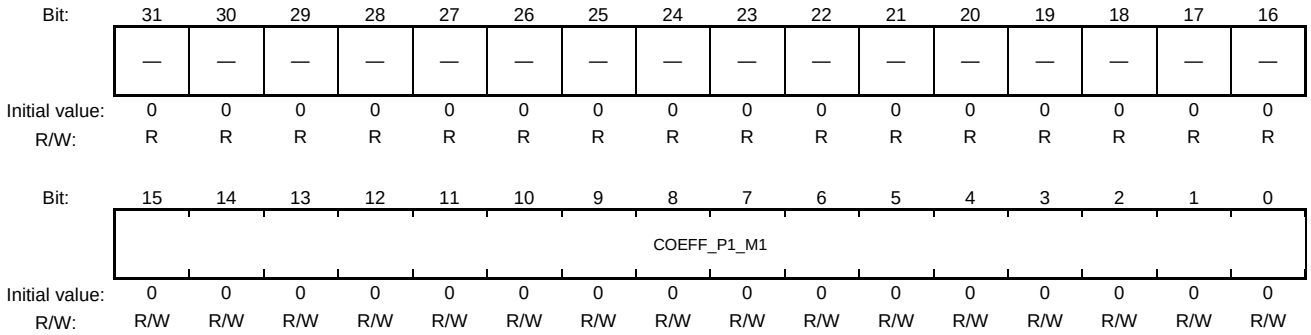

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_00_P2 | H'0 | R/W | Convoution coefficient |

### 54.3.100 ISP RAW Scaler Convolution Coefficient[e] Register_P1_M2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P1_M2_[c])



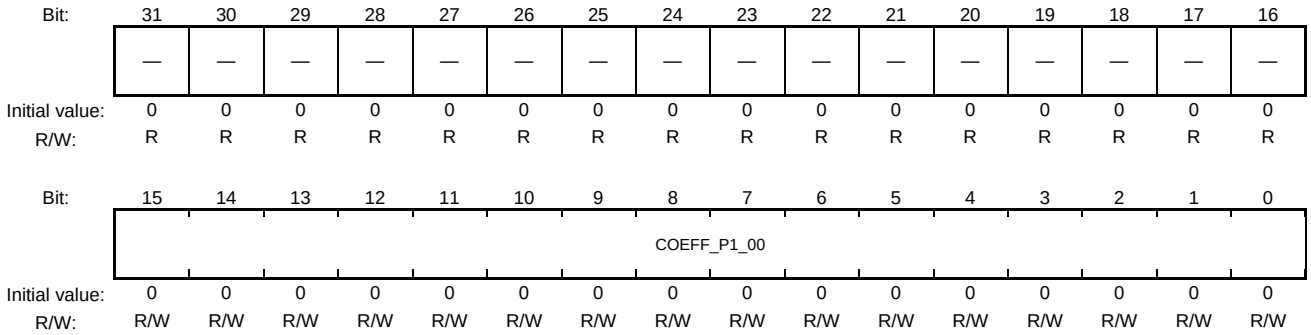

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P1_M2 | H'0 | R/W | Convoution coefficient |

### 54.3.101 ISP RAW Scaler Convolution Coefficient[e] Register_P1_M1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P1_M1_[c])



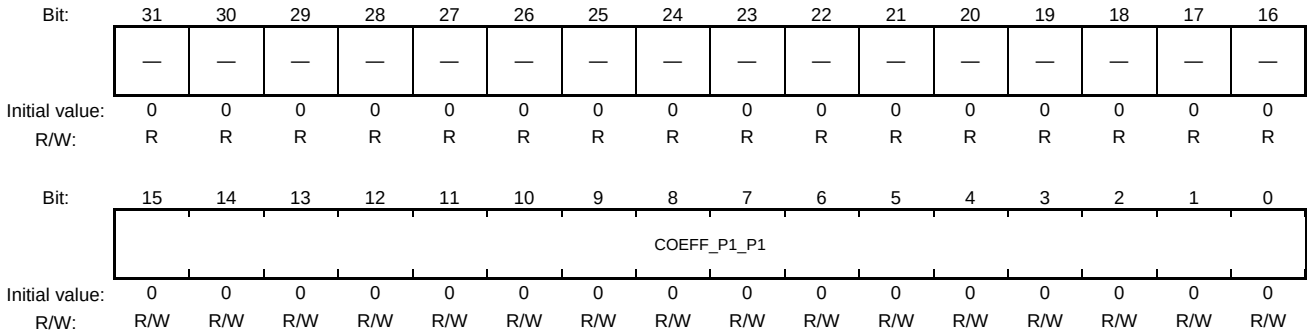

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P1_M1 | H'0 | R/W | Convoution coefficient |

### 54.3.102 ISP RAW Scaler Convolution Coefficient[e] Register_P1_00 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P1_00_[c])



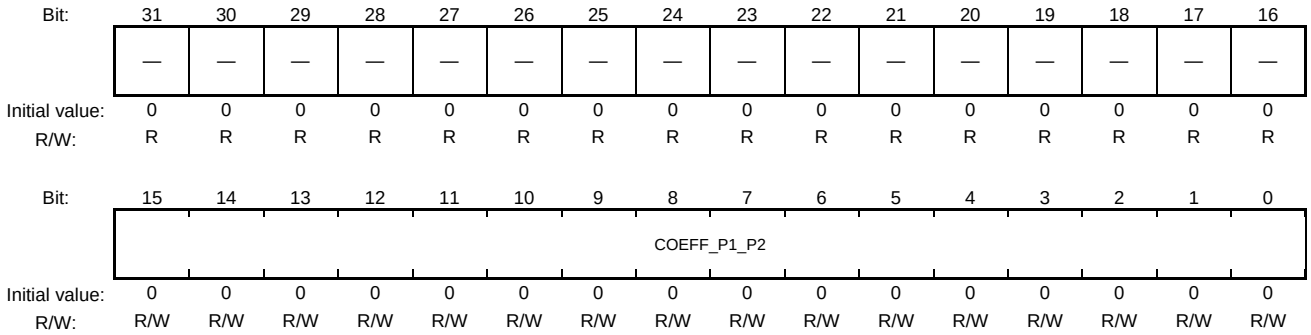

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P1_00 | H'0 | R/W | Convoution coefficient |

### 54.3.103 ISP RAW Scaler Convolution Coefficient[e] Register_P1_P1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P1_P1_[c])



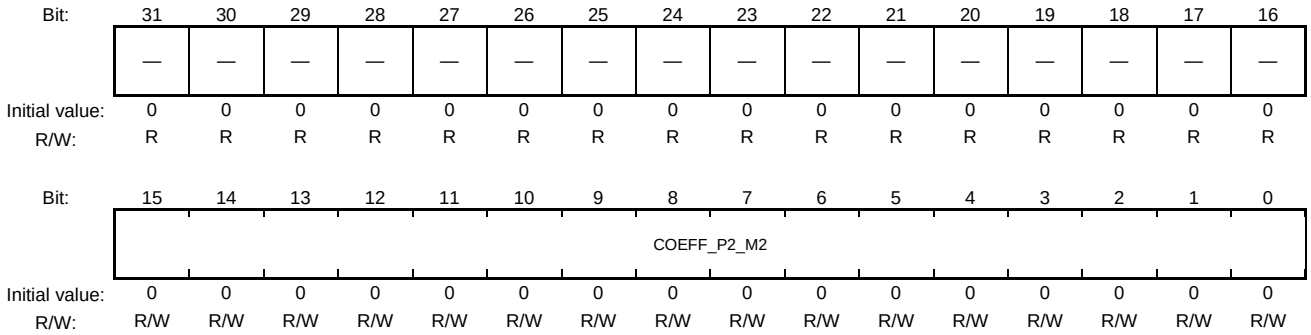

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P1_P1 | H'0 | R/W | Convoution coefficient |

### 54.3.104 ISP RAW Scaler Convolution Coefficient[e] Register_P1_P2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P1_P2_[c])



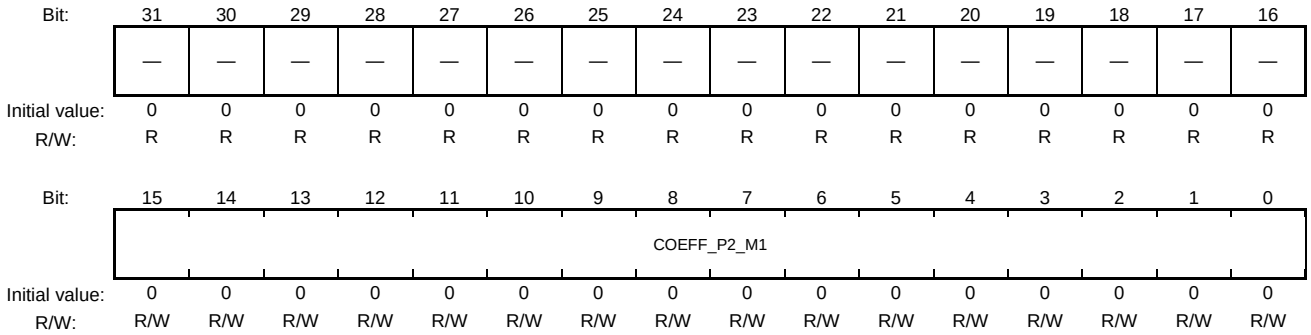

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P1_P2 | H'0 | R/W | Convoution coefficient |

### 54.3.105 ISP RAW Scaler Convolution Coefficient[e] Register_P2_M2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P2_M2_[c])



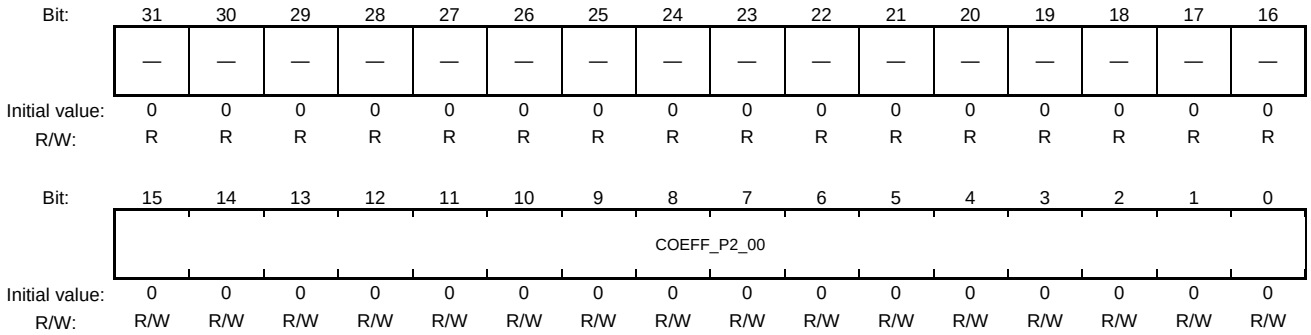

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P2_M2 | H'0 | R/W | Convoution coefficient |

### 54.3.106 ISP RAW Scaler Convolution Coefficient[e] Register_P2_M1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P2_M1_[c])



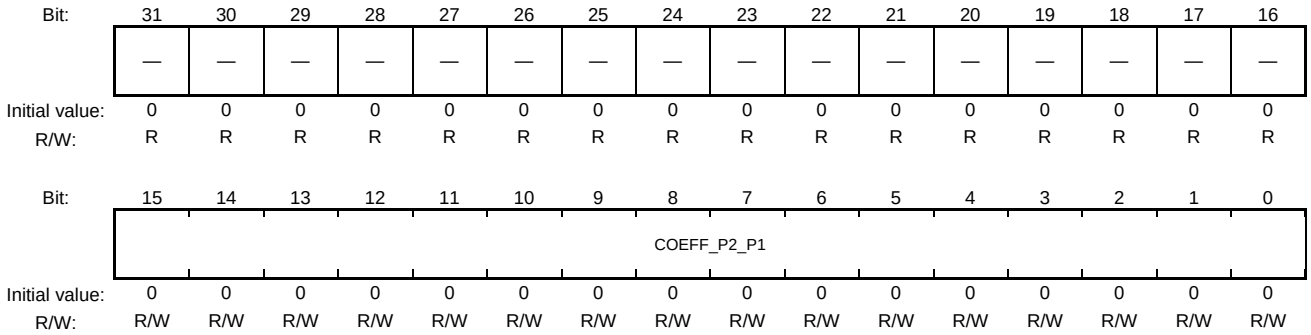

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P2_M1 | H'0 | R/W | Convoution coefficient |

### 54.3.107 ISP RAW Scaler Convolution Coefficient[e] Register_P2_00 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P2_00_[c])



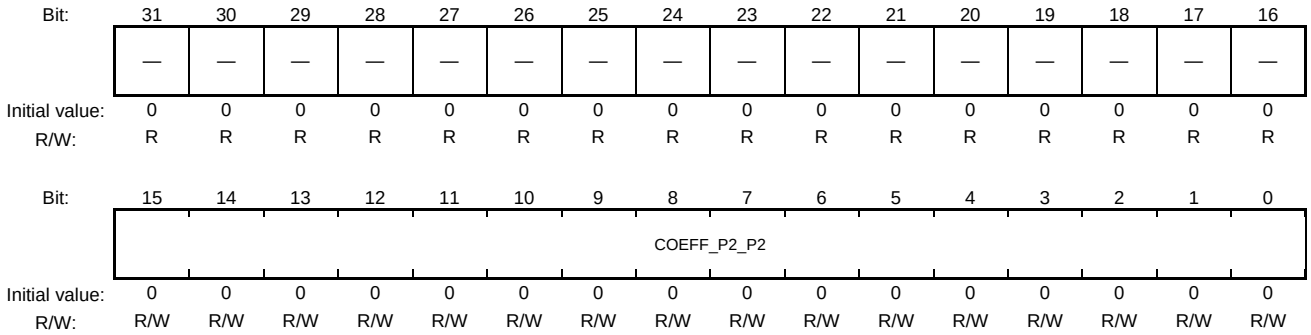

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P2_00 | H'0 | R/W | Convoution coefficient |

### 54.3.108 ISP RAW Scaler Convolution Coefficient[e] Register_P2_P1 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P2_P1_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P2_P1 | H'0 | R/W | Convoution coefficient |

### 54.3.109 ISP RAW Scaler Convolution Coefficient[e] Register_P2_P2 [c](e=0-3)(c=0-1)(RAW_SCALER_COEFF[e]_P2_P2_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | COEFF_P2_P2 | H'0 | R/W | Convoution coefficient |

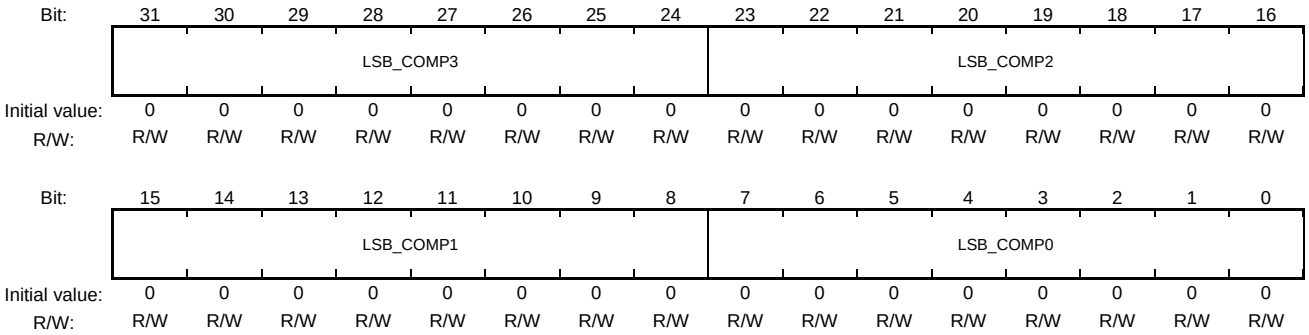
### 54.3.110 ISP Core Image[i] Frame[f] Output DMA Mode Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_MODE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ENABLE_WUP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | NUM_OF_COMPONENT_MINUS1 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 16 | ENABLE_WUP | H'0 | R/W | On/OFF control whether WUP signal is issued in synchronized system with IMR<br>0 : WUP signal is disabled (Synchoronized system with IMR is not used)<br>1: WUP signal is enabled |
| 15 to 2 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 1, 0 | NUM_OF_COMPONENT_MINUS1 | H'0 | R/W | Number of components minus 1.<br>For example, in RGB format, there are 3 components (R, G and B), then NUM_OF_COMPONENT_MINUS1 = 2.<br>In YUV422 or YUV420 or LUV420 or LUV420 case, number of components are 2 (Y and UV/LV), therfore NUM_OF_COMPONENT_MINUS1=1. |

### 54.3.111 ISP Core Image[i] Frame[f] Output DMA Pixel LSB Position Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_PIXEL_POSITION_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | LSB_COMP3 | H'0 | R/W | Bit position of LSB bit of pixel in the output image component3 |
| 23 to 16 | LSB_COMP2 | H'0 | R/W | Bit position of LSB bit of pixel in the output image component2 |
| 15 to 8 | LSB_COMP1 | H'0 | R/W | Bit position of LSB bit of pixel in the output image component1 |
| 7 to 0 | LSB_COMP0 | H'0 | R/W | Bit position of LSB bit of pixel in the output image component0 |

### 54.3.112 ISP Core Image[i] Frame[f] Output DMA Pixel Bitwidth Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_PIXEL_BITWIDTH_MINUS1_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | BW_MINUS1_COMP3 | | | | | | — | — | BW_MINUS1_COMP2 | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | BW_MINUS1_COMP1 | | | | | | — | — | BW_MINUS1_COMP0 | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 29 to 24 | BW_MINUS1_COMP3 | H'0 | R/W | Bit width minus 1 of pixel in the output image component 3 |
| 23, 22 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 21 to 16 | BW_MINUS1_COMP2 | H'0 | R/W | Bit width minus 1 of pixel in the output image component 2 |
| 15, 14 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 13 to 8 | BW_MINUS1_COMP1 | H'0 | R/W | Bit width minus 1 of pixel in the output image component 1 |
| 7, 6 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 5 to 0 | BW_MINUS1_COMP0 | H'0 | R/W | Bit width minus 1 of pixel in the output image component 0 |

### 54.3.113 ISP Core Image[i] Frame[f] Output DMA Pixel BPP register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_PIXEL_BPP_[c])

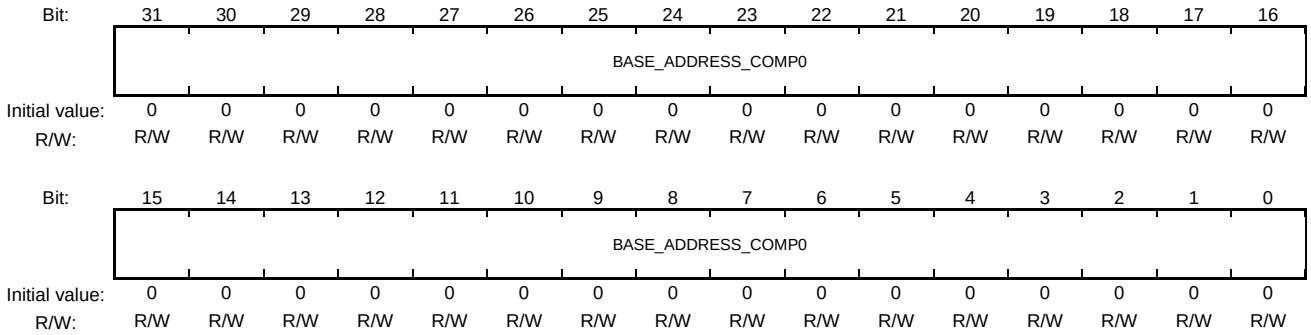
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | LINE_EN_COMP3 | | — | — | BPP_COMP3 | | — | — | LINE_EN_COMP2 | | — | — | BPP_COMP2 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R/W | R/W | R | R | R/W | R/W | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | LINE_EN_COMP1 | | — | — | BPP_COMP1 | | — | — | LINE_EN_COMP0 | | — | — | BPP_COMP0 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R/W | R/W | R | R | R/W | R/W | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 29, 28 | LINE_EN_COMP3 | H'0 | R/W | Flags for ON/OFF control of output for each even/odd line of a frame for the output image component 3.<br>This flag is used for chrominance component in case of YUV420 or LUV420.<br>0 : All lines are not output to AXI<br>1: Only the even-numbered lines are output to AXI<br>2: Only the odd-numbered lines are output to AXI.<br>3 : All lines are output to AXI |
| 27, 26 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 25, 24 | BPP_COMP3 | H'0 | R/W | Bit per pixel of output image component 3<br>0 : 8bpp, 1: 16bpp, 2: 32bpp, 3: Forbidden |
| 23, 22 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 21, 20 | LINE_EN_COMP2 | H'0 | R/W | Flags for ON/OFF control of output for each even/odd line of a frame for the output image component 2.<br>This flag is used for chrominance component in case of YUV420 or LUV420.<br>0 : All lines are not output to AXI<br>1: Only the even-numbered lines are output to AXI<br>2: Only the odd-numbered lines are output to AXI.<br>3 : All lines are output to AXI |
| 19, 18 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 17, 16 | BPP_COMP2 | H'0 | R/W | Bit per pixel of output image component 2<br>0 : 8bpp, 1: 16bpp, 2: 32bpp, 3: Forbidden |
| 15, 14 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 13, 12 | LINE_EN_COMP1 | H'0 | R/W | Flags for ON/OFF control of output for each even/odd line of a frame for the output image component 1.<br>This flag is used for chrominance component in case of YUV420 or LUV420.<br>0 : All lines are not output to AXI<br>1: Only the even-numbered lines are output to AXI<br>2: Only the odd-numbered lines are output to AXI.<br>3 : All lines are output to AXI |
| 11, 10 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 9, 8 | BPP_COMP1 | H'0 | R/W | Bit per pixel of output image component 1<br>0 : 8bpp, 1: 16bpp, 2: 32bpp, 3: Forbidden |
| 7, 6 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 5, 4 | LINE_EN_COMP0 | H'0 | R/W | Flags for ON/OFF control of output for each even/odd line of a frame for the output image component 0.<br>This flag is used for chrominance component in case of YUV420 or LUV420.<br>0 : All lines are not output to AXI<br>1: Only the even-numbered lines are output to AXI<br>2: Only the odd-numbered lines are output to AXI.<br>3 : All lines are output to AXI |
| 3, 2 | — | All 0 | R | Reserved. (These bits are always read as 0.) |

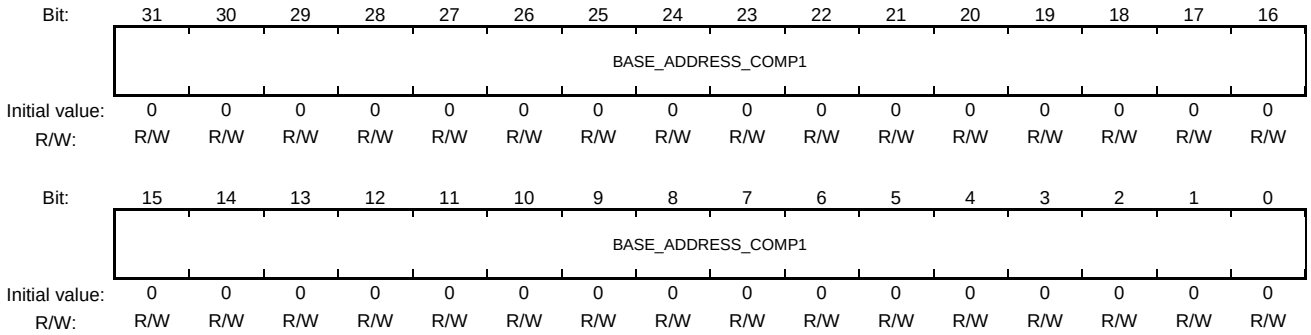
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1, 0 | BPP_COMP0 | H'0 | R/W | Bit per pixel of output image component 0<br>0 : 8bpp, 1: 16bpp, 2: 32bpp, 3: Forbidden |

### 54.3.114 ISP Core Image[i] Frame[f] Output DMA Base Address component0 Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_BASE_ADDRESS_COMP0_[c])



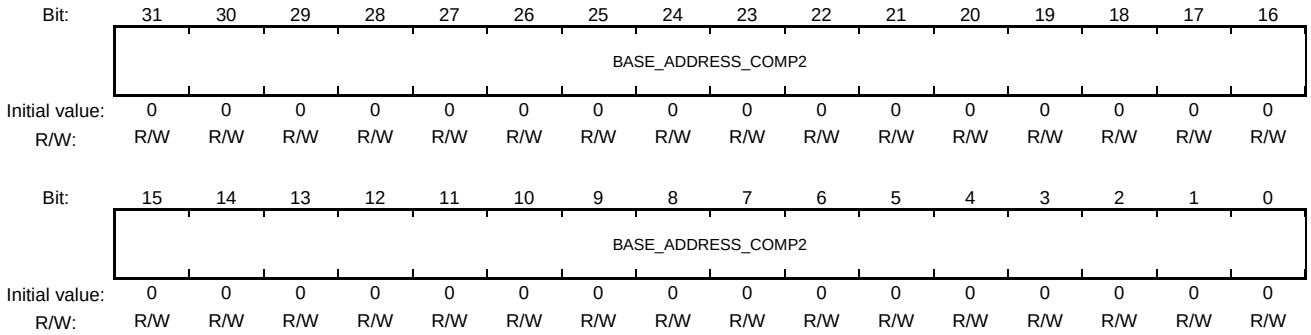

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_COMP0 | H'0 | R/W | Start address ot output image component 0 |

### 54.3.115 ISP Core Image[i] Frame[f] Output DMA Base Address component1 Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_BASE_ADDRESS_COMP1_[c])



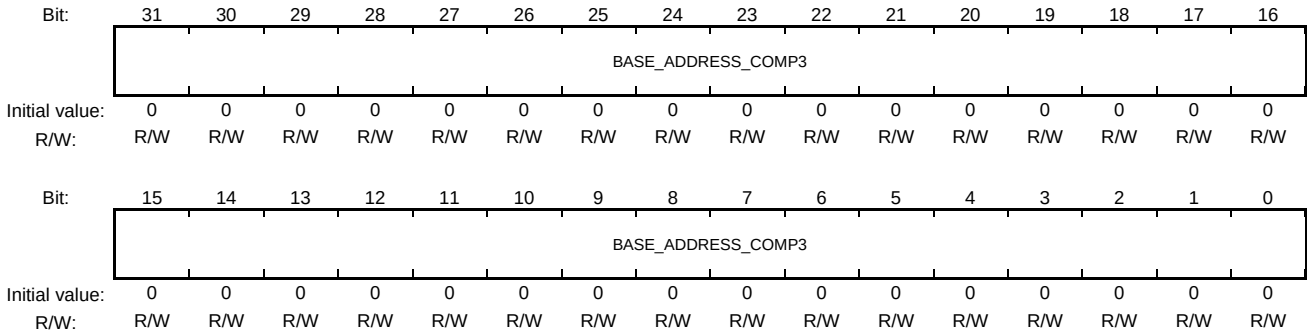

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_COMP1 | H'0 | R/W | Start address ot output image component 1 |

### 54.3.116 ISP Core Image[i] Frame[f] Output DMA Base Address component2 Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_BASE_ADDRESS_COMP2_[c])



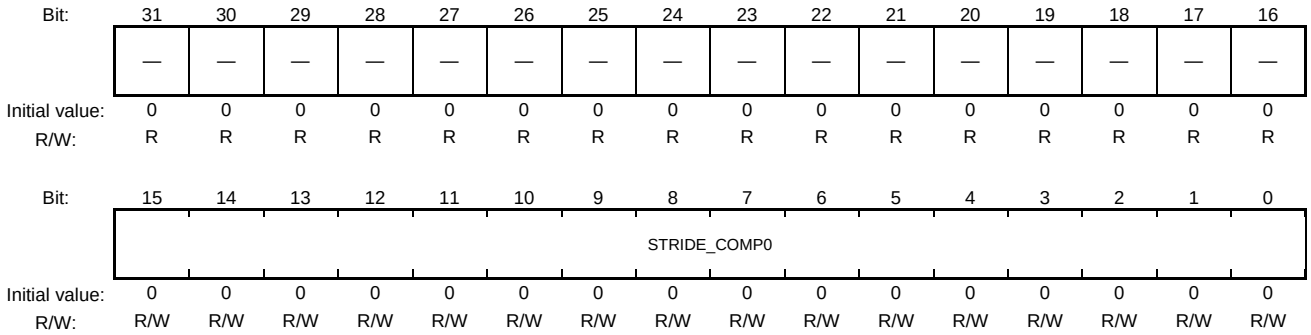

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_COMP2 | H'0 | R/W | Start address ot output image component 2 |

### 54.3.117 ISP Core Image[i] Frame[f] Output DMA Base Address component3 Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_BASE_ADDRESS_COMP3_[c])



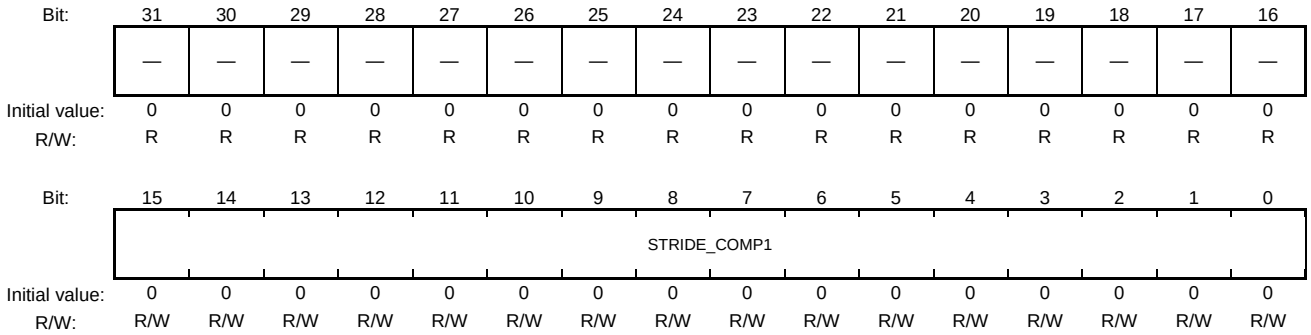

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_COMP3 | H'0 | R/W | Start address ot output image component 3 |

### 54.3.118 ISP Core Image[i] Frame[f] Output DMA Stride component0 Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_STRIDE_COMP0_[c])



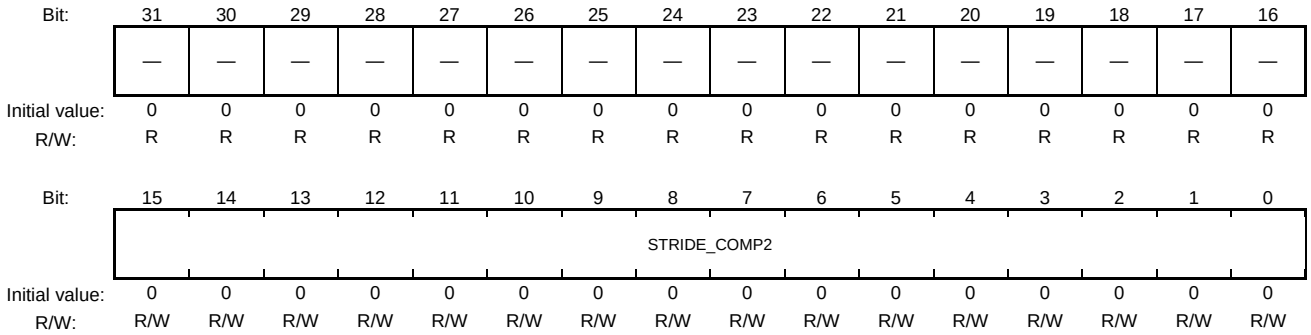

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | STRIDE_COMP0 | H'0 | R/W | Stride (line offset) of output image component 0. Unit is byte |

### 54.3.119 ISP Core Image[i] Frame[f] Output DMA Stride component1 Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_STRIDE_COMP1_[c])



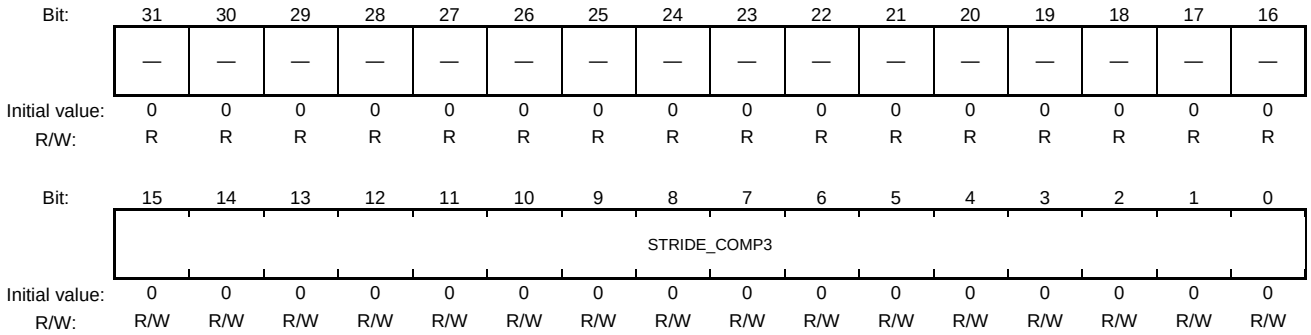

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | STRIDE_COMP1 | H'0 | R/W | Stride (line offset) of output image component 1. Unit is byte |

### 54.3.120 ISP Core Image[i] Frame[f] Output DMA Stride component2 Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_STRIDE_COMP2_[c])



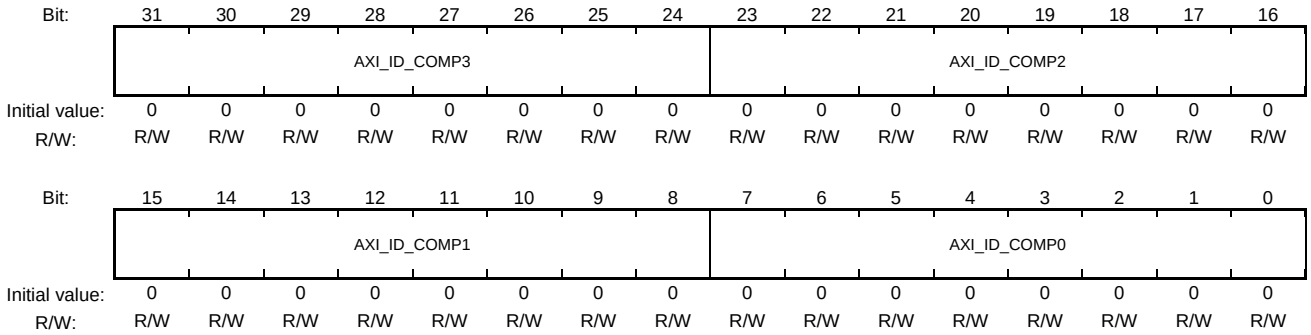

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | STRIDE_COMP2 | H'0 | R/W | Stride (line offset) of output image component 2. Unit is byte |

### 54.3.121 ISP Core Image[i] Frame[f] Output DMA Stride component3 Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_STRIDE_COMP3_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 15 to 0 | STRIDE_COMP3 | H'0 | R/W | Stride (line offset) of output image component 3. Unit is byte |

### 54.3.122 ISP Core Image[i] Frame[f] Output DMA AXI ID Register [c](i=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME[f]_AXI_ID_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | AXI_ID_COMP3 | H'0 | R/W | AXI transaction ID of output image component 3 |
| 23 to 16 | AXI_ID_COMP2 | H'0 | R/W | AXI transaction ID of output image component 2 |
| 15 to 8 | AXI_ID_COMP1 | H'0 | R/W | AXI transaction ID of output image component 1 |
| 7 to 0 | AXI_ID_COMP0 | H'0 | R/W | AXI transaction ID of output image component 0 |

### 54.3.123 ISP Core Image[i] Output DMA AXI Config Register [c](i=0-1)(c=0-0)(ISPCORE_DMA_IMAGE[i]_AXI_CONFIG_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_ENABLE | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | OUTSTANDING | | | | | | — | — | — | — | BURST_LENGTH_MINUS1 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | DMA_ENABLE | H'0 | R/W | Enabling image output DMA<br>0 : Disabled [default]<br>1: Enabled |
| 30 to 14 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 13 to 8 | OUTSTANDING | H'0 | R/W | Maximum number of out standing count of AXI write transaction. Minus 1 value is set to this register |
| 7 to 4 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 3 to 0 | BURST_LENGTH_MINUS1 | H'0 | R/W | Burst length of AXI write data transaction. Minus 1 value is set to this register |

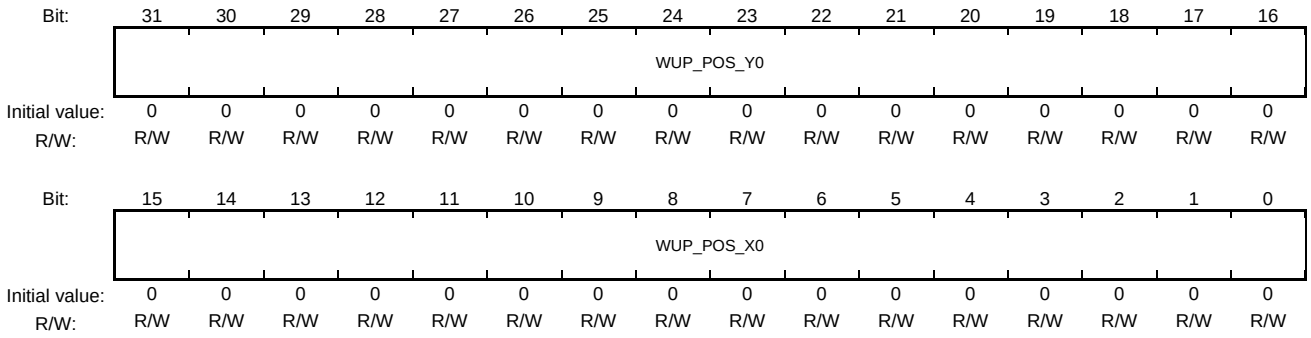
### 54.3.124 ISP Wakeup Config Register [c](c=0-0)(ISPCORE_WUP_CONFIG_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | WUP_PULSE_LENGTH_MINUS1 | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

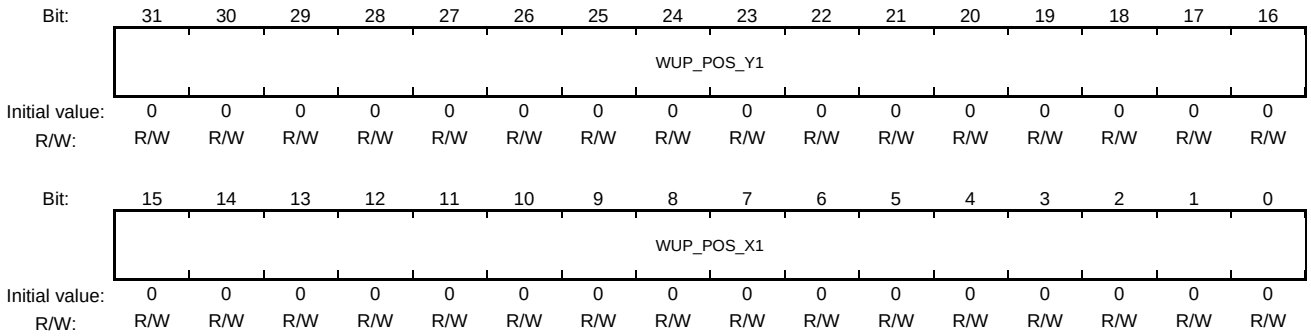
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 26 to 24 | WUP_PULSE_LENGTH_MINUS1 | H'0 | R/W | Pulse length of WUP signal in synchronized system with IMR |
| 23 to 0 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |

### 54.3.125 ISP Wakeup Position0 Register [c](c=0-0)(ISPCORE_WUP_POS0_[c])



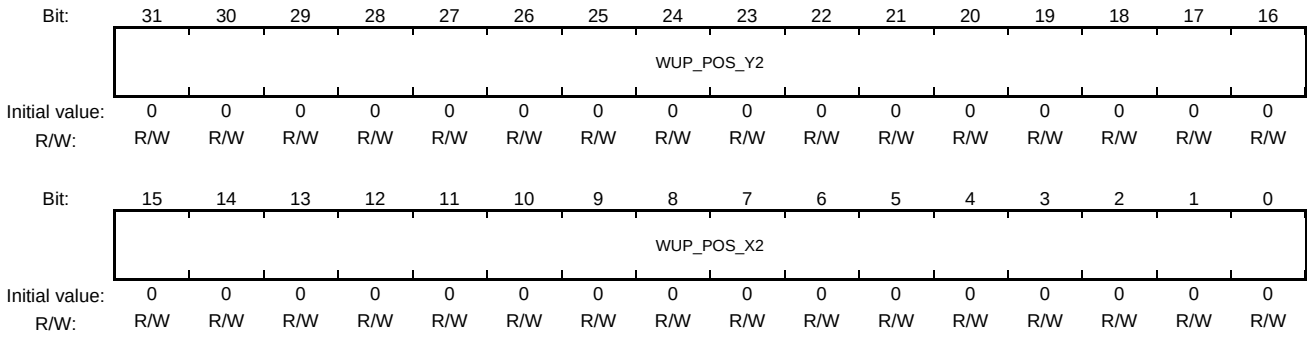

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | WUP_POS_Y0 | H'0 | R/W | Vertical coordinate of pixel which issues the WUP signal |
| 15 to 0 | WUP_POS_X0 | H'0 | R/W | Horizontal coordinate of pixel which issues the WUP signal |

### 54.3.126 ISP Wakeup Position1 Register [c](c=0-0)(ISPCORE_WUP_POS1_[c])



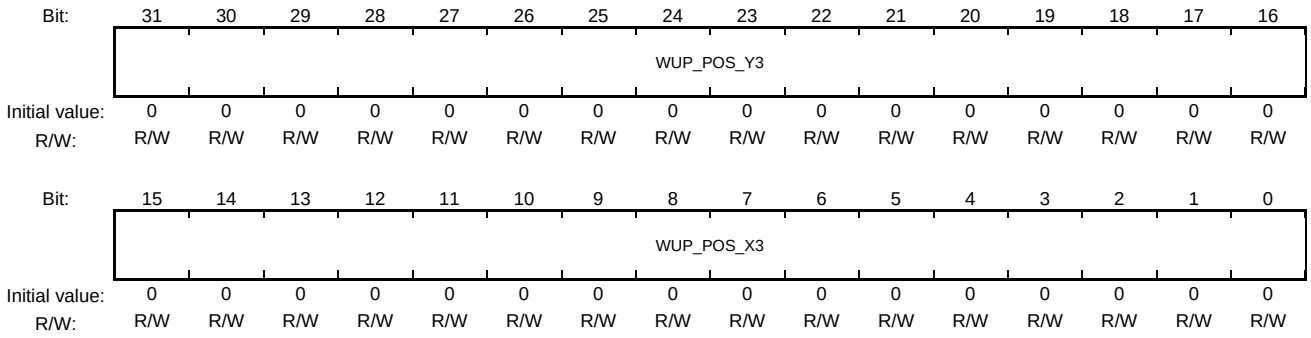

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | WUP_POS_Y1 | H'0 | R/W | Vertical coordinate of pixel which issues the WUP signal |
| 15 to 0 | WUP_POS_X1 | H'0 | R/W | Horizontal coordinate of pixel which issues the WUP signal |

### 54.3.127 ISP Wakeup Position2 Register [c](c=0-0)(ISPCORE_WUP_POS2_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | WUP_POS_Y2 | H'0 | R/W | Vertical coordinate of pixel which issues the WUP signal |
| 15 to 0 | WUP_POS_X2 | H'0 | R/W | Horizontal coordinate of pixel which issues the WUP signal |

### 54.3.128 ISP Wakeup Position3 Register [c](c=0-0)(ISPCORE_WUP_POS3_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | WUP_POS_Y3 | H'0 | R/W | Vertical coordinate of pixel which issues the WUP signal |
| 15 to 0 | WUP_POS_X3 | H'0 | R/W | Horizontal coordinate of pixel which issues the WUP signal |

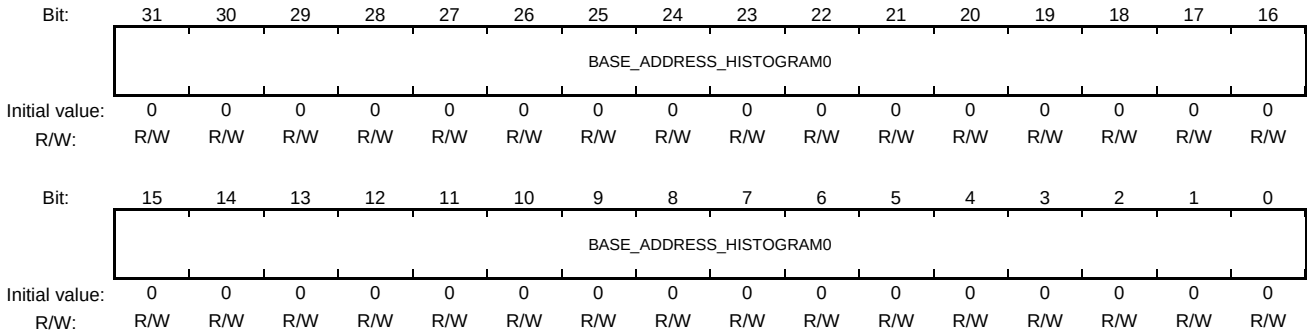
### 54.3.129 ISP Wakeup Enable Register [c](c=0-0)(ISPCORE_WUP_ENABLE_CHANNELS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | WUP_ENABLE_CHANNELS | | | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

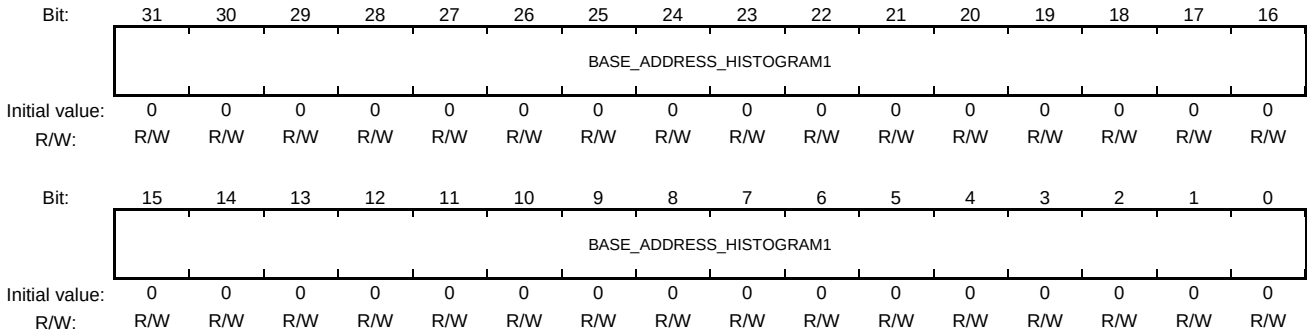
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 13 to 8 | WUP_ENABLE_CHANNELS | H'0 | R/W | Enable control register whether IMR channel receives WUP pulse from ISP. Each bit corresponds to each IMR channel. |
| 7 to 0 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |

### 54.3.130 ISP Core Histogram Output DMA Base Address Frame0 Register [c](c=0-0)(ISPCORE_DMA_HISTOGRAM_BASE_ADDRESS_FRAME0_[c])



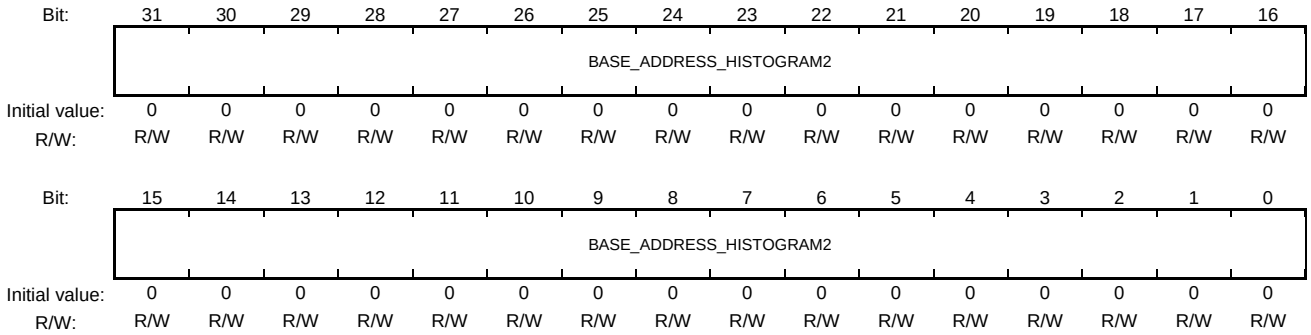

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_HISTOGRAM0 | H'0 | R/W | Start Address of histogram data(frame 0,4,8...) |

### 54.3.131 ISP Core Histogram Output DMA Base Address Frame1 Register [c](c=0-0)(ISPCORE_DMA_HISTOGRAM_BASE_ADDRESS_FRAME1_[c])



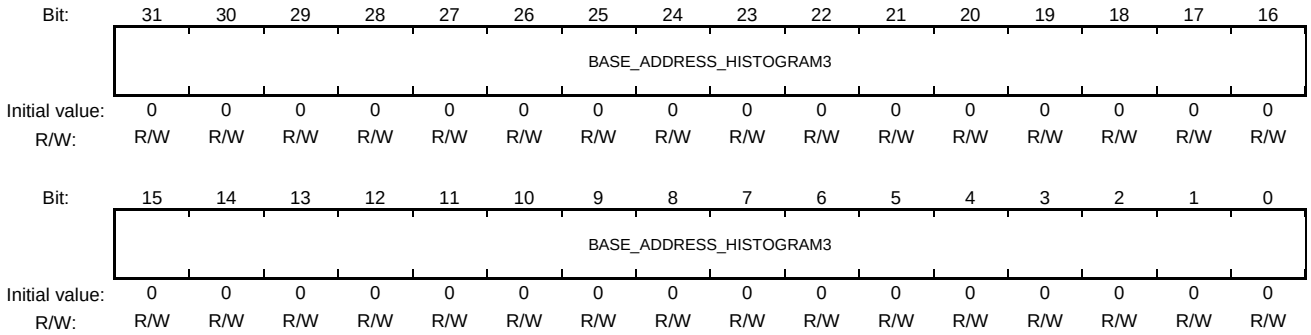

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_HISTOGRAM1 | H'0 | R/W | Start Address of histogram data(frame 1,5,9...) |

### 54.3.132 ISP Core Histogram Output DMA Base Address Frame2 Register [c](c=0-0)(ISPCORE_DMA_HISTOGRAM_BASE_ADDRESS_FRAME2_[c])



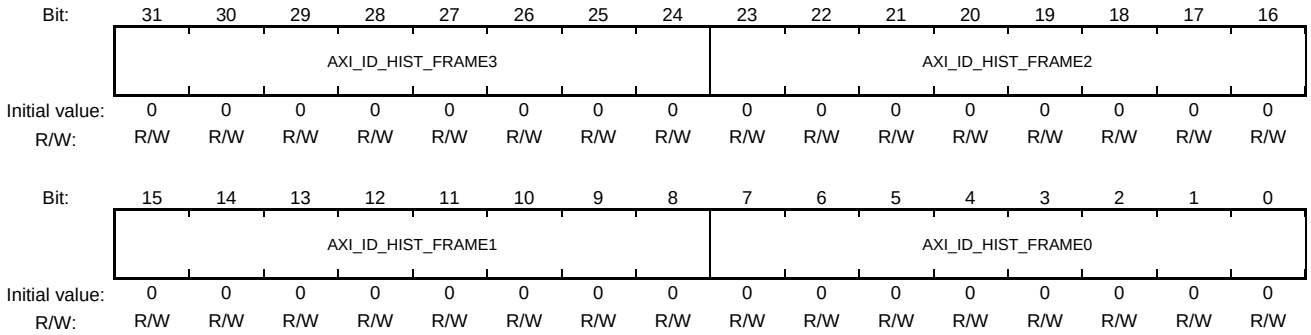

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_HISTOGRAM2 | H'0 | R/W | Start Address of histogram data(frame 2,6,10...) |

### 54.3.133 ISP Core Histogram Output DMA Base Address Frame3 Register [c](c=0-0)(ISPCORE_DMA_HISTOGRAM_BASE_ADDRESS_FRAME3_[c])



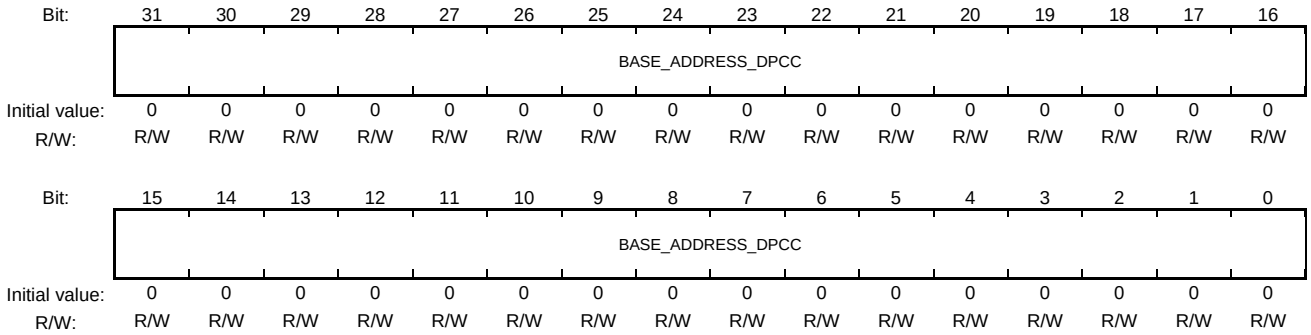

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_HISTOGRAM3 | H'0 | R/W | Start Address of histogram data(frame 3,7,11...) |

### 54.3.134 ISP Core Histogram Output DMA AXI ID Register [c](c=0-0)(ISPCORE_DMA_HISTOGAM_AXI_ID_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | AXI_ID_HIST_FRAME3 | H'0 | R/W | AXI transaction ID of histogram data (frame 3,7,11...) |
| 23 to 16 | AXI_ID_HIST_FRAME2 | H'0 | R/W | AXI transaction ID of histogram data (frame 2,6,10...) |
| 15 to 8 | AXI_ID_HIST_FRAME1 | H'0 | R/W | AXI transaction ID of histogram data (frame 1,5,9...) |
| 7 to 0 | AXI_ID_HIST_FRAME0 | H'0 | R/W | AXI transaction ID of histogram data (frame 0,4,8...) |

### 54.3.135 ISP Core Defect Pixel Information[j] Frame[f] DMA Base Address Register [c](j=0-1)(f=0-3)(c=0-0)(ISPCORE_DMA_DPCC[j]_FRAME[f]_BASE_ADDRESS_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | BASE_ADDRESS_DPCC | H'0 | R/W | Start address of defect pixel information |

### 54.3.136 ISP Core Defect Pixel Information[j] DMA AXI Config Register [c](j=0-1)(c=0-0)(ISPCORE_DMA_DPCC[j]_AXI_CONFIG_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_ENABLE_DPCC | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | BURST_LENGTH_MINUS1_DPCC | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | DMA_ENABLE_DPCC | H'0 | R/W | Enabling defect pixel information DMA<br>0 : Disabled [default]<br>1 : Enabled |
| 30 to 4 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 3 to 0 | BURST_LENGTH_MINUS1_DPCC | H'0 | R/W | Burst length of AXI write data transaction. Minus 1 value is set to this register |

### 54.3.137 ISP Core Interrupt Status [c](c=0-0)(ISPCORE_INT_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | DPCC_PIXEL_COUNT_OVER_DPCC2 | DPCC_PIXEL_COUNT_OVER_DPCC1 | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ST0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 19 | DPCC_PIXEL_COUNT_OVER_DPCC2 | H'0 | R | Asserted when number of detected defect pixel count is over than upper limitation specified in ISPCORE_DMA_DPCC[1]_PIXEL_COUNT_MAX register |
| 18 | DPCC_PIXEL_COUNT_OVER_DPCC1 | H'0 | R | Asserted when number of detected defect pixel count is over than upper limitation specified in ISPCORE_DMA_DPCC[0]_PIXEL_COUNT_MAX register |
| 17 to 1 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 0 | ST0 | H'0 | R | Status of interrupt request signal from ISP core<br>0 : Interrupt request is not issued from ISP core<br>1 : Interrupt request is issued from ISP core |

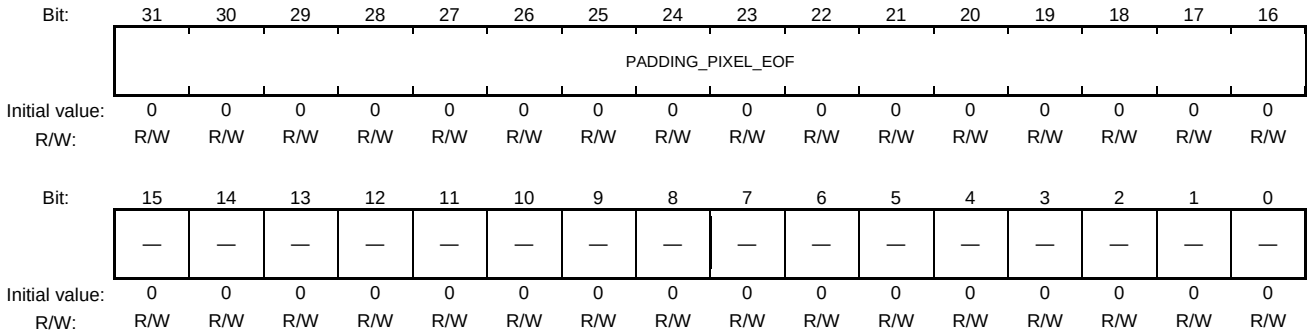
### 54.3.138 ISP Core Interrupt Enable [c](c=0-0)(ISPCORE_INT_ENABLE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | DPCC_PIXEL_COUNT_OVER_DPCC2 | DPCC_PIXEL_COUNT_OVER_DPCC1 | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EN0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 19 | DPCC_PIXEL_COUNT_OVER_DPCC2 | H'0 | R/W | Interrupt enable of ISPCORE_INT_STATUS[19] |
| 18 | DPCC_PIXEL_COUNT_OVER_DPCC1 | H'0 | R/W | Interrupt enable of ISPCORE_INT_STATUS[18] |
| 17 to 1 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 0 | EN0 | H'0 | R/W | Enable bit of interrupt request signal output from ISP core.<br>0 : Interrupt is not notified to INTC [default]<br>1: Interrupt is notified to INTC |

### 54.3.139 ISP Core Image[i] Output DMA Flushing Out Register [c](i=0-1)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FLUSH_OUT_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | PADDING_PIXEL_EOF | H'0 | R/W | Number of pixels to be appended in the end of frame for flushing out the remained data of ISP output image in DMA pipeline. |
| 15 to 0 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |

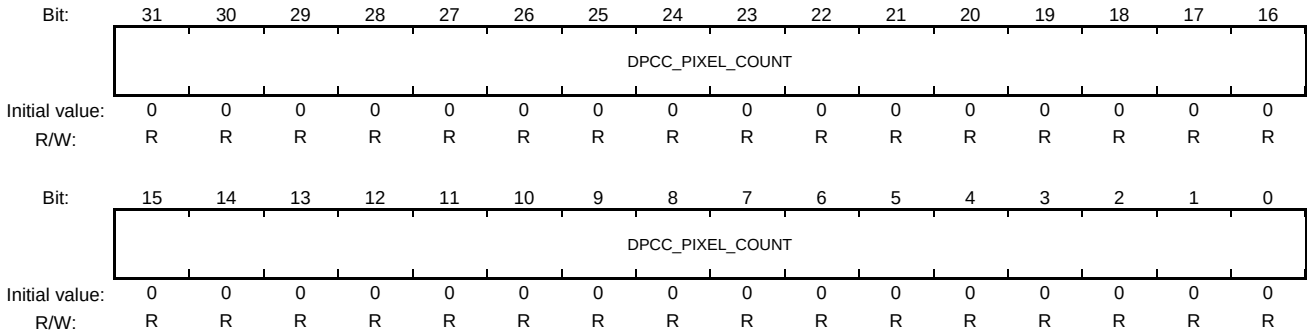
### 54.3.140 ISP Core Back Pressure Setting [c](c=0-0)(ISPCORE_BACK_PRESSURE_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | BP_THRESHOLD | | | | — | — | — | — | — | — | — | BP_STOP_MODE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

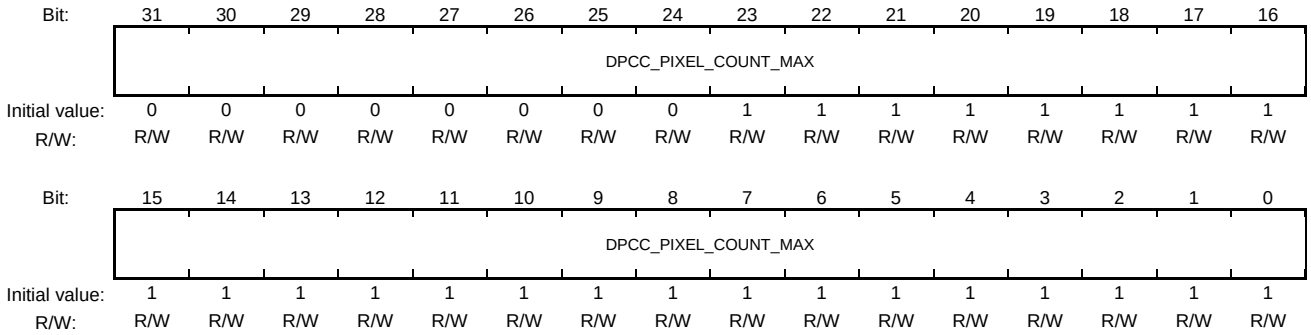
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 11 to 8 | BP_THRESHOLD | H'1 | R/W | Watermak level of stopping threshold for back pressure stopping. |
| 7 to 1 | — | H'0 | R/W | Reserved bits. Do not write any different value from its initial value. |
| 0 | BP_STOP_MODE | H'1 | R/W | Back pressure stopping mode. 0: Back pressure stopping is disabled. 1: Back pressure stopping is enabled(default) |

### 54.3.141 ISP Core Defect Pixel Information[j]. Monitor of the number of detected defect pixel count in previous frame [c](j=0-1)(c=0-0)(ISPCORE_DMA_DPCC[j]_PIXEL_COUNT_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DPCC_PIXEL_COUNT | H'0 | R | Monitor of the number of detected defect pixel count in previous frame. This value includes the number of one word of the end code (H'FFFFFFFF) |

### 54.3.142 ISP Core Defect Pixel Information[j]. Upper limit of number of defect pixel count [c](j=0-1)(c=0-0)(ISPCORE_DMA_DPCC[j]_PIXEL_COUNT_MAX_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DPCC_PIXEL_COUNT_MAX | H'FFFFFFF | R/W | Upper limit of number of defect pixel count |

### 54.3.143 Status of Defect Pixel Information[j] DMA [c](j=0-1)(c=0-0)(ISPCORE_DMA_DPCC[j]_STATUS_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DPCC_COUNT_OVER |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 16 | DPCC_COUNT_OVER | H'0 | R/W | Asserted when number of detected defect pixel count is over than upper limitation specified in ISPCORE_DMA_DPCC[j]_PIXEL_COUNT_MAX register. By writing 0 to this register the status bit ISPCORE_INT_STATUS.DPCC_PIXEL_COUNT_OVER_DPCC[j-1] is cleared. |
| 15 to 0 | — | All 0 | R | Reserved. (These bits are always read as 0.) |

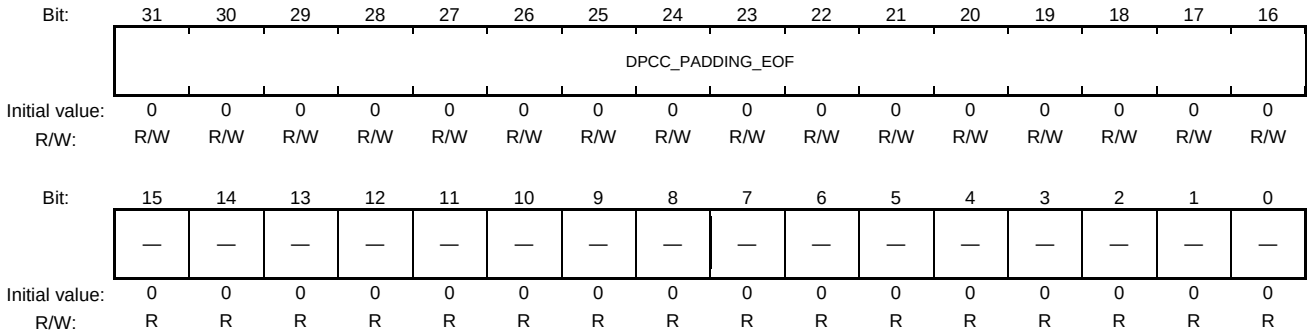
### 54.3.144 ISP Core Image[i] Frame Monitor Register [c](i=0-1)(c=0-0)(ISPCORE_DMA_IMAGE[i]_FRAME_MONITOR_[c])

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | FRAME_COUNT | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved. (These bits are always read as 0.) |
| 3 to 0 | FRAME_COUNT | H'0 | R | Frame count value. Incremented at frame by frame. Wrap around to H'0 after H'F |

### 54.3.145 ISP Core Defect Pixel Information[j] DMA Flushing Out Register [c](j=0-1)(c=0-0)(ISPCORE_DMA_DPCC[j]_FLUSH_OUT_[c])




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | DPCC_PADDING_EOF | H'0 | R/W | Number of words to be appended in the end of frame for flushing out the remained data of defect pixel information. |
| 15 to 0 | — | All 0 | R | Reserved. (These bits are always read as 0.) |

## 54.4 Operation

### 54.4.1 Configuration (ISP Core)

Please refer to the hardware manual of ISP core.

### 54.4.2 Configuration (ISP Interface/Channel Selector)

Register setting for Pixel Reconstructor and Channel Selector is needed to make a though pass from CSI input to the ISP core. This section shows reference register setting for each recommended sensor.

#### 54.4.2.1 Architecture of Channel Selector

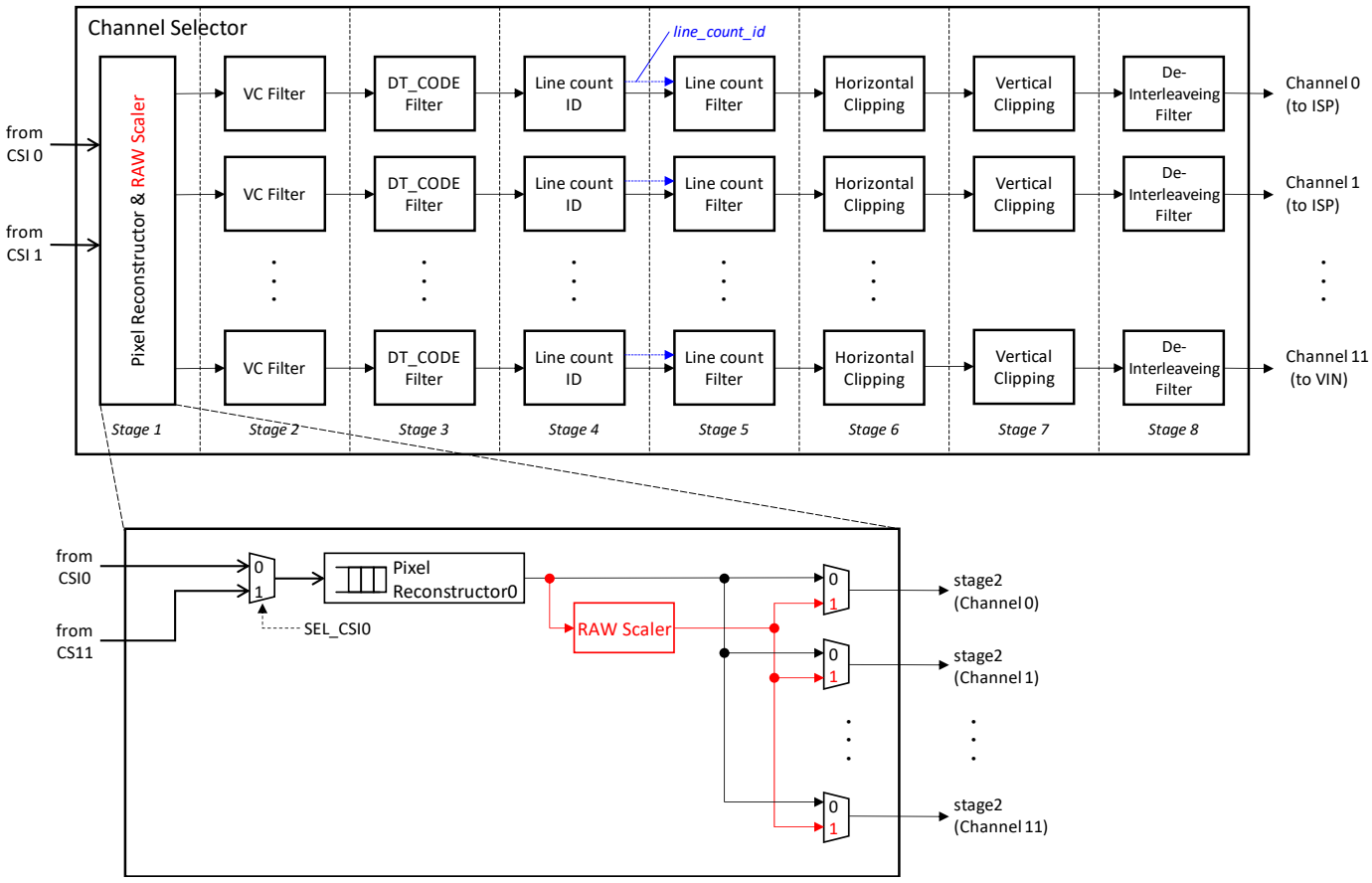



**Figure 54.5 Architecture of the Channel selector**

#### 1. Stage 1: Pixel reconstructor

#### (a) Operations for MIPI CSI-2 based data types

In this stage, packed pixel coming from CSI2 is converted to "1 pixel per clock format". The incoming pixel is packed according to the specification of MIPI CSI-2 standard. Pixel Reconstructor decodes this packed pixel, and converts it to the 1 pixel per one cycle format (This process is called as "pixel unpacking"). Pixel reconstructor has one FIFO (CSI FIFO) for pixel unpacking. If CSI FIFO overflows then FIFO_FULL_CSI_INPUT interrupt is asserted and it is reflected to the ISPINT_STATUS register.

There are several modes of the pixel unpacking shown in Table 54.1. Setting value from 0 to 6 is RAW format, and 7 is embedded data (EMB8). Setting value from 10 to 15 is YUV format. Setting value 20 and 21 is RGB format. Setting value from 56 to 63 is custom format. In this case, unpacking rule is specified by LUT (detail is shown in 54.4.2.3) .

Table 54.1 Meanings of PROC_MODE_VCn register

| Setting Value | Meanings |
|---|---|
| 0 | RAW8 |
| 1 | RAW10 |
| 2 | RAW12 |
| 3 | RAW14 |
| 4 | RAW16 |
| 5 | RAW20 |
| 6 | RAW24 |
| 7 | EMB8 |
| 10 | YUV420 8-bit |
| 11 | YUV420 10-bit |
| 12 | YUV422 8-bit |
| 13 | YUV422 10-bit |
| 14 | YUV422 12-bit |
| 15 | YUV422 Y12-bit, UV 8-bit |
| 20 | RGB565 |
| 21 | RGB888 |
| 56~63 | Custom format for LUT based pixel reconstructor |
| other | Forbidden |

Decision of unpacking mode is based on ISPPROCMODE*n* (*n*=0~63) register. The index *n* is the 6-bit value called as "data_type" or "DT_code" in the MIPI CSI standard, which is stored in the packet header of incoming CSI-2 long packet. Its concrete decision flow as shown as below.

switch (*vc_number %4*) {
  case 0 : *proc_mode* = reg_ISPPROCMODE_DT*n*.PROC_MODE_VC0; break;
  case 1 : *proc_mode* = reg_ISPPROCMODE_DT*n*.PROC_MODE_VC1 ; break;
  case 2 : *proc_mode* = reg_ISPPROCMODE_DT*n*.PROC_MODE_VC2; break;
  case 3 : *proc_mode* = reg_ISPPROCMODE_DT*n*.PROC_MODE_VC3; break;
}

switch (*proc_mode*) {
  0 : Incoming CSI-2 pixel is unpacked by RAW8 format.
  1 : Incoming CSI-2 pixel is unpacked by RAW10 format.
  2 : Incoming CSI-2 pixel is unpacked by RAW12 format.
  3 : Incoming CSI-2 pixel is unpacked by RAW14 format.
  4 : Incoming CSI-2 pixel is unpacked by RAW16 format.
  5 : Incoming CSI-2 pixel is unpacked by RAW20 format.
  6 : Incoming CSI-2 pixel is unpacked by RAW24 format.
  7 : Incoming CSI-2 pixel is unpacked by EMB8 format.
 10: Incoming CSI-s pixel is unpacked by YUV420 8-bit
 11: Incoming CSI-s pixel is unpacked by YUV420 10-bit
 12: Incoming CSI-s pixel is unpacked by YUV422 8-bit
 13: Incoming CSI-s pixel is unpacked by YUV422 10-bit
 14: Incoming CSI-s pixel is unpacked by YUV422 12-bit
 15: Incoming CSI-s pixel is unpacked by YUV422 Y 12-bit, UV 8-bit
 20: Incoming CSI-s pixel is unpacked by RGB565
 21: Incoming CSI-s pixel is unpacked by RGB888
 56~63: Incoming CSI-s pixel is unpacked by YUV420 10-bit
 default : Incoming CSI-2 pixel is discarded (No output)
}

[Defines of variables]
- reg_ISPPROCMODE_DTn : Value of ISPPROCMODE_DT*n*.VC*m*(*n*=0~63, *m*=0~3)
- *vc_number* : Virtual channel of the input CSI packet

Example of ISPPROCMODE_DT*n* registers is shown in Table 54.2

**Table 54.2 Example of ISPPROCMODE register.**

| Virtual Channel | DT_code | Format | Register setting |
|---|---|---|---|
| VC0 | H'12 | EMB8 | ISPPROCMODE_DT18.VC0 = 7 (EMB8) |
| VC0 | H'2C | RAW12 | ISPPROCMODE_DT44.VC0 = 2 (RAW12) |
| VC0 | H'2F | RAW20 | ISPPROCMODE_DT47.VC0 = 5 (RAW20) |
| VC1 | H'2D | RAW14 | ISPPROCMODE_DT45.VC1 = 3 (RAW14) |
| VC1 | H'34 | RAW10 | ISPPROCMODE_DT52.VC1 = 1 (RAW10) |
| VC1 | H'35 | RAW10 | ISPPROCMODE_DT53.VC1 = 1 (RAW10) |
| VC2 | H'2B | RAW16 | ISPPROCMODE_DT43.VC2 = 4 (RAW16) |
| VC2 | H'2C | RAW12 | ISPPROCMODE_DT44.VC2 = 2 (RAW12) |
| VC3 | H'2B | RAW8 | ISPPROCMODE_DT43.VC3 = 0 (RAW8) |

### (b) Operations for custom data types (LUT based pixel reconstructor)

By setting pixel format as 0x30～0x3F, custom data types. Custom data type is set by using ISPPROC_CUSTOM_FORMAT_LUT0_* register (LUT register for custom data types) and ISPCS_DI_FILTER_LUT* register (de-interleaving control register). See the 54.4.2.3 for detail

### 2. Stage 2: VC filter

In this stage, filtering using virtual channel as identifier is performed. [d] means the index of channels (d=0~11).

- Related register
  - ISPCS_FILTER_ID_CH[d].VC_EN (d=0~11)
- Filtering operation: Following process flow is performed for each long packet.

```
if (reg_VC_EN[vc_number] == 1) {
    Passed
} else {
    Discarded
}

[Defines of variables]
・ reg_VC_EN : Value of ISPCS_FILTER_ID_CH[d] register
・ vc_number : Virtual channel of the input CSI packet
```

### 3. Stage 3: DT code filter

In this stage, filtering using incoming dt_code is performed.

- Related register
  - ISPCS_ DT_CODE03_CH[d].DT_CODE*m* ( *d*=0~11, *m*=0~3)
  - ISPCS_ DT_CODE47_CH[d].DT_CODE*m* ( *d*=0~11, *m*=4~7)
  - ISPCS_DT_CODE03_CH[d].ENABLE*m* ( *d*=0~11, *m*=0~3)
  - ISPCS_DT_CODE47_CH[d].ENABLE*m* ( *d*=0~11, *m*=4~7)

- Processing description: Following process flow is performed for each CSI-2 long packet.

```
if ( (reg_ENABLE0==1 && reg_DT_CODE0==dt_code ) || (reg_ENABLE0==1 && reg_DT_CODE1==dt_code) ||
     (reg_ENABLE2==1 && reg_DT_CODE2==dt_code ) || (reg_ENABLE3==1 && reg_DT_CODE3==dt_code) ||
     (reg_ENABLE4==1 && reg_DT_CODE4==dt_code ) || (reg_ENABLE5==1 && reg_DT_CODE5==dt_code) ||
     (reg_ENABLE6==1 && reg_DT_CODE6==dt_code ) || (reg_ENABLE7==1 && reg_DT_CODE7==dt_code) )
{
     Passed
} else {
     Discarded
}
```

[Defines of variables]
- reg_DT_CODE*m* (*m*=0~3) : Value of ISPCS_DT_CODE03_CH[d].DT_CODE*m*
- reg_DT_CODE*m* (*m*=4~7) : Value of ISPCS_DT_CODE47_CH[d].DT_CODE*m*
- reg_ENABLE*m* (*m*=0~3) : Value of ISPCS_DT_CODE03_CH[d].ENABLE*m*
- reg_ENABLE*m* (*m*=4~7) : Value of ISPCS_DT_CODE47_CH[d].ENABLE*m*
- *dt_code* : Value of the "Data Type" code of input CSI packet

### 4. Stage 4: Line count ID

In this stage, the "Line count ID" is added to for each line in one frame by counting number of long packet of incoming CSI input.

- Related registers
  - ISPCS_LC_MODULO_CH[d]. MODULO_MINUS1_VC*n* (*d*=0~11, *n*=0~15)
  - ISPCS_DT_CODE03_CH[d].HOLD*m* ( *d*=0~11, *m*=0~3)
  - ISPCS_DT_CODE47_CH[d].HOLD*m* ( *d*=0~11, *m*=4~7)
- Processing description: Following process flow is performed for each CSI-2 long packet.

```
n = vc_number;
modulo_minus1= reg_MODULO_MINUS1_VCn;

if ( dt_code == H'00 )  {      // FS packet at the start of one frame
       line_count(vc_number) = 0 ;
}

line_count_id = line_count(vc_number) % (modulo_minus1 + 1);

if (reg_HOLDk ==0) {
     line_count(vc_number)++ ;
}
```

[Defines of variables]
- reg_MODULO_MINUS1_VC*n* (*n*=0~15): Value of ISPCS_ MODULO_CH[d].MODULO_MINUS1_VC*n*
- reg_HOLD*k* (*k*=0~3) : Value of ISPCS_DT_CODE03_CH[d]. HOLD*k*
- reg_HOLD*k* (*k*=4~7) : Value of ISPCS_DT_CODE47_CH[d]. HOLD*k*
- *vc_number* : Virtual channel of the input CSI packet
- *dt_code* : Value of the "Data Type" code of input CSI packet
- *line_count(vc_number)* : Value of line count for each virtual channel.
- *line_count_id* : Line count ID (output of this stage)

### 5. Stage 5: Line count filter

In this stage, filtering using the “Line count ID” obtained at stage 3 as identifier is performed.

- Related registers
  - ISPCS_FILTER_ID_CH[d].LC_FILTER_ID (d=0~11)
  - ISPCS_DT_CODE03_CH[d].DT_CODE*m* (d=0~11, *m*=0~3)
  - ISPCS_DT_CODE47_CH[d].DT_CODE*m* (d=0~11, *m*=4~7)
  - ISPCS_DT_CODE03_CH[d].ENABLE*m* ( *n*=0~11, *m*=0~3)
  - ISPCS_DT_CODE47_CH[d].ENABLE*m* ( *n*=0~11, *m*=4~7)
- Processing description: Following process flow is performed for each CSI-2 long packet.

```
switch (dt_code) {
   case reg_DT_CODE0: k = 0; break;
   case reg_DT_CODE1: k = 1; break;
   case reg_DT_CODE2: k = 2; break;
   case reg_DT_CODE3: k = 3; break;
   case reg_DT_CODE4: k = 4; break;
   case reg_DT_CODE5: k = 5; break;
   case reg_DT_CODE6: k = 6; break;
   case reg_DT_CODE7: k = 7; break;
   default: Not happen because of DT code filter operation at stage 3
}

if ( reg_LC_FILTER_ID == line_count_id(vc_number)  && reg_ENABLEk == 1) {
     Passed
} else {
     Discarded
}

[Defines of variables]
・   reg_LC_FILTER_ID       : Value of ISPCS_FILTER_ID CH[d].LC_FILTER_ID
・   reg_DT_CODEm  (m=0~3) :  Value of ISPCS_DT_CODE03_CH[d].DT_CODEm
・   reg_DT_CODEm  (m=4~7) :  Value of ISPCS_DT_CODE47_CH[d].DT_CODEm
・   reg_ENABLEk  (k=0~3) :  Value of ISPCS_DT_CODE03_CH[d].ENABLEk
・   reg_ENABLEk  (k=4~7) :  Value of ISPCS_DT_CODE47_CH[d].ENABLEk
・   vc_number : Virtual channel of the input CSI packet
・   line_count_id : Value of the line count id added at the stage 3
```

### 6. Stage 6: Horizontal clipping

In this stage, pixel based clipping in one line is executed. Clipping operation is performed in accordance with the following registers. Index ‘*m*’ is decided by comparing DT_code in incoming CSI packet with ISPCS_DT_CODE03_CH[d].DT_CODE*m*.

- Related registers
  - ISPCS_DT_CODE03_CH[d].DT_CODE*m* (d=0~11, *m*=0~3)
  - ISPCS_H_CLIP_DT_CODE[*h*]_CH[d].APPENDED_PIXEL (*h*=0~3, d=0~11)

- ISPCS_H_CLIP_DT_CODE[*h*]_CH[d].PREPENDED_PIXEL (*h*=0~3, *d*=0~11)

- Processing description: Following process flow is executed for each pixel.

```
switch (dt_code ) {
   case reg_DT_CODE0 : h  = 0; break;
   case reg_DT_CODE1 : h  = 1; break;
   case reg_DT_CODE2 : h  = 2; break;
   case reg_DT_CODE3 : h  = 3; break;
   default : v  = -1; break;
}

if (h < 0) {
    Passed
} else {
    app_pixel = reg_APPENDED_PIXELh ;
    prep_pixel = reg_PREPENDED_PIXELh ;

    if (pixel_count >= app_pixel  && pixel_count < prep_pixel ) {
        Current pixel is passed
    } else {
        Current pixel is discarded
    }
}

pixel_count ++;
```

[Defines of variables]
- reg_DT_CODE*m* : Value of ISPCS_DT_CODE03_CH[d].DT_CODE*m*
- reg_APPENDED_PIXEL*h* : Value of ISPCS_H_CLIP_DT_CODE[*h*]_CH[d].APPENDED_PIXEL
- reg_PREPENDED_PIXEL*h* : Value of ISPCS_H_CLIP_DT_CODE[*h*]_CH[d].PREPENDED_PIXEL
- *dt_code* : Value of the "Data Type" code of input CSI packet

Example of setting value of APPENDED_PIXEL and PREPENDED_PIXEL is shown in the Figure 54.6

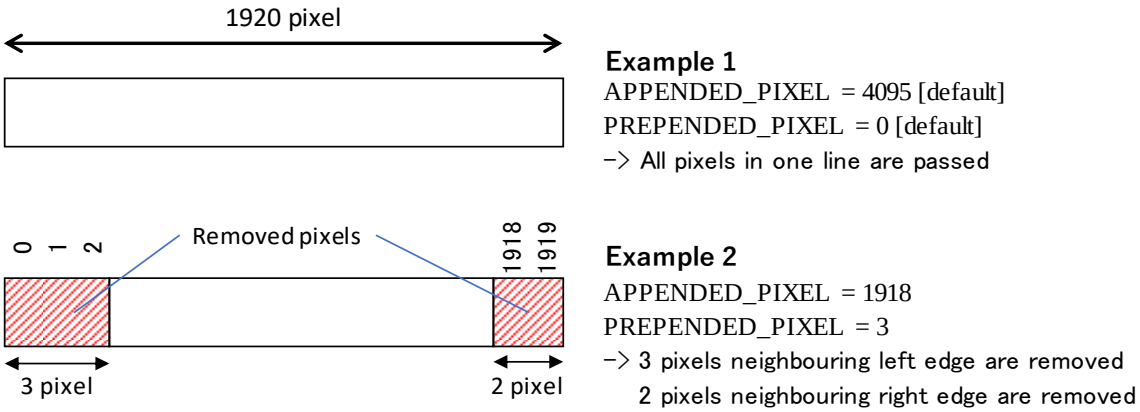



**Figure 54.6 Example of horizontal clipping**

### 7. Stage 7: Vertical clipping

In this stage, line clipping is executed. Clipping operation is performed in accordance with the following register. Index '*m*' is decided by comparing the input DT_code with ISPCS_DT_CODE03_CH[d].DT_CODE*m*.

- Related registers
  - ISPCS_DT_CODE03_CH[d].DT_CODE*m* (*d*=0~11, *m*=0~3)
  - ISPCS_ V_CLIP_DT_CODE[*v*]_CH[d].APPENDED_LINE (*v*=0~3, *n*=0~11)
  - ISPCS_ V_CLIP_DT_CODE[*v*]_CH[d].PREPENDDED_LINE (*v*=0~3, *n*=0~11)
- Processing description: Following processing is executed for each CSI-2 long packet.

```
if ( dt_code == 'H00 )  {      // FS packet at the start of one frame
        line_count = 0 ;
}

switch (dt_code ) {
   case reg_DT_CODE0 : v   = 0; break;
   case reg_DT_CODE1 : v  = 1; break;
   case reg_DT_CODE2 : v   = 2; break;
   case reg_DT_CODE3 : v   = 3; break;
   default : v  = -1; break;
}

if (v  < 0) {
    Passed
} else {
     app_line = reg_APPENDED_LINEv  ;
     prep_line = reg_PREPENDED_LINEv  ;

     if (line_count >= app_line = && line_count < prep_line ) {
          Current line is passed
     else {
          Current line is discarded
     }
}

line_count ++;
```

[Defines of variables]
- reg_DT_CODE*m* : Value of ISPCS_DT_CODE03_CH[d].DT_CODE*m*
- reg_APPENDED_LINE*v* : Value of ISPCS_V_CLIP_DT_CODE[*v*]_CH[d].APPENDED_LINE
- reg_PREPENDED_LINE*v* : Value of ISPCS_V_CLIP_DT_CODE[*v*]_CH[d].PREPENDED_LINE
- *dt_code* : Value of the "Data Type" code of input CSI packet

Example of setting value of APPENDED_PIXEL and PREPENDED_PIXEL are shown in the Figure 54.7.

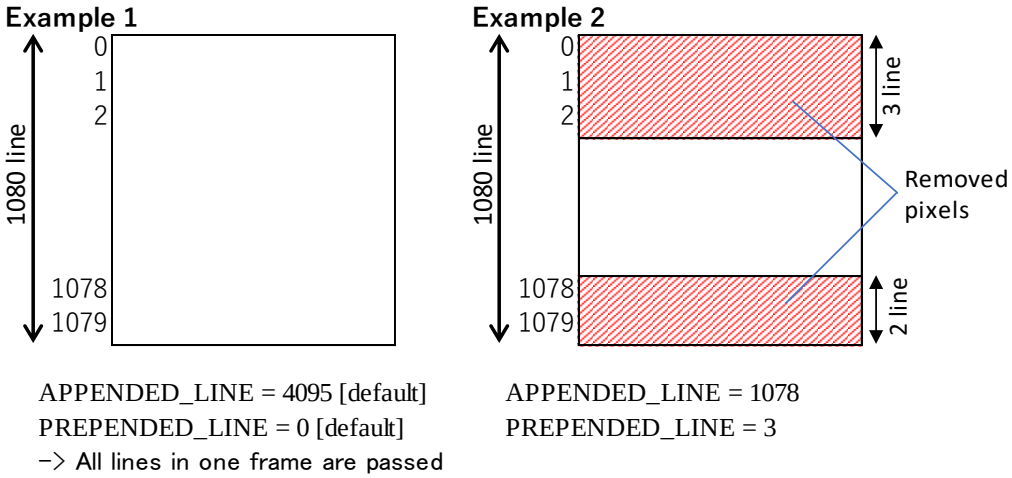



**Figure 54.7 Example of vertical clipping**

### 8. Stage 8: De-interleaving filter

In this stage, pixel-by-pixel filtering in one line is executed. DI_FILTER_LUT register is a configuration register to specify pass or drop for each pixel.

- Related registers
  - ISPCS_DI_FILTER_CTRL_CH[d].CPLX(*n*=0~11)
  - ISPCS_DI_FILTER_CTRL_CH[d].LUT_LENGTH_MINUS1 (*n*=0~11)
  - ISPCS_DI_FILTER_LUT[p]_CH[d].DI_FILTER_LUT (*p*=0~15, *d*=0~11)

```
di_filter_length = reg_DI_FILTER_LENGTH_MINUS1 + 1;
di_pixel_count = pixel_count >> reg_CPLX;

p = ( di_pixel_count % di_filter_length ) / 32;
k = ( di_pixel_count % di_filter_length ) % 32;

if ( reg_DI_FILTER_LUT(p,k)==1) {
   Current pixel is passed
}
else {
   Current pixel is discarded
}

pixel_count++;
```

[Defines of variables]
- *pixel_count* : Pixel count of one line after horizontal clipped (Left edge pixel is numbered as 0).
- reg_CPLX : Value of ISPCS_DI_FILTER_CTRL_CH[d].CPLX register
- reg_DI_FILTER_LENGTH_MINUS1 : Value of ISPCS_DI_FILTER_CTRL_CH[d].LUT_LENGTH_MINUS1 registeter
- reg_DI_FILTER_LUT(*p*,*k*): Value of k-th bit of ISPCS_DI_FILTER_CTRL_LUT[p]_CH[d].DI_FILTER_LUT register

## 9. RAW Scaler

RAW scaler block is newly added between Pixel Reconstructor (stage1) and VC filter (stage2).

RAW scaler scales incoming RAW image in RAW domain. Scaled ratio is only both 1 or 1/2 horizontal size and 1 or 1/2 vertical size. Horizontal only scaling or vertical only are not supported. Scale factor other than 1/2 is not supported.

Pixel value of RAW scaled image is calculated by linear convolution operation. Sliding window size is 5x5, and convolution coefficient is fixed, and is set by register. There are up to 4 sets of 5x5 convolution coefficients.

Switching of sets of convolution coeffient is done by co-ordinate of input pixel. Typically in RGGB Bayer array, R is located at (x%2,y%2) = (0,0) locatoin, where ((x,y) is co-ordinate of input RAW image, G is located at (x%2, y%2)=(0,1) and (1,0) location, B is located at (x%2,y%2) = (0,0) locatoin. So, user shjuld set 3 sets (R,G,B) of coefficient values to the register in advance. After that, while RAW scaling is running, if current RAW pixel is R, then coefficient set for R pixel is selected. Similarlly, if current RAW pixel is G, then coefficient set for G is selected.

Each channel can select independently either direct connection path (bypassed path) from Pixel Reconstructor or RAW scaled path by setting ISPCS_FILTER_ID.SEL_RAW_SCALED register.

If ISPCS_FILTER_ID.SEL_RAW_SCALED==0, then the bypassed path is selected (default). This means that RAW scaler is not enabled at this channel.

If ISPCS_FILTER_ID.SEL_RAW_SCALED==1, then RAW scaled image is selected. This means that RAW scaler is available at this channel.

#### 54.4.2.2 Example of configuration

**1. Example 1: Single sensor, single exposure**

In this example, image sensor transfers one long packet per one line with RAW20 or RAW12 format, its *dt_code* is H'2F(RAW20) or H'2C(RAW12) respectively. At the first of frame, the "Front Embedded Data" is transferred. Continuously, Ignored OB packets and Vertical OB packets are transferred.

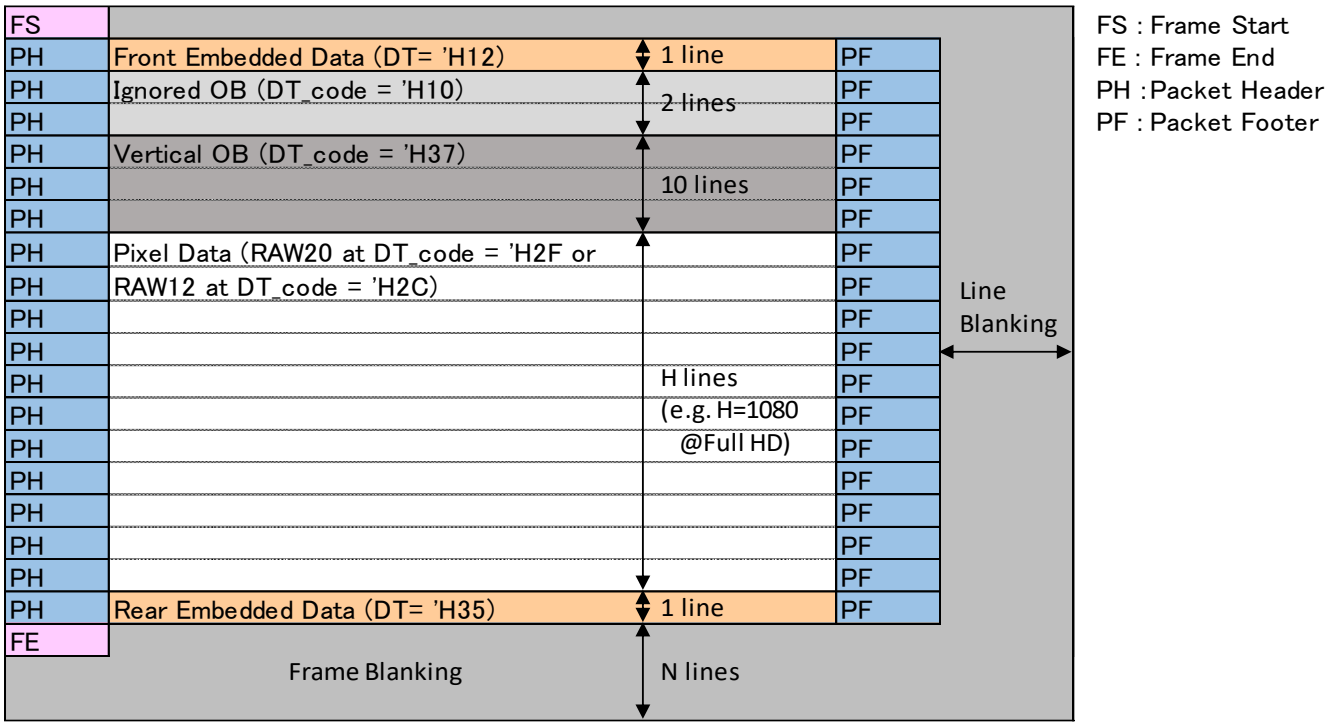



**Figure 54.8 Frame structure of example 1**

Register configuration of the Pixel Reconstructor and Channel Selector according to the frame structure in Figure 54.8 is shown in Table 54.3. Transferred data on each channel is shown in the Table 54.4

Regarding DT filter setting, up to eight different DT CODE can be set. The first 4 DT CODE values are set in the ISPCS_DT_CODE03_CH[d] (d=0~11) register. The second 4 DT CODE values are set in the ISPCS_DT_CODE47_CH[d] (d=0~11) register. In the following description, the bit fields in which these DT CODE values are set will be referred to as 'slot' or 'Filter slot m (m=0~7)', etc.

At the ISPCS_ DT_CODE03_CH[0] register, DT_CODE0 = H'2C is set as "Filter slot 0" to pass the incoming RAW12 pixel, and DT_CODE1 = H'2F is set as "Filter slot 1" to pass the incoming RAW20 pixel. By this configuration, any cases of either RAW12 case or RAW20 case can be passed by one configuration.

In this example, embedded data (meta data) transfer is also enabled. Incoming pixel data (DT_code = H'2C or H'2F) is transferred on the Channel 0 to the ISP core The Front Embedded Data and Rear Embedded Data are transferred on the Channel 5 to DRAM via VIN. Ignored OB and Vertical OB are discarded.

**Table 54.3 Register setting for the example 1**

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[18] | H'0000_1148 | H'0000_0007 | Front embedded data is processed as EMB8 |
| ISPPROCMODE_DT[44] | H'0000_11B0 | H'0000_0002 | RAW12 pixel data is processed as RAW12 |
| ISPPROCMODE_DT[47] | H'0000_11BC | H'0000_0005 | RAW20 pixel data is processed as RAW20 |
| ISPPROCMODE_DT[53] | H'0000_11D4 | H'0000_0007 | Rear embedded data is processed as EMB8 |
| ISPCS_FILTER_ID_CH[0] | H'0000_3000 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[0] | H'0000_3004 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[0] | H'0000_3008 | H'0000_AFAC | Filtering for the port0 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2C,<br>Filter slot1 : ENABLE1=1, DT_CODE1= 'H2F<br>Filter slot2 : ENABLE2=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0 and Filter slot1 are available.<br>CSI packet of *dt_code* = 'H2C or 'H2F are passed through |
| ISPCS_DT_CODE47_CH[0] | H'0000_300C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[0] | H'0000_3020 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[0] | H'0000_3024 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[0] | H'0000_3028 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[0] | H'0000_302C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[0] | H'0000_3030 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[0] | H'0000_3034 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[0] | H'0000_3038 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[0] | H'0000_303C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_FILTER_ID_CH[5] | H'0000_3500 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[5] | H'0000_3504 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[5] | H'0000_3508 | H'0000_B592 | Embedded data transfer<br>Filter slot0 : ENABLE0=1, HOLD=0, DT_CODE0= 'H12,<br>Filter slot1 : ENABLE1=1, HOLD=0, DT_CODE1= 'H35<br>Filter slot2 and filter slot3 are not used<br>Filter slot0 and Filter slot1 are available for the embedded data transfer. |
| ISPCS_DT_CODE47_CH[5] | H'0000_350C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[5] | H'0000_3520 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[5] | H'0000_3524 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[5] | H'0000_3528 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[5] | H'0000_352C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[5] | H'0000_3530 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[5] | H'0000_3534 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[5] | H'0000_3538 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[5] | H'0000_353C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPFIFOCTL | H0000_0004 | H'0000_0004 | FIFO for CSI input is enabled |
| ISPINPUTSEL0 | H0000_0008 | H'0000_0000 | Default value |
| ISPSTART | H0000_0014 | H'0000_FFFF | ISP processing start. |

**Table 54.4 Transferred data for each channel (Example 1)**

| Channel | Transferred data |
|---|---|
| Channel 0 | Pixel data (RAW20 *dt_code*=H'2F or RAW12 *dt_code*=H'2C) for ISP processing |
| Channel 1~4 | (nothing) |
| Channel 5 | Meta data (embedded data )<br>・ Front embedded data (*dt_code* = H'12)<br>・ Rear embedded data (*dt_code* = H'35) |
| Channel 6~11 | (nothing) |

### 2. Example 2: Single sensor, triple exposure, same DT_code for each exposure, line interleaved (only used VC0).

In this example, image sensor transfers 3 types of long packet with different exposures by line interleaved order (so-called as line-by-line). The same format (RAW12) is used and common *dt_code* (H'2C) is used for each exposure (H'2C

is MIPI CSI-2 standard defined value for RAW12 format). Configuration of line count ID (stage4) and line count filter (stage5) is needed for filtering each exposure data to each input port of ISP core because one common *dt_code* is shared around each exposure. 4 pixels at the beginning of one packet is not pixel data. These should be eliminated at the horizontal clipping (stage6).

If ISP core cannot receive 3 exposures (e.g. in case of up to 2 exposure), pixel data transferred on Channel2 is not available even if Channel 2 is configured properly.

Register configuration of the Pixel Reconstructor and Channel Selector according to the Figure 54.9 is shown in Table 54.5, and Table 54.6. Transferred data on each channel in ISP Interface/Channel Selector is shown in Table 54.7.

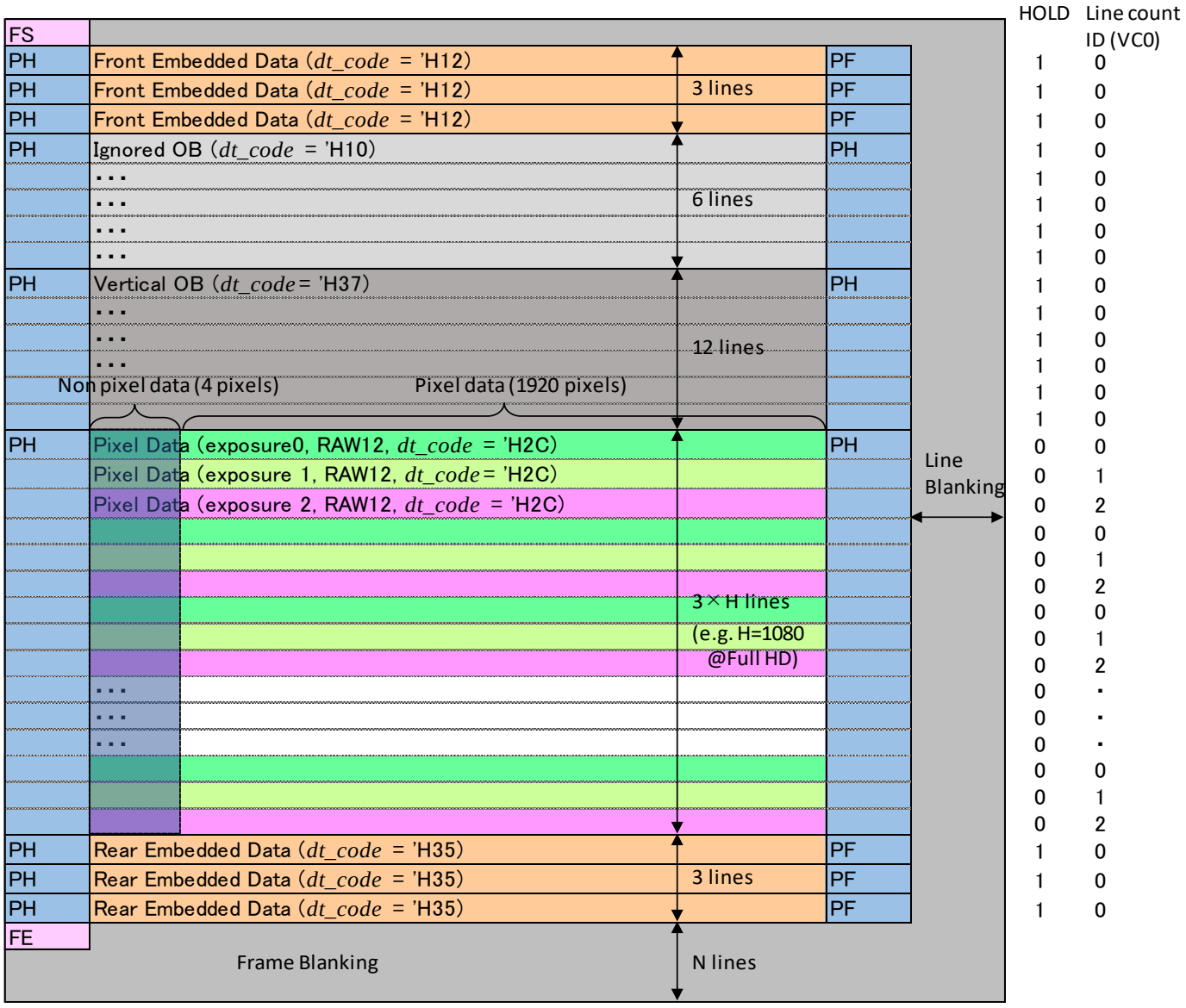



**Figure 54.9 Frame structure of example 2**

Table 54.5 Register setting for the Example 2 (1)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[18] | H'0000_1148 | H'0000_0007 | Front embedded data is processed as EMB8 |
| ISPPROCMODE_DT[44] | H'0000_11B0 | H'0000_0002 | RAW12 pixel data is processed as RAW12 |
| ISPPROCMODE_DT[53] | H'0000_11D4 | H'0000_0007 | Rear embedded data is processed as EMB8 |
| ISPCS_FILTER_ID_CH[0] | H'0000_3000 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[0] | H'0000_3004 | H'0000_0002 | MODULO_MINUS1_VC0='B010. |
| ISPCS_DT_CODE03_CH[0] | H'0000_3008 | H'0000_00AC | Filtering for the port0 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2C,<br>Filter slot1 : ENABLE1=0, DT_CODE1= 'H00 (not used)<br>Filter slot2 : ENABLE2=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0 are available.<br>CSI packet of *dt_code* = 'H2C are passed through |
| ISPCS_DT_CODE47_CH[0] | H'0000_300C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[0] | H'0000_3020 | H'0784_0004 | PREPENDED_PIXEL = 4, APPENDED_PIXEL = 1924. |
| ISPCS_H_CLIP_DT_CODE1_CH[0] | H'0000_3024 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[0] | H'0000_3028 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[0] | H'0000_302C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[0] | H'0000_3030 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[0] | H'0000_3034 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[0] | H'0000_3038 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[0] | H'0000_303C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_FILTER_ID_CH[1] | H'0000_3100 | H'0001_0001 | LC_FILTER_ID=1, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[1] | H'0000_3104 | H'0000_0002 | MODULO_MINUS1_VC0='B010. |
| ISPCS_DT_CODE03_CH[1] | H'0000_3108 | H'0000_00AC | Filtering for the port1 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2C,<br>Filter slot1 : ENABLE1=0, DT_CODE1= 'H00 (not used)<br>Filter slot2 : ENABLE2=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0 are available.<br>CSI packet of *dt_code* = 'H2C are passed through |
| ISPCS_DT_CODE47_CH[1] | H'0000_310C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[1] | H'0000_3120 | H'0784_0004 | PREPENDED_PIXEL = 4,APPENDED_PIXEL = 1924. |
| ISPCS_H_CLIP_DT_CODE1_CH[1] | H'0000_3124 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[1] | H'0000_3128 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[1] | H'0000_312C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[1] | H'0000_3130 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[1] | H'0000_3134 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[1] | H'0000_3138 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[1] | H'0000_313C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |

Table 54.6 Register setting of the Example 2 (2)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPCS_FILTER_ID_CH[2] | H'0000_3200 | H'0002_0001 | LC_FILTER_ID=2, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[2] | H'0000_3204 | H'0000_0002 | MODULO_MINUS1_VC0='B010. |
| ISPCS_DT_CODE03_CH[2] | H'0000_3208 | H'0000_00AC | Filtering for the port2 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2C,<br>Filter slot1 : ENABLE1=0, DT_CODE1= 'H00 (not used)<br>Filter slot2 : ENABLE2=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0 are available.<br>CSI packet of *dt_code* = 'H2C are passed through |
| ISPCS_DT_CODE47_CH[2] | H'0000_320C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[2] | H'0000_3220 | H'0784_0004 | PREPENDED_PIXEL = 4, APPENDED_PIXEL = 1924. |
| ISPCS_H_CLIP_DT_CODE1_CH[2] | H'0000_3224 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[2] | H'0000_3228 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[2] | H'0000_322C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[2] | H'0000_3230 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[2] | H'0000_3234 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[2] | H'0000_3238 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[2] | H'0000_323C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_FILTER_ID_CH[5] | H'0000_3500 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[5] | H'0000_3504 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[5] | H'0000_3508 | H'0000_B592 | Embedded data transfer<br>Filter slot0 : ENABLE0=1, HOLD=0, DT_CODE0= 'H12,<br>Filter slot1 : ENABLE1=1, HOLD=0, DT_CODE1= 'H35<br>Filter slot2 and filter slot3 are not used<br>Filter slot0 and Filter slot1 are available for the embedded data transfer. |
| ISPCS_DT_CODE47_CH[5] | H'0000_350C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[5] | H'0000_3520 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[5] | H'0000_3524 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[5] | H'0000_3528 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[5] | H'0000_352C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[5] | H'0000_3530 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[5] | H'0000_3534 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[5] | H'0000_3538 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[5] | H'0000_353C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPFIFOCTL | H0000_0004 | H'0000_0004 | FIFO for CSI input is enabled |
| ISPINPUTSEL0 | H0000_0008 | H'0000_0000 | Default value |
| ISPSTART | H0000_0014 | H'0000_FFFF | ISP processing start. |

Table 54.7 Data for transferred on each channel (Example 2)

| Channel | Transferred data |
|---|---|
| Channel 0 | Pixel data (exposure 0, RAW12, *dt_code* = H'2C) for ISP processing |
| Channel 1 | Pixel data (exposure 1, RAW12, *dt_code* = H'2C) for ISP processing |
| Channel 2 | Unavailable pixel data (exposure 2, RAW12, *dt_code* = H'2C)<br>(Reasion: Channel 2 is not connected to the ISP core) |
| Channel 3~4 | (nothing) |
| Channel 5 | Meta data (embedded data )<br>・ Front embedded data (*dt_code* = H'12)<br>・ Rear embedded data (*dt_code* = H'35) |
| Channel 6~11 | (nothing) |

### 3. Example 3: Single sensor, triple exposure with different *dt_code* for each exposure, line interleaved (only VC0).

In this example, image sensor transfers 3 types of long packet with different exposures by line interleaved order. Pixel format of each exposure is RAW14 (exposure0), RAW10 (exposure1) and RAW10 (exposure2) respectively. The *dt_code* of each exposure are particular values, which does not equal to the MIPI CSI-2 standard defined ones. Because

different *dt_code* values are assigned for each exposure packet filtering can be done by referring *dt_code* instead of line count ID. Therefore, configuration of line count ID (stage4) and line count filter (stage5) are not needed.

In this configuration, the ASIL Lines and the Voltage ASIL Lines are transferred to VIN via channel 4 and stored to the DRAM with the Embedded Data.

If ISP core cannot receive 3 exposures (e.g. in case of up to 2 exposure), pixel data transferred on Channel2 is not available even if Channel2 is configured properly.

Register configuration of the Pixel Reconstructor and Channel Selector according to the Figure 54.10 is shown in Table 54.8 and Table 54.9. Transferred data on each channels in ISP Interface/Channel Selector is shown in Table 54.10.

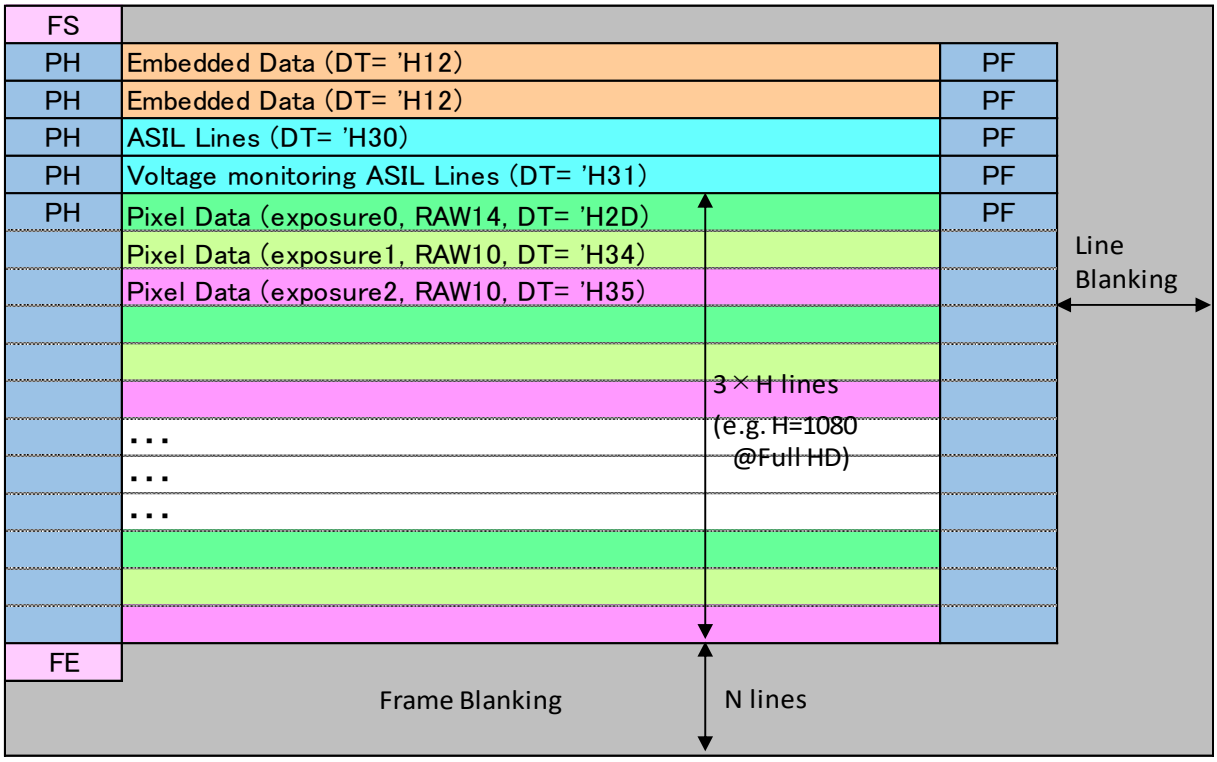



**Figure 54.10 Frame structure of example 3**

Table 54.8 Register setting of the Example 3 (1)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[18] | H'0000_1148 | H'0000_0002 | Front embedded data is processed as RAW12 |
| ISPPROCMODE_DT[45] | H'0000_11B4 | H'0000_0003 | RAW14 pixel data is processed as RAW14 |
| ISPPROCMODE_DT[48] | H'0000_11C0 | H'0000_0002 | ASIL line is processed as RAW12 |
| ISPPROCMODE_DT[49] | H'0000_11C4 | H'0000_0002 | ASIL line is processed as RAW12 |
| ISPPROCMODE_DT[52] | H'0000_11D0 | H'0000_0001 | RAW10 pixel data is processed as RAW10 |
| ISPPROCMODE_DT[53] | H'0000_11D4 | H'0000_0001 | RAW10 pixel data is processed as RAW10 |
| ISPCS_FILTER_ID_CH[0] | H'0000_3000 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[0] | H'0000_3004 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[0] | H'0000_3008 | H'0000_00AD | Filtering for the port0 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2D<br>Filter slot1 : ENABLE1=0, DT_CODE1= 'H00 (not used)<br>Filter slot2 : ENABLE2=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0 are available.<br>CSI packet of *dt_code* = 'H2D are passed through |
| ISPCS_DT_CODE47_CH[0] | H'0000_300C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[0] | H'0000_3020 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE1_CH[0] | H'0000_3024 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE2_CH[0] | H'0000_3028 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE3_CH[0] | H'0000_302C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_V_CLIP_DT_CODE0_CH[0] | H'0000_3030 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE1_CH[0] | H'0000_3034 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE2_CH[0] | H'0000_3038 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE3_CH[0] | H'0000_303C | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_FILTER_ID_CH[1] | H'0000_3100 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[1] | H'0000_3104 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[1] | H'0000_3108 | H'0000_B400 | Filtering for the port1 of ISP core<br>Filter slot0 : ENABLE0=0, DT_CODE0= 'H00 (not used)<br>Filter slot1 : ENABLE1=1, DT_CODE1= 'H34<br>Filter slot2 : ENABLE2=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot1 are available.<br>CSI packet of *dt_code* = 'H34 are passed through |
| ISPCS_DT_CODE47_CH[1] | H'0000_310C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[1] | H'0000_3120 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE1_CH[1] | H'0000_3124 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE2_CH[1] | H'0000_3128 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE3_CH[1] | H'0000_312C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_V_CLIP_DT_CODE0_CH[1] | H'0000_3130 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE1_CH[1] | H'0000_3134 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE2_CH[1] | H'0000_3138 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE3_CH[1] | H'0000_313C | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |

Table 54.9 Register setting of the Example 3 (2)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPCS_FILTER_ID_CH[2] | H'0000_3200 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[2] | H'0000_3204 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[2] | H'0000_3208 | H'00B5_0000 | Filtering for the port2 of ISP core<br>Filter slot0 : ENABLE0=0, DT_CODE0= 'H00 (not used)<br>Filter slot1 : ENABLE1=0, DT_CODE1= 'H00 (not used)<br>Filter slot2 : ENABLE2=1, DT_CODE2= 'H35<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot2 is available.<br>CSI packet of *dt_code* = 'H35 are passed through |
| ISPCS_DT_CODE47_CH[2] | H'0000_320C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[2] | H'0000_3220 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[2] | H'0000_3224 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[2] | H'0000_3228 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[2] | H'0000_322C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[2] | H'0000_3230 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[2] | H'0000_3234 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[2] | H'0000_3238 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[2] | H'0000_323C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_FILTER_ID_CH[5] | H'0000_3500 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[5] | H'0000_3504 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[5] | H'0000_3508 | H'00B1_B092 | Embedded data transfer<br>Filter slot0 : ENABLE0=1, HOLD0=0, DT_CODE0= 'H12<br>Filter slot1 : ENABLE1=1, HOLD1=0, DT_CODE1= 'H30<br>Filter slot2 : ENABLE2=1, HOLD2=0, DT_CODE2= 'H31<br>Filter slot3 : ENABLE3=0, HOLD3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0, slot1 and slot 2 are available. |
| ISPCS_DT_CODE47_CH[5] | H'0000_350C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[5] | H'0000_3520 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[5] | H'0000_3524 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[5] | H'0000_3528 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[5] | H'0000_352C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[5] | H'0000_3530 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[5] | H'0000_3534 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[5] | H'0000_3538 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[5] | H'0000_353C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPFIFOCTL | H0000_0004 | H'0000_0004 | FIFO for CSI input is enabled. |
| ISPINPUTSEL0 | H0000_0008 | H'0000_0000 | Default value |
| ISPSTART | H0000_0014 | H'0000_FFFF | ISP processing start. |

Table 54.10 Transferred data for each channel (Example 3)

| Channel | Transferred data |
|---|---|
| Channel 0 | Pixel data (exposure 0, RAW14, *dt_code* = H'2D) for ISP processing |
| Channel 1 | Pixel data (exposure 1, RAW10, *dt_code* = H'34) for ISP processing |
| Channel 2 | Unavailable pixel data (exposure 2, RAW10, *dt_code* = H'35)<br>(Reasion: Channel 2 is not connected to the ISP core) |
| Channel 3~4 | (nothing) |
| Channel 5 | Meta data<br>・ Embedded data (*dt_code* = H'12)<br>・ ASIL Lines (*dt_code* = H'30)<br>・ Voltage ASIL Lines (*dt_code* = H'31) |
| Channel 6~11 | (nothing) |

### 4. Example 4: Single sensor, triple exposure with virtual channel based interleaving (VC0, VC1 and VC2 are used for each exposure).

In this example, image sensor transfers 3 types of long packet with different exposures by VC based interleaving. Pixel format of each exposure is RAW14 (exposure0), RAW10 (exposure1) and RAW10 (exposure2) respectively. Each exposure uses different virtual channels (VC) each other. Filtering can be done by referring VC. Configuration of VC

filtering (stage2) is needed to enable VC based filtering. Configuration of line count ID and line count filtering are not needed.

If ISP core cannot receive 3 exposures (e.g. in case of up to 2 exposures), pixel data transferred on Channel2 is not available even if Channel2 is configured properly.

Register configuration of the Pixel Reconstructor and Channel Selector according to the Figure 54.11 is shown in Table 54.11 and Table 54.12. Transferred data on each channels in ISP Interface/Channel Selector is shown in Table 54.13.

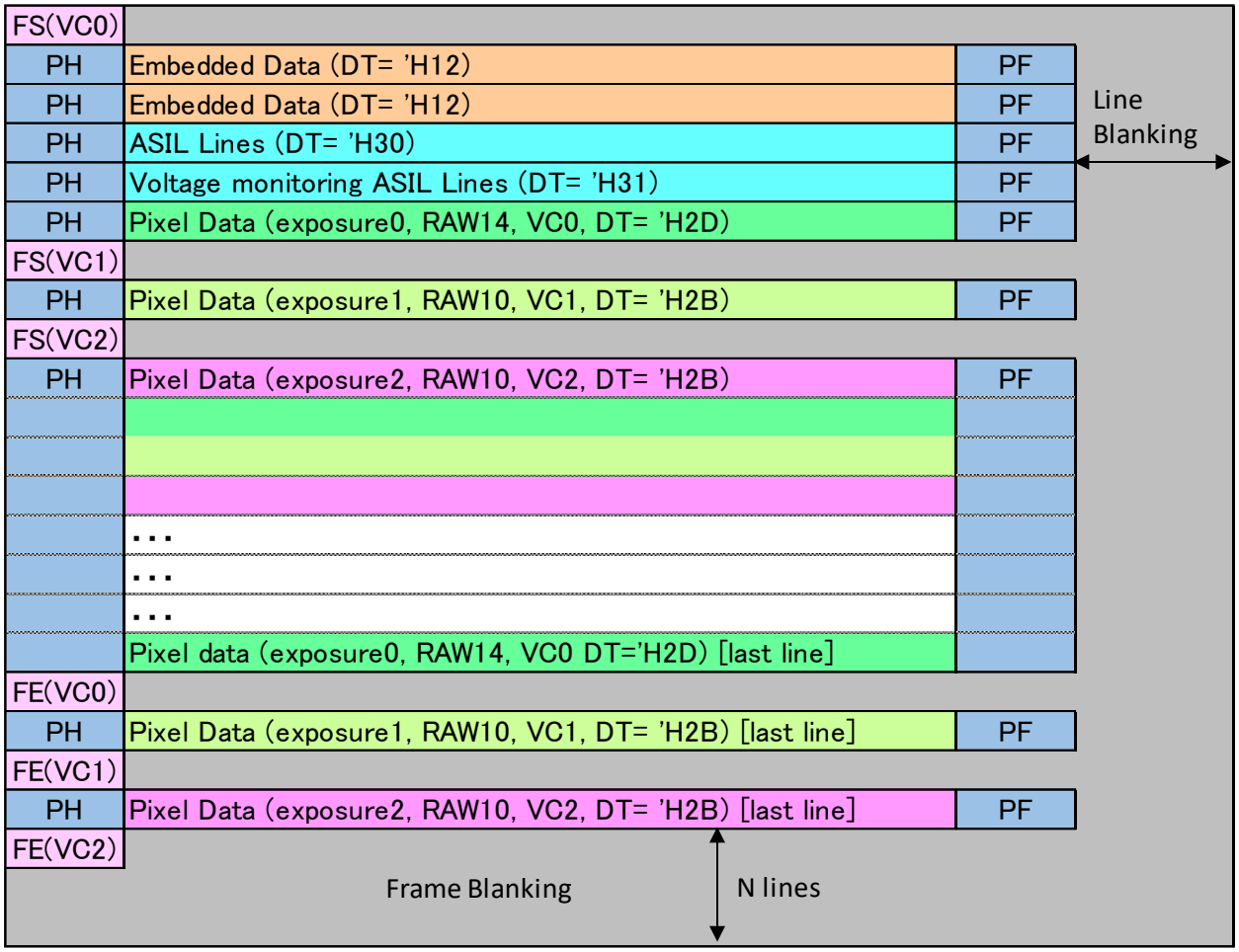



**Figure 54.11 Frame structure of the example 4**

Table 54.11 Register setting of the Example 4 (1)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[16] | H'0000_1140 | H'0000_0000 | Ignored OB is processed as RAW8 |
| ISPPROCMODE_DT[18] | H'0000_1148 | H'0000_0002 | Front embedded data is processed as RAW12 |
| ISPPROCMODE_DT[43] | H'0000_11AC | H'0001_0100 | RAW14 pixel data is processed as RAW10 |
| ISPPROCMODE_DT[45] | H'0000_11B4 | H'0000_0003 | RAW14 pixel data is processed as RAW14 |
| ISPPROCMODE_DT[48] | H'0000_11C0 | H'0000_0002 | ASIL line is processed as RAW12 |
| ISPPROCMODE_DT[49] | H'0000_11C4 | H'0000_0002 | ASIL line is processed as RAW12 |
| ISPCS_FILTER_ID_CH[0] | H'0000_3000 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[0] | H'0000_3004 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[0] | H'0000_3008 | H'0000_00AD | Filtering for the port0 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2D<br>Filter slot1 : ENABLE1=0, DT_CODE1= 'H00 (not used)<br>Filter slot2 : ENABLE2=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0 are available.<br>CSI packet of *dt_code* = 'H2D are passed through |
| ISPCS_DT_CODE47_CH[0] | H'0000_300C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[0] | H'0000_3020 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE1_CH[0] | H'0000_3024 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE2_CH[0] | H'0000_3028 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE3_CH[0] | H'0000_302C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_V_CLIP_DT_CODE0_CH[0] | H'0000_3030 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE1_CH[0] | H'0000_3034 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE2_CH[0] | H'0000_3038 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE3_CH[0] | H'0000_303C | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_FILTER_ID_CH[1] | H'0000_3100 | H'0008_0002 | LC_FILTER_ID=8, VC_EN = 'B0010. VC1 is passed |
| ISPCS_LC_MODULO_CH[1] | H'0000_3104 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[1] | H'0000_3108 | H'0000_AB00 | Filtering for the port1 of ISP core<br>Filter slot0 : ENABLE0=0, DT_CODE0= 'H00 (not used)<br>Filter slot1 : ENABLE1=1, DT_CODE1= 'H2B<br>Filter slot2 : ENABLE2=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot1 are available.<br>CSI packet of *dt_code* = 'H2D are passed through |
| ISPCS_DT_CODE47_CH[1] | H'0000_310C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[1] | H'0000_3120 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE1_CH[1] | H'0000_3124 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE2_CH[1] | H'0000_3128 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE3_CH[1] | H'0000_312C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_V_CLIP_DT_CODE0_CH[1] | H'0000_3130 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE1_CH[1] | H'0000_3134 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE2_CH[1] | H'0000_3138 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE3_CH[1] | H'0000_313C | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |

Table 54.12 Register setting of Example 4 (2)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPCS_FILTER_ID_CH[2] | H'0000_3200 | H'0010_0004 | LC_FILTER_ID=16, VC_EN = 'B0100. VC2 is passed |
| ISPCS_LC_MODULO_CH[2] | H'0000_3204 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[2] | H'0000_3208 | H'00AB_0000 | Filtering for the port2 of ISP core<br>Filter slot0 : ENABLE0=0, DT_CODE0= 'H00 (not used)<br>Filter slot1 : ENABLE1=0, DT_CODE1= 'H00 (not used)<br>Filter slot2 : ENABLE2=1, DT_CODE2= 'H2B<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H00 (not used)<br>Filter slot2 are available.<br>CSI packet of *dt_code* = 'H2B are passed through |
| ISPCS_DT_CODE47_CH[2] | H'0000_320C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[2] | H'0000_3220 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[2] | H'0000_3224 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[2] | H'0000_3228 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[2] | H'0000_322C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[2] | H'0000_3230 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[2] | H'0000_3234 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[2] | H'0000_3238 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[2] | H'0000_323C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_FILTER_ID_CH[5] | H'0000_3500 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[5] | H'0000_3504 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[5] | H'0000_3508 | H'00B1_B092 | Embedded data transfer<br>Filter slot0 : ENABLE0=1, HOLD0=0, DT_CODE0= 'H12<br>Filter slot1 : ENABLE1=1, HOLD1=0, DT_CODE1= 'H30<br>Filter slot2 : ENABLE2=1, HOLD2=0, DT_CODE2= 'H31<br>Filter slot3 : ENABLE3=0, HOLD3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0, slot1 and slot 2 are available. |
| ISPCS_DT_CODE47_CH[5] | H'0000_350C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[5] | H'0000_3520 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[5] | H'0000_3524 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[5] | H'0000_3528 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[5] | H'0000_352C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[5] | H'0000_3530 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[5] | H'0000_3534 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[5] | H'0000_3538 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[5] | H'0000_353C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPFIFOCTL | H0000_0004 | H'0000_0004 | FIFO for CSI input is enabled. |
| ISPINPUTSEL0 | H0000_0008 | H'0000_0000 | Default value |
| ISPSTART | H0000_0014 | H'0000_FFFF | ISP processing start. |

Table 54.13 Transferred data for each channel (Example 4)

| Channel | Transferred data |
|---|---|
| Channel 0 | Pixel data (exposure 0, RAW14, VC0, *dt_code* = H'2D) for ISP processing |
| Channel 1 | Pixel data (exposure 1, RAW10, VC1, *dt_code* = H'2B) for ISP processing |
| Channel 2 | Unavailable pixel data (exposure 2, RAW10, VC2, *dt_code* = H'2B)<br>(Reasion: Channel 2 is not connected to the ISP core) |
| Channel 3~4 | (nothing) |
| Channel 5 | Meta data<br>・ Embedded data (*dt_code* = H'12)<br>・ ASIL Lines (*dt_code* = H'30)<br>・ Voltage ASIL Lines (*dt_code* = H'31) |
| Channel 6~11 | (nothing) |

### 5. Example 5: Single sensor, quad exposure with line interleaved (only used VC0)

In this example, image sensor transfers 4 types of long packet with different exposure by line interleaved order. Pixel format of each exposure is RAW20 (exposure0), RAW16 (exposure1), RAW12 (exposure2) and RAW12 (exposure 3) respectively. *dt_code* value of each exposure is in accordance with MIPI CSI-2 standard. However, sending timing of

each exposure is different. Some vertical blanking (V-blanking) packet are transferred for padding of transfer timing of each exposure. Because the same dt_code is used between exposure 2 and exposure 3, configuration of line count ID (stage4) and line count filter (stage5) is needed.

If ISP core cannot receive 3 exposures (e.g. in case of up to 2 exposure), pixel data transferred on Channel2 and Channel3 is not available even if Channel2 and Channel3 are configured properly.

Register configuration of the Pixel Reconstructor and Channel Selector according to the Figure 54.12 is shown in the Table 54.14, Table 54.15 and Table 54.16. Transferred data on each channels in ISP Interface/Channel Selector is shown in Table 54.17.

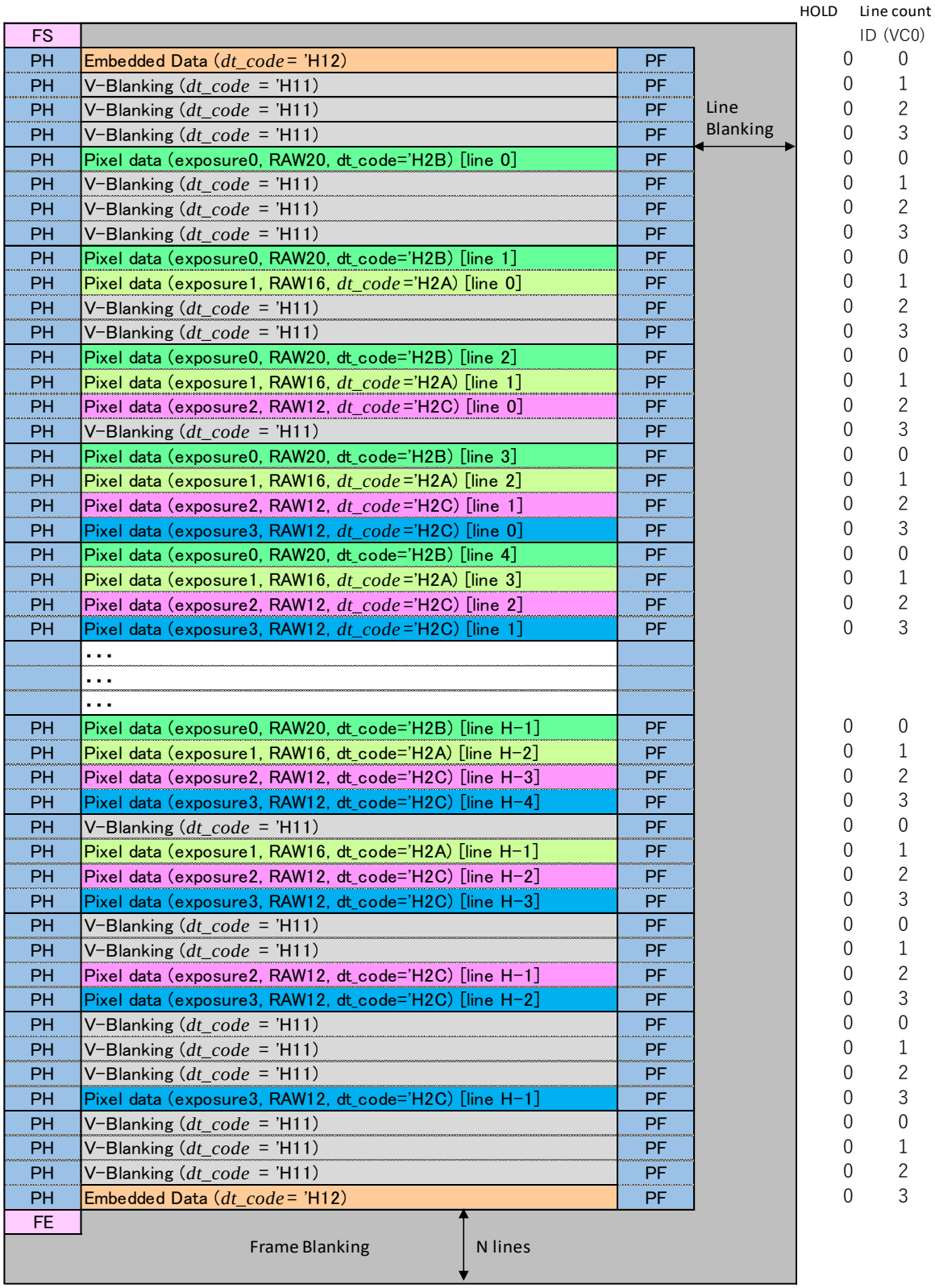



**Figure 54.12 Frame structure of the Example 5**

Table 54.14 Register setting of the Example 5 (1)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[17] | H'0000_1144 | H'0000_0007 | Vblanking is processed as EMB8 |
| ISPPROCMODE_DT[18] | H'0000_1148 | H'0000_0007 | Front embedded data is processed as EMB8 |
| ISPPROCMODE_DT[42] | H'0000_11A8 | H'0000_0004 | RAW16 pixel data is processed as RAW16 |
| ISPPROCMODE_DT[43] | H'0000_11AC | H'0000_0005 | RAW20 pixel data is processed as RAW20 |
| ISPPROCMODE_DT[44] | H'0000_11B0 | H'0000_0002 | RAW12 pixel data is processed as RAW12 |
| ISPCS_FILTER_ID_CH[0] | H'0000_3000 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[0] | H'0000_3004 | H'0000_0003 | MODULO_MINUS1_VC0='B011. |
| ISPCS_DT_CODE03_CH[0] | H'0000_3008 | H'11AA_ABAC | Filtering for the port0 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2C,<br>Filter slot1 : ENABLE1=1, DT_CODE1= 'H2B,<br>Filter slot2 : ENABLE2=1, DT_CODE2= 'H2A,<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H11,<br>Filter slot0, Filter slot1, Filter slot2 and Filter slot3 are available.<br>CSI packet of *dt_code* = 'H2C or 'H2B or 'H2A or 'H11 are passed through |
| ISPCS_DT_CODE47_CH[0] | H'0000_300C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[0] | H'0000_3020 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE1_CH[0] | H'0000_3024 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE2_CH[0] | H'0000_3028 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE3_CH[0] | H'0000_302C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_V_CLIP_DT_CODE0_CH[0] | H'0000_3030 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE1_CH[0] | H'0000_3034 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE2_CH[0] | H'0000_3038 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE3_CH[0] | H'0000_303C | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_FILTER_ID_CH[1] | H'0000_3100 | H'0001_0001 | LC_FILTER_ID=1, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[1] | H'0000_3104 | H'0000_0003 | MODULO_MINUS1_VC0='B011. |
| ISPCS_DT_CODE03_CH[1] | H'0000_3108 | H'11AA_ABAC | Filtering for the port1 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2C,<br>Filter slot1 : ENABLE1=1, DT_CODE1= 'H2B,<br>Filter slot2 : ENABLE2=1, DT_CODE2= 'H2A,<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H11,<br>Filter slot0, Filter slot1, Filter slot2 and Filter slot3 are available.<br>CSI packet of *dt_code* = 'H2C or 'H2B or 'H2A or 'H11 are passed through |
| ISPCS_DT_CODE47_CH[1] | H'0000_310C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[1] | H'0000_3120 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE1_CH[1] | H'0000_3124 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE2_CH[1] | H'0000_3128 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_H_CLIP_DT_CODE3_CH[1] | H'0000_312C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used. |
| ISPCS_V_CLIP_DT_CODE0_CH[1] | H'0000_3130 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE1_CH[1] | H'0000_3134 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE2_CH[1] | H'0000_3138 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE3_CH[1] | H'0000_313C | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |

Table 54.15 Register setting of the Example 5 (2)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPCS_FILTER_ID_CH[2] | H'0000_3200 | H'0002_0001 | LC_FILTER_ID=2, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[2] | H'0000_3204 | H'0000_0003 | MODULO_MINUS1_VC0='B011. |
| ISPCS_DT_CODE03_CH[2] | H'0000_3208 | H'11AA_ABAC | Filtering for the port2 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2C,<br>Filter slot1 : ENABLE1=1, DT_CODE1= 'H2B,<br>Filter slot2 : ENABLE2=1, DT_CODE2= 'H2A,<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H11,<br>Filter slot0, Filter slot1, Filter slot2 and Filter slot3 are available.<br>CSI packet of *dt_code* = 'H2C or 'H2B or 'H2A or 'H11 are passed through |
| ISPCS_DT_CODE47_CH[2] | H'0000_320C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[2] | H'0000_3220 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[2] | H'0000_3224 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[2] | H'0000_3228 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[2] | H'0000_322C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[2] | H'0000_3230 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE1_CH[2] | H'0000_3234 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE2_CH[2] | H'0000_3238 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE3_CH[2] | H'0000_323C | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_FILTER_ID_CH[3] | H'0000_3300 | H'0003_0001 | LC_FILTER_ID=3, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[3] | H'0000_3304 | H'0000_0003 | MODULO_MINUS1_VC0='B011. |
| ISPCS_DT_CODE03_CH[3] | H'0000_3308 | H'11AA_ABAC | Filtering for the port3 of ISP core<br>Filter slot0 : ENABLE0=1, DT_CODE0= 'H2C,<br>Filter slot1 : ENABLE1=1, DT_CODE1= 'H2B,<br>Filter slot2 : ENABLE2=1, DT_CODE2= 'H2A,<br>Filter slot3 : ENABLE3=0, DT_CODE3= 'H11,<br>Filter slot0, Filter slot1, Filter slot2 and Filter slot3 are available.<br>CSI packet of *dt_code* = 'H2C or 'H2B or 'H2A or 'H11 are passed through |
| ISPCS_DT_CODE47_CH[3] | H'0000_330C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[3] | H'0000_3320 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[3] | H'0000_3324 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[3] | H'0000_3328 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[3] | H'0000_332C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[3] | H'0000_3330 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE1_CH[3] | H'0000_3334 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE2_CH[3] | H'0000_3338 | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |
| ISPCS_V_CLIP_DT_CODE3_CH[3] | H'0000_333C | H'0FFF_0000 | Default value. Line clipping in one frame is not used. |

Table 54.16 Register setting of the Example 5 (3)

| Register Name | Offset | Value | Remarks |
|---|---|---|---|
| ISPCS_FILTER_ID_CH[5] | H'0000_3500 | H'0000_0001 | LC_FILTER_ID=0, VC_EN = 'B0001. VC0 is passed |
| ISPCS_LC_MODULO_CH[5] | H'0000_3504 | H'0000_0000 | Line count filter is not used |
| ISPCS_DT_CODE03_CH[5] | H'0000_3508 | H'0000_0092 | Embedded data transfer<br>Filter slot0 : ENABLE0=1, HOLD0=0, DT_CODE0= 'H12<br>Filter slot1 : ENABLE1=0, HOLD3=0, DT_CODE1= 'H00 (not used)<br>Filter slot2 : ENABLE2=0, HOLD3=0, DT_CODE2= 'H00 (not used)<br>Filter slot3 : ENABLE3=0, HOLD3=0, DT_CODE3= 'H00 (not used)<br>Filter slot0 is available. |
| ISPCS_DT_CODE47_CH[5] | H'0000_350C | H'0000_0000 | Filer slot4,5,6,7 are not used |
| ISPCS_H_CLIP_DT_CODE0_CH[5] | H'0000_3520 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE1_CH[5] | H'0000_3524 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE2_CH[5] | H'0000_3528 | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_H_CLIP_DT_CODE3_CH[5] | H'0000_352C | H'0FFF_0000 | Default value. Pixel clipping in one line is not used |
| ISPCS_V_CLIP_DT_CODE0_CH[5] | H'0000_3530 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE1_CH[5] | H'0000_3534 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE2_CH[5] | H'0000_3538 | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPCS_V_CLIP_DT_CODE3_CH[5] | H'0000_353C | H'0FFF_0000 | Default value. Line clipping in one frame is not used |
| ISPFIFOCTL | H'0000_0004 | H'0000_0004 | FIFO for CSI input is enabled. |
| ISPINPUTSEL0 | H'0000_0008 | H'0000_0000 | Default value |
| ISPSTART | H'0000_0014 | H'0000_FFFF | ISP processing start. |

Table 54.17 Transferred data for each channel (Example 5)

| Channel | Transferred data |
|---|---|
| Channel 0 | Pixel data (exposure 0, RAW20, *dt_code* = H'2B) for ISP processing |
| Channel 1 | Pixel data (exposure 1, RAW16, *dt_code* = H'2A) for ISP processing |
| Channel 2 | Unavilable pixel data (exposure 2, RAW12, *dt_code* = H'2C) .<br>(Reason: Channel2 is not connected to the ISP core) |
| Channel 3 | Unavailable Pixel data (exposure 3, RAW12, *dt_code* = H'2C)<br>(Reason: Channel3 is not connected to the ISP core) |
| Channel 4 | (nothing) |
| Channel 5 | Embedded data (*dt_code* = H'12) |
| Channel 6~11 | (nothing) |

#### 54.4.2.3 Configuration of LUT based pixel reconstructor

LUT based pixel reconstructor supports some packing formats which are not defined in MIPI CSI-2. Supporting format are the following types. However, only 1 format are available simultaneously.

- YUV444: Y16, U8, V8 (Y 16bit, U 8bit, V 8bit)
- YUV444: Y16, U10, V10 (Y 16bit, U 10bit, V 10bit)
- YUV444: Y16, U12, V12 (Y 16bit, U 12bit, V 12bit)
- YUV444: Y12, U8, V8 (Y 16bit, U 8bit, V 8bit)
- YUV422: Y12, UV8 (Y 12bit, UV 8bit)

This function can be used by setting H'3F to the ISPPCS_PROC_MODE_DT[n].PROC_MODE_VCm (n=0~63, m=0~3) register.

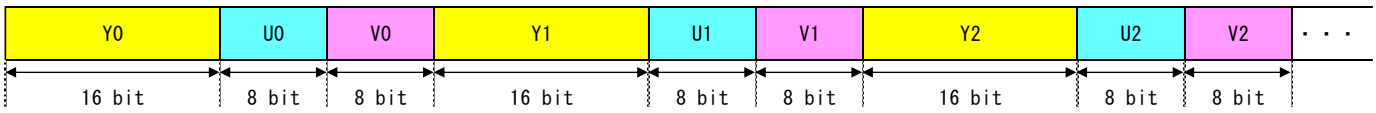
Each color components (Y, U, V) should be transferred on separate channels. Configuration to transfer with mixing several components on one channel are forbidden. Color component separation is done at the De-interleaving filter. So, configuration of De-interleavling filter is also needed.

Following configuration uses the Channel 4 for Y-component, Channel 5 for U-component and the Channel 6 for V-component, respectively.

In addition to the following configuration, configuration which relates to the Stage1~ Stage8.

### 1. YUV444, Y16, U8, V8

Incoming pixel sequence which is packed in one CSI long packet is shown as below.



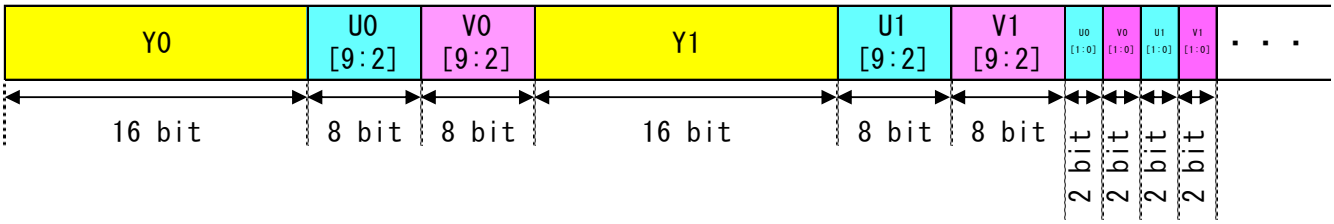

Configuration of LUT based pixel reconstructor

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[n] (n: DT-code of incoming CSI packet) | 'H1100 + 4*n | 'H38383838 | |
| ISPPROC_CUSTOM_FORMAT[0]_CTRL | 'H1010 | 'H02000000 | LUT_LENGTH_MINUS1=2<br>LUT_START_ADDRESS=0<br>POPNUM_MINUS1=0 |
| ISPCS_DI_FILTER_CTRL_CH[4] | 'H3440 | 'H00000002 | Ch4, Y |
| ISPCS_DI_FILTER_LUT[0]_CH[4] | 'H3480 | 'H00000001 | Ch4, Y |
| ISPCS_DI_FILTER_CTRL_CH[5] | 'H3540 | 'H00000002 | Ch5, U |
| ISPCS_DI_FILTER_LUT[0]_CH[5] | 'H3580 | 'H00000002 | Ch5, U |
| ISPCS_DI_FILTER_CTRL_CH[6] | 'H3640 | 'H00000002 | Ch6, V |
| ISPCS_DI_FILTER_LUT[0]_CH[6] | 'H3680 | 'H00000004 | Ch6, V |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[0] | 'H8000 | 'H00A02408 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[1] | 'H8004 | 'H00D0300B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[2] | 'H8008 | 'H00003C0E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[3] | 'H800C | 'H00300801 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[4] | 'H8010 | 'H00601404 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[5] | 'H8014 | 'H20080007 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[6] | 'H8018 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[7] | 'H801C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[0] | 'H8020 | 'H01204410 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[1] | 'H8024 | 'H01505013 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[2] | 'H8028 | 'H20005C16 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[3] | 'H802C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[4] | 'H8030 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[5] | 'H8034 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[6] | 'H8038 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[7] | 'H803C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[0] | 'H8040 | 'H01A06418 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[1] | 'H8044 | 'H01D0701B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[2] | 'H8048 | 'H20007C1E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[3] | 'H804C | 'H20080200 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[4] | 'H8050 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[5] | 'H8054 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[6] | 'H8058 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[7] | 'H805C | 'H20080200 | |

### 2. YUV444, Y16, U10, V10

Incoming pixel sequence which is packed in one CSI long packet is shown as below.




Configuration of LUT based pixel reconstructor

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[n]<br>(n: DT-code of incoming CSI packet) | 'H1100 + 4*n | 'H38383838 | |
| ISPPROC_CUSTOM_FORMAT[0]_CTRL | 'H1010 | 'H17000008 | LUT_LENGTH_MINUS1=23<br>LUT_START_ADDRESS=0<br>POPNUM_MINUS1=8 |
| ISPCS_DI_FILTER_CTRL_CH[4] | 'H3440 | 'H00000002 | Ch4, Y |
| ISPCS_DI_FILTER_LUT[0]_CH[4] | 'H3480 | 'H00000001 | Ch4, Y |
| ISPCS_DI_FILTER_CTRL_CH[5] | 'H3540 | 'H00000002 | Ch5, U |
| ISPCS_DI_FILTER_LUT[0]_CH[5] | 'H3580 | 'H00000002 | Ch5, U |
| ISPCS_DI_FILTER_CTRL_CH[6] | 'H3640 | 'H00000002 | Ch6, V |
| ISPCS_DI_FILTER_LUT[0]_CH[6] | 'H3680 | 'H00000002 | Ch6, V |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[0] | 'H8000 | 'H00A02408 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[1] | 'H8004 | 'H00D0300B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[2] | 'H8008 | 'H00003C0E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[3] | 'H800C | 'H00300801 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[4] | 'H8010 | 'H00601404 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[5] | 'H8014 | 'H20080007 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[6] | 'H8018 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[7] | 'H801C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[0] | 'H8020 | 'H01010440 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[1] | 'H8024 | 'H01304811 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[2] | 'H8028 | 'H01605414 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[3] | 'H802C | 'H20080017 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[4] | 'H8030 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[5] | 'H8034 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[6] | 'H8038 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[7] | 'H803C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[0] | 'H8040 | 'H01810C42 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[1] | 'H8044 | 'H01B06819 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[2] | 'H8048 | 'H01E0741C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[3] | 'H804C | 'H2008001F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[4] | 'H8050 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[5] | 'H8054 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[6] | 'H8058 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[7] | 'H805C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[0] | 'H8060 | 'H02A0A428 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[1] | 'H8064 | 'H02D0B02B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[2] | 'H8068 | 'H0200BC2E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[3] | 'H806C | 'H02308821 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[4] | 'H8070 | 'H02609424 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[5] | 'H8074 | 'H20080027 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[6] | 'H8078 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[7] | 'H807C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[0] | 'H8080 | 'H03011444 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[1] | 'H8084 | 'H0330C831 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[2] | 'H8088 | 'H0360D434 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[3] | 'H808C | 'H20080037 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[4] | 'H8090 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[5] | 'H8094 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[6] | 'H8098 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[7] | 'H809C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[0] | 'H80A0 | 'H03811C46 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[1] | 'H80A4 | 'H03B0E839 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[2] | 'H80A8 | 'H03E0F43C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[3] | 'H80AC | 'H2008003F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[4] | 'H80B0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[5] | 'H80B4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[6] | 'H80B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[7] | 'H80BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[0] | 'H80C0 | 'H05214450 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[1] | 'H80C4 | 'H05515053 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[2] | 'H80C8 | 'H04815C56 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[3] | 'H80CC | 'H04B12849 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[4] | 'H80D0 | 'H04E1344C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[5] | 'H80D4 | 'H2008004F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[6] | 'H80D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[7] | 'H80DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[0] | 'H80E0 | 'H05822488 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[1] | 'H80E4 | 'H05B16859 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[2] | 'H80E8 | 'H05E1745C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[3] | 'H80EC | 'H2008005F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[4] | 'H80F0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[5] | 'H80F4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[6] | 'H80F8 | 'H20080200 | |

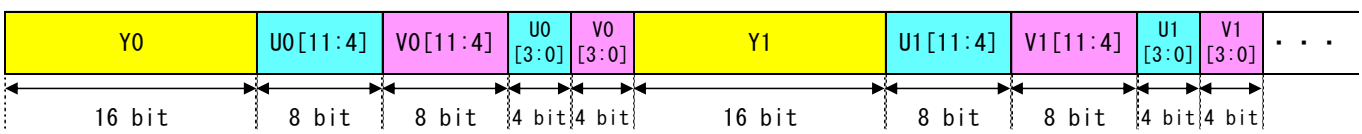
| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[7] | 'H80FC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[0] | 'H8100 | 'H06022C8A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[1] | 'H8104 | 'H06318861 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[2] | 'H8108 | 'H06619464 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[3] | 'H810C | 'H20080067 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[4] | 'H8110 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[5] | 'H8114 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[6] | 'H8118 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[7] | 'H811C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[0] | 'H8120 | 'H0721C470 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[1] | 'H8124 | 'H0751D073 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[2] | 'H8128 | 'H0681DC76 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[3] | 'H812C | 'H06B1A869 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[4] | 'H8130 | 'H06E1B46C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[5] | 'H8134 | 'H2008006F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[6] | 'H8138 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[7] | 'H813C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[0] | 'H8140 | 'H0782348C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[1] | 'H8144 | 'H07B1E879 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[2] | 'H8148 | 'H07E1F47C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[3] | 'H814C | 'H2008007F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[4] | 'H8150 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[5] | 'H8154 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[6] | 'H8158 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[7] | 'H815C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[0] | 'H8160 | 'H08023C8E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[1] | 'H8164 | 'H08320881 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[2] | 'H8168 | 'H08621484 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[3] | 'H816C | 'H20080087 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[4] | 'H8170 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[5] | 'H8174 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[6] | 'H8178 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[7] | 'H817C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[0] | 'H8180 | 'H09A26498 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[1] | 'H8184 | 'H09D2709B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[2] | 'H8188 | 'H09027C9E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[3] | 'H818C | 'H09324891 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[4] | 'H8190 | 'H09625494 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[5] | 'H8194 | 'H20080097 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[6] | 'H8198 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[7] | 'H819C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[0] | 'H81A0 | 'H0A0344D0 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[1] | 'H81A4 | 'H0A3288A1 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[2] | 'H81A8 | 'H0A6294A4 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[3] | 'H81AC | 'H200800A7 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[4] | 'H81B0 | 'H20080200 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[5] | 'H81B4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[6] | 'H81B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[7] | 'H81BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[0] | 'H81C0 | 'H0A834CD2 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[1] | 'H81C4 | 'H0AB2A8A9 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[2] | 'H81C8 | 'H0AE2B4AC | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[3] | 'H81CC | 'H200800AF | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[4] | 'H81D0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[5] | 'H81D4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[6] | 'H81D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[7] | 'H81DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[0] | 'H81E0 | 'H0BA2E4B8 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[1] | 'H81E4 | 'H0BD2F0BB | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[2] | 'H81E8 | 'H0B02FCBE | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[3] | 'H81EC | 'H0B32C8B1 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[4] | 'H81F0 | 'H0B62D4B4 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[5] | 'H81F4 | 'H200800B7 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[6] | 'H81F8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[7] | 'H81FC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[0] | 'H8200 | 'H0C0354D4 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[1] | 'H8204 | 'H0C3308C1 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[2] | 'H8208 | 'H0C6314C4 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[3] | 'H820C | 'H200800C7 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[4] | 'H8210 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[5] | 'H8214 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[6] | 'H8218 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[7] | 'H821C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[0] | 'H8220 | 'H0C835CD6 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[1] | 'H8224 | 'H0CB328C9 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[2] | 'H8228 | 'H0CE334CC | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[3] | 'H822C | 'H200800CF | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[4] | 'H8230 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[5] | 'H8234 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[6] | 'H8238 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[7] | 'H823C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[0] | 'H8240 | 'H0E2384E0 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[1] | 'H8244 | 'H0E5390E3 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[2] | 'H8248 | 'H0D839CE6 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[3] | 'H824C | 'H0DB368D9 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[4] | 'H8250 | 'H0DE374DC | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[5] | 'H8254 | 'H200800DF | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[6] | 'H8258 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[7] | 'H825C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[0] | 'H8260 | 'H0E846518 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[1] | 'H8264 | 'H0EB3A8E9 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[2] | 'H8268 | 'H0EE3B4EC | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[3] | 'H826C | 'H200800EF | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[4] | 'H8270 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[5] | 'H8274 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[6] | 'H8278 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[7] | 'H827C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[0] | 'H8280 | 'H0F046D1A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[1] | 'H8284 | 'H0F33C8F1 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[2] | 'H8288 | 'H0F63D4F4 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[3] | 'H828C | 'H200800F7 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[4] | 'H8290 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[5] | 'H8294 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[6] | 'H8298 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[7] | 'H829C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[0] | 'H82A0 | 'H10240500 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[1] | 'H82A4 | 'H10541103 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[2] | 'H82A8 | 'H0F841D06 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[3] | 'H82AC | 'H0FB3E8F9 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[4] | 'H82B0 | 'H0FE3F4FC | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[5] | 'H82B4 | 'H200800FF | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[6] | 'H82B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[7] | 'H82BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[0] | 'H82C0 | 'H1084751C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[1] | 'H82C4 | 'H10B42909 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[2] | 'H82C8 | 'H10E4350C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[3] | 'H82CC | 'H2008010F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[4] | 'H82D0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[5] | 'H82D4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[6] | 'H82D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[7] | 'H82DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[0] | 'H82E0 | 'H11047D1E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[1] | 'H82E4 | 'H11344911 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[2] | 'H82E8 | 'H11645514 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[3] | 'H82EC | 'H20080117 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[4] | 'H82F0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[5] | 'H82F4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[6] | 'H82F8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[7] | 'H82FC | 'H20080200 | |

### 3. YUV444, Y16, U12, V12

Incoming pixel sequence which is packed in one CSI long packet is shown as below.



Configuration of LUT based pixel reconstructor

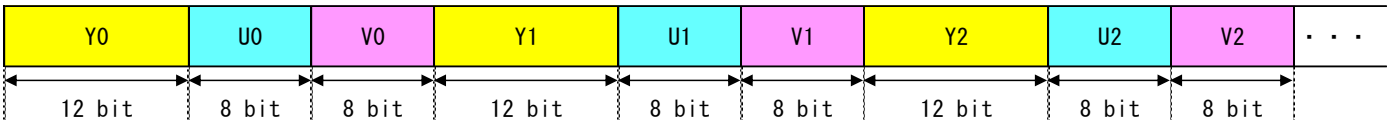
| Register Name | Offset | Setting Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[n]<br>(n: DT-code of incoming CSI packet) | 'H1100 + 4*n | 'H38383838 | |
| ISPPROC_CUSTOM_FORMAT[0]_CTRL | 'H1010 | 'H0B000004 | LUT_LENGTH_MINUS1=11<br>LUT_START_ADDRESS=0<br>POPNUM_MINUS1=4 |
| ISPCS_DI_FILTER_CTRL_CH[4] | 'H3440 | 'H00000002 | Ch4, Y |
| ISPCS_DI_FILTER_LUT[0]_CH[4] | 'H3480 | 'H00000001 | Ch4, Y |
| ISPCS_DI_FILTER_CTRL_CH[5] | 'H3540 | 'H00000002 | Ch5, U |
| ISPCS_DI_FILTER_LUT[0]_CH[5] | 'H3580 | 'H00000002 | Ch5, U |
| ISPCS_DI_FILTER_CTRL_CH[6] | 'H3640 | 'H00000002 | Ch6, V |
| ISPCS_DI_FILTER_LUT[0]_CH[6] | 'H3680 | 'H00000004 | Ch6, V |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[0] | 'H8000 | 'H00A02408 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[1] | 'H8004 | 'H00D0300B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[2] | 'H8008 | 'H00003C0E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[3] | 'H800C | 'H00300801 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[4] | 'H8010 | 'H00601404 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[5] | 'H8014 | 'H20080007 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[6] | 'H8018 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[7] | 'H801C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[0] | 'H8020 | 'H02609424 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[1] | 'H8024 | 'H01104027 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[2] | 'H8028 | 'H01404C12 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[3] | 'H802C | 'H01705815 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[4] | 'H8030 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[5] | 'H8034 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[6] | 'H8038 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[7] | 'H803C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[0] | 'H8040 | 'H02208420 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[1] | 'H8044 | 'H01906023 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[2] | 'H8048 | 'H01C06C1A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[3] | 'H804C | 'H01F0781D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[4] | 'H8050 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[5] | 'H8054 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[6] | 'H8058 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[7] | 'H805C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[0] | 'H8060 | 'H0320C430 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[1] | 'H8064 | 'H0350D033 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[2] | 'H8068 | 'H0280DC36 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[3] | 'H806C | 'H02B0A829 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[4] | 'H8070 | 'H02E0B42C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[5] | 'H8074 | 'H2008002F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[6] | 'H8078 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[7] | 'H807C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[0] | 'H8080 | 'H04E1344C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[1] | 'H8084 | 'H0390E04F | |

| Register Name | Offset | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[2] | 'H8088 | 'H03C0EC3A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[3] | 'H808C | 'H03F0F83D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[4] | 'H8090 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[5] | 'H8094 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[6] | 'H8098 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[7] | 'H809C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[0] | 'H80A0 | 'H04A12448 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[1] | 'H80A4 | 'H0411004B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[2] | 'H80A8 | 'H04410C42 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[3] | 'H80AC | 'H04711845 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[4] | 'H80B0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[5] | 'H80B4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[6] | 'H80B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[7] | 'H80BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[0] | 'H80C0 | 'H05A16458 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[1] | 'H80C4 | 'H05D1705B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[2] | 'H80C8 | 'H05017C5E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[3] | 'H80CC | 'H05314851 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[4] | 'H80D0 | 'H05615454 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[5] | 'H80D4 | 'H20080057 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[6] | 'H80D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[7] | 'H80DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[0] | 'H80E0 | 'H0761D474 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[1] | 'H80E4 | 'H06118077 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[2] | 'H80E8 | 'H06418C62 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[3] | 'H80EC | 'H06719865 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[4] | 'H80F0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[5] | 'H80F4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[6] | 'H80F8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[7] | 'H80FC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[0] | 'H8100 | 'H0721C470 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[1] | 'H8104 | 'H0691A073 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[2] | 'H8108 | 'H06C1AC6A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[3] | 'H810C | 'H06F1B86D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[4] | 'H8110 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[5] | 'H8114 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[6] | 'H8118 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[7] | 'H811C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[0] | 'H8120 | 'H08220480 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[1] | 'H8124 | 'H08521083 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[2] | 'H8128 | 'H07821C86 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[3] | 'H812C | 'H07B1E879 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[4] | 'H8130 | 'H07E1F47C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[5] | 'H8134 | 'H2008007F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[6] | 'H8138 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[7] | 'H813C | 'H20080200 | |

| Register Name | Offset | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[0] | 'H8140 | 'H09E2749C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[1] | 'H8144 | 'H0892209F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[2] | 'H8148 | 'H08C22C8A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[3] | 'H814C | 'H08F2388D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[4] | 'H8150 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[5] | 'H8154 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[6] | 'H8158 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[7] | 'H815C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[0] | 'H8160 | 'H09A26498 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[1] | 'H8164 | 'H0912409B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[2] | 'H8168 | 'H09424C92 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[3] | 'H816C | 'H09725895 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[4] | 'H8170 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[5] | 'H8174 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[6] | 'H8178 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[7] | 'H817C | 'H20080200 | |

### 4. YUV444, Y12, U8, V8

Incoming pixel sequence which is packed in one CSI long packet is shown as below.




Configuration of LUT based pixel reconstructor

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[n]<br>(n: DT-code of incoming CSI packet) | 'H1100 + 4*n | 'H38383838 | |
| ISPPROC_CUSTOM_FORMAT[0]_CTRL | 'H1010 | 'H17000006 | LUT_LENGTH_MINUS1=23<br>LUT_START_ADDRESS=0<br>POPNUM_MINUS1=6 |
| ISPCS_DI_FILTER_CTRL_CH[4] | 'H3440 | 'H00000002 | Ch4, Y |
| ISPCS_DI_FILTER_LUT[0]_CH[4] | 'H3480 | 'H00000001 | Ch4, Y |
| ISPCS_DI_FILTER_CTRL_CH[5] | 'H3540 | 'H00000002 | Ch5, U |
| ISPCS_DI_FILTER_LUT[0]_CH[5] | 'H3580 | 'H00000002 | Ch5, U |
| ISPCS_DI_FILTER_CTRL_CH[6] | 'H3640 | 'H00000002 | Ch6, V |
| ISPCS_DI_FILTER_LUT[0]_CH[6] | 'H3680 | 'H00000004 | Ch6, V |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[0] | 'H8000 | 'H00E0340C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[1] | 'H8004 | 'H0010000F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[2] | 'H8008 | 'H17000006 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[3] | 'H800C | 'H00701805 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[4] | 'H8010 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[5] | 'H8014 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[6] | 'H8018 | 'H20080200 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[7] | 'H801C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[0] | 'H8020 | 'H01605414 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[1] | 'H8024 | 'H00902017 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[2] | 'H8028 | 'H20002C0A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[3] | 'H802C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[4] | 'H8030 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[5] | 'H8034 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[6] | 'H8038 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[7] | 'H803C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[0] | 'H8040 | 'H01E0741C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[1] | 'H8044 | 'H0110401F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[2] | 'H8048 | 'H20004C12 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[3] | 'H804C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[4] | 'H8050 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[5] | 'H8054 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[6] | 'H8058 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[7] | 'H805C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[0] | 'H8060 | 'H02208420 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[1] | 'H8064 | 'H02509023 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[2] | 'H8068 | 'H01809C26 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[3] | 'H806C | 'H01B06819 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[4] | 'H8070 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[5] | 'H8074 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[6] | 'H8078 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[7] | 'H807C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[0] | 'H8080 | 'H02A0A428 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[1] | 'H8084 | 'H02D0B02B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[2] | 'H8088 | 'H2000BC2E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[3] | 'H808C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[4] | 'H8090 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[5] | 'H8094 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[6] | 'H8098 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[7] | 'H809C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[0] | 'H80A0 | 'H0320C430 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[1] | 'H80A4 | 'H0350D033 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[2] | 'H80A8 | 'H2000DC36 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[3] | 'H80AC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[4] | 'H80B0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[5] | 'H80B4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[6] | 'H80B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[7] | 'H80BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[0] | 'H80C0 | 'H04611444 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[1] | 'H80C4 | 'H0390E047 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[2] | 'H80C8 | 'H03C0EC3A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[3] | 'H80CC | 'H03F0F83D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[4] | 'H80D0 | 'H20080200 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[5] | 'H80D4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[6] | 'H80D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[7] | 'H80DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[0] | 'H80E0 | 'H04E1344C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[1] | 'H80E4 | 'H0411004F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[2] | 'H80E8 | 'H20010C42 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[3] | 'H80EC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[4] | 'H80F0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[5] | 'H80F4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[6] | 'H80F8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[7] | 'H80FC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[0] | 'H8100 | 'H05615454 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[1] | 'H8104 | 'H04912057 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[2] | 'H8108 | 'H20012C4A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[3] | 'H810C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[4] | 'H8110 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[5] | 'H8114 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[6] | 'H8118 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[7] | 'H811C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[0] | 'H8120 | 'H05A16458 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[1] | 'H8124 | 'H05D1705B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[2] | 'H8128 | 'H05017C5E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[3] | 'H812C | 'H05314851 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[4] | 'H8130 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[5] | 'H8134 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[6] | 'H8138 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[7] | 'H813C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[0] | 'H8140 | 'H06218460 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[1] | 'H8144 | 'H06519063 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[2] | 'H8148 | 'H20019C66 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[3] | 'H814C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[4] | 'H8150 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[5] | 'H8154 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[6] | 'H8158 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[7] | 'H815C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[0] | 'H8160 | 'H06A1A468 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[1] | 'H8164 | 'H06D1B06B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[2] | 'H8168 | 'H2001BC6E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[3] | 'H816C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[4] | 'H8170 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[5] | 'H8174 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[6] | 'H8178 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[7] | 'H817C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[0] | 'H8180 | 'H07E1F47C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[1] | 'H8184 | 'H0711C07F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[2] | 'H8188 | 'H0741CC72 | |

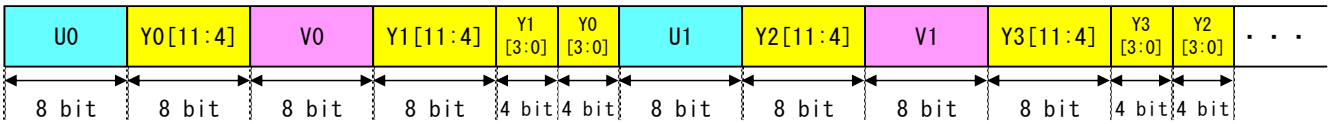
| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[3] | 'H818C | 'H0771D875 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[4] | 'H8190 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[5] | 'H8194 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[6] | 'H8198 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[7] | 'H819C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[0] | 'H81A0 | 'H08621484 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[1] | 'H81A4 | 'H0791E087 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[2] | 'H81A8 | 'H2001EC7A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[3] | 'H81AC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[4] | 'H81B0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[5] | 'H81B4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[6] | 'H81B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[7] | 'H81BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[0] | 'H81C0 | 'H08E2348C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[1] | 'H81C4 | 'H0812008F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[2] | 'H81C8 | 'H20020C82 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[3] | 'H81CC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[4] | 'H81D0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[5] | 'H81D4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[6] | 'H81D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[7] | 'H81DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[0] | 'H81E0 | 'H09224490 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[1] | 'H81E4 | 'H09525093 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[2] | 'H81E8 | 'H08825C96 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[3] | 'H81EC | 'H08B22889 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[4] | 'H81F0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[5] | 'H81F4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[6] | 'H81F8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[7] | 'H81FC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[0] | 'H8200 | 'H09A26498 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[1] | 'H8204 | 'H09D2709B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[2] | 'H8208 | 'H20027C9E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[3] | 'H820C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[4] | 'H8210 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[5] | 'H8214 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[6] | 'H8218 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[16]_B[7] | 'H821C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[0] | 'H8220 | 'H0A2284A0 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[1] | 'H8224 | 'H0A5290A3 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[2] | 'H8228 | 'H20029CA6 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[3] | 'H822C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[4] | 'H8230 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[5] | 'H8234 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[6] | 'H8238 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[17]_B[7] | 'H823C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[0] | 'H8240 | 'H0B62D4B4 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[1] | 'H8244 | 'H0A92A0B7 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[2] | 'H8248 | 'H0AC2ACAA | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[3] | 'H824C | 'H0AF2B8AD | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[4] | 'H8250 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[5] | 'H8254 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[6] | 'H8258 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[18]_B[7] | 'H825C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[0] | 'H8260 | 'H0BE2F4BC | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[1] | 'H8264 | 'H0B12C0BF | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[2] | 'H8268 | 'H2002CCB2 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[3] | 'H826C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[4] | 'H8270 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[5] | 'H8274 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[6] | 'H8278 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[19]_B[7] | 'H827C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[0] | 'H8280 | 'H0C6314C4 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[1] | 'H8284 | 'H0B92E0C7 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[2] | 'H8288 | 'H2002ECBA | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[3] | 'H828C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[4] | 'H8290 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[5] | 'H8294 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[6] | 'H8298 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[20]_B[7] | 'H829C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[0] | 'H82A0 | 'H0CA324C8 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[1] | 'H82A4 | 'H0CD330CB | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[2] | 'H82A8 | 'H0C033CCE | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[3] | 'H82AC | 'H0C3308C1 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[4] | 'H82B0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[5] | 'H82B4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[6] | 'H82B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[21]_B[7] | 'H82BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[0] | 'H82C0 | 'H0D2344D0 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[1] | 'H82C4 | 'H0D5350D3 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[2] | 'H82C8 | 'H20035CD6 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[3] | 'H82CC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[4] | 'H82D0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[5] | 'H82D4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[6] | 'H82D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[22]_B[7] | 'H82DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[0] | 'H82E0 | 'H0DA364D8 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[1] | 'H82E4 | 'H0DD370DB | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[2] | 'H82E8 | 'H20037CDE | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[3] | 'H82EC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[4] | 'H82F0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[5] | 'H82F4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[6] | 'H82F8 | 'H20080200 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[23]_B[7] | 'H82FC | 'H20080200 | |

### 5. YUV422, Y12, UV8

Incoming pixel sequence which is packed in one CSI long packet is shown as below.




Configuration of LUT based pixel reconstructor

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROCMODE_DT[n]<br>(n: DT-code of incoming CSI packet) | 'H1100 + 4*n | 'H38383838 | |
| ISPPROC_CUSTOM_FORMAT[0]_CTRL | 'H1010 | 'H0F000004 | LUT_LENGTH_MINUS1=15<br>LUT_START_ADDRESS=0<br>POPNUM_MINUS1=4 |
| ISPCS_DI_FILTER_CTRL_CH[4] | 'H3440 | 'H00000003 | Ch4, Y0,Y1 |
| ISPCS_DI_FILTER_LUT[0]_CH[4] | 'H3480 | 'H00000005 | Ch4, Y0,Y1 |
| ISPCS_DI_FILTER_CTRL_CH[5] | 'H3540 | 'H00000003 | Ch5, U0 |
| ISPCS_DI_FILTER_LUT[0]_CH[5] | 'H3580 | 'H00000002 | Ch5, U0 |
| ISPCS_DI_FILTER_CTRL_CH[6] | 'H3640 | 'H00000003 | Ch6, V0 |
| ISPCS_DI_FILTER_LUT[0]_CH[6] | 'H3680 | 'H00000008 | Ch6, V0 |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[0] | 'H8000 | 'H02208420 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[1] | 'H8004 | 'H00902023 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[2] | 'H8008 | 'H0F000004 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[3] | 'H800C | 'H00F0380D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[4] | 'H8010 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[5] | 'H8014 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[6] | 'H8018 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[0]_B[7] | 'H801C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[0] | 'H8020 | 'H00200400 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[1] | 'H8024 | 'H00501003 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[2] | 'H8028 | 'H20001C06 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[3] | 'H802C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[4] | 'H8030 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[5] | 'H8034 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[6] | 'H8038 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[1]_B[7] | 'H803C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[0] | 'H8040 | 'H02609424 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[1] | 'H8044 | 'H01906027 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[2] | 'H8048 | 'H01C06C1A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[3] | 'H804C | 'H01F0781D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[4] | 'H8050 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[5] | 'H8054 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[6] | 'H8058 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[2]_B[7] | 'H805C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[0] | 'H8060 | 'H01204410 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[1] | 'H8064 | 'H01505013 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[2] | 'H8068 | 'H20005C16 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[3] | 'H806C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[4] | 'H8070 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[5] | 'H8074 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[6] | 'H8078 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[3]_B[7] | 'H807C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[0] | 'H8080 | 'H04A12448 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[1] | 'H8084 | 'H0310C04B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[2] | 'H8088 | 'H0340CC32 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[3] | 'H808C | 'H0370D835 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[4] | 'H8090 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[5] | 'H8094 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[6] | 'H8098 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[4]_B[7] | 'H809C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[0] | 'H80A0 | 'H02A0A428 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[1] | 'H80A4 | 'H02D0B02B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[2] | 'H80A8 | 'H2000BC2E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[3] | 'H80AC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[4] | 'H80B0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[5] | 'H80B4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[6] | 'H80B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[5]_B[7] | 'H80BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[0] | 'H80C0 | 'H04E1344C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[1] | 'H80C4 | 'H0411004F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[2] | 'H80C8 | 'H04410C42 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[3] | 'H80CC | 'H04711845 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[4] | 'H80D0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[5] | 'H80D4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[6] | 'H80D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[6]_B[7] | 'H80DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[0] | 'H80E0 | 'H03A0E438 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[1] | 'H80E4 | 'H03D0F03B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[2] | 'H80E8 | 'H2000FC3E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[3] | 'H80EC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[4] | 'H80F0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[5] | 'H80F4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[6] | 'H80F8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[7]_B[7] | 'H80FC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[0] | 'H8100 | 'H0721C470 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[1] | 'H8104 | 'H05916073 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[2] | 'H8108 | 'H05C16C5A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[3] | 'H810C | 'H05F1785D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[4] | 'H8110 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[5] | 'H8114 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[6] | 'H8118 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[8]_B[7] | 'H811C | 'H20080200 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[0] | 'H8120 | 'H05214450 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[1] | 'H8124 | 'H05515053 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[2] | 'H8128 | 'H20015C56 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[3] | 'H812C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[4] | 'H8130 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[5] | 'H8134 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[6] | 'H8138 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[9]_B[7] | 'H813C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[0] | 'H8140 | 'H0761D474 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[1] | 'H8144 | 'H0691A077 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[2] | 'H8148 | 'H06C1AC6A | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[3] | 'H814C | 'H06F1B86D | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[4] | 'H8150 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[5] | 'H8154 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[6] | 'H8158 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[10]_B[7] | 'H815C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[0] | 'H8160 | 'H06218460 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[1] | 'H8164 | 'H06519063 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[2] | 'H8168 | 'H20019C66 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[3] | 'H816C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[4] | 'H8170 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[5] | 'H8174 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[6] | 'H8178 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[11]_B[7] | 'H817C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[0] | 'H8180 | 'H09A26498 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[1] | 'H8184 | 'H0812009B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[2] | 'H8188 | 'H08420C82 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[3] | 'H818C | 'H08721885 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[4] | 'H8190 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[5] | 'H8194 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[6] | 'H8198 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[12]_B[7] | 'H819C | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[0] | 'H81A0 | 'H07A1E478 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[1] | 'H81A4 | 'H07D1F07B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[2] | 'H81A8 | 'H2001FC7E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[3] | 'H81AC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[4] | 'H81B0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[5] | 'H81B4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[6] | 'H81B8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[13]_B[7] | 'H81BC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[0] | 'H81C0 | 'H09E2749C | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[1] | 'H81C4 | 'H0912409F | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[2] | 'H81C8 | 'H09424C92 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[3] | 'H81CC | 'H09725895 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[4] | 'H81D0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[5] | 'H81D4 | 'H20080200 | |

| Register Name | Offset Address | Setting Value | Remarks |
|---|---|---|---|
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[6] | 'H81D8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[14]_B[7] | 'H81DC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[0] | 'H81E0 | 'H08A22488 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[1] | 'H81E4 | 'H08D2308B | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[2] | 'H81E8 | 'H20023C8E | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[3] | 'H81EC | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[4] | 'H81F0 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[5] | 'H81F4 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[6] | 'H81F8 | 'H20080200 | |
| ISPPROC_CUSTOM_FORMAT_LUT0_W[15]_B[7] | 'H81FC | 'H20080200 | |

### 54.4.3 Boot up sequence

#### 54.4.3.1 Boot up sequence type 1

Flow of boot up sequence Type1 is shown in the Figure 54.19

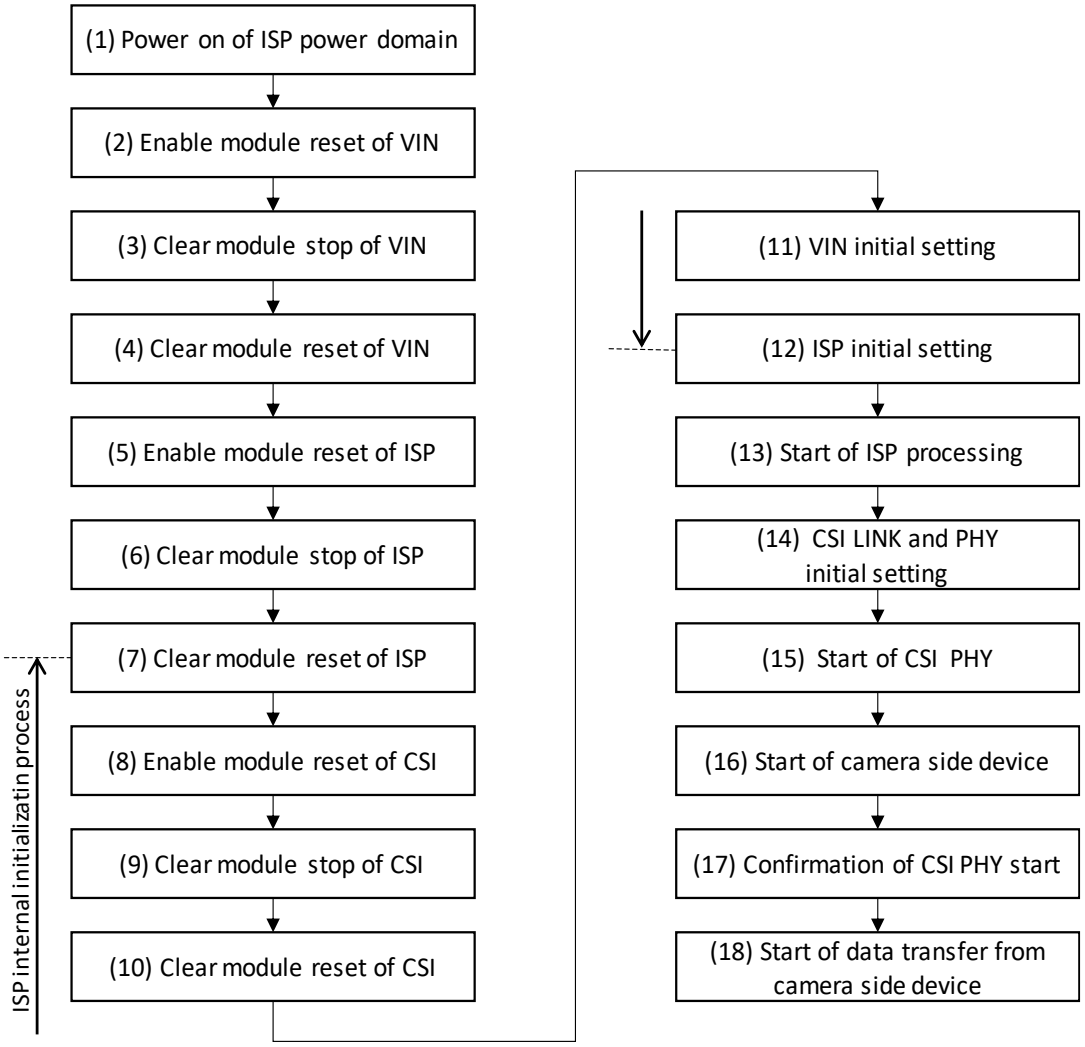



Figure 54.19 Flow of boot up sequence type 1

**1. Power on of ISP**

ISP is located in a dedicated power domain. When the boot up of SoC, ISP power domain is shut-off state. Before using ISP, an power supply operation for ISP power domain is required. Refer to the “Power Domain power-ON Control Register n (PDRONCRn)” register section in the hardware manual of SYSC.

**2. Enable module reset of VIN**

An explicit module resetting to VIN is done before clear module stop. See the hardware manual of CPG for detail.

**3. Clear module stop of VIN**

At the boot up of SoC, VIN, ISP and CSI are in the state of module stop. During module stop state, clock signal is not supplied. Explicitly clear the module stop state to supply clock to VIN. If this operation is skipped, any access to VIN are locked and it may bring hang up.See the hardware manual of CPG for detail.

**4. Clear module reset of VIN**

At this stage clock supply is enabled. Explicitly clear the reset of VIN.

**5. Enable module reset of ISP**

An explicit module resetting to ISP is done before clear module stop. See the hardware manual of CPG for detail.

**6. Clear module stop of ISP**

Explicitly clear the module stop state to supply clock to ISP. If this operation is skipped, any access to ISP are locked and it may bring hang up. See the hardware manual of CPG for detail.

**7. Clear module reset of ISP**

At this stage clock supply is enabled. Explicitly clear the reset of ISP. See the hardware manual of CPG for detail. At the timing of reset clear, ISP starts internal initialization processing. User shall wait for completion of this process, and see 54.4.3.1.12 or 54.4.3.2.16 for the actual value of it.

**8. Enable module reset of CSI**

An explicit module resetting to CSI is done before clear module stop. See the hardware manual of CPG for detail.

**9. Clear module stop of CSI**

Explicitly clear the module stop state to supply clock to CSI. If this operation is skipped, any access to CSI are locked and it may bring hang up. See the hardware manual of CPG for detail.

**10. Clear module reset of CSI**

At this stage clock supply is enabled. Explicitly clear the reset of CSI.

**11. VIN initial setting**

Register setting to capture using VIN, such as image size, stride and base address, color format, and etc. See the hardware manual of VIN for more information

**12. ISP initial setting**

Register setting of both ISP core and ISP Interface. See 54.4.1 and 54.4.2 for more information.

Before starting this step, over 6.250 [us] (*) waiting period from clear timing of module reset is required due to internal initialization processing of the ISP. User shall wait for completion of this period before configuring the ISP. Since there are no registers to indicate the completion, user shall use a system timer to obtain a time.

(*) This waiting period corresponds to 5000 cycles when frequency of S0D1_VIO $\phi$ clock is 800 MHz. If the frequency of S0D1_VIO $\phi$ clock is not 800 MHz, then it is required to recalculate this waiting period by the following formula. In this formula, *clock_frquency* denotes the frequency of S0D1_VIO $\phi$ clock and *waiting_period* denotes the waiting period to be recalculated.

*waiting_period* [us] = 5000 / *clock_frequency* [MHz]

**13. Start of ISP processing**

ISP processing is started by writing ISPSTART register. For register access while ISP is running, see 54.5.2.

**14. CSI LINK and PHY initial setting**

**15. Start of cSI PHY**

**16. Start of camera side device**

**17. Confirmation of CSI PHY start**

(14)-(17) are steps to use external camera device which are connected via CSI interface. See the hardware manual of CSI for more information.

**18. Start of data transfer from external camera side device**

This step is to enable data transfer from external camera side devices to provide an image data to the SoC. See the manual of camera(sensor) module for more information.

#### 54.4.3.2 Boot up sequence type 2

In the “Boot up sequence Type 1”, “Confirmation of CSI PHY available” takes longer time than other steps. This method overlaps ISP initialization process on the “Confirmation of CSI PHY available” step in order to shorten boot up time. Boot up sequence of this method is shown **Figure 54.20**

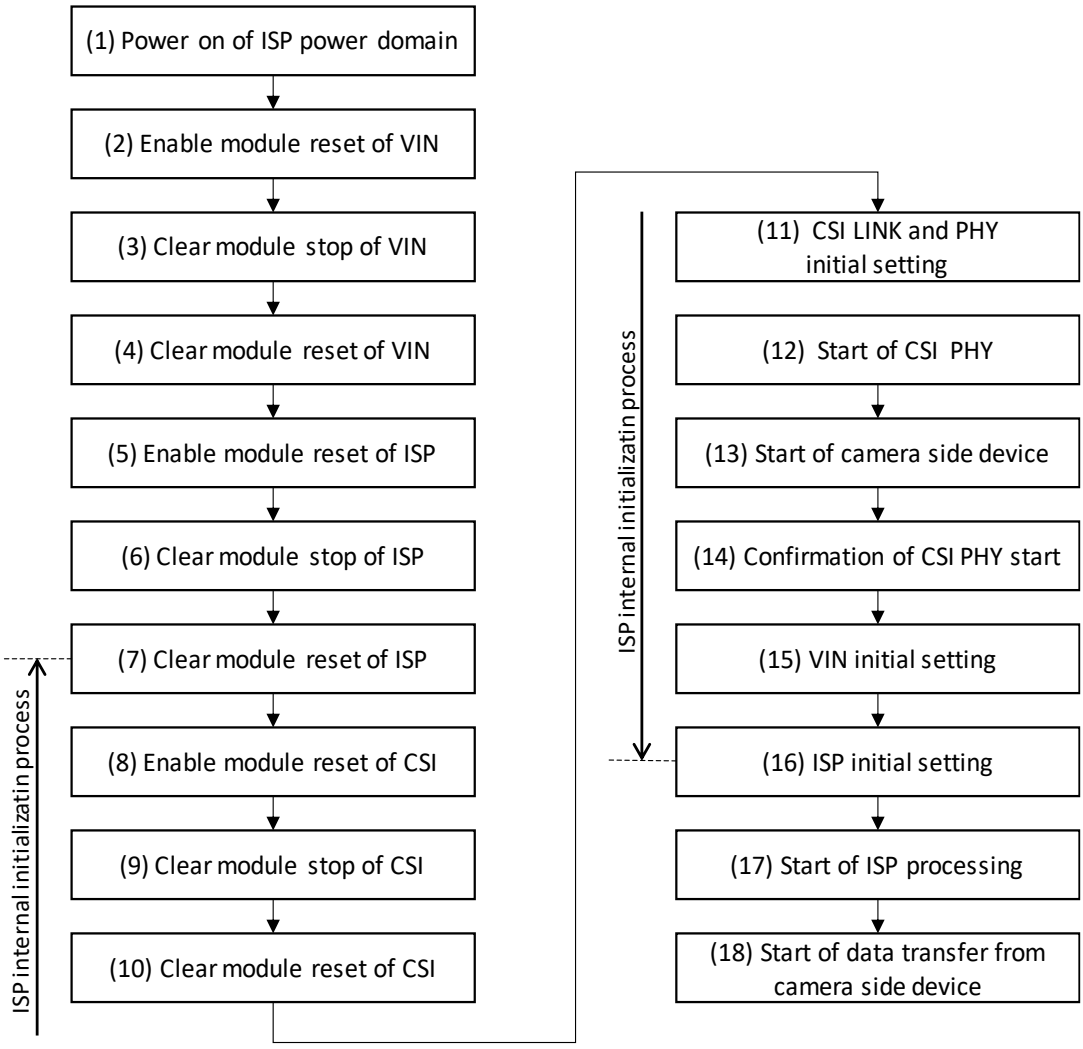



**Figure 54.20 Flow of boot up sequence type 2**

1. **Power on of ISP**
2. **Enable module reset of VIN**
3. **Clear module reset of VIN**
4. **Clear module reset of VIN**
5. **Enable module reset of ISP**
6. **Clear module stop of ISP**
7. **Clear module reset of ISP**
8. **Enable module reset of CSI**
9. **Clear module stop of CSI**
10. **Clear module reset of CSI**

Detail of each step of (1)-(10) are same as the “Boot up sequence type 1”.

11. **CSI LINK and PHY initial setting**
12. **Start of CSI PHY**
13. **Start of camera side device**
14. **Confirmation of CSI PHY start**

(11)-(14) are steps to use external camera device which are connected via CSI interface. See the hardware manual of CSI for more information.

15. **VIN initial setting**

Register setting to capture using VIN, such as image size, stride and base address, color format, and etc. See the hardware manual of VIN for more information.

16. **ISP initial setting**

Register setting of both ISP core and ISP Interface. See 54.4.1 and 54.4.2 for more information.

Before starting this step, over 6.250 [us] (*) waiting period from clear timing of module reset is required due to internal initialization processing of the ISP. User shall wait for completion of this period before configuring the ISP. Since there are no registers to indicate the completion, user shall use a system timer to obtain a time.

(*) This waiting period corresponds to 5000 cycles when frequency of S0D1_VIO $\phi$ clock is 800 MHz. If the frequency of S0D1_VIO $\phi$ clock is not 800 MHz, then it is required to recalculate this waiting period by the following formula. In this formula, *clock_frquency* denotes the frequency of S0D1_VIO $\phi$ clock and *waiting_period* denotes the waiting period to be recalculated.

*waiting_period* [us] = 5000 / *clock_frequency* [MHz]

17. **Start of ISP processing**

ISP processing is started by writing ISPSTART register. For register access while ISP is running, see 54.5.2.

18. **Start of data transfer from external camera side device**

This step is to enable data transfer from external camera side devices to provide an image data to the SoC. See the manual of camera(sensor) module for more information.

### 54.4.4 Stop sequence

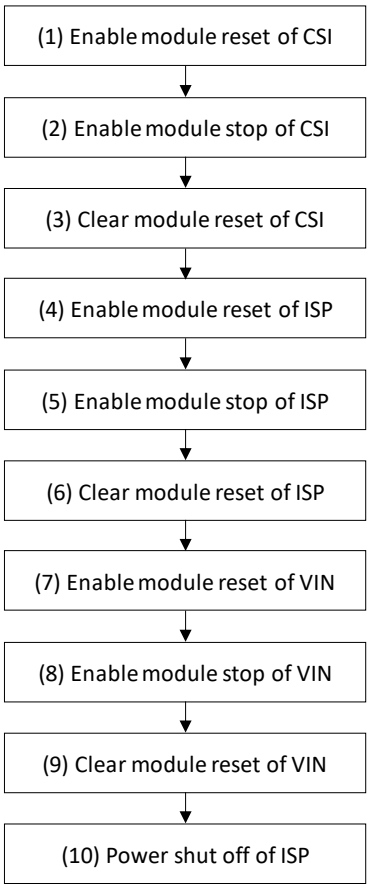



Figure 54.21 Stop sequence

**1. Enable module reset of CSI**

By enabling module reset, CSI processing is stopped. Detailed operation is same as the corresponding step in the boot up sequence flow.

**2. Enable module stop of CSI**

By enabling module stop, clock supply is stopped. See the hardware manual of CPG for detail.

**3. Clear module reset of CSI**

Module reset is cleared for the boot up at next time. Detailed operation is same as the corresponding stop in boot up sequence flow

**4. Enable module reset of ISP**

By enabling module reset, ISP processing is stopped. Detailed operation is same as the corresponding step in the boot up sequence flow.

**5. Enable module stop of ISP**

By enabling module stop, clock supply is stopped. See the hardware manual of CPG for detail.

**6. Clear module reset of ISP**

Module reset is cleared for the boot up at next time. Detailed operation is same as the corresponding stop in boot up sequence flow

**7. Enable module reset of VIN**

By enabling module reset, ISP processing is stopped. Detailed operation is same as the corresponding step in the boot up sequence flow.

**8. Enable module stop of VIN**

By enabling module stop, clock supply is stopped. See the hardware manual of CPG for detail.

**9. Clear module reset of VIN**

Module reset is cleared for the boot up at next time. Detailed operation is same as the corresponding stop in boot up sequence flow

**10. Power shut off of ISP**

After ISP has been stopped, power domain of ISP shall be shut-off. Refer to the "Power Domain power-OFF Control Register n (PDROFFCRn)" register section in the hardware manual of SYSC.

See the hardware manual of SYSC for detail.

Note: If this procedure applied, VIN memory writing finish in the middle of the frame. If it needs to finish VIN in frames, see the hardware manual of VIN for detail.

### 54.4.5 Synchronized system with IMR

ISP supports synchronized system with IMR. In this function, ISP issues a start signal called as "WUP signal" to IMR during one frame processing is still on-going. IMR can start image rendering processing without waiting a completion of one frame. Therefore, low delay processing can be realized. However, take care that you should boot up IMR before ISP when you use this function. See the hardware manual of IMR for more information.

#### 54.4.5.1 Configuration register

- ENABLE_WUP
  - ENABLE_WUP reigster specifies enabling of synchronized system with IMR. In order to activate this function, ENABLE_WUP should be set to 1.
- WUP_ENABLE_CHANNELS
  - WUP_ENABLE_CHANNELS register controls whether IMR channel receives WUP pulse from ISP. Each bit corresponds to one IMR channel. In order to issue a WUP signal to IMR, corresponsinding bits of WUP_ ENABLE_CHANNELS should be set to 1 in addition to set ENABLE_WUP = 1. For example, when WUP_ENABLE_CHANNELS = H'1F , all IMR channels receive the WUP signal
  - In R-Car V4M, WUP_ENABLE_CHANNELS[5:4] is no effect (dummy bit) . Only WUP_ENABLE_ CHANNEL[3:0] is available because number of IMR channels is 4 in R-Car V4M.
- WUP_PULSE_LENGTH_MINUS1
  - This register specifies a subtraced value of number of cycles of WUP signal pulse (WUP signal is N-cycled pulse). For example, when WUP_PULSE_LENGTH=1, pulse length is 2 cycles.
  - 5 cycles is recommended as WUP signal pulse length. Recommended value for WUP_PULSE_LENGTH register is 4.
- WUP_POS_Xn (n=0,1,2,3), WUP_POS_Yn (n=0,1,2,3)

- This register specifies the timing for issuing WUP pulse. Issue timing is specified by the coordinate value of pixel of output human vision image. However, issue timing at (WUP_POS_Xn, WUP_POS_Yn)=(0, 0) is not supported. Therefore, setting of both WUP_POS_Xn==0 and WUP_POS_Yn=0 is forbidden.
- The coordinates for issuing WUP pulses must be spaced a certain distance (at least several lines) from each other. So, setting to specify multiple WUP pulse coordinates on a single line is prohibited.
- WUP pulse can be issued up to 4 times per frame.
- For example, in order to issue WUP pulse at 3 times per one frame and its issueing timing is at outputting (x0,y0) pixels, (x1,y1) pixels or (x2,y2) pixels, setting of WUP_POS_Xn and WUP_POS_Yn register is whowns as below. The number of WUP pulse can be specified by setting a value larger than output image size in the WUP_POS_Xn or WUP_POS_Yn. In this example, (WUP_POS_X3, WUP_POS_Y3) is set as (H'FFFF, H'FFFF), so WUP_POS_X3 and WUP_POS_Y3 is not effected to the WUP pulse issuing.

| Register | Setting value | Register name | Setting value |
|---|---|---|---|
| WUP_POS_X0 | x0 | WUP_POS_Y0 | y0 |
| WUP_POS_X1 | x1 | WUP_POS_Y1 | y1 |
| WUP_POS_X2 | x2 | WUP_POS_Y2 | y2 |
| WUP_POS_X3 | H'FFFF | WUP_POS_Y3 | H'FFFF |

Figure 54.22 shows a timing chart of WUP pulse issue timing in one frame

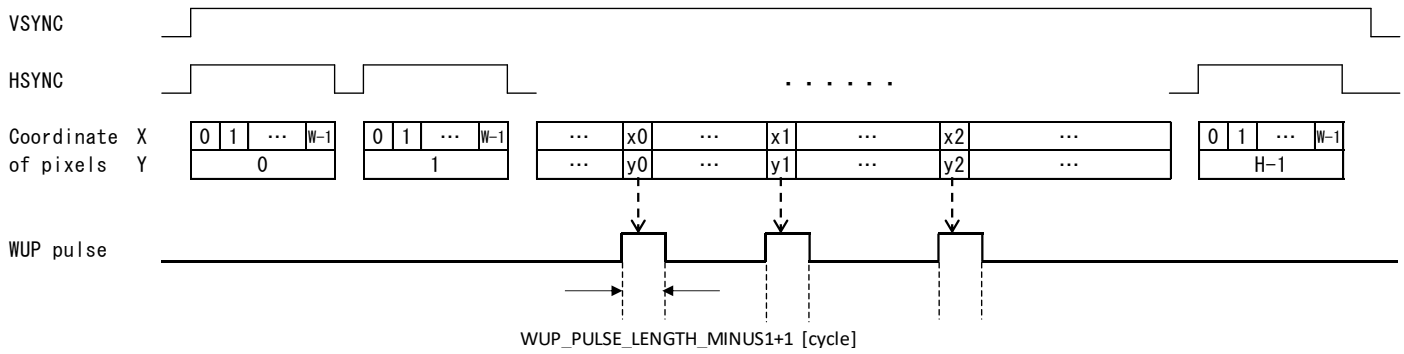



Figure 54.22 Timing chart of WUP pulse issue

### 54.4.5.2 WOVF and WERR error at IMR

IMR may notify either WOVF error or WERR error to CPU as interrupt request. When the synchronization timing between ISP and IMR is not correct. CPU should apply appropriate action when receiving these errors.

1. WOVF error

Processing start timing for each frame of ISP is in accordance with incoming video timing from external connected sensors. ISP does not have a function which corresponds to SLP instruction in IMR (SLP instruction is to wait start of image rendering processing). Because ISP cannot wait image processing there are cases where ISP issues 2 WUP signals continuously. In this case IMR notifies WOVF error. If WOVF error is issued, you should stop ISP processing after confirming that IMR has been terminated by TRAP instruction. Refer to the HW manual of IMR for more information.

2. WERR error

WERR error is notified when the following either (i) or (ii) cases.

(i) WUP signal is issued when boot up sequence and initialization of IMR has not been completed.

(ii) WUP signal is issued when the interrupt service routine caused by IMR interrupt has not been completed.

If WERR is issued in case of (i), then you should stop the ISP processing without boot up of IMR. If WERR error is issued in case of (ii), then you should stop the ISP processing after confirming that IMR has been terminated by TRAP instruction. Refer to the HW manual of IMR for more information.

### 54.4.6 Interrupts

#### 54.4.6.1 Interrupt from ISP Interface/Channel Selector

**1. Interrupt sources**

Interrupt source is stored in the ISPINT_STATUS register. This register is read only.

**2. Enabling method**

When writing 1 into a particular bit in ISPINT_ENABLE register, then interrupt signal is asserted to INTC (Interrupt Controller) at the timing of assertion of corresponding interrupt source.

Example： Setting to notify INTC of an interrupt when tither RAW_SCALER_PIXEL_DROP or FIFO_FULL_CSI_INPUT interrupt source occurs. In this setting, even if other interrupt source is asserted, interrupt is not notified to INTC.

| Register Name | Writing value | Remarks |
|---|---|---|
| ISPINT_ENABLE | H’0001_00004 | bit16: RAW_SCALER_PIXEL_DROP<br>bit2: FIFO_FULL_CSI_INPUT |

**3. Clear method**

When writing 1 into a particular bit in ISPINT_CLEAR, then corresponding interrupt status bits in ISPINT_STATUS register are cleared.

Example：To clear RAW_SCALER_PIXEL_DROP and FIFO_FULL_CSI_INPUT interrupt status.

| Register name | Writing value | Remarks |
|---|---|---|
| ISPINT_CLEAR | H’0001_0004 | bit16: RAW_SCALER_PIXEL_DROP<br>bit2: FIFO_FULL_CSI_INPUT |

**4. Connection structure**

Connection structure of interrupt status register (ISPINT_STATUS) and interrupt enable register (ISPINT_ENABLE) is shown in Figure 54.23

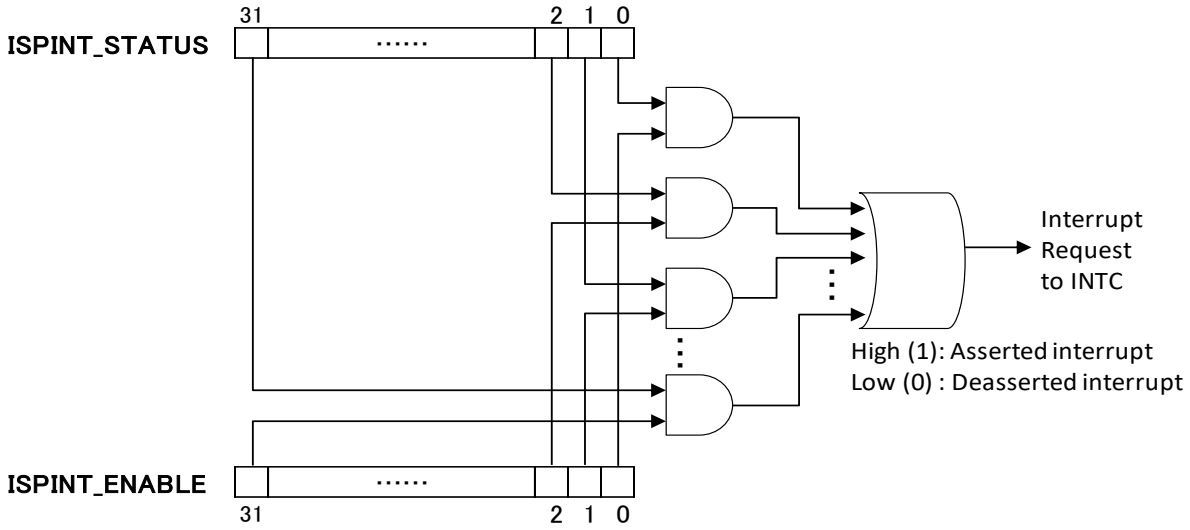



**Figure 54.23 Connection structure of interrupt register signals**

### 5. Bit assignment of Interrupt

| Bit | Status Register | Clear Register | Enable Register | Interrupt Source |
|---|---|---|---|---|
| 0 | ST0 | CLR0 | EN0 | (Not used) |
| 1 | ST1 | CLR1 | EN1 | (Not used) |
| 2 | ST2 | CLR2 | EN2 | FIFO_FULL_CSI_INPUT. Full status of the input fifo for CSI interface |
| 3 | ST3 | CLR3 | EN3 | (Not used) |
| 4 | ST4 | CLR4 | EN4 | (Not used) |
| 5 | ST5 | CLR5 | EN5 | (Not used) |
| 6 | ST6 | CLR6 | EN6 | (Not used) |
| 7 | ST7 | CLR7 | EN7 | (Not used) |
| 8 | ST8 | CLR8 | EN8 | (Not used) |
| 9 | ST9 | CLR9 | EN9 | (Not used) |
| 10 | ST10 | CLR10 | EN10 | (Not used) |
| 11 | ST11 | CLR11 | EN11 | (Not used) |
| 12 | ST12 | CLR12 | EN12 | (Not used) |
| 13 | ST13 | CLR13 | EN13 | (Not used) |
| 14 | ST14 | CLR14 | EN14 | (Not used) |
| 15 | ST15 | CLR15 | EN15 | (Not used) |
| 16 | ST16 | CLR16 | EN16 | RAW_SCALER_PIXEL_DROP<br>Asserted when a pixel drop at RAW scaler input is happened. |
| 17 | ST17 | CLR17 | EN17 | (Not used) |
| 18 | ST18 | CLR18 | EN18 | (Not used) |
| 19 | ST19 | CLR19 | EN19 | (Not used) |
| 20 | ST20 | CLR20 | EN20 | (Not used) |
| 21 | ST21 | CLR21 | EN21 | (Not used) |
| 22 | ST22 | CLR22 | EN22 | (Not used) |
| 23 | ST23 | CLR23 | EN23 | (Not used) |
| 24 | ST24 | CLR24 | EN24 | (Not used) |
| 25 | ST25 | CLR25 | EN25 | (Not used) |
| 26 | ST26 | CLR26 | EN26 | (Not used) |
| 27 | ST27 | CLR27 | EN27 | (Not used) |
| 28 | ST28 | CLR28 | EN28 | (Not used) |
| 29 | ST29 | CLR29 | EN29 | (Not used) |
| 30 | ST30 | CLR30 | EN30 | (Not used) |
| 31 | ST31 | CLR31 | EN31 | (Not used) |

Figure 54.24 Bit assignment of ISP interrupt

#### 54.4.6.2 Interrupt from ISP Core

Interrupt request from ISP core is connected to ISPCORE_INT_STATUS.ST0 register (bit 0). By reading ISPCORE_INT_STATUS.ST0, it is possible to obverve whether interrupt request from ISP core is issueed or not.

In order to issue interrupt to INTC, ISPCORE_INT_ENABLE.EN0 is set to 1.

Regarding how to clear interrupt request or more detail, refer to the manual of ISP core released from IP vendor.

In addition to interrupt from ISP core, interrupt from “Defect Pixel Information DMA” is connected to ISPCORE_INT_ STATUS.

Bit 18 is assigned to ISPCORE_INT_STATUS.DPCC_PIXEL_COUNT_ OVER_DPCC1 interrupt. In order to issue an interrupt to INTC from this interrupt source, ISPCORE_INT_ENABLE. DPCC_PIXEL_COUNT_ OVER_DPCC1 is set to 1.

Bit 19 is is assigned to ISPCORE_INT_STATUS.DPCC_PIXEL_COUNT_ OVER_DPCC2 interrupt. In order to issue an interrupt to INTC from this interrupt source, ISPCORE_INT_ENABLE. DPCC_PIXEL_COUNT_ OVER_DPCC2 is set to 1.

See 54.4.11.3 for more detail.

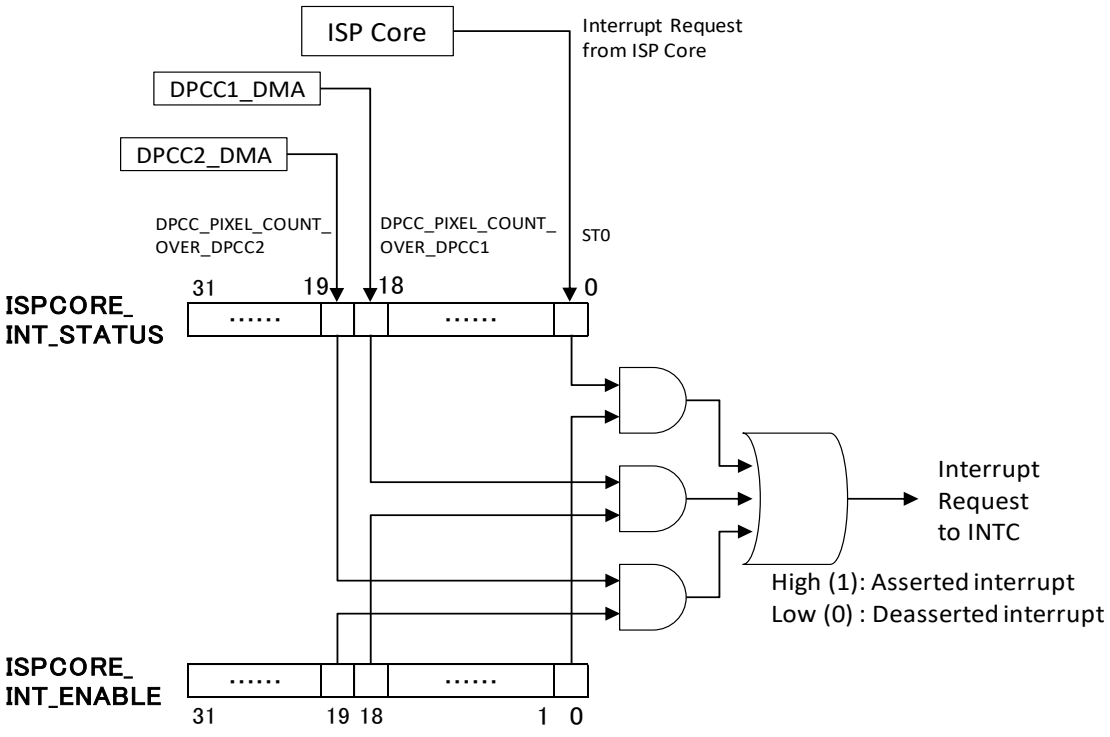



Figure 54.25 Connection structure of interrupt register from ISP core

### 54.4.7 Error Request

#### 54.4.7.1 Error request of ISP Interface/Chanel Selector

**1. Error request sources**

Error request sources from ISP Interface/Channel Selector are stored in ISPERR1_STATUS register or ISPERR2_STATUS register. This registers are read only.

**2. Enabling method**

When writing 1 to particular bits in ISPERR1_ENABLE or ISPERR2_ENABLE register, then an error request signal is asserted to ECM (Error Control Module) at the timing of assertion of corresponding error source.

Example: Setting to notify ECM of an error request when either CHANNEL_SELECT_CH9_PIXEL_COUNT_ MIN_ERROR (bit 31 in ISPERR1) or SYNC_ERROR_VC14 (bit 14 in ISPERR2) error source occurs.

| Register Name | Writing value | Remarks |
|---|---|---|
| ISPERR1_ENABLE | H’8000_0000 | bit31: CHANNEL_SELECT_CH9_PIXELCOUNT_MIN_ERROR |
| ISPERR2_ENABLE | H’0000_4000 | bit 14: SYNC_ERROR_VC14 |

**3. Clear method**

When writing 1 to particular bits in ISPERR1_CLEAR or ISPERR2_CLEAR register, then corresponding error status bits in ISPERR1_STATUS or ISPERR2_STATUS register are cleared to 0.

Example: To clear CHANNEL_SELECT_CH9_PIXEL_COUNT_ MIN_ERROR (bit 31 in ISPERR1) or SYNC_ERROR_VC14 (bit 14 in ISPERR2) error status

| Register Name | Writing value | Remarks |
|---|---|---|
| ISPERR1_CLEAR | H’8000_0000 | bit31: CHANNEL_SELECT_CH9_PIXELCOUNT_MIN_ERROR |
| ISPERR2_CLEAR | H’0000_4000 | bit 14: SYNC_ERROR_VC14 |

**4. Connection structure**

Connection structure of error status register (ISPERR1_STATUS, ISPERR2_STATUS) and error enable register (ISPERR1_ENABLE, ISPERR2_ENALBLE) is shown in Figure 54.26.

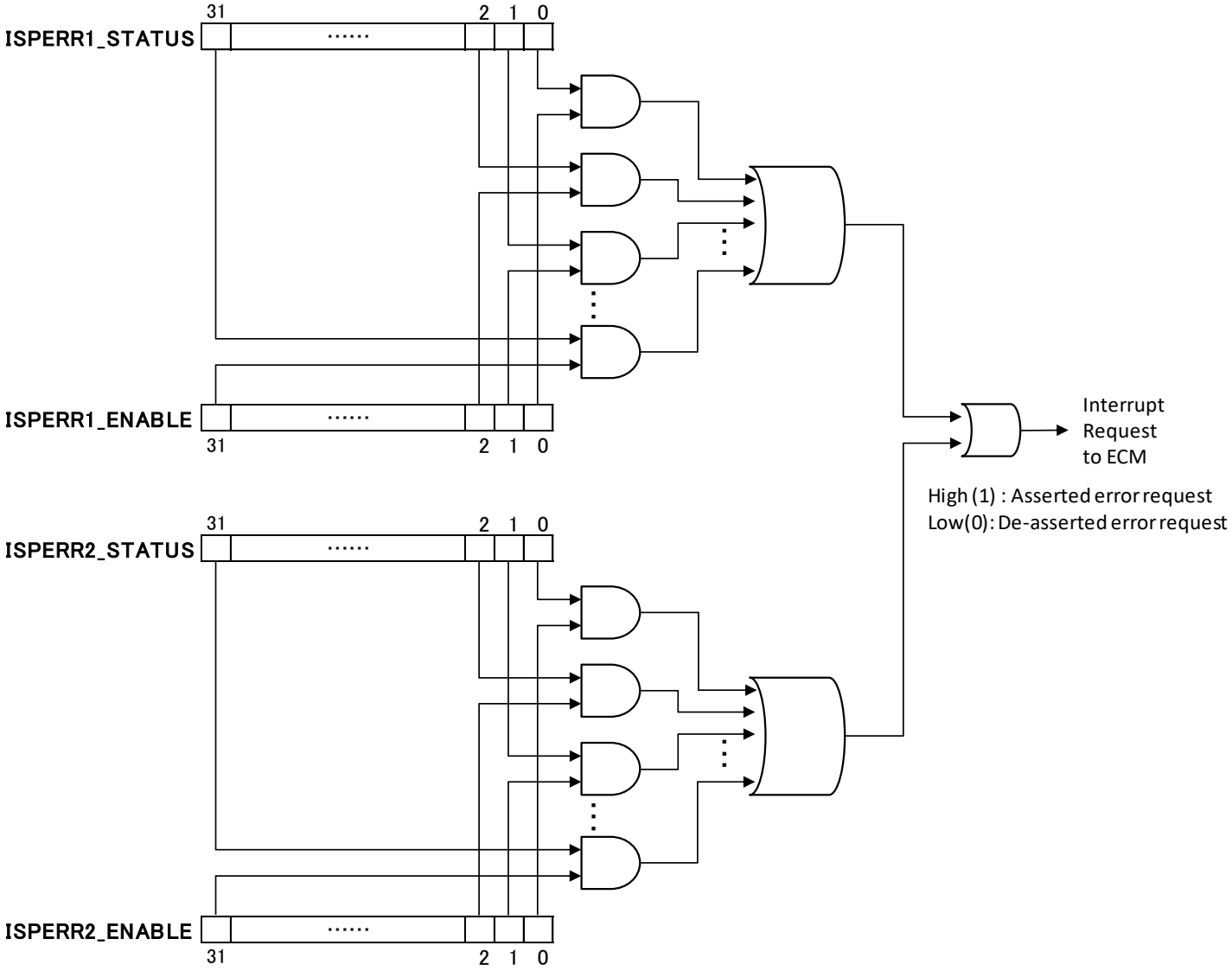



**Figure 54.26 Connection structure of error request from ISP Interface**

#### 54.4.7.2 Error request sourced from ISP core

**1. Error request sources**

Error request sources from ISP core are stored in ISPERR0_STATUS or ISPERR3STATUS or ISPERR4_STATUS register. There registers are read only.

**2. Enabling method**

When writing 1 to particular bits in ISPERR0_ENABLE or ISPERR3_ENABLE or ISPERR4_ENABLE register, then an error request signal is asserted to ECM (Error Control Module) at the timing of assertion of corresponding error source

Example: Setting to notify ECM of an error request when either ISP Core SRAM EDC error (RPP_MAIN_PRE2_LSC _GB_MEM) (bit 12 in ISPERR0) or ISP Core SRAM EDC error (RPP_MAIN_PRE1_LSC_B_MEM) (bit 3 in ISPERR0) or ISP Core SRAM EDC error (RPP_MAIN_POST_SHRPCNR_MEM) (bit 7 in ISPERR3) error source occurs.

| Register Name | Writing value | Remarks |
|---|---|---|
| ISPERR0_ENABLE | H'0000_1008 | bit12: ISP Core SRAM EDC error ( RPP_MAIN_PRE2_LSC_GB_MEM) |

| | | |
|---|---|---|
| | | bit 3: ISP Core SRAM EDC error (RPP_MAIN_PRE1_LSC_B_MEM) |
| ISPERR3_ENABLE | H'0000_0080 | bit 7: ISP Core SRAM EDC error (RPP_MAIN_POST_SHRPCNR_MEM) |

### 3. Clear method

When writing 1 to particular bits in ISPERR0_CLEAR or ISPERR3_CLEAR or ISPERR4_CLEAR register, then corresponding error status bits in ISPERR0_STATUS or ISPERR3_STATUS or ISPERR4_STATUS register are cleared to 0.

Example: To clear ISP Core SRAM EDC error (RPP_MAIN_PRE2_LSC _GB_MEM) (bit 12 in ISPERR0) and ISP Core SRAM EDC error (RPP_MAIN_PRE1_LSC_B_MEM) (bit 3 in ISPERR0) and ISP Core SRAM EDC error (RPP_MAIN_POST_SHRPCNR_MEM) (bit 7 in ISPERR3) error status

| Register Name | Writing value | Remarks |
|---|---|---|
| ISPERR0_CLEAR | H'0000_1008 | bit12: ISP Core SRAM EDC error ( RPP_MAIN_PRE2_LSC_GB_MEM)<br>bit 3: ISP Core SRAM EDC error (RPP_MAIN_PRE1_LSC_B_MEM) |
| ISPERR3_CLEAR | H'0000_0080 | bit 7: ISP Core SRAM EDC error (RPP_MAIN_POST_SHRPCNR_MEM) |

### 4. Connection structure

Connection structure of error status register (ISPERR0_STATUS, ISPERR3_STATUS, ISPERR4_STATUS) and error enable register (ISPERR0_ENABLE, ISPERR3_ENABLE, ISPERR4_ENALBLE) is shown in Figure 54.27.

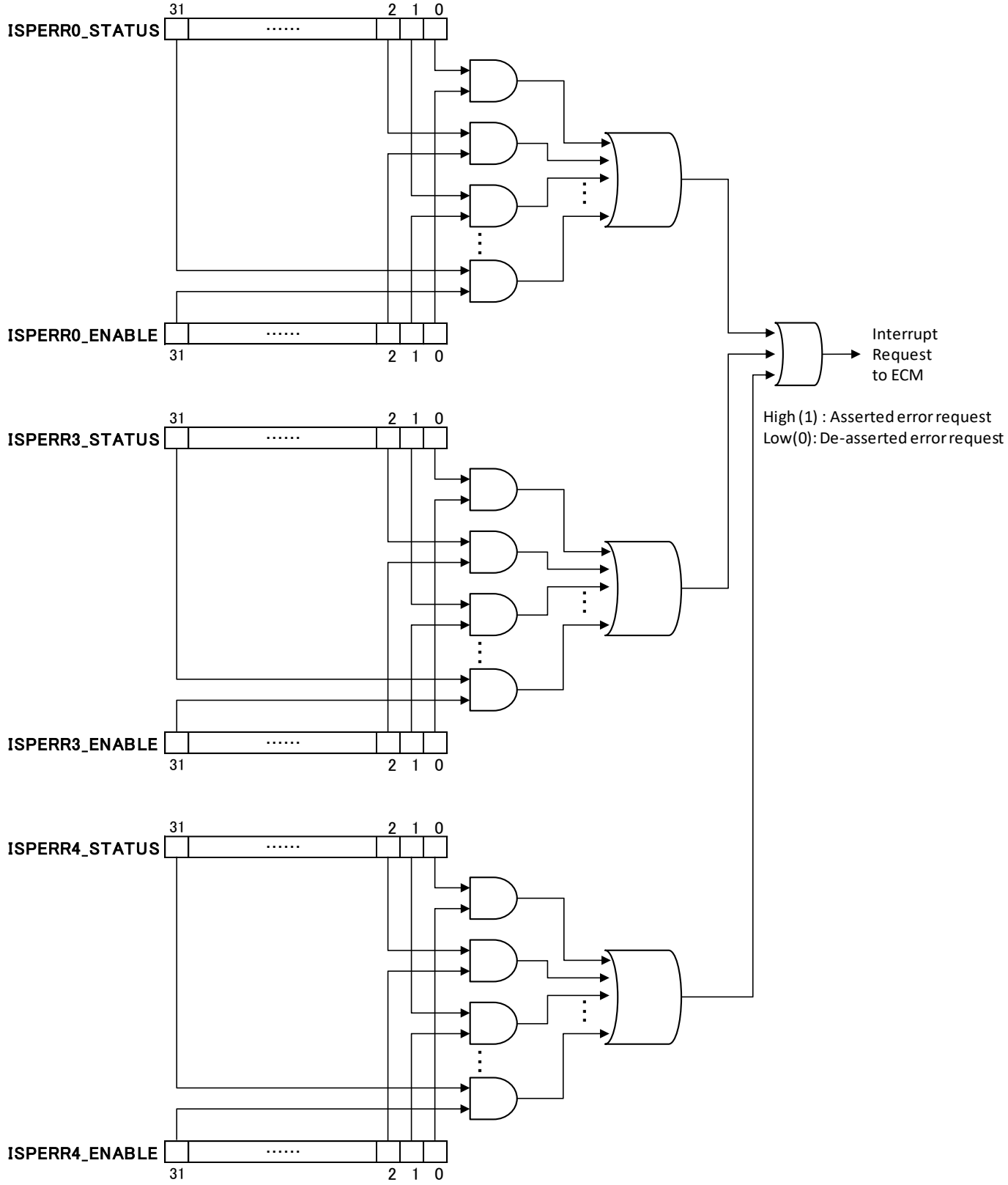



**Figure 54.27 Connection structure of error request from ISP core**

#### 54.4.7.3 Bit assignment of Error Request

**1. ISPERR0**

| n | Bit | Status Register | Clear Register | Enable Register | Interrupt Source |
|---|---|---|---|---|---|
| 0 | 0 | ST0 | CLR0 | EN0 | ISP Core SRAM EDC error (RPP_MAIN_PRE1_ACQ_FIFO_MEM) |
| 0 | 1 | ST1 | CLR1 | EN1 | ISP Core SRAM EDC error (RPP_MAIN_PRE1_LSC_R_MEM) |
| 0 | 2 | ST2 | CLR2 | EN2 | ISP Core SRAM EDC error (RPP_MAIN_PRE1_LSC_GR_MEM) |
| 0 | 3 | ST3 | CLR3 | EN3 | ISP Core SRAM EDC error (RPP_MAIN_PRE1_LSC_B_MEM) |
| 0 | 4 | ST4 | CLR4 | EN4 | ISP Core SRAM EDC error (RPP_MAIN_PRE1_LSC_GB_MEM) |
| 0 | 5 | ST5 | CLR5 | EN5 | ISP Core SRAM EDC error (RPP_MAIN_PRE1_DPCC_MEM) |
| 0 | 6 | ST6 | CLR6 | EN6 | ISP Core SRAM EDC error (RPP_MAIN_PRE1_DPCC_BPRAM_MEM) |
| 0 | 7 | ST7 | CLR7 | EN7 | ISP Core SRAM EDC error (RPP_MAIN_PRE1_DPF_MEM) |
| 0 | 8 | ST8 | CLR8 | EN8 | ISP Core SRAM EDC error (RPP_MAIN_PRE2_ACQ_FIFO_MEM) |
| 0 | 9 | ST9 | CLR9 | EN9 | ISP Core SRAM EDC error (RPP_MAIN_PRE2_LSC_R_MEM) |
| 0 | 10 | ST10 | CLR10 | EN10 | ISP Core SRAM EDC error (RPP_MAIN_PRE2_LSC_GR_MEM) |
| 0 | 11 | ST11 | CLR11 | EN11 | ISP Core SRAM EDC error (RPP_MAIN_PRE2_LSC_B_MEM) |
| 0 | 12 | ST12 | CLR12 | EN12 | ISP Core SRAM EDC error (RPP_MAIN_PRE2_LSC_GB_MEM) |
| 0 | 13 | ST13 | CLR13 | EN13 | ISP Core SRAM EDC error (RPP_MAIN_PRE2_DPCC_MEM) |
| 0 | 14 | ST14 | CLR14 | EN14 | ISP Core SRAM EDC error (RPP_MAIN_PRE2_DPCC_BPRAM_MEM) |
| 0 | 15 | ST15 | CLR15 | EN15 | ISP Core SRAM EDC error (RPP_MAIN_PRE2_DPF_MEM) |
| 0 | 16 | ST16 | CLR16 | EN16 | (Not used) |
| 0 | 17 | ST17 | CLR17 | EN17 | (Not used) |
| 0 | 18 | ST18 | CLR18 | EN18 | (Not used) |
| 0 | 19 | ST19 | CLR19 | EN19 | (Not used) |
| 0 | 20 | ST20 | CLR20 | EN20 | (Not used) |
| 0 | 21 | ST21 | CLR21 | EN21 | (Not used) |
| 0 | 22 | ST22 | CLR22 | EN22 | (Not used) |
| 0 | 23 | ST23 | CLR23 | EN23 | (Not used) |
| 0 | 24 | ST24 | CLR24 | EN24 | ISP Core SRAM EDC error (RPP_RMAP_PINTL_MEM) |
| 0 | 25 | ST25 | CLR25 | EN25 | ISP Core SRAM EDC error (RPP_RMAP_PINTS_MEM) |
| 0 | 26 | ST26 | CLR26 | EN26 | ISP Core SRAM EDC error (RPP_RMAP_PINTVS_MEM) |
| 0 | 27 | ST27 | CLR27 | EN27 | ISP Core SRAM EDC error (RPP_RMAP_BLMFL_MEM) |
| 0 | 28 | ST28 | CLR28 | EN28 | ISP Core SRAM EDC error (RPP_RMAP_BLMFS_MEM) |
| 0 | 29 | ST29 | CLR29 | EN29 | ISP Core SRAM EDC error (RPP_RMAP_BLSFL_MEM) |
| 0 | 30 | ST30 | CLR30 | EN30 | ISP Core SRAM EDC error (RPP_RMAP_BLSFS_MEM) |
| 0 | 31 | ST31 | CLR31 | EN31 | ISP Core SRAM EDC error (RPP_RMAP_DELAYL_MEM) |

Figure 54.28 Bit assignment of ISPERR0

### 2. ISPERR1

| n | Bit | Status Register | Clear Register | Enable Register | Interrupt Source |
|---|---|---|---|---|---|
| 1 | 0 | ST0 | CLR0 | EN0 | CSI_FIFO0_MEM_ERROR (EDC error of the CSI FIFO SRAM) |
| 1 | 1 | ST1 | CLR1 | EN1 | (Not used) |
| 1 | 2 | ST2 | CLR2 | EN2 | (Not used) |
| 1 | 3 | ST3 | CLR3 | EN3 | (Not used) |
| 1 | 4 | ST4 | CLR4 | EN4 | CSI_ECC_ERROR. CSI input ECC error. |
| 1 | 5 | ST5 | CLR5 | EN5 | CSI_CRC_ERROR. CSI input CRC error |
| 1 | 6 | ST6 | CLR6 | EN6 | (Not used) |
| 1 | 7 | ST7 | CLR7 | EN7 | (Not used) |
| 1 | 8 | ST8 | CLR8 | EN8 | CHANNEL_SELECT_CH0_ERROR (EDC error of pixel stream) |
| 1 | 9 | ST9 | CLR9 | EN9 | CHANNEL_SELECT_CH1_ERROR |
| 1 | 10 | ST10 | CLR10 | EN10 | CHANNEL_SELECT_CH2_ERROR |
| 1 | 11 | ST11 | CLR11 | EN11 | CHANNEL_SELECT_CH3_ERROR |
| 1 | 12 | ST12 | CLR12 | EN12 | CHANNEL_SELECT_CH4_ERROR |
| 1 | 13 | ST13 | CLR13 | EN13 | CHANNEL_SELECT_CH5_ERROR |
| 1 | 14 | ST14 | CLR14 | EN14 | CHANNEL_SELECT_CH6_ERROR |
| 1 | 15 | ST15 | CLR15 | EN15 | CHANNEL_SELECT_CH7_ERROR |
| 1 | 16 | ST16 | CLR16 | EN16 | CHANNEL_SELECT_CH8_ERROR |
| 1 | 17 | ST17 | CLR17 | EN17 | CHANNEL_SELECT_CH9_ERROR |
| 1 | 18 | ST18 | CLR18 | EN18 | CHANNEL_SELECT_CH10_ERROR |
| 1 | 19 | ST19 | CLR19 | EN19 | CHANNEL_SELECT_CH11_ERROR |
| 1 | 20 | ST20 | CLR20 | EN20 | CHANNEL_SELECT_CH4_PIXELCOUNT_MAX_ERROR |
| 1 | 21 | ST21 | CLR21 | EN21 | CHANNEL_SELECT_CH5_PIXELCOUNT_MAX_ERROR |
| 1 | 22 | ST22 | CLR22 | EN22 | CHANNEL_SELECT_CH6_PIXELCOUNT_MAX_ERROR |
| 1 | 23 | ST23 | CLR23 | EN23 | CHANNEL_SELECT_CH7_PIXELCOUNT_MAX_ERROR |
| 1 | 24 | ST24 | CLR24 | EN24 | CHANNEL_SELECT_CH8_PIXELCOUNT_MAX_ERROR |
| 1 | 25 | ST25 | CLR25 | EN25 | CHANNEL_SELECT_CH9_PIXELCOUNT_MAX_ERROR |
| 1 | 26 | ST26 | CLR26 | EN26 | CHANNEL_SELECT_CH4_PIXELCOUNT_MIN_ERROR |
| 1 | 27 | ST27 | CLR27 | EN27 | CHANNEL_SELECT_CH5_PIXELCOUNT_MIN_ERROR |
| 1 | 28 | ST28 | CLR28 | EN28 | CHANNEL_SELECT_CH6_PIXELCOUNT_MIN_ERROR |
| 1 | 29 | ST29 | CLR29 | EN29 | CHANNEL_SELECT_CH7_PIXELCOUNT_MIN_ERROR |
| 1 | 30 | ST30 | CLR30 | EN30 | CHANNEL_SELECT_CH8_PIXELCOUNT_MIN_ERROR |
| 1 | 31 | ST31 | CLR31 | EN31 | CHANNEL_SELECT_CH9_PIXELCOUNT_MIN_ERROR |

Figure 54.29 Bit assignment of ISPERR1

### 3. ISPERR2

| n | Bit | Status Register | Clear Register | Enable Register | Interrupt Source |
|---|---|---|---|---|---|
| 2 | 0 | ST0 | CLR0 | EN0 | CHANNEL_SELECT_CH10_PIXELCOUNT_MAX_ERROR |
| 2 | 1 | ST1 | CLR1 | EN1 | CHANNEL_SELECT_CH11_PIXELCOUNT_MAX_ERROR |
| 2 | 2 | ST2 | CLR2 | EN2 | SYNC_ERROR_VC4 |
| 2 | 3 | ST3 | CLR3 | EN3 | SYNC_ERROR_VC5 |
| 2 | 4 | ST4 | CLR4 | EN4 | SYNC_ERROR_VC6 |
| 2 | 5 | ST5 | CLR5 | EN5 | SYNC_ERROR_VC7 |
| 2 | 6 | ST6 | CLR6 | EN6 | SYNC_ERROR_VC8 |
| 2 | 7 | ST7 | CLR7 | EN7 | SYNC_ERROR_VC9 |
| 2 | 8 | ST8 | CLR8 | EN8 | CHANNEL_SELECT_CH10_PIXELCOUNT_MIN_ERROR |
| 2 | 9 | ST9 | CLR9 | EN9 | CHANNEL_SELECT_CH11_PIXELCOUNT_MIN_ERROR |
| 2 | 10 | ST10 | CLR10 | EN10 | SYNC_ERROR_VC10 |
| 2 | 11 | ST11 | CLR11 | EN11 | SYNC_ERROR_VC11 |
| 2 | 12 | ST12 | CLR12 | EN12 | SYNC_ERROR_VC12 |
| 2 | 13 | ST13 | CLR13 | EN13 | SYNC_ERROR_VC13 |
| 2 | 14 | ST14 | CLR14 | EN14 | SYNC_ERROR_VC14 |
| 2 | 15 | ST15 | CLR15 | EN15 | SYNC_ERROR_VC15 |
| 2 | 16 | ST16 | CLR16 | EN16 | (Not used) |
| 2 | 17 | ST17 | CLR17 | EN17 | RAW_SCALER_MEMORY_ERROR |
| 2 | 18 | ST18 | CLR18 | EN18 | RAW_SCALER_MISMATCH_ERROR |
| 2 | 19 | ST19 | CLR19 | EN19 | (Not used) |
| 2 | 20 | ST20 | CLR20 | EN20 | SYNC_ERROR_VC0 |
| 2 | 21 | ST21 | CLR21 | EN21 | SYNC_ERROR_VC1 |
| 2 | 22 | ST22 | CLR22 | EN22 | SYNC_ERROR_VC2 |
| 2 | 23 | ST23 | CLR23 | EN23 | SYNC_ERROR_VC3 |
| 2 | 24 | ST24 | CLR24 | EN24 | (Not used) |
| 2 | 25 | ST25 | CLR25 | EN25 | (Not used) |
| 2 | 26 | ST26 | CLR26 | EN26 | (Not used) |
| 2 | 27 | ST27 | CLR27 | EN27 | CSI_LOCK_STEP_ERROR |
| 2 | 28 | ST28 | CLR28 | EN28 | (Not used) |
| 2 | 29 | ST29 | CLR29 | EN29 | (Not used) |
| 2 | 30 | ST30 | CLR30 | EN30 | (Not used) |
| 2 | 31 | ST31 | CLR31 | EN31 | (Not used) |

Figure 54.30 Bit assignment of ISPERR2

### 4. ISPERR3

| n | Bit | Status Register | Clear Register | Enable Register | Interrupt Source |
|---|---|---|---|---|---|
| 3 | 0 | ST0 | CLR0 | EN0 | ISP Core SRAM EDC error (RPP_RMAP_DELAYS_MEM) |
| 3 | 1 | ST1 | CLR1 | EN1 | ISP Core SRAM EDC error (RPP_RMAP_DELAYVS_MEM) |
| 3 | 2 | ST2 | CLR2 | EN2 | ISP Core SRAM EDC error (RPP_MAIN_POST_FILT_MEM) |
| 3 | 3 | ST3 | CLR3 | EN3 | ISP Core SRAM EDC error (RPP_MAIN_POST_LTM0_MEM) |
| 3 | 4 | ST4 | CLR4 | EN4 | ISP Core SRAM EDC error (RPP_MAIN_POST_LTM1_MEM) |
| 3 | 5 | ST5 | CLR5 | EN5 | ISP Core SRAM EDC error (RPP_MAIN_POST_LTM2_MEM) |
| 3 | 6 | ST6 | CLR6 | EN6 | ISP Core SRAM EDC error (RPP_MAIN_RGBDENOISE_MEM) |
| 3 | 7 | ST7 | CLR7 | EN7 | ISP Core SRAM EDC error (RPP_MAIN_POST_SHRPCNR_MEM) |
| 3 | 8 | ST8 | CLR8 | EN8 | ISP Core SRAM EDC error (RPP_HDR_MV_OUT_RAM_MEM) |
| 3 | 9 | ST9 | CLR9 | EN9 | ISP Core SRAM EDC error (RPP_OUT_CONV_420_MEM) |
| 3 | 10 | ST10 | CLR10 | EN10 | (Not used) |
| 3 | 11 | ST11 | CLR11 | EN11 | (Not used) |
| 3 | 12 | ST12 | CLR12 | EN12 | (Not used) |
| 3 | 13 | ST13 | CLR13 | EN13 | (Not used) |
| 3 | 14 | ST14 | CLR14 | EN14 | (Not used) |
| 3 | 15 | ST15 | CLR15 | EN15 | (Not used) |
| 3 | 16 | ST16 | CLR16 | EN16 | (Not used) |
| 3 | 17 | ST17 | CLR17 | EN17 | (Not used) |
| 3 | 18 | ST18 | CLR18 | EN18 | (Not used) |
| 3 | 19 | ST19 | CLR19 | EN19 | (Not used) |
| 3 | 20 | ST20 | CLR20 | EN20 | (Not used) |
| 3 | 21 | ST21 | CLR21 | EN21 | (Not used) |
| 3 | 22 | ST22 | CLR22 | EN22 | (Not used) |
| 3 | 23 | ST23 | CLR23 | EN23 | (Not used) |
| 3 | 24 | ST24 | CLR24 | EN24 | (Not used) |
| 3 | 25 | ST25 | CLR25 | EN25 | (Not used) |
| 3 | 26 | ST26 | CLR26 | EN26 | (Not used) |
| 3 | 27 | ST27 | CLR27 | EN27 | (Not used) |
| 3 | 28 | ST28 | CLR28 | EN28 | (Not used) |
| 3 | 29 | ST29 | CLR29 | EN29 | (Not used) |
| 3 | 30 | ST30 | CLR30 | EN30 | (Not used) |
| 3 | 31 | ST31 | CLR31 | EN31 | (Not used) |

Figure 54.31 Bit assignment of ISPERR3

### 5. ISPERR4

| n | Bit | Status Register | Clear Register | Enable Register | Interrupt Source |
|---|---|---|---|---|---|
| 4 | 0 | ST0 | CLR0 | EN0 | FAULT_FLAG. |
| 4 | 1 | ST1 | CLR1 | EN1 | (Not used) |
| 4 | 2 | ST2 | CLR2 | EN2 | (Not used) |
| 4 | 3 | ST3 | CLR3 | EN3 | (Not used) |
| 4 | 4 | ST4 | CLR4 | EN4 | (Not used) |
| 4 | 5 | ST5 | CLR5 | EN5 | (Not used) |
| 4 | 6 | ST6 | CLR6 | EN6 | (Not used) |
| 4 | 7 | ST7 | CLR7 | EN7 | (Not used) |
| 4 | 8 | ST8 | CLR8 | EN8 | (Not used) |
| 4 | 9 | ST9 | CLR9 | EN9 | (Not used) |
| 4 | 10 | ST10 | CLR10 | EN10 | (Not used) |
| 4 | 11 | ST11 | CLR11 | EN11 | (Not used) |
| 4 | 12 | ST12 | CLR12 | EN12 | (Not used) |
| 4 | 13 | ST13 | CLR13 | EN13 | (Not used) |
| 4 | 14 | ST14 | CLR14 | EN14 | (Not used) |
| 4 | 15 | ST15 | CLR15 | EN15 | (Not used) |
| 4 | 16 | ST16 | CLR16 | EN16 | (Not used) |
| 4 | 17 | ST17 | CLR17 | EN17 | (Not used) |
| 4 | 18 | ST18 | CLR18 | EN18 | (Not used) |
| 4 | 19 | ST19 | CLR19 | EN19 | (Not used) |
| 4 | 20 | ST20 | CLR20 | EN20 | (Not used) |
| 4 | 21 | ST21 | CLR21 | EN21 | (Not used) |
| 4 | 22 | ST22 | CLR22 | EN22 | (Not used) |
| 4 | 23 | ST23 | CLR23 | EN23 | (Not used) |
| 4 | 24 | ST24 | CLR24 | EN24 | (Not used) |
| 4 | 25 | ST25 | CLR25 | EN25 | (Not used) |
| 4 | 26 | ST26 | CLR26 | EN26 | (Not used) |
| 4 | 27 | ST27 | CLR27 | EN27 | (Not used) |
| 4 | 28 | ST28 | CLR28 | EN28 | (Not used) |
| 4 | 29 | ST29 | CLR29 | EN29 | (Not used) |
| 4 | 30 | ST30 | CLR30 | EN30 | (Not used) |
| 4 | 31 | ST31 | CLR31 | EN31 | (Not used) |

Figure 54.32 Bit assignment of ISPERR4

### 54.4.8 Streaming Bridge

Streaming Bridge function realizes “Memory-to_memory” operation, which gets the input image data from DRAM, processes it and output to the DRAM by collabolated operation with VSPX.

Number of exposures is up to 2 exposures. So, two VSPX RPFs (Read Pixel Formatter), i.e. RPF0 and RPF1 are available for input image transfer use (RPF2 and RPF3 are not used).

**1. Configuration of VSPX**

Parameter of input image (base address, width, height and pixel format, etc.) are set to VSPX register or VSPX display list. When multiple exposure is used then, image parameter should be set for each exposure.

**2. Configuration of CSI, Channel Selector and VIN for image capture path.**

In order to activate image capture path from CSI to DRAM via VIN, configuration of CSI, Channel Selector and VIN is needed. Regarding configuration of Channel Selector, refer to the 54.4.2 for detail. Regarding configuration CSI and VIN, refer to the users manual of CSI or VIN.

**3. Configuration of Streaming Bridge**

Write B’1111 to ISP_STREAMER_MODE.STREAMER_ENABLE register. Streaming path is activated after that.

Write vertical blanking period (unit : cycle) to STREAMER_V_BLANK register.

Write horizonal blanking period (unit : cycle) to the STREAMER_H_BLANK register.

During the Streaming Bridge is feeding an input image to the ISP core, Streaming Bridge generates horizontal blanking cycles at the line boundary (end of one line) of every line. Period of horizontal blanking cycle is set by ISP_STREAMER_H_BLANK register

During the Streaming Bridge is feeding an input image to the ISP core, Streaming Bridge generates virtual blanking cycles at the frame boundary (end of one frame) of every frame. Period of virtual blanking cycle is set by ISP_STREAMER_V_BLANK register

**4. Configuration of ISP core**

Set the ISP core register according to ISP processing.

**5. Start image input**

Write H’FFFF to ISPSTART register to activate image capturing from CSI.

Write H’00000001 to VI6_CMD0 register (start register of VSPX) to trigger image input from DRAM to VSPX.

**6. Waiting complete of one frame processing**

ISP core and VSPX asserts end of frame interrupts. To notify complete of one frame processing, wait both interrupts from ISP core and VSPX (FRE interrupt).

### 54.4.9 RAW Domain Scaler (RAW Scaler)

RAW domain scaling function (RAW scaler) is newly added. RAW scaler is placed between Pixel Reconstructor (stage1) and VC filter(stage2) in pixel processing pipeline.

#### 54.4.9.1 Overview

RAW scaler scales incoming RAW image from CSI in RAW domain. There are 2 use-case in RAW scaler, one is 1/2 scaling and the another one is RGB-Ir conversion.

In 1/2 scaling usecase, scaled ratio is only both 1/2 horizontal and 1/2 vertical. Scale ratio other than 1/2 is not supported. And, horizontal only scaling or vertical only scaling is not supported.

In RGB-IR conversion usecase, scaled ratio is only 1.0 horizontal and 1.0 vertical. This usecase converts from non-Bayer and 4x4 based RGB-IR color filter array into Bayer color filter array by using convolution operation.

#### 54.4.9.2 Preparation of of RAW Scaler

Before setting convolutional operation in RAW scaler, following settings are needed.

RAW_SCALER_MODE.ENABLE should be set to 1.

ISPCS_FILTER_ID_ID_CHc.SEL_RAW_SCALED should be set to 1, where c (c=0~11) is the channel number through which the output from the RAW Scaler is to be passed. For example, if the RAW scaler output is to be passed through the channel 2 and channel6 , then ISPCS_FILTER_ID_ID_CH2.SEL_RAW_SCALED =1 ISPCS_FILTER_ID_ID_CH6.SEL_RAW_SCALED =1.

Set the virtual channel number to be input to the RAW scaler to the RAW_SCALER_MODE.TARGET_VC register

Set the data type code to be input to the RAW scaler to the RAW_SCALER_MODE.TARGET_DT_CODE register

Set the image width to the RAW_SCALER_IMAGE_SIZE.IMAGE_WIDTH register. Its unit is number of pixels per line

Set the image height to the RAW_SCALER_IMAGE_SIZE.IMAGE_HEIGHT register. Its unit is number of lines in one frame to be processed by RAW scaler.

Set the toggle timing of the VSYNC, HSYNC and DE signals of RAW scaler output to the RAW_SCALER_OUT_VLANK and RAW_SCALER_OUT_HBLANK and RAW_SCALER_OUT_VS_HS register.

In the following description, register symbol name such as “RAW_SCAER_MODE” or RAW_SCALER_COEFF_SEL_ROWn”, etc which are defined as the RAW scaler-related registers is omitted.

#### 54.4.9.3 Calculation operation

RAW scaler performs convolutional calculation on the input pixels. Convolution coefficient is switched by pixel-by-pixel according to the co-ordinate of input RAW pixel. The formula of this convolution operation is as follows.

$$Q'(x,y) = \left[\sum_{m=-2}^{2}\sum_{n=-2}^{2}\{P(x+m,y+n) << input_lshift\} * C(s,m,n) + rounding(s)\right] >> conv_shift(s)$$

$$s = coeff_select_table(x\%4, y\%4)$$

$$Q(x,y) = Sat\{Q'(x,y), sat_min, sat_max\} << final_lshift$$

$$Sat\{Q', m, n\} = \begin{cases} Q' \ if \ sat_min \le Q \le sat_max \\ sat_min \ if \ Q' < sat_min \\ sat_max \ if \ Q' > sat_max \end{cases}$$

$P(x,y)$ : Input pixel to RAW scaler
$Q(x,y)$ : Output pixel from RAW scaler

$C(s,m,n)$ : The s-th convolution coefficient( its size is 5x5). The number of convolution coefficient set is 4 (s=0,1,2,3). Size of a convolution coefficient set is 5x5 *(m=-2,-1,0,1,2. n=-2,-1,0,1,2).* $C(s,m,n)$ is set by using RAW_SCALER_ COEFF[s]_* register (hereafter, "RAW_SCALER_" will be omitted and will be referred to as COEFF[s]_*) . Correspondence between $C(s,m,n)$ and register is shown as below.

| $C(s,m,n)$ | Register | $C(s,m,n)$ | Register |
|---|---|---|---|
| *C(s,-2,-2)* | COEFF[s]_M2_M2 | *C(s,-1,-2)* | COEFF[s]_M1_M2 |
| *C(s,-2,-1)* | COEFF[s]_M2_M1 | *C(s,-1,-1)* | COEFF[s]_M1_M1 |
| *C(s,-2,0)* | COEFF[s]_M2_00 | *C(s,-1, 0)* | COEFF[s]_M1_00 |
| *C(s,-2,1)* | COEFF[s]_M2_P1 | *C(s,-1, 1)* | COEFF[s]_M1_P1 |
| *C(s,-2,2)* | COEFF[s]_M2_P2 | *C(s,-1, 2)* | COEFF[s]_M1_P2 |
| *C(s, 0,-2)* | COEFF[s]_00_M2 | *C(s, 1,-2)* | COEFF[s]_P1_M2 |
| *C(s, 0,-1)* | COEFF[s]_00_M1 | *C(s, 1,-1)* | COEFF[s]_P1_M1 |
| *C(s, 0, 0)* | COEFF[s]_00_00 | *C(s, 1, 0)* | COEFF[s]_P1_00 |
| *C(s, 0, 1)* | COEFF[s]_00_P1 | *C(s, 1, 1)* | COEFF[s]_P1_P1 |
| *C(s, 0, 2)* | COEFF[s]_00_P2 | *C(s, 1, 2)* | COEFF[s]_P1_P2 |
| *C(s, 2,-2)* | COEFF[s]_P2_M2 | | |
| *C(s, 2,-1)* | COEFF[s]_P2_M1 | | |
| *C(s, 2, 0)* | COEFF[s]_P2_00 | | |
| *C(s, 2, 1)* | COEFF[s]_P2_P1 | | |
| *C(s, 2, 2)* | COEFF[s]_P2_P2 | | |

$input_lshift$: Setting value of INPUT_LSHIFT register

$rounding(s)$: Setting value of ROUNDINGs register( s=0,1,2,3)

$conv_rshift(s)$ : Setting value of CONVs_RHSHIFT register (s=0,1,2,3)

$coeff_select_table(x\%4, y\%4)$: This indicates the index 's' of the convolution coefficient. The available value is any of 0,1,2,3, and 15. However, in case of *coeff_select_table(x%4,y%4)*=15, no pixel will be output corresponding to this input pixel. Setting as *coeff_select_table(x%4,y%4)*=15 is used for 1/2 scaling usecase

$Sat\{Q'(x,y), sat_min, sat_max\}$ : Saturation operation. If Q'(x,y) value exceeds in range of *[sat_min, sat_max]* then saturation output is limited to *sat_min* or *sat_max*. *sat_min* is a setting value of SATURATION_MIN register. *sat_max* is a setting value of SATURATION_MAX register.

$final_lshift$ : Setting value of FINAL_LSHIFT register. FINAL_LSHIFT register is used when packing output pixel as MSB aligned. For LSB aligned case, FINAL_LSHIFT register is set to 0.

#### 54.4.9.4 Configuration for each usecase

### 1. Scaling usecase

This section describes an example of convolution coefficient setting in 1/2 scaling case. Output pixels in RAW scaled image is calculated by convolution operation using the pixel surrounding the current pixel. Its convolution coefficient is switched on a pixel-by-pixel of the output image. Figure 54.33 shows an example of calculation of pixels of RAW scaled image.

This section describes the principle of calculation for each pixel of a RAW scaled image using Figure 54.33. However, convolution kernel shown in this section is just an example. Actual setting of convolution coefficients are configurable. If there are any other more optimized coefficients or kernel window setting, it is possible to apply these optimized settings.

**(a) Pixel calculation of RAW scaled image**

To calculate the pixels corresponding to the R component, extract the R component pixels of the input RAW image and perform convolution operation. For example in Figure 54.33, the output pixel Q(4,2) at “Output Image” is regarded as R component in the Bayer array. Therefore, in calculating Q(4,2), a convolution operation is performed with the input pixel P(8,4) (=R4) as the center, and the R component pixels around it are extracted.

The calculation of the G and B component pixels of RAW scaled output image is the same as for the R component

In Figure 54.33, the output pixel Q(3,2) is regarded as G component in the Bayer array. Therefore, in calculating Q(3,2), a convolution operation is performed with the input pixel P(7,4) (=G2) and P(6,5) (=G3) as the center and the G-component pixels around it are extracted.

In Figure 54.33, the output pixel Q(3,1) is regarded as B component in the Bayer array. Therefore, in calculating Q(3,1), a convolution operation is performed with the input pixel P(7,3) (=B4) as the center, and the B component pixels around it are extracted.

**(b) Selection of convolution coefficient**

Convolution coefficient is switched by pixel-by-pixel. To specify appropriate convolution coefficients coefficient select table (*coeff_select_table*)is used. The *coeff_select_table* determines the index number of convolution coefficient set to be applied in the above convolution operation, using the remainder of the X and Y coordinates of the input RAW image divided by 4 as the index of this table (X denotes a horizontal location, and Y denotes a vertical location). Here, *coeff_select_ table(0,0)* denotes the case where the X and Y coordinates of the input RAW image are both multiples of 4.

Since R component pixels are located at coordinates *(4*I,4*j), (4*i+2,4*j), (4*i+2,4*j)* and *(4*i+2,4*j)* in the input image (*i* and *j* are integer) , *coeff_select_table(0,0), (2,0), (0,2), (2,2)* specifies the index number of the convolution coefficient for calculating the R component. However, considering 1/2 scaling, coordinate of R component pixels in the input image which will be output after 1/2 scaling is only *(4*i,4*j).* Other R component pixels located at *(4*i+2,4*j), (4*i+2,4*j)* and *(4*i+2,4*j)* will not be output. Therefore, in the setting of coeff_seletct_table, coeff_select_table(2,0), (0,2), (2,2) should be set to 15, which represents the pixel output OFF.

Regarding *coeff_select_table* for G and B component is same as for the R component.

Figure 54.34 is an example of *coeff_select_table* setting, where 0-th convolution coefficient set is used to calculate R-component, 1st convolution coefficient set is used to calculate G-component, and the 2nd convolution coefficient set is used to calculate B component of output image.

Considering 1/2 scaling, *(4*i,4*j)* in input RAW image corresponds to the *(2*i,2*j)* in output RAW image, and it is regarded as R-component. *(4*i+3,4*j+3)* in input image corresponds to the *(2*i+1,2*j+1)* in output image, and it is regarded as B-component. *(4*i+3,4*j)* and *(4*i+3,4*j)* in input image corresponds to the *(2*i+1,2*j) and (2*i,2*j+1)* respectively in output image, and these are regarded as G-component. Other pixels are not output, so coeff_select_table for other pixels are set to 15.

**(c) Register setting**

Figure 54.35 is a register setting of convolution coefficient and coeff_select_table in 1/2 scaling use-case. Here, r0,r1,…r8 and g0,g1,…g8 and b0, b1, … b8 in Figure 54.34 and Figure 54.35 are setting value of convolution coefficient. The following is an example of a specific value of convolution coefficient. This example shows decimal point of coefficient value is represented with 12 bit precision. Note that this is just one example, and the coefficient value itself is configurable.

| coef. | setting value | real number value |
|---|---|---|
| r0 | 6 | 0.001 |
| r1 | 592 | 0.145 |
| r2 | 6 | 0.001 |
| r3 | 592 | 0.145 |
| r4 | 1657 | 0.405 |
| r5 | 592 | 0.145 |
| r6 | 6 | 0.001 |
| r7 | 592 | 0.145 |
| r8 | 6 | 0.001 |

| coef. | setting value | real number value |
|---|---|---|
| b0 | 6 | 0.001 |
| b1 | 592 | 0.145 |
| b2 | 6 | 0.001 |
| b3 | 592 | 0.145 |
| b4 | 1657 | 0.405 |
| b5 | 592 | 0.145 |
| b6 | 6 | 0.001 |
| b7 | 592 | 0.145 |
| b8 | 6 | 0.001 |

| coef. | setting value | real number value |
|---|---|---|
| g0 | 1205 | 0.294 |
| g1 | 1205 | 0.294 |
| g2 | 717 | 0.175 |
| g3 | 717 | 0.175 |

If the Bayer array is not RGGB order, such BGGR, etc., it can still be handled by setting the coeff_select_table to an appropriate value.

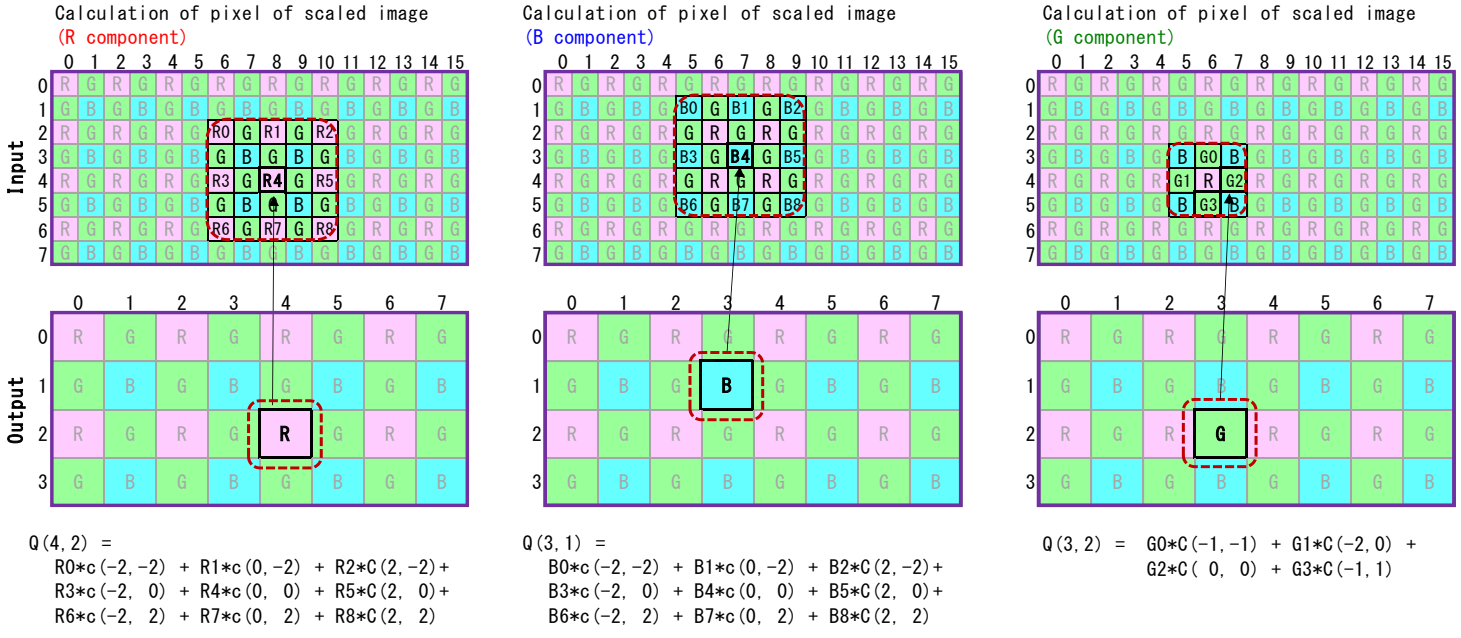



Figure 54.33 Calculation of RAW scaled pixel

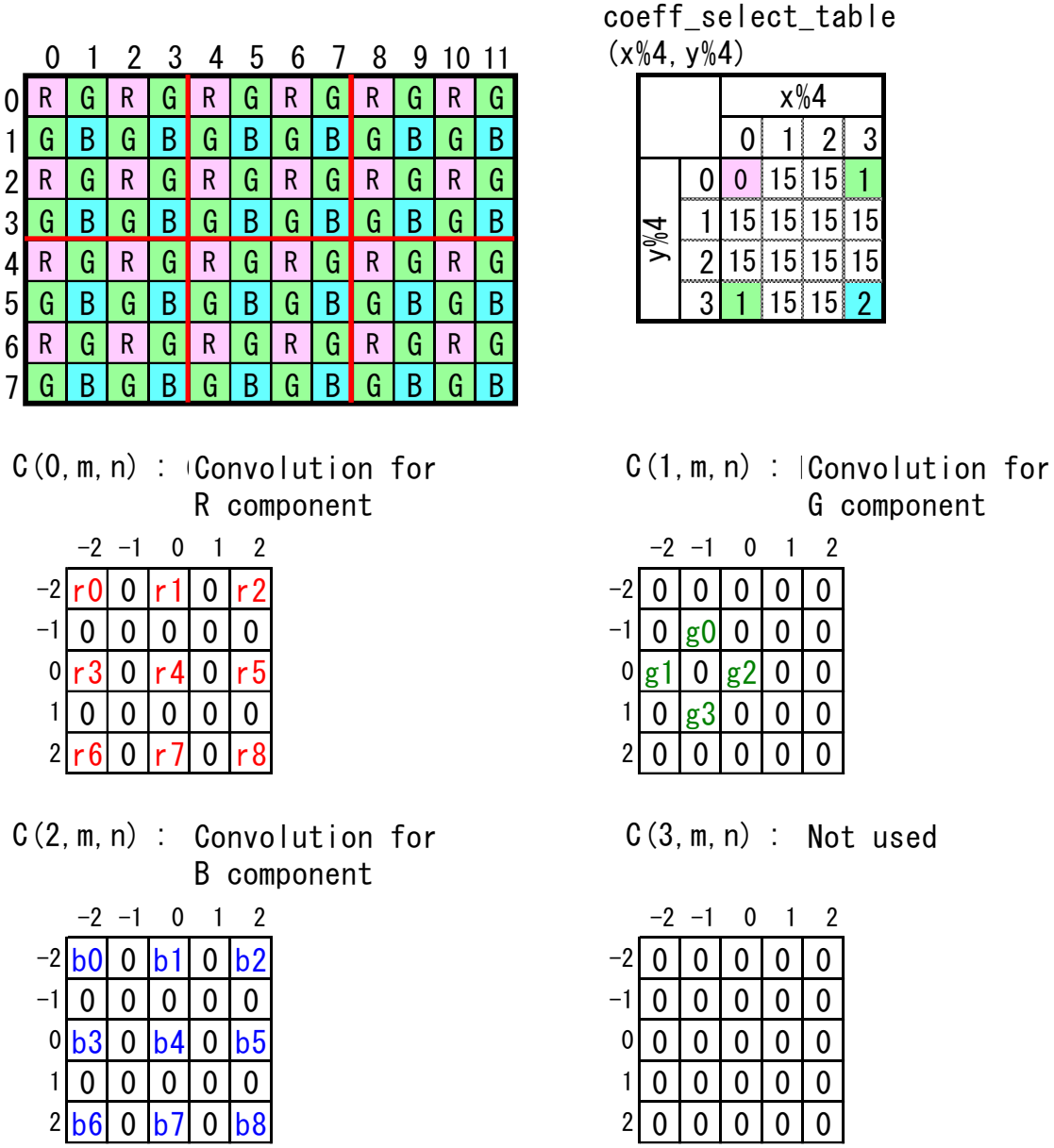



Figure 54.34 Example of coeff_select_table

Setting value of convolution coefficient register

| Register Name | Setting Value | Register Name | Setting Value | Register Name | Setting Value | Register Name | Setting Value |
|---|---|---|---|---|---|---|---|
| COEFF[0]_M2_M2 | r0 | COEFF[1]_M2_M2 | 0 | COEFF[2]_M2_M2 | b0 | COEFF[3]_M2_M2 | don't care |
| COEFF[0]_M2_M1 | 0 | COEFF[1]_M2_M1 | 0 | COEFF[2]_M2_M1 | 0 | COEFF[3]_M2_M1 | don't care |
| COEFF[0]_M2_00 | r1 | COEFF[1]_M2_00 | 0 | COEFF[2]_M2_00 | b1 | COEFF[3]_M2_00 | don't care |
| COEFF[0]_M2_P1 | 0 | COEFF[1]_M2_P1 | 0 | COEFF[2]_M2_P1 | 0 | COEFF[3]_M2_P1 | don't care |
| COEFF[0]_M2_P2 | r2 | COEFF[1]_M2_P2 | 0 | COEFF[2]_M2_P2 | b2 | COEFF[3]_M2_P2 | don't care |
| COEFF[0]_M1_M2 | 0 | COEFF[1]_M1_M2 | 0 | COEFF[2]_M1_M2 | 0 | COEFF[3]_M1_M2 | don't care |
| COEFF[0]_M1_M1 | 0 | COEFF[1]_M1_M1 | g0 | COEFF[2]_M1_M1 | 0 | COEFF[3]_M1_M1 | don't care |
| COEFF[0]_M1_00 | 0 | COEFF[1]_M1_00 | 0 | COEFF[2]_M1_00 | 0 | COEFF[3]_M1_00 | don't care |
| COEFF[0]_M1_P1 | 0 | COEFF[1]_M1_P1 | 0 | COEFF[2]_M1_P1 | 0 | COEFF[3]_M1_P1 | don't care |
| COEFF[0]_M1_P2 | 0 | COEFF[1]_M1_P2 | 0 | COEFF[2]_M1_P2 | 0 | COEFF[3]_M1_P2 | don't care |
| COEFF[0]_00_M2 | r3 | COEFF[1]_00_M2 | g1 | COEFF[2]_00_M2 | b3 | COEFF[3]_00_M2 | don't care |
| COEFF[0]_00_M1 | 0 | COEFF[1]_00_M1 | 0 | COEFF[2]_00_M1 | 0 | COEFF[3]_00_M1 | don't care |
| COEFF[0]_00_00 | r4 | COEFF[1]_00_00 | g2 | COEFF[2]_00_00 | b4 | COEFF[3]_00_00 | don't care |
| COEFF[0]_00_P1 | 0 | COEFF[1]_00_P1 | 0 | COEFF[2]_00_P1 | 0 | COEFF[3]_00_P1 | don't care |
| COEFF[0]_00_P2 | r5 | COEFF[1]_00_P2 | 0 | COEFF[2]_00_P2 | b5 | COEFF[3]_00_P2 | don't care |
| COEFF[0]_P1_M2 | 0 | COEFF[1]_P1_M2 | 0 | COEFF[2]_P1_M2 | 0 | COEFF[3]_P1_M2 | don't care |
| COEFF[0]_P1_M1 | 0 | COEFF[1]_P1_M1 | g3 | COEFF[2]_P1_M1 | 0 | COEFF[3]_P1_M1 | don't care |
| COEFF[0]_P1_00 | 0 | COEFF[1]_P1_00 | 0 | COEFF[2]_P1_00 | 0 | COEFF[3]_P1_00 | don't care |
| COEFF[0]_P1_P1 | 0 | COEFF[1]_P1_P1 | 0 | COEFF[2]_P1_P1 | 0 | COEFF[3]_P1_P1 | don't care |
| COEFF[0]_P1_P2 | 0 | COEFF[1]_P1_P2 | 0 | COEFF[2]_P1_P2 | 0 | COEFF[3]_P1_P2 | don't care |
| COEFF[0]_P2_M2 | r6 | COEFF[1]_P2_M2 | 0 | COEFF[2]_P2_M2 | b6 | COEFF[3]_P2_M2 | don't care |
| COEFF[0]_P2_M1 | 0 | COEFF[1]_P2_M1 | 0 | COEFF[2]_P2_M1 | 0 | COEFF[3]_P2_M1 | don't care |
| COEFF[0]_P2_00 | r7 | COEFF[1]_P2_00 | 0 | COEFF[2]_P2_00 | b7 | COEFF[3]_P2_00 | don't care |
| COEFF[0]_P2_P1 | 0 | COEFF[1]_P2_P1 | 0 | COEFF[2]_P2_P1 | 0 | COEFF[3]_P2_P1 | don't care |
| COEFF[0]_P2_P2 | r8 | COEFF[1]_P2_P2 | 0 | COEFF[2]_P2_P2 | b8 | COEFF[3]_P2_P2 | don't care |

Setting value of coeffient select table register and bit shift control register

| Register Name | Setting Value | Register Name | Setting Value | Register Name | Setting Value | Register Name | Setting Value |
|---|---|---|---|---|---|---|---|
| COEFF_SEL_0_0 | 0 | COEFF_SEL_1_0 | 15 | COEFF_SEL_2_0 | 15 | COEFF_SEL_3_0 | 1 |
| COEFF_SEL_0_1 | 15 | COEFF_SEL_1_1 | 15 | COEFF_SEL_2_1 | 15 | COEFF_SEL_3_1 | 15 |
| COEFF_SEL_0_2 | 15 | COEFF_SEL_1_2 | 15 | COEFF_SEL_2_2 | 15 | COEFF_SEL_3_2 | 15 |
| COEFF_SEL_0_3 | 1 | COEFF_SEL_1_3 | 15 | COEFF_SEL_2_3 | 15 | COEFF_SEL_3_3 | 2 |
| CONV0_RSHIFT | 24 | CONV1_RSHIFT | 24 | CONV2_RSHIFT | 24 | CONV3_RSHIFT | 24 |
| INPUT_LSHIFT | 12 | FINAL_LSHIFT | 0 | | | | |

**Figure 54.35 Register setting of coefficient value and coefficient select table**

## 2. RGB-IR conversion

This usecase targets color filter array conversion from 4x4 RGB-Ir to Bayer. This conversion is called as “RGB-IR conversion” in the following description. Similar to the “1.Scaling usecase”, actual setting value of convolution coefficients in this case are also configurable. It is possible to adapt its value according to the applications.

### (a) B component pixels

In RGB-Ir conversion, the Ir component is located at the coordinates where the B component is located in RGGB type Bayer array. Therefore, the B component pixel at the same position is calculated by interpolating from rounding B component pixels in RGB-Ir format.

There are 2 formulas for interpolation according to coordinate of B component pixels in RGB-Ir format. Regarding B component pixels located at *(4*i+1,4*j+1)* and *(4*i+3,4*j+3)*, B component pixel for interpolation is placed at upper right and lower left from its center in RGB-Ir format. Therefore B component pixel is calculated by (B_ur+B_ll)/2 in this case, where B_ur denotes upper right pixel and B_ll denotes lower left pixel. Regarding B component pixel located at *(4*i+3,4*j+1)* and *(4*i+1,4*j+3)*, B component pixel for interpolation is placed at upper left and lower right from its center. Therefore B component pixel is calculated by (B_ul+B_lr)/2 in this case, where B_ul denotes upper left pixel and B_lr denotes lower right pixel

### (b) R component pixels

In 4x4 RGB-Ir format, R component pixels are not fully existing. In Bayer format, R component pixels exist in 2 step increments in the horizontal and vertical directions, such as at (0,0), (0,2), (2,0), (2,2) coordinate In RGB-Ir format, R component pixels exists in 4 step increments, such as at (0,0), (0,4), (4,0), (4,4) coordinate. In order to convert to Bayer format, interpolation using the surrounding R component pixels in RGB-Ir format is needed for R-component pixels located at at *(4*i,4*j+2)* or *(4*i+2,4*j)* or *(4*i,4*j+2)* such as (0,2), (2,0) or (2,2). On the other hands, regarding R component pixels located at *(4*i,4*j)* or *(4*i+4,4*j)* or *(4*i,4*j+4)* , interpolation is not needed, and the output R component pixel can be get by pass-though without any calculation.

### (c) G component pixels

Regarding G component pixels, since the location of G component pixels do not change before and after conversion, it is treated as a pass-through. Therefore, convolution coefficient should be set as pass-through, i.e. center coefficient is set to 1 and other ones are set to 0.

### (d) Register setting

Register setting for RGB-Ir conversion to realize above mentioned interpolation is shown in Figure 54.38. In Figure 54.38, each coefficient value is represented by 8 bit fixed-point number. Therefore, H’100 represents 1.0, H’80 represents 0.5, and H’40 represents 0.25 respectively.

### (e) Ir pixel handling

When applied above mentioned settings, RAW scaler itself discards Ir pixels. In order to capture Ir pixels, use a other channel which is not enabled RAW scaler output (channel with SEL_RAW_SCALED=0) and apply line count filter setting and de-interleave filter setting into this channel.

Regarding line count filtering, apply the settings that only odd-numbered lines in one frame are acquired. Regarding de-interleaving filter, apply the settings to pick up only odd-numbered pixels in one line.

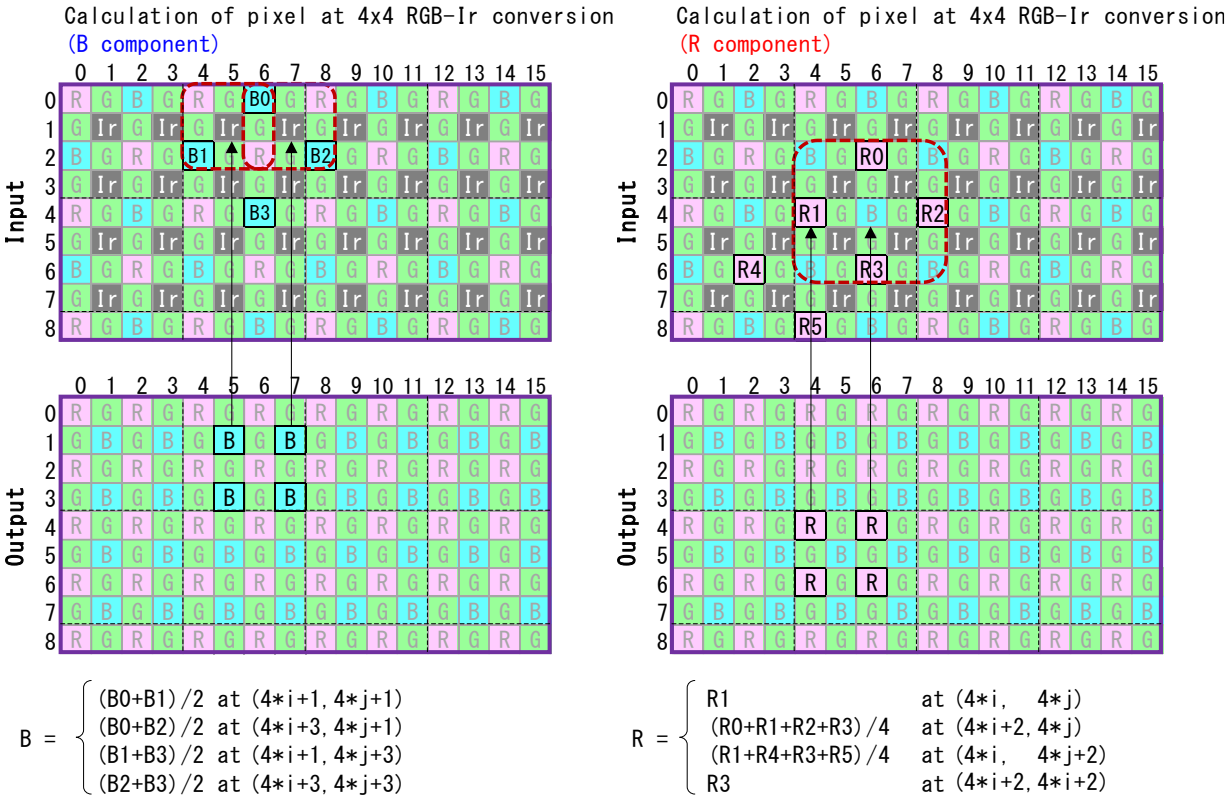
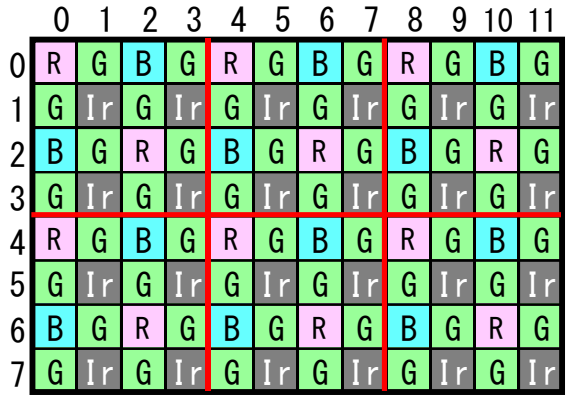



Figure 54.36 Pixel calculation at RGB-Ir conversion




coeff_select_table
(x%4, y%4)

| y%4 \ x%4 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 2 | 0 | 3 |
| 2 | 1 | 0 | 0 | 0 |
| 3 | 0 | 3 | 0 | 2 |

C(0, m, n) : Pass through

| | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| -2 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |

C(1, m, n) : Interpolation for R component

| | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| -2 | 0 | 0 | 1/4 | 0 | 0 |
| -1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1/4 | 0 | 0 | 0 | 1/4 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1/4 | 0 | 0 |

C(2, m, n) : Interpolation for B component

| | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| -2 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 1/2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1/2 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |

C(3, m, n) : Interpolation for B component

| | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| -2 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 1/2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1/2 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |

Figure 54.37 Example of coeff_select_table at RGB-Ir conversion

Setting value of convolution coefficient register

| Register Name | Setting Value | Register Name | Setting Value | Register Name | Setting Value | Register Name | Setting Value |
|---|---|---|---|---|---|---|---|
| COEFF[0]_M2_M2 | 0 | COEFF[1]_M2_M2 | 0 | COEFF[2]_M2_M2 | 0 | COEFF[3]_M2_M2 | 0 |
| COEFF[0]_M2_M1 | 0 | COEFF[1]_M2_M1 | 0 | COEFF[2]_M2_M1 | 0 | COEFF[3]_M2_M1 | 0 |
| COEFF[0]_M2_00 | 0 | COEFF[1]_M2_00 | H'40 | COEFF[2]_M2_00 | 0 | COEFF[3]_M2_00 | 0 |
| COEFF[0]_M2_P1 | 0 | COEFF[1]_M2_P1 | 0 | COEFF[2]_M2_P1 | 0 | COEFF[3]_M2_P1 | 0 |
| COEFF[0]_M2_P2 | 0 | COEFF[1]_M2_P2 | 0 | COEFF[2]_M2_P2 | 0 | COEFF[3]_M2_P2 | 0 |
| COEFF[0]_M1_M2 | 0 | COEFF[1]_M1_M2 | 0 | COEFF[2]_M1_M2 | 0 | COEFF[3]_M1_M2 | 0 |
| COEFF[0]_M1_M1 | 0 | COEFF[1]_M1_M1 | 0 | COEFF[2]_M1_M1 | 0 | COEFF[3]_M1_M1 | H'80 |
| COEFF[0]_M1_00 | 0 | COEFF[1]_M1_00 | 0 | COEFF[2]_M1_00 | 0 | COEFF[3]_M1_00 | 0 |
| COEFF[0]_M1_P1 | 0 | COEFF[1]_M1_P1 | 0 | COEFF[2]_M1_P1 | H'80 | COEFF[3]_M1_P1 | 0 |
| COEFF[0]_M1_P2 | 0 | COEFF[1]_M1_P2 | 0 | COEFF[2]_M1_P2 | 0 | COEFF[3]_M1_P2 | 0 |
| COEFF[0]_00_M2 | 0 | COEFF[1]_00_M2 | H'40 | COEFF[2]_00_M2 | 0 | COEFF[3]_00_M2 | 0 |
| COEFF[0]_00_M1 | 0 | COEFF[1]_00_M1 | 0 | COEFF[2]_00_M1 | 0 | COEFF[3]_00_M1 | 0 |
| COEFF[0]_00_00 | H'100 | COEFF[1]_00_00 | 0 | COEFF[2]_00_00 | 0 | COEFF[3]_00_00 | 0 |
| COEFF[0]_00_P1 | 0 | COEFF[1]_00_P1 | 0 | COEFF[2]_00_P1 | 0 | COEFF[3]_00_P1 | 0 |
| COEFF[0]_00_P2 | 0 | COEFF[1]_00_P2 | H'40 | COEFF[2]_00_P2 | 0 | COEFF[3]_00_P2 | 0 |
| COEFF[0]_P1_M2 | 0 | COEFF[1]_P1_M2 | 0 | COEFF[2]_P1_M2 | 0 | COEFF[3]_P1_M2 | 0 |
| COEFF[0]_P1_M1 | 0 | COEFF[1]_P1_M1 | 0 | COEFF[2]_P1_M1 | H'80 | COEFF[3]_P1_M1 | 0 |
| COEFF[0]_P1_00 | 0 | COEFF[1]_P1_00 | 0 | COEFF[2]_P1_00 | 0 | COEFF[3]_P1_00 | 0 |
| COEFF[0]_P1_P1 | 0 | COEFF[1]_P1_P1 | 0 | COEFF[2]_P1_P1 | 0 | COEFF[3]_P1_P1 | H'80 |
| COEFF[0]_P1_P2 | 0 | COEFF[1]_P1_P2 | 0 | COEFF[2]_P1_P2 | 0 | COEFF[3]_P1_P2 | 0 |
| COEFF[0]_P2_M2 | 0 | COEFF[1]_P2_M2 | 0 | COEFF[2]_P2_M2 | 0 | COEFF[3]_P2_M2 | 0 |
| COEFF[0]_P2_M1 | 0 | COEFF[1]_P2_M1 | 0 | COEFF[2]_P2_M1 | 0 | COEFF[3]_P2_M1 | 0 |
| COEFF[0]_P2_00 | 0 | COEFF[1]_P2_00 | H'40 | COEFF[2]_P2_00 | 0 | COEFF[3]_P2_00 | 0 |
| COEFF[0]_P2_P1 | 0 | COEFF[1]_P2_P1 | 0 | COEFF[2]_P2_P1 | 0 | COEFF[3]_P2_P1 | 0 |
| COEFF[0]_P2_P2 | 0 | COEFF[1]_P2_P2 | 0 | COEFF[2]_P2_P2 | 0 | COEFF[3]_P2_P2 | 0 |

Setting value of coeffient select table register and bit shift control register.

| Register Name | Setting Value | Register Name | Setting Value | Register Name | Setting Value | Register Name | Setting Value |
|---|---|---|---|---|---|---|---|
| COEFF_SEL_0_0 | 0 | COEFF_SEL_1_0 | 0 | COEFF_SEL_2_0 | 1 | COEFF_SEL_3_0 | 0 |
| COEFF_SEL_0_1 | 0 | COEFF_SEL_1_1 | 2 | COEFF_SEL_2_1 | 0 | COEFF_SEL_3_1 | 3 |
| COEFF_SEL_0_2 | 1 | COEFF_SEL_1_2 | 0 | COEFF_SEL_2_2 | 0 | COEFF_SEL_3_2 | 0 |
| COEFF_SEL_0_3 | 0 | COEFF_SEL_1_3 | 3 | COEFF_SEL_2_3 | 0 | COEFF_SEL_3_3 | 2 |
| CONV0_RSHIFT | 16 | CONV1_RSHIFT | 16 | CONV2_RSHIFT | 16 | CONV3_RSHIFT | 16 |
| INPUT_LSHIFT | 8 | FINAL_LSHIFT | 8 | | | | |

**Figure 54.38 Register setting of coefficient value and coefficient select table (RGB Ir conversion and 8 bit fixed-point precision)**

#### 54.4.9.5 Precision of convolution coefficient

The convolution coefficients which are set to the COEFF[s]_* registers are defined as fixed-point number. The bit width of the decimal area in fixed-point representation is determined by the $conv_rshift(s)$ and $input_lshift$.

$conv_rshift(s)$ and $input_lshift$ are configured by CONVs_RSHIFT (s=0,1,2,3) and INPUT_LSHIFT reigster respectively. For example, in the case of 8-bit precision, the INPUT_LSHIFT and CONVs_RHISFT registers should be set to 8. In this case, the lower 8 bits of the coefficient value in COEFF[s]_* are regarded as the decimal point area.

#### 54.4.9.6 Toggle timing of VSYNC, HSYNC and DE in RAW scaler output pixel stream

RAW scaler discards incoming VSYNC, HSYNC and DE signals in input pixel stream. RAW scaler newly generates VSYNC, HSYNC and DE signals of output pixel stream from RAW scaler. Toggle timing of these sync signals are controlled by VBLANK_FE_CYCLE, VBLANK_FS_CYCLE, HBLANK_CYCLE, FRONT_PORCH_CYCLE, BACK_PORCH_CYCLE, VS_HS_RISE_CYCLE, VS_HS_FALL_CYCLE registers. Unit of each register is “cycle”.

It is necessary to set the value such that subsequent IPs (VIN or ISP core) can correctly capture the output pixel stream from RAW scaler. See Figure 54.39 for the effect of each register.

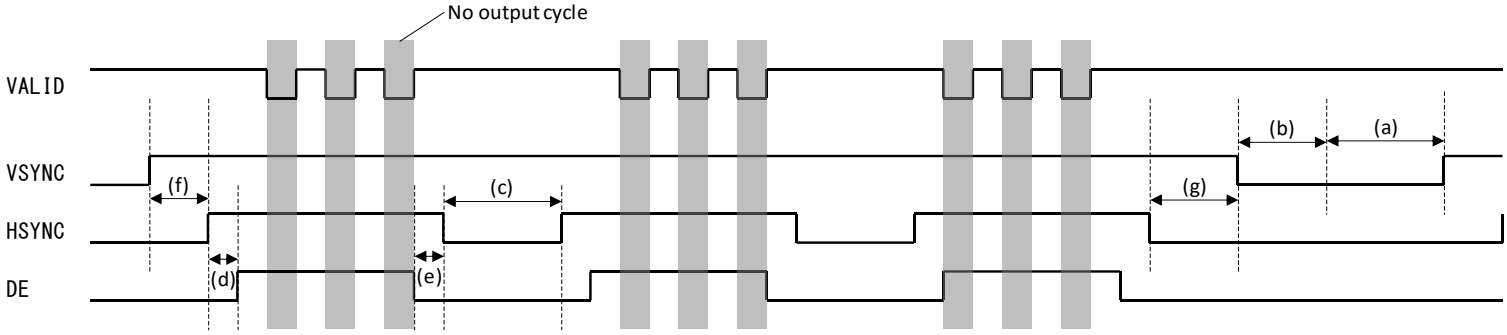



Figure 54.39 Toggle timing of VSYNC, HSYNC and DE of RAW scaler output

Correspondence of symbol (a), (b), (c), (d), (e), (f), (g) in Figure 54.39 to each register is shown as below

| Symbol | Register |
|---|---|
| (a) | VBLANK_FS_CYCLE |
| (b) | VBLANK_FE_CYCLE |
| (c) | HBLANK_CYCLE |
| (d) | FRONT_PORCH_CYCLE |
| (e) | BACK_PORCH_CYCLE |
| (f) | VS_HS_RISE_CYCLE |
| (g) | VS_HS_FALL_CYCLE |

### 54.4.10 Configuration DMA

Configuration DMA (Config DMA) is a function which realizes ISP core register setting by using DMA. Configuration data (a sequence of pair of register address and write data for writing to the ISP core) which is stored in main memory (DRAM) is read by DMA, and this configuration data is applied to the ISP core.

DMA transfer is realized by VSPX. VSPX reads configuration data via RPF0 (Read Pixel Formatter), and read RAW image via RPF0 and RPF1. RPF0 is used both configuration data and RAW image. Image data transfer and configuration data transfer via RPF0 is done at separate times and cannot be done simultaneously.

-RPF0 : ISP input RAW image (long exposure). Register configuration data

- RPF1 : ISP input RAW image (short exposure)

- RPF2 : Not used

- RPF3: Not used

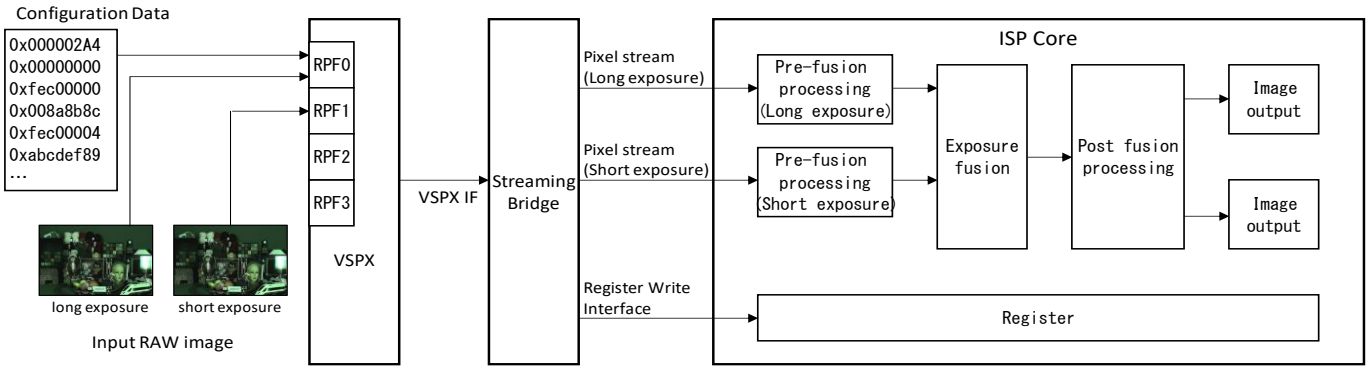



Figure 54.40 Concept of configuration DMA

- Structure of configuration data
  - Configuration data is a sequence of pair of register address and register writing data for ISP core. As a memory image, it is a sequence of alternating register address and writing data are
  - The first word denotes a length of configuration data (number of pairs of register address and register writing data)
  - The second word denotes Reserved Data. It should be H'0000 0000.
  - Sequence of pairs of (Register Address, Write data) is followed by the Reserved Data. The number of pair is shown by the first word.
  - Following is an example which shows that configuration data is consist of 674 pairs of (Register Address, Write data) .

| | |
|---|---|
| H'0000_02A4 | Length of Configuration data. 674 (=H'2A4) pairs of (Register Address, Write Data) are followe by the "Reserved Data") |
| H'0000_0000 | Reserved Data (Should be H'0000_0000) |
| H'FEC0_0000 | Register address 0 |
| H'008A_8B8C | Write data 0 |
| H'FEC0_0004 | Register address 1 |
| H'ABCD_EF89 | Write data 0 |
| ... | ... |
| ... | ... |
| H'FEC0_1384 | Register address 673 |
| H'FFFF_FFFF | Write data 673 |

Number of (Register Address, Write Data) pair is 6

Figure 54.41 Data structure of config data

- Display list for VSPX (VSPX-DL) for configuration DMA
  - A dedicated VSPX-DL for configuration data transfer is needed. This VSPX-DL is defined as "Configuration VSPX-DL" in the following description. Therefore, 2 types of VSPX-DL should be prepared. One is a VSPX-DL for input image transfer, -- this is defined as "Image Data VSPX-DL" in the following description --, and the another one is the Configuration VSPX-DL.
  - Start address of Image Data VSPX-DL is set into pNext_header field in the Configuration VSPX-DL.
  - Start address of Configuration VSPX-DL for subsequent frame is set into pNext_header field in the Image Data VSPX-DL, when different configuration from a previous frame is applied.
  - By setting pNext_header field for each VSPX-DL, sequence structure of VSPX-DLis realized

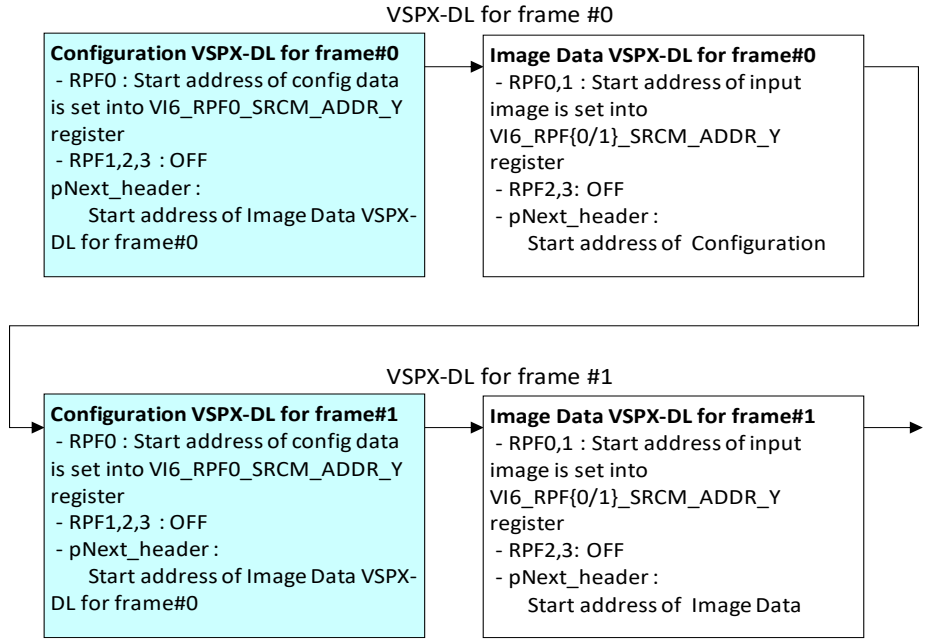



Figure 54.42 Sequence structure of VSPX-DL

- Setting of VI6_WPFn_SRCRPF register
  - Setting of RPFn_ACT[1:0] (n=0,1,2,3) field in VI6_WPFn_SRCRPF register is required in order to transfer input image and configuration data.

- RPFn_ACT[1:0] setting for Image Data VSPX-DL and Configuration VSPX-DL is shown as following table. Regarding other bit field, VIR_ACT[1:0] should be 0 and VIR_ACT2[1:0] should be 0

| | RPF3_ACT[1:0] | RPF2_ACT[1:0] | RPF1_ACT[1:0] | RPF0_ACT[1:0] |
|---|---|---|---|---|
| Image Data VSPX-DL (single exposure case) | B'00 | B'00 | B'00 | B'10 |
| Image Data VSPX-DL (dual exposure case) | B'00 | B'00 | B'01 | B'10 |
| Configuration Data VSPX-DL | B'00 | B'00 | B'00 | B'10 |

- According to the above table, VI6_WPFn_SRCRPF in Image Data VSPX-DL should be set as H'0000_0002 (single exposure) or H'0000_0006 (dual exposure).
- According to the above table, VI6_WPFn_SRCRPF in Configuration VSPX-DL should be set as H'0000_0002.
- See the VSPX manual for bit field definition of VI6_WPFn_SRCRPF register.

- Setting of VI6_DPR_RPF0_ROUTE
  - For routing configuration data read via RPF0 to the register write interface of ISP core, VI6_DPR_RPF0_ROUTE register should be set to H'1A.
- Setting of VI6_IIF_CTRL.CTRL register and ISP_STREAMER_CONFIG_DMA_CONTROL.BYTE_PER_ WORD register
  - For setting of data format of configuration data, VI6_IIF_CTRL.CTRL and ISP_STREAMER_CONFIG_ DMA_CONTROL.BYTE_PER_WORD should be set according to the following.

| Register | Setting value |
|---|---|
| VI6_IIF_CTRL.CTRL | H'13 |
| ISP_STREAMER_CONFIG_DMA_CONTROL.BYTE_PER_WORD | B'00 |

- When using Config DMA, display list of VSPX and ISP_STREAMER_CONFIG_DMA register should be set. Regarding register setting, both ISP_STREAMER_CONFIG_DMA_CONTROL.ENABLE0 register and ISP_STREAMER_CONFIG_DMA_CONTROL1.ENABLE1 register should be set to 1.
- Only memory-to-memory operation is supported when using Config DMA. Start trigger of input image under control from CPU.
- Register access from CPU to the ISP core during config DMA is running are forbidden. By reading ISP_STREAMER_CONFIG_DMA_STATUS.CONFIG_DMA_READY_BUSY register, status whether config DMA is running or not can be known. If register access to the ISP core during config DMA running is needed, it is necessary to wait completion of config DMA. See the 54.5.2 for detail.

### 54.4.11 Output DMA

A DMA function is implemented on the output side of ISP. This DMA function is a rectangle-based DMA, which transfers stream data output from ISP to the external memory (DRAM) in the form of 2D rectangle data via AXI interface. This DMA has three types of functions, DMA for output image transfer (Image Output DMA), DMA for histogram data transfer (Histogram Output DMA) and DMA for defect pixel information transfer (Defect Pixel Information DMA) . To use the output DMA, it is necessary to set related registers. However, do not overwrite registers related to DMA during ISP running.

#### 54.4.11.1 Image Output DMA

### 1. Operation

Image output DMA setting is used to transfer and store ISP output image to the external memory (DRAM).

- Register set for each image

There are two types of ISP output image (human vision image and machine vision image), and DMA settings are required for each of this two types of output images. Use the ISPCORE_DMA_IMAGE[0]_* register to set the human vision image (HV image), and use the ISPCORE_DMA_IMAGE[1]_* register to set the DMA for the machine vision image (MV image).

- Pixel format settings

Figure 54.43 shows example settings for the various formats listed under "Pixel Format" in the table. However, it does not describe settings for all case of pixel bit width. The number of pixel bit width for each color component can be set independently. In this case, pixel bit width for each color component should be less or equal to the maximal depth provided by the ISP shown in Figure 54.45. For example, “YUV422, 12Y, 12UV for HV image” pixel format in HV image, the bit depth that can be set for Y and UV components must be less than 12 bits.

PIXEL_POSITION.LSB_COMPc, PIXEL_BITWIDTH_MINUS1.BW_MINUS1_COMPc, and PIXEL_BPP.BPP_COMPc (c is index of color components, c=0,1,2,3) register should be set according to the pixel format of the HV images and MV images.

Pixel bit width for each color component is set by using PIXEL_POSITION and PIXEL_BITWIDTH_MINUS1 registers . For example, in 12 bit pixel case such as Y12 in YUV422 in HV image, Y component is packed between bit-23(MSB) and bit-12(LSB) according to Figure 54.43. Therefore, PIXEL_POSITION.LSB_COMP1=12 and PIXEL_BITWIDTH_MINUS 1.BW_MINUS1_COMP1 = 11 (=12 -1) .

- Base address and stride

For the start address of the output image, use the BASE_ADDRESS register to specify the start address of the output image for each component. For example, in the case of RGB images, set the starting address of the R component image to the BASE_ADDRESS_COMP0 register, the starting address of the G component image to the BASE_ADDRESS_COMP1 register, and the B component image to the BASE_ADDRESS_COMP2 register. In this case, the output image is 3-plannar RGB format.

Base address of output image must be aligned to 256 bytes boundary. So, setting value of BASE_ADDRESS_COMPc (c=0,1,2,3 ) must be multiple of 256.

As for the stride (line offset) of the output image, use the STRIDE register to set the value of the stride of the output image in bytes for each component. The setting value of this register does not depend on the pixel format or the number of bytes per pixel. For example, in the case of the RGB image, set the stride of the R component image to the STRIDE_COMP0 register, the stride of the G component image to the STRIDE_COMP1 register, and the stride of the B component image to the STRIDE_COMP2 register.

Stride, i.e., the setting value of STRIDE_COMPc (c=0,1,2,3) register, must be a multiple of 256.

- Register settings for each frame

The pixel format, start address, and stride settings described above are set individually for each frame. The FRAME[n] keyword in the register name identifies the frame number. The [n] in FRAME[n] denotes the remainder of the frame number divided by 4, i.e., use FRAME[0] for DMA settings for frames 0, 4, 8, …, and FRAME[1] for frames 1, 5, 9, …..

- AXI transaction ID

AXI transaction ID (AWID, WID, BID) can be set individually for each component. AXI_ID.AXI_ID_COMP0 register is the AXI transaction ID for component 0 (R component for RGB images), AXI_ID.AXI_ID_COMP1 register is the AXI transaction ID for component 1 (G component for RGB images), and AXI_ID.COMP2 register is the AXI transaction ID for component 2 (B component for RGB images).

- AXI burst length

The AXI burst length is fixed and is specified by ISPCORE_DMA_IMAGE[k]_AXI_CONFIG.BURST_LENGTH_ MINUS1 register. This register is set to the AXI burst length minus one. The supported burst length is only 8. Therefore, the ISPCORE_DMA_IMAGE[k]_AXI_CONFIG.BURST_ LENGTH_MINUS1 should be set to 7.

- Activation (enabling) of DMA

By setting 1 to the ISPCORE_DMA_IMAGE[k]_AXI_CONFIG.DMA_ENABLE register, the image output DMA is activated.

- Flush out setting

If ISP output image is used immediately after the end of frame interrupt is asserted, a flush out setting is needed for flushing out the remained data in image output DMA pipeline at the end of frame. To use flushing out, set the PADDING_EOF register to a non-zero value.

The minimum value for ISPCORE_DMA_IMAGE[k]_FLUSH_OUT.PADDING_PIXEL_EOF (hereafter, called PADDING_EOF) register is 512, and setting value of this register must be a multiple of 256. On the other hand, considering the data transfer of output image on SoC memory bus, larger value for this register is better. In this case, the setting value of this register should be obtained based on the image width (unit is pixel).

If image width is not a multiple of 256, the PADDING_PIXEL_EOF register should be set to a value that is both greater than image width and a smallest value in a multiple of 256. For example, if image width is 770 (=256x3+2) pixel, then the setting value of PADDING_EOF register is 1024 (256x4). If image width is not a multiple of 256, set the PADDING_EOF register to the image width value.

- Back Pressure Stopping

ISPCORE_BACK_PRESSURE.BP_STOP_MODE register is used to set enable/disable of back pressure stopping control. By default, back pressure stopping is enabled.

When back pressure stopping control is enabled, if the memory bandwidth of AXI write from ISP is insufficient, the ISP processing pipeline is automatically paused according to the AXI write response latency. In addition, during ISP pipeline is paused stopping signal is asserted to VSPX, and pixel streaming from VSPX to ISP is also stopped. That is, pixel streaming from VSPX is stopped in conjunction with ISP pipeline pause. This control can prevent ISP output images from collapsing due to insufficient memory bandwidth for ISP.

When back pressure stopping control is enabled, ISPCORE_BACK_PRESSURE.BP_THRESHOLD register must be set to 1

| | Pixel Format | Register Settings | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MODE | PIXEL_POSITION | | | | PIXEL_BITWIDTH_MINUS1 | | | | PIXEL_BPP (BPP) | | | | PIXEL_BPP (LINE_EN) | | | |
| | | NUM_OF_COMPONENT_MINUS1 | LSB_COMP3 | LSB_COMP2 | LSB_COMP1 | LSB_COMP0 | BW_MINUS1_COMP3 | BW_MINUS1_COMP2 | BW_MINUS1_COMP1 | BW_MINUS1_COMP0 | BPP_COMP3 | BPP_COMP2 | BPP_COMP1 | BPP_COMP0 | LINE_EN_COMP3 | LINE_EN_COMP2 | LINE_EN_COMP1 | LINE_EN_COMP0 |
| HV Image | RGB888 (3 plannar) | 2 | 0 | 16 | 8 | 0 | 0 | 7 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 3 |
| | RGB888 (interleaved) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 23 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 3 |
| | YUV422, 12Y, 12UV for HV | 1 | 0 | 0 | 12 | 0 | 0 | 0 | 11 | 11 | 0 | 0 | 1 | 1 | 0 | 0 | 3 | 3 |
| | YUV422, 12Y, 8UV for HV | 1 | 0 | 0 | 12 | 4 | 0 | 0 | 11 | 7 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 3 |
| | YUV422, 8Y, 8UV for HV | 1 | 0 | 0 | 16 | 4 | 0 | 0 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 |
| | YUV420, 12Y, 12UV for HV | 1 | 0 | 0 | 12 | 0 | 0 | 0 | 11 | 11 | 0 | 0 | 1 | 1 | 0 | 0 | 3 | 2 |
| | YUV420, 8Y, 8UV for HV | 1 | 0 | 0 | 16 | 4 | 0 | 0 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 |
| | | | | | | | | | | | | | | | | | | |
| MV Image | RGB16 | 2 | 0 | 32 | 16 | 0 | 0 | 15 | 15 | 15 | 0 | 1 | 1 | 1 | 0 | 3 | 3 | 3 |
| | YUV422, 24Y, 16UV | 1 | 0 | 0 | 24 | 8 | 0 | 0 | 23 | 15 | 0 | 0 | 2 | 1 | 0 | 0 | 3 | 3 |
| | YUV420, 24Y, 16UV | 1 | 0 | 0 | 24 | 8 | 0 | 0 | 23 | 15 | 0 | 0 | 2 | 1 | 0 | 0 | 3 | 2 |
| | LUV444, 16L, 8U, 8V | 2 | 0 | 32 | 24 | 16 | 0 | 15 | 7 | 7 | 0 | 1 | 0 | 0 | 0 | 3 | 3 | 3 |
| | LUV422, 16L, 8UV | 1 | 0 | 0 | 32 | 24 | 0 | 0 | 15 | 7 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 3 |
| | LUV420, 16L, 8UV | 1 | 0 | 0 | 32 | 24 | 0 | 0 | 15 | 7 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 2 |
| | RGB12 | 2 | 0 | 36 | 24 | 12 | 0 | 11 | 11 | 11 | 0 | 1 | 1 | 1 | 0 | 3 | 3 | 3 |
| | YUV422, 8Y, 8UV for MV | 1 | 0 | 0 | 40 | 28 | 0 | 0 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 |
| | YUV420, 8Y, 8UV for MV | 1 | 0 | 0 | 40 | 28 | 0 | 0 | 7 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 |
| | YUV422, 12Y, 12UV for MV | 1 | 0 | 0 | 36 | 24 | 0 | 0 | 11 | 11 | 0 | 0 | 1 | 1 | 0 | 0 | 3 | 3 |
| | YUV420, 12Y, 12UV for MV | 1 | 0 | 0 | 36 | 24 | 0 | 0 | 11 | 11 | 0 | 0 | 1 | 1 | 0 | 0 | 3 | 2 |
| | LUV444, 12L, 8U, 8V | 2 | 0 | 36 | 28 | 20 | 0 | 11 | 7 | 7 | 0 | 1 | 0 | 0 | 0 | 3 | 3 | 3 |
| | LUV422, 12L, 8UV | 1 | 0 | 0 | 36 | 28 | 0 | 0 | 11 | 7 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 3 |
| | LUV420, 12L, 8UV | 1 | 0 | 0 | 36 | 28 | 0 | 0 | 11 | 7 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 2 |
| | RGBY888 | 3 | 40 | 32 | 24 | 16 | 7 | 7 | 7 | 7 | 0 | 0 | 0 | 0 | 3 | 3 | 3 | 3 |

**Figure 54.43 Setting examples for some pixel format**

## 2. Packing Format and storing image on external memory

This section describes data image for the 3 observation points (A), (B) and (C). The location of each observation points for (A), (B) and (C) is shown in Figure 54.44.

Observation point (A) is the pixel stream port that connects the ISP processing pipeline to the DMA function, and is the data path for one packed pixel per one clock.

Observation point (B) is AXI Write channel, indicating the packing of the WDATA[255:0] signal.

Observation point (C) indicates the rectangle based storing format of the image data in the external memory (DRAM).

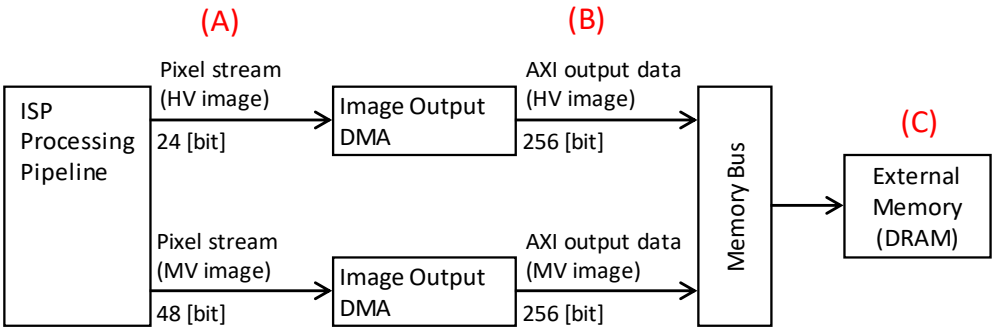



**Figure 54.44 Observation points of pixel packing**

Packing format on pixel stream port (Observation point (A)) for each pixel format is shown in the Figure 54.45. This packing shown in the Figure 54.45 is fixed by the ISP IP specification, and the allocation of each bit cannot be changed.

Note: There are no specific setting for "YUV422, 8Y,8V" in the ISP IP. In case of "YUV422, 8Y,8V", the same setting as "YUV422 12Y, 12UV" should be applied to the ISP IP. By applying the setting shown in the Figure 54.43 to the image output DMA, upper 8 bit value in 12 bit pixel is extracted and transferred to the external memory.

Note: There are no specific setting for "YUV420, 8Y,8V" in the ISP IP. In this case, the same settings as for "YUV420 12Y, 12UV" should be applied to the ISP IP. This is the same concept as in case of "YUV422, 8Y, 8UV", i.e., extracting upper 8 bit value in 12 bit pixel.

| | Pixel Format | Each bit in pixel stream 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|---|---|---|
| HV Image | RGB888 (3 plannar) | R R R R R R R R G G G G G G G G B B B B B B B B |
| | RGB888 (interleaved) | R R R R R R R R G G G G G G G G B B B B B B B B |
| | YUV422, 12Y, 12UV for HV | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| | YUV422, 12Y, 8UV for HV (*1) | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| | YUV422, 8Y, 8UV for HV (*1) | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| | YUV420, 12Y, 12UV for HV (Even lines) | Y Y Y Y Y Y Y Y Y Y Y Y |
| | (Odd lines) | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| | YUV420, 8Y, 8UV for HV (*2) (Even lines) | Y Y Y Y Y Y Y Y Y Y Y Y |
| | (Odd lines) | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| MV Image | RGB16 | R R R R R R R R R R R R R R R R G G G G G G G G G G G G G G G G B B B B B B B B B B B B B B B B |
| | YUV422, 24Y, 16UV | Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV UV UV UV UV |
| | YUV420, 24Y, 16UV (Even lines) | Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y |
| | (Odd lines) | Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV UV UV UV UV |
| | LUV444, 16L, 8U, 8V | L L L L L L L L L L L L L L L L U U U U U U U U V V V V V V V V |
| | LUV422, 16L, 8UV | L L L L L L L L L L L L L L L L UV UV UV UV UV UV UV UV |
| | LUV420, 16L, 8UV (Even lines) | L L L L L L L L L L L L L L L L |
| | (Odd lines) | L L L L L L L L L L L L L L L L UV UV UV UV UV UV UV UV |
| | RGB12 | R R R R R R R R R R R R G G G G G G G G G G G G B B B B B B B B B B B B |
| | YUV422, 8Y, 8UV for MV (*3) | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| | YUV420, 8Y, 8UV for MV (*4) (Even lines) | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| | (Odd lines) | Y Y Y Y Y Y Y Y Y Y Y Y |
| | YUV422, 12Y, 12UV for MV | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| | YUV420, 12Y, 12UV for MV (Even lines) | Y Y Y Y Y Y Y Y Y Y Y Y |
| | (Odd lines) | Y Y Y Y Y Y Y Y Y Y Y Y UV UV UV UV UV UV UV UV UV UV UV UV |
| | LUV444, 12L, 8U, 8V | L L L L L L L L L L L L U U U U U U U U V V V V V V V V |
| | LUV422, 12L, 8UV | L L L L L L L L L L L L UV UV UV UV UV UV UV UV |
| | LUV420, 12L, 8UV (Even lines) | L L L L L L L L L L L L |
| | (Odd lines) | L L L L L L L L L L L L UV UV UV UV UV UV UV UV |
| | RGBY888 | R R R R R R R R G G G G G G G G B B B B B B B B Y Y Y Y Y Y Y Y |

(*1): Should be applied the same setting as YUV422, 12Y, 12UV for HV into the ISP-IP
(*2): Should be applied the same setting as YUV420, 12Y, 12UV for HV into the ISP-IP
(*3): Should be applied the same setting as YUV422, 12Y, 12UV for MV into the ISP-IP
(*4): Should be applied the same setting as YUV420, 12Y, 12UV for MV into the ISP-IP

**Figure 54.45 Pixel packing format on observation point (A) (24bit@HV image, 48bit@MV image)**

Packing format on the AXI interface (observation point (B)) for each pixel format and component is shown in Figure 54.46 and Figure 54.47. These figures show the bit allocation in the form of a 32-bit area extracted from the AXI WDATA[255:0]. This bit allocation appears when the value shown in Figure 54.43 is applied to the Image Output DMA.

Figure 54.48 shows the storing format on the external memory (observation point (C)) in the RGB16 format case. Each of the components is assigned a separated memory area. B, G and R components are separated and expanded into three rectangular areas in the external memory.

In this format, one pixel uses 2 bytes (2 byte per pixel). One line (W pixel) in one frame uses 2xW bytes. Line offset (stride) for each components are specified by the STRIDE_COMPn (n=0,1,2) register (unit is byte). Therefore, start address of each line is calculated by BASE_ADDRESS_COMPn + STRDE_COMPn * h (h is line index).

| | Pixel Format | AXI output data WDATA[32*n+31:32*n] (n=0～7, Component0) 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | bit per pixel |
|---|---|---|---|
| HV Image | RGB888 (3 plannar) | B3 \| B2 \| B1 \| B0 | 8 |
| | RGB888 (interleaved) | 0 \| R0 \| G0 \| B0 | 24 |
| | YUV422, 12Y, 12UV for HV | 0 \| UV1 \| 0 \| UV0 | 12 |
| | YUV422, 12Y, 8UV for HV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| | YUV422, 8Y, 8UV for HV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| | YUV420, 12Y, 12UV for HV | 0 \| UV1 \| 0 \| UV0 | 12 |
| | YUV420, 8Y, 8UV for HV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| MV Image | RGB16 | B1 \| B0 | 16 |
| | YUV422, 24Y, 16UV | UV1 \| UV0 | 16 |
| | YUV420, 24Y, 16UV | UV1 \| UV0 | 16 |
| | LUV444, 16L, 8U, 8V | V3 \| V2 \| V1 \| V0 | 8 |
| | LUV422, 16L, 8UV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| | LUV420, 16L, 8UV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| | RGB12 | 0 \| B1 \| 0 \| B0 | 12 |
| | YUV422, 8Y, 8UV for MV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| | YUV420, 8Y, 8UV for HV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| | YUV422, 12Y, 12UV for MV | 0 \| UV1 \| 0 \| UV0 | 12 |
| | YUV420, 12Y, 12UV for MV | 0 \| UV1 \| 0 \| UV0 | 12 |
| | LUV444, 12L, 8U, 8V | V3 \| V2 \| V1 \| V0 | 8 |
| | LUV422, 12L, 8UV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| | LUV420, 12L, 8UV | UV3 \| UV2 \| UV1 \| UV0 | 8 |
| | RGBY888 | Y3 \| Y2 \| Y1 \| Y0 | 8 |

| | Pixel Format | AXI output data WDATA[32*n+31:32*n] (n=0～7, Component1) 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | bit per pixel |
|---|---|---|---|
| HV Image | RGB888 (3 plannar) | G3 \| G2 \| G1 \| G0 | 8 |
| | RGB888 (interleaved) | (No output data) | 0 |
| | YUV422, 12Y, 12UV for HV | 0 \| Y1 \| 0 \| Y0 | 12 |
| | YUV422, 12Y, 8UV for HV | 0 \| Y1 \| 0 \| Y0 | 12 |
| | YUV422, 8Y, 8UV for HV | Y3 \| Y2 \| Y1 \| Y0 | 8 |
| | YUV420, 12Y, 12UV for HV | 0 \| Y1 \| 0 \| Y0 | 12 |
| | YUV420, 8Y, 8UV for HV | Y3 \| Y2 \| Y1 \| Y0 | 8 |
| MV Image | RGB16 | G1 \| G0 | 16 |
| | YUV422, 24Y, 16UV | 0 \| Y0 | 24 |
| | YUV420, 24Y, 16UV | 0 \| Y0 | 24 |
| | LUV444, 16L, 8U, 8V | U3 \| U2 \| U1 \| U0 | 8 |
| | LUV422, 16L, 8UV | L1 \| L0 | 16 |
| | LUV420, 16L, 8UV | L1 \| L0 | 16 |
| | RGB12 | 0 \| G1 \| 0 \| G0 | 12 |
| | YUV422, 8Y, 8UV for MV | Y3 \| Y2 \| Y1 \| Y0 | 8 |
| | YUV420, 8Y, 8UV for HV | Y3 \| Y2 \| Y1 \| Y0 | 8 |
| | YUV422, 12Y, 12UV for MV | 0 \| Y1 \| 0 \| Y0 | 12 |
| | YUV420, 12Y, 12UV for MV | 0 \| Y1 \| 0 \| Y0 | 12 |
| | LUV444, 12L, 8U, 8V | U3 \| U2 \| U1 \| U0 | 8 |
| | LUV422, 12L, 8UV | 0 \| L1 \| 0 \| L0 | 12 |
| | LUV420, 12L, 8UV | 0 \| L1 \| 0 \| L0 | 12 |
| | RGBY888 | B3 \| B2 \| B1 \| B0 | 8 |

**Figure 54.46 Pixel packing format on observation point (B) (Component0, Component1)**

| | Pixel Format | AXI output data WDATA[32*n+31:32*n] (n=0～7,Component2) 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | bit per pixel |
|---|---|---|---|
| HV Image | RGB888 (3 plannar) | R3 \| R2 \| R1 \| R0 | 8 |
| | RGB888 (interleaved) | (No output data) | 0 |
| | YUV422, 12Y, 12UV for HV | (No output data) | 0 |
| | YUV422, 12Y, 8UV for HV | (No output data) | 0 |
| | YUV422, 8Y, 8UV for HV | (No output data) | 0 |
| | YUV420, 12Y, 12UV for HV | (No output data) | 0 |
| | YUV420, 8Y, 8UV for HV | (No output data) | 0 |
| MV Image | RGB16 | R1 \| R0 | 16 |
| | YUV422, 24Y, 16UV | (No output data) | 0 |
| | YUV420, 24Y, 16UV | (No output data) | 0 |
| | LUV444, 16L, 8U, 8V | L1 \| L0 | 16 |
| | LUV422, 16L, 8UV | (No output data) | 0 |
| | LUV420, 16L, 8UV | (No output data) | 0 |
| | RGB12 | 0 \| R1 \| 0 \| R0 | 12 |
| | YUV422, 8Y, 8UV for MV | (No output data) | 0 |
| | YUV420, 8Y, 8UV for HV | (No output data) | 0 |
| | YUV422, 12Y, 12UV for MV | (No output data) | 0 |
| | YUV420, 12Y, 12UV for MV | (No output data) | 0 |
| | LUV444, 12L, 8U, 8V | 0 \| L1 \| 0 \| L0 | 12 |
| | LUV422, 12L, 8UV | (No output data) | 0 |
| | LUV420, 12L, 8UV | (No output data) | 0 |
| | RGBY888 | G3 \| G2 \| G1 \| G0 | 8 |

| | Pixel Format | AXI output data WDATA[32*n+31:32*n] (n=0～7,Component3) 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | bit per pixel |
|---|---|---|---|
| HV Image | RGB888 (3 plannar) | (No output data) | 0 |
| | RGB888 (interleaved) | (No output data) | 0 |
| | YUV422, 12Y, 12UV for HV | (No output data) | 0 |
| | YUV422, 12Y, 8UV for HV | (No output data) | 0 |
| | YUV422, 8Y, 8UV for HV | (No output data) | 0 |
| | YUV420, 12Y, 12UV for HV | (No output data) | 0 |
| | YUV420, 8Y, 8UV for HV | (No output data) | 0 |
| MV Image | RGB16 | (No output data) | 0 |
| | YUV422, 24Y, 16UV | (No output data) | 0 |
| | YUV420, 24Y, 16UV | (No output data) | 0 |
| | LUV444, 16L, 8U, 8V | (No output data) | 0 |
| | LUV422, 16L, 8UV | (No output data) | 0 |
| | LUV420, 16L, 8UV | (No output data) | 0 |
| | RGB12 | (No output data) | 0 |
| | YUV422, 8Y, 8UV for MV | (No output data) | 0 |
| | YUV420, 8Y, 8UV for HV | (No output data) | 0 |
| | YUV422, 12Y, 12UV for MV | (No output data) | 0 |
| | YUV420, 12Y, 12UV for MV | (No output data) | 0 |
| | LUV444, 12L, 8U, 8V | (No output data) | 0 |
| | LUV422, 12L, 8UV | (No output data) | 0 |
| | LUV420, 12L, 8UV | (No output data) | 0 |
| | RGBY888 | R3 \| R2 \| R1 \| R0 | 8 |

Figure 54.47 Pixel packing format on observation point (B) (Component2, Component3)

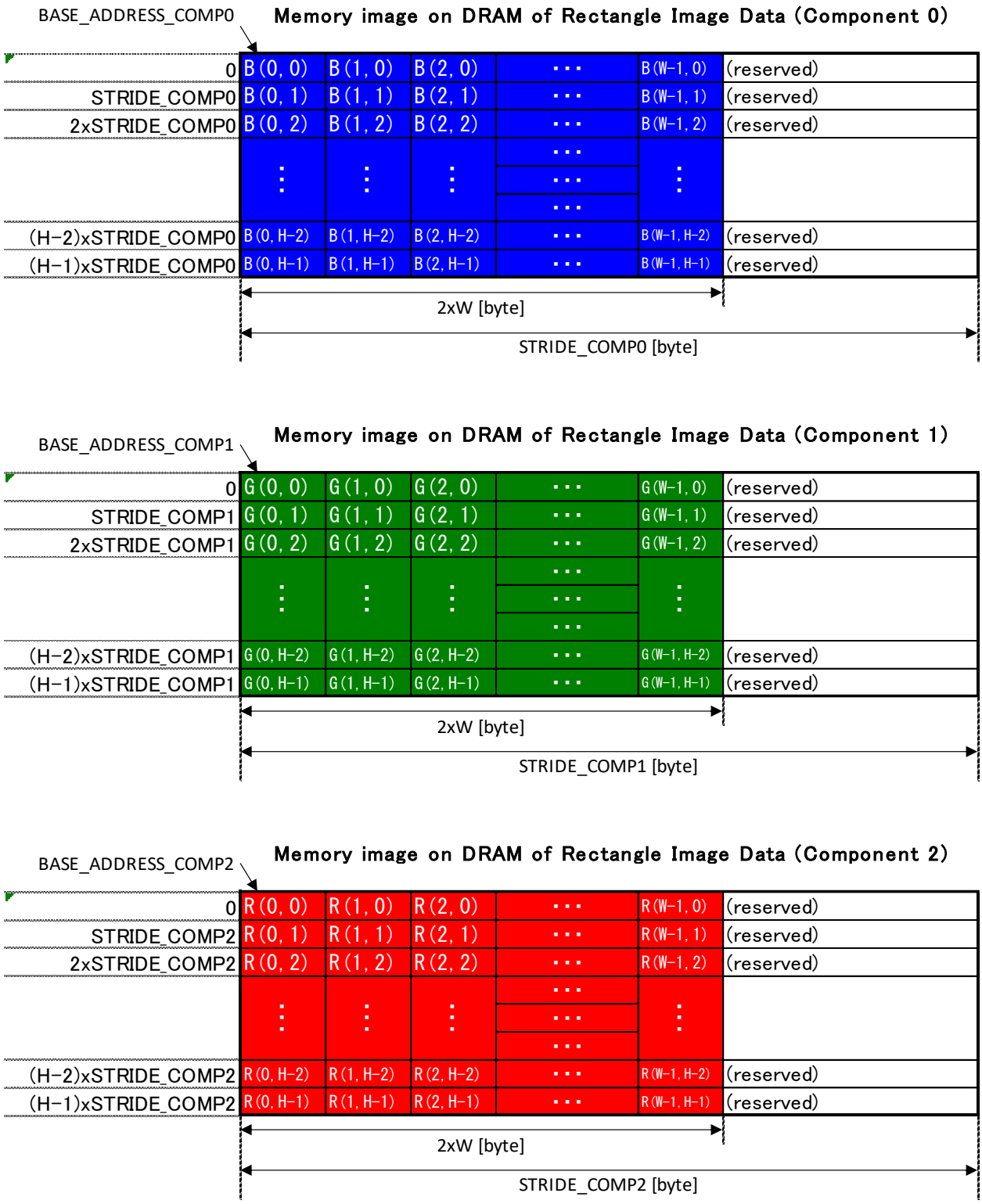



Figure 54.48 Storing format on observation point (C) (external memory(DRAM), RGB16 case)

### 3. Operation of Frame Count Register

Image Output DMA has 4 planed cyclic frame buffer. This is implemented in both Image DMA for HV (HV image DMA) and Image Output DMA for MV (MV image DMA). Each plane is switched frame by frame. Thus, plane index (frame buffer index) is incremented at the timing when the final pixel in one frame is transferred from ISP Core.

By using the ISPCORE_DMA_IMAGE[k]_FRAME_MONITOR.FRAME_COUNT register (hereafter it is referred to as FRAME_COUNT register), it is possible to know which buffer was used (frame buffer index) most recently. This section describes a recommended operation to use FRAME_COUNT register.

- Assumptions

There are several prerequisites to use FRAME_COUNT register.

- The memory bandwidth for the writing output image from ISP must be sufficient to transfer the real-time input image.
- The flush completion time (T1) of the previous frame must be before the start timing of pixel input (T2) of the following frame (T1<T2). For detail of T1 and T2, refer to Figure 54.49. If this assumption is not satisfied, pixels of subsequent frames are input during processing the previous frame. It results an overflow of the pixel acquisition FIFO in the ISP core.

- Operations

Before starting ISP processing, an interrupt which is asserted at VSYNC rising edge (v_start_edge interrupt) as a interrupt source is enabled. By this setting, an interrupt to be issued when the first pixel is input to the ISP Core. When detecting interrupt from the ISP core, the processing shown below is performed depending on each interrupt factor.

(1) If interrupt source is VSYNC rising edge interrupt, read the FRAME_COUNT register of HV image DMA and MV image DMA. The value of each read should be retained.

(2) If interrupt source is HV image output completed (rpp_out_frame_out), the modulo divided by 4 of the read value at (1) is used as a frame buffer index for the HV image

(3) If interrupt source is MV image output completed (mv_out_frame_out), the modulo divided by 4 of the the read value at (1) is used as a frame buffer index for the MV image.

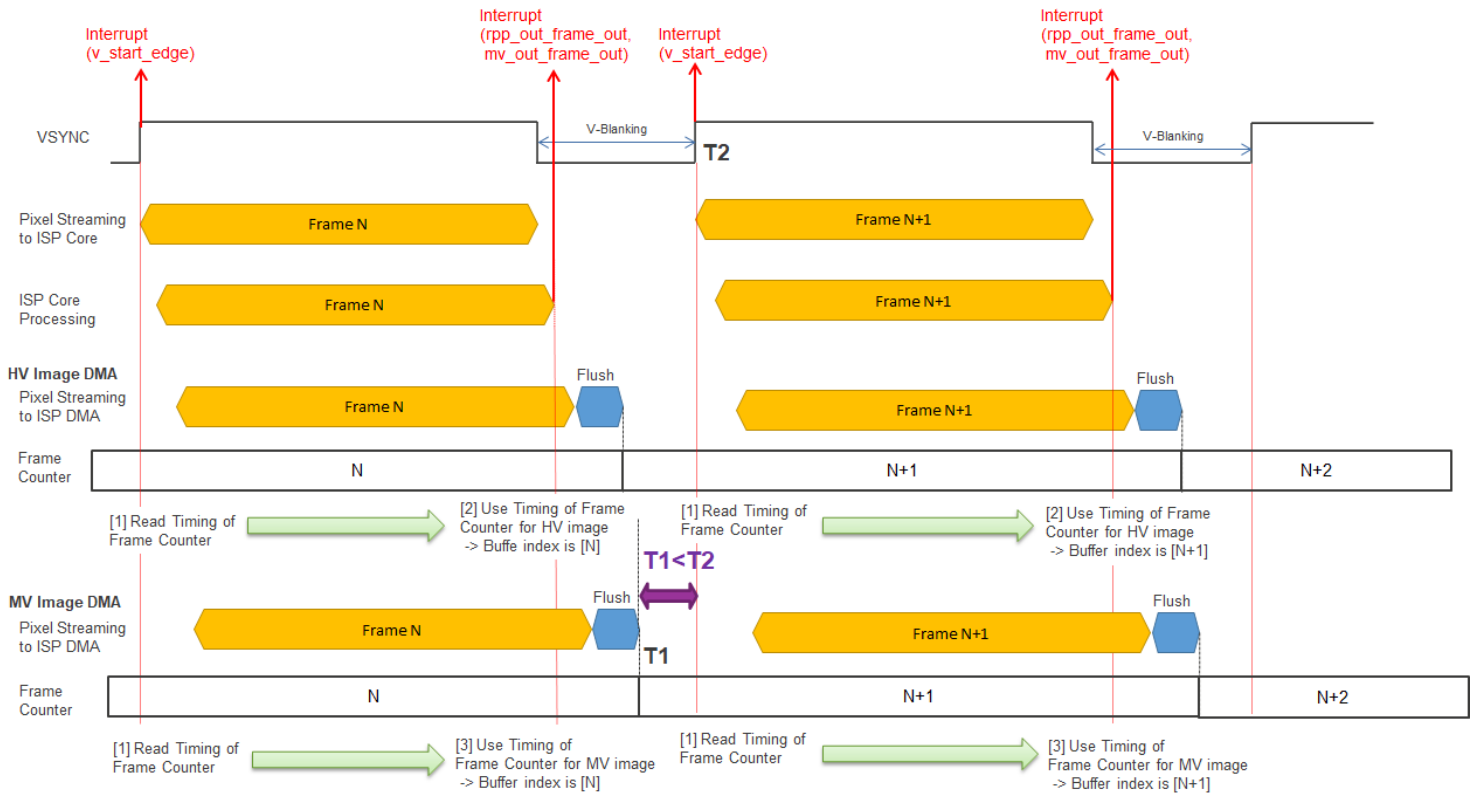



**Figure 54.49 Timing chart of frame count register operation.**

### 54.4.11.2 Histogram Output DMA

Histogram Output DMA transfers 1024 bin histogram data which is streamed out from ISP to the external memory via AXI as 2D rectangle data. The stored histogram data on the external memory consists of [64 bin per line] x [16 lines] rectangle format, and its stride is fixed at 512 bytes

#### 1. Operation

- Activation of histogram output

  In order to stream out histogram data from ISP, it is necessary to activate histogram output function. For more information, please refer to the user's manual released from ISP IP developer.

- Base address (start address)

Base address (start address) of histogram data is specified by ISPCORE_DMA_HISTOGRAM_BASE_ADDRESS_ FRAME[n] register. Same as Image Output DMA, [n] in "FRAME[n]" denotes the remainder of the frame number divided by 4.

Set the start adderss of the histogram data for frame 0, 4, 8.,,, to the ISPCORE_DMA_HISTOGRAM_BASE_ ADDRESS_ FRAME[0]. The same applies to the other frames.

- AXI burst length

Setting register for AXI burst length is shared to the Image output DMA (human vision image). So, ISPCORE_DMA_ IMAGE[0]_AXI_CONFIG.BURST_LENGTH_MINUS1 is also used for histogram. Same as image output DMA, this register is also should be set to7.

- Activation (enabling) of DMA

Setting register of activation of histogram output DMA is shared to the Image Output DMA (human vision image). So, when enabling Image output DMA by setting 1 to ISPCORE_DMA_IMAGE[0]_AXI_CONFIG.DMA_ENABLE register, then histogram output DMA is activated automatically

### 2. Memory image

Memory image stored on external memory is shown in Figure 54.50. One word (1 bin) of histogram data is 4 bytes, and 64 bins (256 byte data) make up one line. By aligning one line histogram data (64 bins) for 16 lines, the histogram data of one frame (1024bins) is composed and output to the external memory. The line offset, i.e. stride is always 512 bytes (fixed). Therefore, the latter 256 bytes area in one line is reserved area.

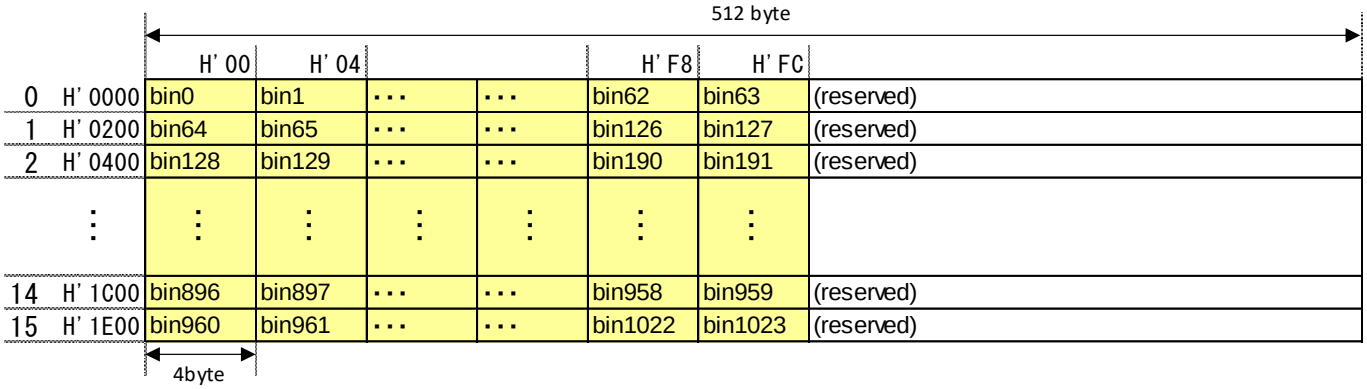



Figure 54.50 Memory image on DRAM of Histogram Data

#### 54.4.11.3 Defect Pixel Information DMA

Defect pixel information is a sequence of coordinate of defect pixels included in one input frame. Defect pixel information DMA transfers defect pixel information which is streamed out from ISP to the external memory via AXI.

Defect pixel information DMA uses 64 bit type AXI (AXI64) interface. There are 2 ports of AXI 64 interface. Each AXI64 port is assigned each pre-fusion pipe of ISP IP.

### 1. Operation

- Activation of defect pixel correction

  In order to stream out defect pixel information from ISP, it is necessary to activate defect pixel correction. For more information, please refer to the user's manual released from ISP IP developer.

- Base address (start address)

Base address (start address) of defect pixel information is specified by ISPCORE_DMA_DPCC[k]_BASE_ADDRESS_ FRAME[n] register. The index [k] in "DPCC[k]" denotes a index of pre-fusion pipe of ISP IP. DPCC[0] means pre-fusion pipe1 (long exposure pipe) and DPCC[1] is pre-fusion pipe2 (short exposure pipe), respectively.

The index [n] in "FRAME[n]" denotes the remainder of the frame number divided by 4. Set the start adderss of the defect pixel information for frame 0, 4, 8.,,, to the ISPCORE_DMA_DPCC[k]_BASE_ ADDRESS_ FRAME[0]. The same applies to the other frames.

- AXI burst length

AXI burst length is fixed and is specified by ISPCORE_DMA_DPCC[k]_AXI_CONFIG.BURST_LENGTH_MINUS1 register. The supported burst length is only 8. Therefore, ISPCORE_DMA_DPCC[k]_AXI_CONFIG.BURST_ LENGTH_MINUS1 must be set to 7.

- Flush out setting

  After all defect pixel information in one frame has been output from ISP, flush out operation should be executed for the DMA. For the flushing out operation, DPCC_PADDING_EOF register should be set to 16.

  With this setting, after the final word of defect pixel information is output from the ISP at the end of one frame, the DMA writes the defect pixel information data that remains in the DMA (such as internal FIFO of DMA) to the memory by writing padding data.

- Activation (enabling) of DMA

By setting 1 to the ISPCORE_DMA_DPCC[k]_AXI_CONFIG.DMA_ENABLE register, the defect pixel information DMA is activated.

### 2. Memory image

Memory image stored on external memory is shown in Figure 54.52, Figure 54.53 and Figure 54.54. In these figures, M is the number of defect pixel information (length of defect pixel information).

One word (4bytes) of defect pixel information is described as DP(*i*) (*i*=0,1,…M-1). DP(*i*) shows the coordinate of detected defect pixel in the ISP. Therefore, the defect pixel information that is output to the the external memory is composed a sequence of DP(*i*).

Following the final data of defect pixel information, an end code (EndC: H’FFFFFFFF) is output from ISP. Even if length of defect pixel information is 0 (M=0), this end code H’FFFFFFFF is output from ISP when ISP is configured as defect pixel detection function is enabled.

Following the end code, padding code (PadC:H’FFFFFFFE) is output from DMA. These padding codes are generated by the DMA for the purpose of flushing out the defect pixel information which are remained at the internal FIFO of DMA.

As shown in these figures, padding codes are output until it reaches the aligned length to the AXI burst length boundary (8 burst: 64 byte). Notes that 16 must be set in the DPCC_PADDING_EOF register in order for padding data to be output as shown in these figures. However, in case of M=16×n+15, padding code is not output because the length M including the end code is aligned to the AXI burst length.

The 64-byte area following the padding area should be reserved as a reserved area (forbidden use). This area is prohibited to use.

EndC = H'FFFFFFFF
PadC = H'FFFFFFFE

In case of M=16×n (n=0, 1, ···, N-1)

| Offset | +H'00 | +H'04 | +H'08 | +H'0C | +H'10 | +H'14 | +H'18 | +H'1C | +H'20 | +H'24 | +H'28 | +H'2C | +H'30 | +H'34 | +H'38 | +H'3C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H'0000 | DP(0) | DP(1) | DP(2) | DP(3) | DP(4) | DP(5) | DP(6) | DP(7) | DP(8) | DP(9) | DP(10) | DP(11) | DP(12) | DP(13) | DP(14) | DP(15) |
| H'0040 | DP(16) | DP(17) | DP(18) | DP(19) | DP(20) | DP(21) | DP(22) | DP(23) | DP(24) | DP(25) | DP(26) | DP(27) | DP(28) | DP(29) | DP(30) | DP(31) |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| H'0040*(N-1) | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | DP(M-4) | DP(M-3) | DP(M-2) | DP(M-1) |
| H'0040*N | EndC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC |
| H'0040*(N+1) | Reserved area (cannot be used) | | | | | | | | | | | | | | | |

In case of M=16×n+1 (n=0, 1, ···, N-1)

| Offset | +H'00 | +H'04 | +H'08 | +H'0C | +H'10 | +H'14 | +H'18 | +H'1C | +H'20 | +H'24 | +H'28 | +H'2C | +H'30 | +H'34 | +H'38 | +H'3C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H'0000 | DP(0) | DP(1) | DP(2) | DP(3) | DP(4) | DP(5) | DP(6) | DP(7) | DP(8) | DP(9) | DP(10) | DP(11) | DP(12) | DP(13) | DP(14) | DP(15) |
| H'0040 | DP(16) | DP(17) | DP(18) | DP(19) | DP(20) | DP(21) | DP(22) | DP(23) | DP(24) | DP(25) | DP(26) | DP(27) | DP(28) | DP(29) | DP(30) | DP(31) |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| H'0040*(N-1) | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | DP(M-4) | DP(M-3) | DP(M-2) |
| H'0040*N | DP(M-1) | EndC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC |
| H'0040*(N+1) | Reserved area (cannot be used) | | | | | | | | | | | | | | | |

Figure 54.52 Memory image on defect pixel information (In case of M=16×n and M=16×n+1)

In case of M=16×n+2 (n=0, 1, ···, N-1)

| Offset | +H'00 | +H'04 | +H'08 | +H'0C | +H'10 | +H'14 | +H'18 | +H'1C | +H'20 | +H'24 | +H'28 | +H'2C | +H'30 | +H'34 | +H'38 | +H'3C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H'0000 | DP(0) | DP(1) | DP(2) | DP(3) | DP(4) | DP(5) | DP(6) | DP(7) | DP(8) | DP(9) | DP(10) | DP(11) | DP(12) | DP(13) | DP(14) | DP(15) |
| H'0040 | DP(16) | DP(17) | DP(18) | DP(19) | DP(20) | DP(21) | DP(22) | DP(23) | DP(24) | DP(25) | DP(26) | DP(27) | DP(28) | DP(29) | DP(30) | DP(31) |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| H'0040*(N-1) | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | DP(M-4) | DP(M-3) |
| H'0040*N | DP(M-2) | DP(M-1) | EndC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC |
| H'0040*(N+1) | Reserved area (cannot be used) | | | | | | | | | | | | | | | |

In case of M=16×n+3 (n=0, 1, ···, N-1)

| Offset | +H'00 | +H'04 | +H'08 | +H'0C | +H'10 | +H'14 | +H'18 | +H'1C | +H'20 | +H'24 | +H'28 | +H'2C | +H'30 | +H'34 | +H'38 | +H'3C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H'0000 | DP(0) | DP(1) | DP(2) | DP(3) | DP(4) | DP(5) | DP(6) | DP(7) | DP(8) | DP(9) | DP(10) | DP(11) | DP(12) | DP(13) | DP(14) | DP(15) |
| H'0040 | DP(16) | DP(17) | DP(18) | DP(19) | DP(20) | DP(21) | DP(22) | DP(23) | DP(24) | DP(25) | DP(26) | DP(27) | DP(28) | DP(29) | DP(30) | DP(31) |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| H'0040*(N-1) | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | ··· | DP(M-4) |
| H'0040*N | DP(M-3) | DP(M-2) | DP(M-1) | EndC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC | PadC |
| H'0040*(N+1) | Reserved area (cannot be used) | | | | | | | | | | | | | | | |

Figure 54.53 Memory image on defect pixel information (In case of M=16×n+2 and M=16×n+3)

In case of M=16×n+14 (n=0, 1, ···, N-1)

| Offset | +H' 00 | +H' 04 | +H' 08 | +H' 0C | +H' 10 | +H' 14 | +H' 18 | +H' 1C | +H' 20 | +H' 24 | +H' 28 | +H' 2C | +H' 30 | +H' 34 | +H' 38 | +H' 3C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H' 0000 | DP(0) | DP(1) | DP(2) | DP(3) | DP(4) | DP(5) | DP(6) | DP(7) | DP(8) | DP(9) | DP(10) | DP(11) | DP(12) | DP(13) | DP(14) | DP(15) |
| H' 0040 | DP(16) | DP(17) | DP(18) | DP(19) | DP(20) | DP(21) | DP(22) | DP(23) | DP(24) | DP(25) | DP(26) | DP(27) | DP(28) | DP(29) | DP(30) | DP(31) |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| H' 0040*(N-1) | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| H' 0040*N | DP(M-14) | DP(M-13) | DP(M-12) | DP(M-11) | DP(M-10) | DP(M-9) | DP(M-8) | DP(M-7) | DP(M-6) | DP(M-5) | DP(M-4) | DP(M-3) | DP(M-2) | DP(M-1) | EndC | PadC |
| H' 0040*(N+1) | Reserved area (cannot be used) | | | | | | | | | | | | | | | |

In case of M=16×n+15 (n=0, 1, ···, N-1)

| Offset | +H' 00 | +H' 04 | +H' 08 | +H' 0C | +H' 10 | +H' 14 | +H' 18 | +H' 1C | +H' 20 | +H' 24 | +H' 28 | +H' 2C | +H' 30 | +H' 34 | +H' 38 | +H' 3C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H' 0000 | DP(0) | DP(1) | DP(2) | DP(3) | DP(4) | DP(5) | DP(6) | DP(7) | DP(8) | DP(9) | DP(10) | DP(11) | DP(12) | DP(13) | DP(14) | DP(15) |
| H' 0040 | DP(16) | DP(17) | DP(18) | DP(19) | DP(20) | DP(21) | DP(22) | DP(23) | DP(24) | DP(25) | DP(26) | DP(27) | DP(28) | DP(29) | DP(30) | DP(31) |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| H' 0040*(N-1) | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| H' 0040*N | DP(M-15) | DP(M-14) | DP(M-13) | DP(M-12) | DP(M-11) | DP(M-10) | DP(M-9) | DP(M-8) | DP(M-7) | DP(M-6) | DP(M-5) | DP(M-4) | DP(M-3) | DP(M-2) | DP(M-1) | EndC |
| H' 0040*(N+1) | Reserved area (cannot be used) | | | | | | | | | | | | | | | |

**Figure 54.54 Memory image on defect pixel information** (In case of M=16×n+14 and M=16×n+15)

### 3. Upper limitation of length of defect pixel information.

- By setting the ISPCORE_DMA_DPCC[j]_PIXEL_COUNT.DPCC_PIXEL_COUNT_MAX register (hereafter it denoted as DPCC_PIXEL_COUNT_MAX) , it is possible to set the upper limit of length of defect pixel information to be written to the memory.
  - The maximum value of that can be set in this register is 16384
- With this setting, even if the length of defect pixel information (number of defect pixel coordinate) output from the ISP IP exceeds the DPCC_PIXEL_COUNT_MAX register setting, no more than the DPCC_PIXEL_COUNT_MAX setting value will be written to the memory.
- If the length of defect pixel information output from the ISP IP exceeds the DPCC_PIXEL_COUNT_MAX register, then interrupt status ISPCORE_INT_STATUS.DPCC_PIXEL_COUNT_OVER_DPCCn (n=1, 2) is asserted.
  - Setting of ISPCORE_DMA_DPCC[0]_PIXEL_COUNT.DPCC_PIXEL_COUNT_MAX corresponds to the ISPCORE_INT_STATUS.DPCC_PIXEL_COUNT_OVER_DPCC1
  - Setting of ISPCORE_DMA_DPCC[1]_PIXEL_COUNT.DPCC_PIXEL_COUNT_MAX corresponds to the ISPCORE_INT_STATUS.DPCC_PIXEL_COUNT_OVER_DPCC2
- By writing 0 to the ISPCORE_DMA_DPCC[0]_STATUS.DPCC_COUNT_OVER register, then the status bit of interrupt status register ISPCORE_INT_STATUS.DPCC_PIXEL_COUNT_OVER_DPCC1 can be cleared to 0.
- By writing 0 to the ISPCORE_DMA_DPCC[1]_STATUS.DPCC_COUNT_OVER register, then the status bit of interrupt status register ISPCORE_INT_STATUS.DPCC_PIXEL_COUNT_OVER_DPCC2 can be cleared to 0.
- By reading ISPCORE_DMA_DPCC[j]_PIXEL_COUNT.DPCC_PIXEL_COUNT register, it is possible to observe the actual length of defect pixel information output from the ISP IP. The read value of this register includes the number of one word of the end code (H'FFFFFFFF). This register is not affected by the upper limit setting by the

DPCC_PIXEL_COUNT_MAX register. Therefore, the true length of defect pixel information output from the ISP IP is observed from this register.

### 54.4.12 Initialization of SRAM EDC error flag

The SRAM used in ISP has an error detection function by using EDC (Error Detection Code). After powering (*1) on the ISP and before starting the ISP processing, undefined values are stored in these SRAM. Therefore, an EDC error of the SRAM (hereafter, SRAM EDC error) is always detected when the first frame of the ISP is processed.

To avoid these phenomena, it is necessary to initialize the SRAM and clear notification flag of the SRAM EDC error which are defined in ISPERRn_STATUS register, before starting the original processing of the ISP. The procedure is shown below.

- After power on of the ISP, set a larger image width value than the image width of target image for processing by the ISP, and then execute one frame processing.
  - For example, if the image width of the target image for the original ISP processing is W [pixel], set a value of W+2 [pixel] or higher as the image width.
  - At this time, setting of the image height is arbitrary, but at least 40 lines or more and with minimal impact on processing time is recommended. Also, the content of the input image at this time is arbitrary. This procedure corresponds to the processing with a dummy image input.
- After one frame processing of the ISP at this setting is finished, refer to the ISPERRn_STATUS register and check the bit that is set to 1 in this register. Regarding detail of ISPERRn_STATUS register, see the 54.4.7.
- By writing 1 to the target bit of the ISPERRn_CLEAR register, clear the target bit in the ISPERRn_STAUS register. Regarding the clearing method, see the 54.4.7.
  - When clearing the error flag, all bits in the ISPERRn_STATUS register can be cleared at once by writing H'FFFFFFFF to ISPERRn_CLEAR register.
- Thereafter, the original processing by the ISP is executed.

(*1) This initialization needs to after the module stop (MSTP) ON/OFF and the real time test (RTT).

## 54.5 Usage Notes

When using the ISP core, note the following points.

### 54.5.1 Notes of register access

#### 54.5.1.1 Access size of register

Access size of each register should be 32 bit. Access size other than 32 bit is not supported

#### 54.5.1.2 Unassigned offset address

Write access to the un-assigned offset address in register address area is forbidden. Do not write to address which offset is not assigned.

#### 54.5.1.3 Write access of reserved bits or un-used bits

Do not write any different value from its initial value to reserved bits or un-used bits. Note that there are some writable bits (R/W) in reserved bits or un-used bits in each register.

### 54.5.2 Register access from host CPU during running ISP

#### 54.5.2.1 Basic principle of availability of register access

The availability of register access from host CPU while the ISP is running depends on the address region to be accessed and whether Config DMA is used or not, as specified in the Table 54.55 and Table 54.56.

However, all of register settings (configuration of ISP) should be completed before ISP running. Write access during ISP running is prohibited regardless of the address region or whether Config DMA is used or not. During ISP running, do not change any settings.

Table 54.55 Availability of register access during ISP running (Config DMA is not used)

| | Address region (ISP ch0) | Address region (ISP ch1) | Read | Write |
|---|---|---|---|---|
| ISP Interface | H'FED00000 ~ H'FED0FFFF | H'FED20000 ~ H'FED2FFFF | Available | Forbidden (*2)(*3) |
| ISP Core | H'FEC00000 ~ H'FEC7FFFF | H'FEE00000 ~ H'FEE7FFFF | Available | Forbidden(*2) |
| | H'FEC80000 ~ H'FEC8FFFF | H'FEE80000 ~ H'FEE8FFFF | Available | Forbidden(*2) |

Table 54.56 Availability of register access during ISP running (Config DMA is used)

| | Address region (ISP ch0) | Address region (ISP ch1) | Read | Write |
|---|---|---|---|---|
| ISP Interface | H'FED00000 ~ H'FED0FFFF | H'FED20000 ~ H'FED2FFFF | Available | Forbidden (*2)(*3) |
| ISP Core | H'FEC00000 ~ H'FEC7FFFF | H'FEE00000 ~ H'FEE7FFFF | NotAvailable (*1) | Forbidden (*1)(*2) |
| | H'FEC80000 ~ H'FEC8FFFF | H'FEE80000 ~ H'FEE8FFFF | NotAvailable (*1) | Forbidden (*1)(*2) |

(*1) When Config DMA is used register access is not available. However, in order to access this area during ISP running, following procedure is needed. Set the ISP_STREAMER_CONFIG_DMA_CONTROL.ENABLE0 register to 0 after receiving an interrupt for the notification of Config DMA transfer completion from VSPX. In addition, set the ISP_STREAMER_CONFIG_DMA_CONTROL.CONFIG_DMA_ENABLE register to 1 before restarting the Config DMA transfer when resuming ISP processing, such as when starting the processing of the next frame. However when using this procedure, do not use the display list link function in VSPX. In this case, switching between configuration data transfer and image data transfer should be done manually.

(*2) This "Forbidden" means that it is prohibited to overwrite the setting of ISP during ISP is running. However, regarding registers related to interrupt functions (e.g. interrupt clear register, interrupt mask register, etc.) and registers such as those with shadow register structure (see the IP manual for details), write access is available.

(*3) However, write access is available regarding registers shown in 54.5.2.2.

#### 54.5.2.2 Configuration of Streaming Bridge during running capture operation

Regarding registers listed in the following, the values themselves can be written even during the Channel Selector is running (after writing 0xFFFF to START_ISP register, and while the Channel Selector routes pixel streams from CSI).

However, rewriting is not allowed while configuration data transfer and image data transfer to the ISP core are in operation. It is basically recommended to complete the setting of these registers before starting and to hold its value.

- ISP_STREAMER_MODE
- ISP_STREAMER_V_BLANK
- ISP_STREAMER_H_BLANK
- ISP_STREAMER_CONFIG_DMA_CONTROL
- ISP_STREAMER_CONFIG_DMA_CONTROL1
- ISP_STREAMER_CONFIG_DMA_CONTROL2

### 54.5.3 Waiting period from clear timing of module reset to the register access to ISP.

At the timing of reset clear, ISP starts internal initialization processing. User shall wait for completion of this process, and see 54.4.3.1.12 or 54.4.3.2.16 for the actual value of it.

### 54.5.4 Synchronized system with IMR

#### 54.5.4.1 Boot up sequence between ISP and IMR

Boot up sequence should keep the first is IMR and the second is ISP. You should finish boot up sequence and configuration of IMR before starting boot up operation of ISP.

#### 54.5.4.2 Power control of A3IR domain

You should turn on the power of A3IR domain when inter module synchronization with IMR is used. It is because that wake up signals are connected to IMR via A3IR domain. Refer to the HW manual of the System Controller (SYSC) for more information.

#### 54.5.4.3 Module standby mode

When forcibly stopping the clock of the related modules to synchronization, a critical inconsistency may occurs and it may bring hung-up. Take care not to bring inconsistency when executing module stop operation.

#### 54.5.4.4 Configuration period at frame boundary timing

IMR should be configured at frame boundary timing of every frame. You should keep enough period for this configuration. You should restrict content of configuration so that this configuration has been completed during vertical blanking period. Mainly, configuration at this time is consist of address register setting.

### 54.5.5 Restrictions of maximum pixel throughput

#### 54.5.5.1 Restrictions in MIPI CSI-2 input

Restriction in MIPI CSI-2 input case is based on the input pixel rate. It is defined as the value obtained by the division number of active pixels per one horizontal line (1H period) by total time per 1H period including horizontal blanking for each inputs. It depends on parameters defined per 1 line for each 4 input ports. Actual parameters are image width [pixel], number of bit per one pixel [bit/pixel] and horizontal blanking period [ns] as differentiated parameters for each inputs, and CSI bit rate, number of lanes and number of padding cycles at the end of long packet to push CSI FIFO (PADDING_NUM register) as common parameters. Considering these parameters, input pixel rate is calculated by the following equation.

$$Input_pixel_rate\ [Mpixel/s] = \frac{Number\ of\ pixel\ in\ 1H\ peiord}{Time\ in\ 1H\ period}$$
$$= \frac{\sum_{k=1}^{4}\left\{WIDTH(k)\ +\ \frac{32 \times A}{bpp(k)}\ \right\}}{\sum_{k=1}^{4}\left\{\frac{WIDTH(k)\ \times\ bpp(k))}{CSI_rate\ \times\ lane} + h_blank(k)\ \right\}} \ldots (1)$$

Take care that the input pixel rate should be less than upper limitation, which is shown in the 'Maximum pixel throughput' in 54.1.1. If input pixel rate exceeds this limitation, then pixel stream to input the ISP core is corrupted because CSI FIFO inside Channel Selector block (see the Figure 54.5) overflows. Overflow of CSI FIFO is notified as FIFO_FULL_CSI_INPUT interrupt. See 1.4.7 for more information.

Definition of each variable in equation (1) is shown in Table 54.27. When developing imaging application/system with the ISP, user should confirm that the input pixel rate for each use-case does not exceed the upper limit.

**Table 54.27 Definition of variables in equation (1)**

| Variables | Definition | Unit |
|---|---|---|
| *N* | Number of active input ports (*N*=1,2,3,4) | |
| *WIDTH(k)* | Number of pixels per one line (image width) of the k-th input (*k*=1,⋯*N*) | [pixel] |
| *bpp(k)* | Number of bits per one pixel of the k-th input (*k*=1,⋯*N*).<br>When pixel format of input port 1 is RAW20 format, then *bpp(k)* = 20 | [bit/pixel] |
| *H_blank (k)* | Horizontal blanking period of the k-th input image (*k*=1,⋯*N*)<br>It is obtained by specification of connected image sensors. | [s] |
| *A* | Value of PADDING_NUM register. Number of padding cycles to push CSI FIFO at the end of every long packet. | |
| *CSI_rate* | Bit rate of CSI interface per one lane | [bit/s] |
| *Lane* | Number of active lane of CSI interface. | |

Examples of calculation are shown in Table 54.28. Notes that if calculated input pixel rate exceeds the upper limit, then its configuration is invalid.

**Table 54.28 Example of input pixel rate**

| Parameters | Variables | Example1 | Example2 | Example 3 |
|---|---|---|---|---|
| Number of active input | *N* | 4 | 4 | 4 |
| Image width [pixel] | *WIDTH(k)* | 1920 (common) | 1920 (common) | 1920 (common) |
| Number of bits per pixel [bit/pixel] | *bpp(1)* | 20 | 20 | 20 |
| | *bpp(2)* | 20 | 16 | 16 |
| | *bpp(3)* | 20 | 16 | 16 |
| | *bpp(4)* | 20 | 12 | 12 |
| Horizontal Blanking [ns] | *H_blank(1)* | 500 | 500 | 500 |
| | *H_blank(2)* | 500 | 2000 | 500 |
| | *H_blank(3)* | 500 | 2000 | 500 |
| | *H_blank(4)* | 500 | 2500 | 500 |
| PADDING_NUM register | *A* | 256 | 256 | 256 |
| Bit rate of CSI interface [Gbit/s] | *CSI_rate* | 1.5 | 1.5 | 1.5 |
| Number of lane | *Lane* | 4 | 4 | 4 |
| Input pixel rate [Mpixel/s] | | 337.6 | 356.5 | 435.8 |

#### 54.5.5.2 Impact on memory bandwidth

ISP performance is affected by the memory bus bandwidth of SoC. The available memory bandwidth for ISP can be configured by QoS setting. Therefore, note that pixel throughput (pixel rate) of ISP may vary depending on the QoS setting.

### 54.5.6 Pixel format of CSI-2 long packet

Figure 54.57 ~ Figure 54.63 show packed pixel format of incoming CSI-2 long packet for each pixel format. Pixel reconstructor converts it to the “1 pixel per clock” format so that ISP core can receive. In MIPI CSI-2 V1.1 standard, RAW16, RAW20 and RAW24 are not defined. Therefore, these formats are treated as combination of two sequential received pixels.

**1. Pixel reconstruction of RAW16 format**

Pixel reconstructor processes RAW16 pixel as combination of two sequential received RAW8 pixels

The 1st RAW8 pixel corresponds to the upper half of RAW16 pixel (A[15:8] or B[15:8]).

The 2nd RAW8 pixel corresponds to the lower half of RAW16 pixel (A[7:0] or B[7:0]) .

**2. Pixel reconstruction of RAW20 format**

Pixel reconstructor processes RAW20 pixel as combination of two sequential received RAW10 pixels

The 1st RAW10 pixel corresponds to the upper half of RAW20 pixel (A[19:10] or B[19:10]).

The 2nd RAW10 pixel corresponds to the lower half of RAW16 pixel (A[9:0] or B[9:0] ).

**3. Pixel reconstruction of RAW24 format**

Pixel reconstructor processes RAW24 pixel as combination of two sequential received RAW12 pixels

The 1st RAW12 pixel corresponds to the upper half of RAW24 pixel (A[23:12] or B[23:12]).

The 2nd RAW12 pixel corresponds to the lower half of RAW16 pixel (A[11:0] or B[11:0]).

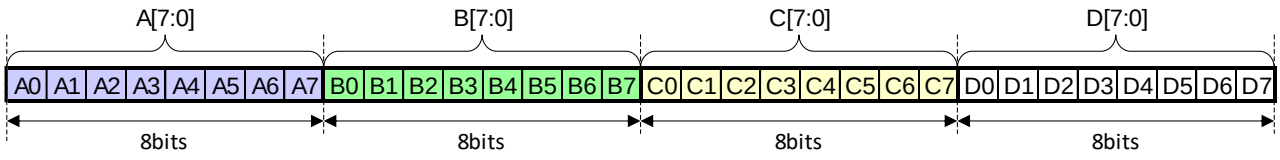



Figure 54.57 Packed pixel format of RAW8

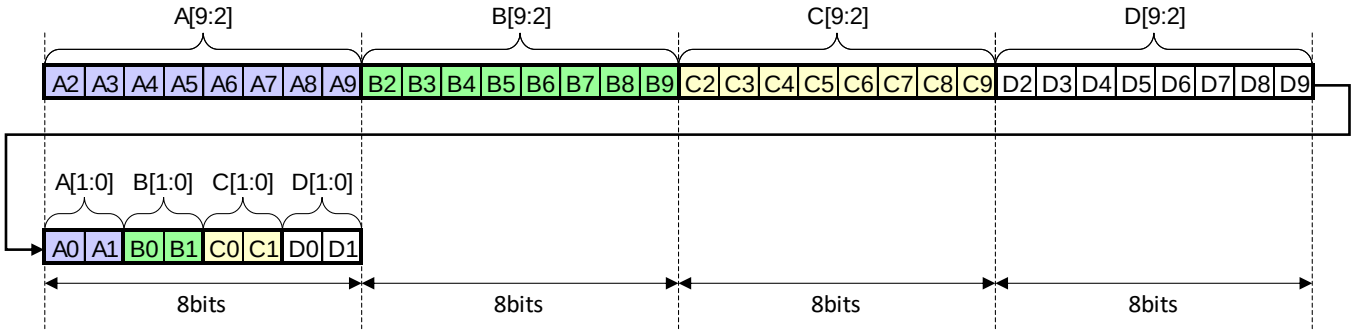



Figure 54.58 Packed pixel format of RAW10

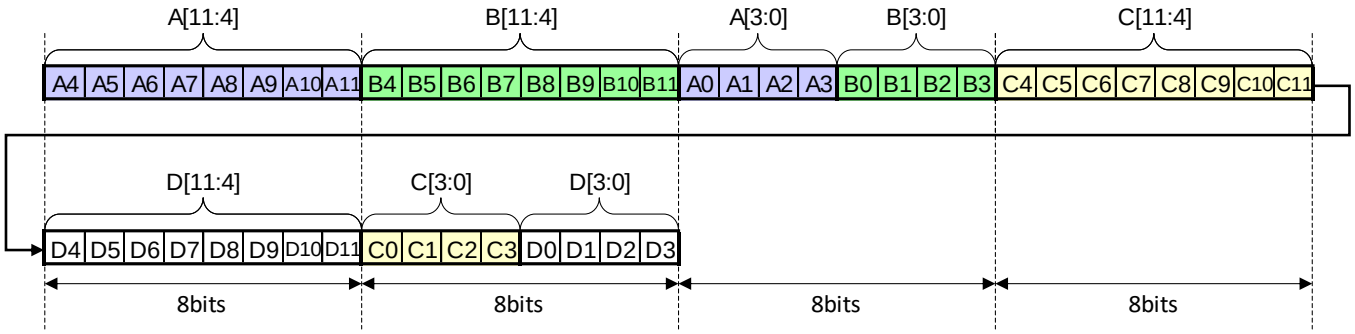



Figure 54.59 Packed pixel format of RAW12

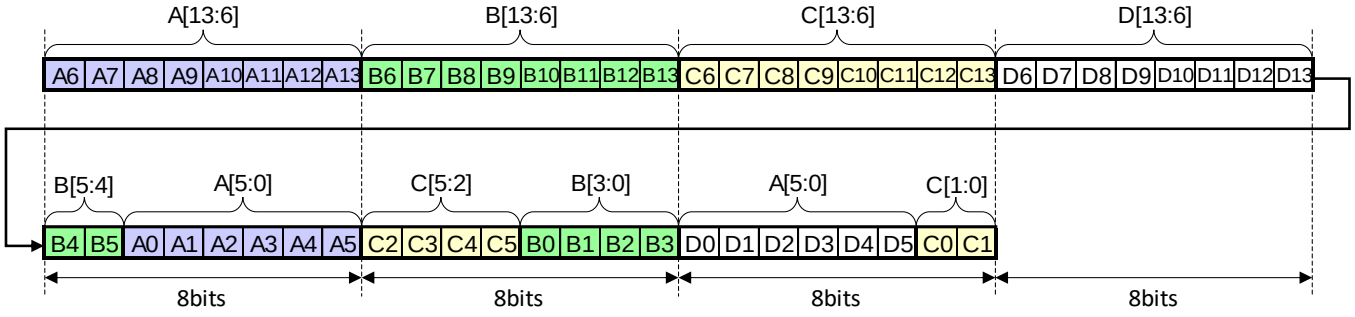



Figure 54.60 Packed pixel format of RAW14

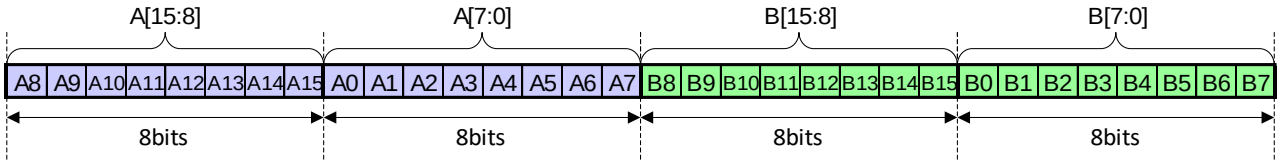



**Figure 54.61 Packed pixel format of RAW16**

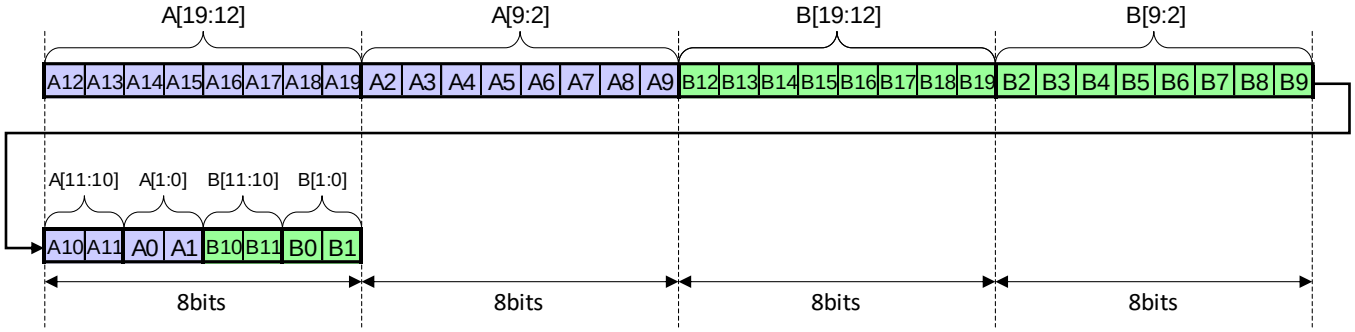



**Figure 54.62 Packed pixel format of RAW20**

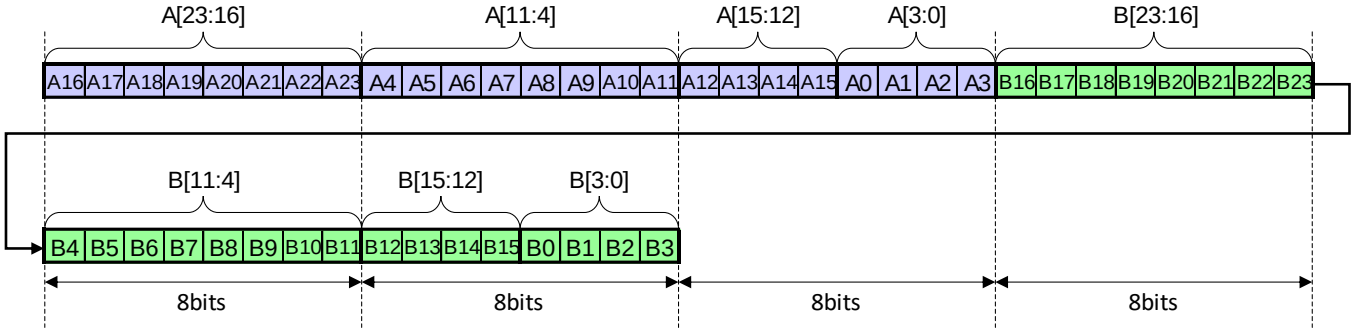



**Figure 54.63 Packed pixel format of RAW24**

### 54.5.7 Notes resuming operation after Runtime Test

After the Runtime Test is executed following (i) and (ii) must be applied
(i) Explicitly module resetting by using CPG register
(ii) After (i), 5000 [cycle] waiting

### 54.5.8 Notes related to RAW Scaler

- When using RAW scaler, RAW_SCALER.IMAGE_WIDTH register should be set a value with adding +1 from image width of input image
- RAW scaler is available at single exposure only. It does not support multiple exposure use-case.

### 54.5.9 Notes related to image width

When RAW scaling is disabled, image width of incoming image from CSI should always be multiple of 2. Only even value is available (odd value is forbidden)

When RAW scaling is enabled, image width of incoming image from CSI should always be multiple of 4 in order to keep even number of image width after scaled to 1/2 at RAW scaler.

### 54.5.10 Notes related to setting of Output DMA (Image Output DMA, Histogram Output DMA and Defect Pixel Information DMA)

Image output DMA does not have any mechanism to avoid 4k boundary address crossing. 4k boundary address crossing shall be avoided by setting of base address and stride registers.

Setting value of base address register of the output image (BASE_ADDRESs_COMPc (c=0,1,2,3)) must be a multiple of 256.

Setting value of STRIDE_COMPc (c=0,1,2,3) must be a multiple of 256. Since the unit of the STRIDE_COMPc (c=0,1,2,3) is 'byte', so only multiples of 256 bytes can be set for image stride.

Supported AXI burst length is 8 beat only. Therefore, setting value of AXI_CONFIG.BURST_LENGTH_MINUS1 must be set to 7.

### 54.5.11 Notes of additional memory area of flush out (Image Output DMA)

When using flush out setting (54.4.11.11), an additional memory area is required to store the pixels to be output by flush out operation to the memory area for rectangle image data. Pixel output by flush out operation is called as "padding pixel". This additional area is required for each component of the output image data (rectangle data) and must be a predetermined length starting from the next address of the final data of rectangle data (rightmost of bottom edge) .

Length of additional area *L(c)* for each component *c* is calculated by following formula.

$$L(c) = ceiling(BPP(c) \times WIDTH,\ 256) - BPP(c) \times WIDTH + BPP(c) \times FLUSH[byte]$$

$$ceiling(x, base) = \left\lceil \frac{x}{base} \right\rceil \times base$$

*BPP(c)* is byte per pixel of component *c* (In RGB, *c=0,1,2*. In YUV422, *c=0,1*).

WIDTH is number of pixels per one line, i.e. image width (horizontal length) of ISP output image.

*FLUSH* is setting value of ISPCORE_DMA_IMAGE[k]_FLUSH_OUT.PADDING_PIXEL_EOF register (*k*=0, 1). The value of *FLUSH* is always a multiple of 256.

Figure 54.64 shows a memory image of padding area.

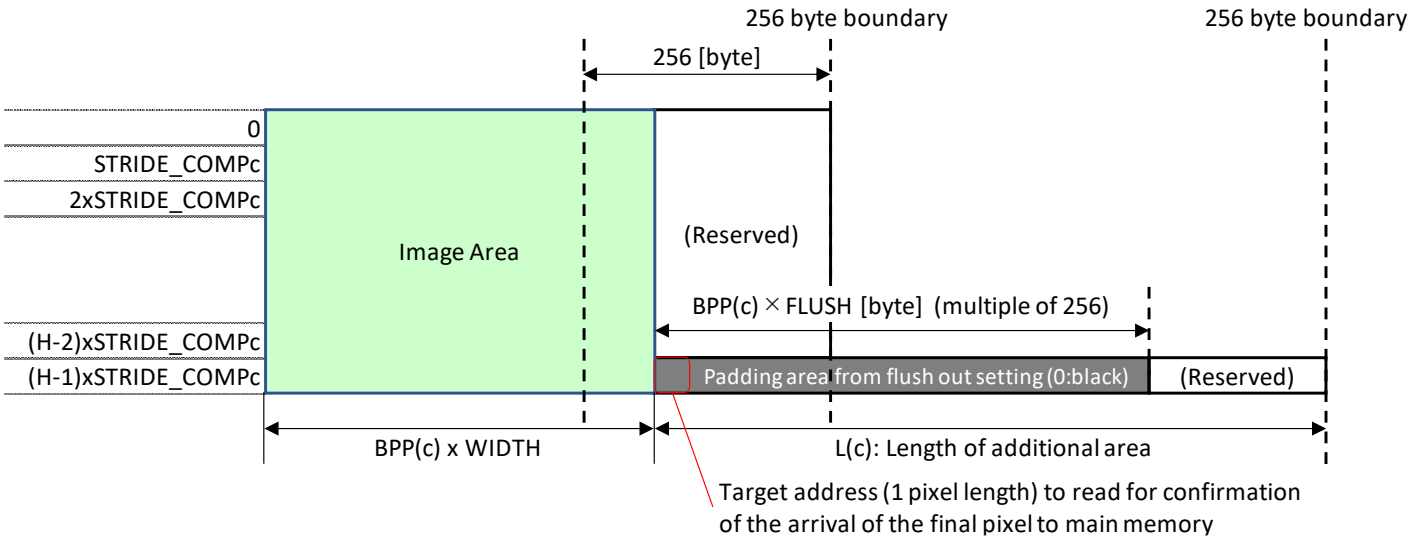



Figure 54.64 Memory Image of padding pixel

### 54.5.12 Notes on judgement of completion of ISP processing.

In order to accurately detect the completion of ISP processing after asserting the end-of-frame interrupt from ISP, flush out setting (54.4.11.11) is needed.

By flush out setting, remained pixels in DMA pipeline is pushed out to the main memory by padding pixel generated in DMA. Therefore, it is necessary to wait until the final pixel in one frame (the rightmost pixel of the bottom edge line in one frame) arrives in the main memory.

Pixel value of padding area is always 0 (black) (see Figure 54.64). Read the next address area to the final pixel (the area where padding pixel is stored), and confirm that the read value is 0. However, this address area is filled by 0 at every frame, so it is necessary to write explicitly non-zero value to this address area before starting one frame processing of ISP. Therefore, process flow of this sequence is shown in the following.

1. Setting non-zero value to the next address area to the final pixel.
2. Start to run ISP (one frame processing) with flush out setting.
3. Waiting end-of-frame interrupt from ISP
4. Read the next address area to the final pixel
5. If read value is 0 (black), then finish the current frame. If read value is not 0 return to 4.

Actual address to set checking value (non-zero value) and to read check is derived from rectangle size (width and height) of ISP output and stride according to Figure 54.64.

### 54.5.13 Notes on Module Reset/Module Stop

Before applying Module Reset/Module Stop to the ISP, execute the procedure described in 54.5.12. Module Reset/ Module Stop shall be asserted after confirming that the final pixel of the ISP output image has been written to the main memory. Assert of Module Reset/Module Stop without following the procedure described in 54.5.12 is prohibited.

### 54.5.14 Notes on using Channel Selector with Sync Interval Checker in VIN or Pixel/Metadata Counter

When using Sync Interval Checker in VIN or Pixel/Metadata Counter to execute capturing operation while counting the number of pixels in one frame, configuration of Horizontal clipping described in 54.4.2.16 is required.

This is because the Pixel Reconstructor processing generates padding pixels at the end of one line, which are transferred to VIN following the active pixel data. This padding data is generated in the process of unpacking the packed pixel data at the rightmost area in one frame according to the MIPI CSI-2 standard, and its amount depends on the pixel format. This phenomenon does not affect the capture operation of the active pixel area, but when observing the image width, it may make a wrong judgment due to the padding pixel.

By using the Horizontal clipping function, it is possible to prevent padding pixels from being transferred. Set the ISPCS_H_CLIP_DT_CODE[h]_CH[d].PREPENDED_PIXEL register (hereafter, denoted as APPENDED_PIXEL) and ISPCS_H_CLIP_DT_CODE[h]_CH[d].APPENDED_PIXEL register (hereafter, denoted as APPENDED_PIXEL) according to the image width. For example, if the image width is 1920 pixels, setting PREPENDED_PIXEL=0 and APPENDED_PIXEL=1920 will transmit only 1920 pixel areas starting from the leftmost pixel. This prevents the transfer of padding pixels generated in the Pixel Reconstructor processing.

Therefore, when using Sync Interval Checker or Pixel/Metadata Counter together, activate the Horizontal clipping in order to pass only valid pixel areas. Note that in the setting examples listed in 54.4.2.2, Examples 1~5, the Horizontal clipping register are examples with default values and these settings do not activate Horizontal clipping operation.

### 54.5.15 Notes on HSYNC Rise Count Checker at Sync Interval Checker in VIN

Regarding the behavior of the filter function described in 54.4.2.1, the Channel Selector sets an invalid value (low) to the Data Enable signal (named as CLKENB at VIN) at the line or pixel blocked by the filter. In the other hand, the HSYNC signal is passed through even if the corresponding line or pixel are blocked. That is, the HSYNC signal is propagated to VIN regardless of whether it is blocked or passed through.

For example, as shown in Figure 54.65, this example shows that 5 lines (a), (b), (c), (d), and (e) are input to the filter. Line (a), (c) and (e) represent one line which are sent as a long packet belonging to VC0. Line (b) and (d) represent one line which are sent as a long packet belonging to VC1. And Figure 54.65 shows an example where only the lines corresponding to VC0 is passed through and transferred to VIN.

At this time, when blocking the line (b) and (d), the Channel Selector controls the CLKENB signal on lines (b) and (d) to the invalid side (low), but the HSYNC signal during the same period remains valid value (high).

This example shows the result of filter processing by the VC Filter, but the same applies to other filters. That is, even filters other than the VC Filter control the CLKENB signal, and the HSYNC signal is passed through regardless of the filter processing result.

Considering above mentioned behavior, when counting the rise edge of HSYNC using the Sync Interval Checker in VIN, the number of rise edge of HSYNC is counted at the timing of line (b) and (d) even if these lines are blocked by the filter.

Therefore, it is necessary to set the number of HSYNC rise edge count at the HSYNC Rise Time Measurement in VIN considering the number of long packets (*) transmitted from CSI during the observation period (active period of VSYNC).

(* The number of long packets is equal to the rise count of HSYNC)

Similarly, registers related to the HSYNC Toggle Cycle Measurement in VIN shall also be set based on the above specification. See the section 55. Video Input Module (VIN) for more information on these registers.

If 5 lines (a), (b), (c), (d), and (e) are input to CSI, as shown in Figure 54.65, the maximum number of rise edge of HSYNC should be set as 5, even if the image size (number of line) to be captured by VIN is 3 lines. In particular, when using multiple sensors, the maximum number of rise edge of HSYNC should be set to the total number of lines transmitted from each sensor within the observation period.

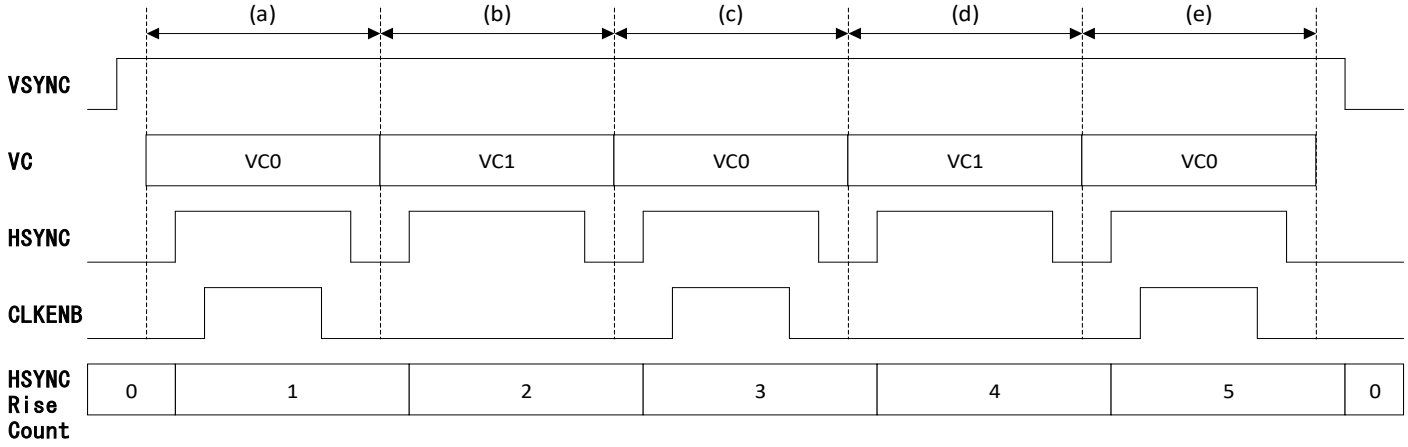



Figure 54.65 Timing example of VSYNC, HSYNC and CLKENB for VIN

### 54.5.16 Notes on using the HIST256/AWB/LSC function

**Notes on using the Histogram (HIST256) function:**

- Note that for continuous processing of frames, there is no direct support for on-the-fly switching between the different inputs. For input channel switching with intermediate reset, it is mandatory to execute the following programming sequence after frame processing and before reset:
  RPP_MAIN_PRE1_HIST256_CHANNEL_SEL = 0
  RPP_HDR_UPD = 1
  It is recommended to also set all other registers of the module RPP_MAIN_PRE1_HIST256 to their reset values and confirm with RPP_HDR_UPD = 1 before the reset.

**Notes on using the Auto White Balance Measurement function:**

- For correct operation it is mandatory that the module is disabled during configuration. The generic recommended way is to disable the module before configuration with RPP_MAIN_POST_AWB_MEAS_PROP::awb_mode = 0 and enable after configuration with RPP_MAIN_POST_AWB_MEAS_PROP::awb_mode = 2.
  Alternatively, a not-recommended disable/enable cycle after configuration works as well.

**Notes on using the Lens Shade Correction function:**

- Programming sequence for retriggering the prefetch: For disable:
  RPP_MAIN_PRE1_LSC_CTRL = 0x0
  RPP_HDR_UPD::regs_cfg_upd = 0x1
  This sets the internal status of LSC to disabled.

  Set each parameter configuration for LSC For enable:
  RPP_MAIN_PRE1_LSC_CTRL = 0x1
  RPP_HDR_UPD::regs_cfg_upd = 0x1
  This sets the internal status of LSC to enabled.

  In option to perform the same results without CPG Reset, use the following setting order:
  1. CPG Reset
  2. LSC Table(Previous)
  3. LSC Enable
  4. LSC Table(Current)

## 54.6 Safety Mechanism (SM)

### 54.6.1 EDC for SRAM in the ISP and Channel Selector

#### 1. Overview

EDC protection is applied to SRAMs in the ISP and Channel Selector.

#### 2. Hardware Description

Table 54.66 shows a SRAM list in the ISP core which are protected by EDC.
If an EDC error in these SRAM is detected, then ISPERR*n*_STATUS register (*n*=0,3) is updated.

Table 54.67 shows a SRAM list in the Channel Selector which are protected by EDC. If an EDC error in these SRAM is detected, then ISPERR*n*_STATUS register (*n*=1,2) is updated.

To use this function, the initialization of SRAM EDC error flag is required before operating the ISP. See the 54.4.12 for details.

Table 54.66 SRAM list in ISP Core

| SRAM Name | Block | Category | EDC protected |
|---|---|---|---|
| RPP_MAIN_PRE1_ACQ_FIFO_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for input FIFO | Yes |
| RPP_MAIN_PRE1_LSC_R_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE1_LSC_GR_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE1_LSC_B_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE1_LSC_GB_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE1_DPCC_MEM | rpp_main_pre1 | SRAM for local storage | Yes |
| RPP_MAIN_PRE1_DPCC_BPRAM_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE1_DPF_MEM | rpp_main_pre1 | SRAM for local storage | Yes |
| RPP_MAIN_PRE2_ACQ_FIFO_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for input FIFO | Yes |
| RPP_MAIN_PRE2_LSC_R_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE2_LSC_GR_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE2_LSC_B_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE2_LSC_GB_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE2_DPCC_MEM | rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE2_DPCC_BPRAM_MEM | rpp_main_pre1, rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_MAIN_PRE2_DPF_MEM | rpp_main_pre2 | SRAM for local storage | Yes |
| RPP_RMAP_PINTL_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_RMAP_PINTS_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_RMAP_PINTVS_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_RMAP_BLMFL_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_RMAP_BLMFS_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_RMAP_BLSFL_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_RMAP_BLSFS_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_RMAP_DELAYL_MEM | rpp_rmap | SRAM for local storage | Yes |

| | | | |
|---|---|---|---|
| RPP_RMAP_DELAYS_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_RMAP_DELAYVS_MEM | rpp_rmap | SRAM for local storage | Yes |
| RPP_MAIN_POST_FILT_MEM | rpp_main | SRAM for local storage | Yes |
| RPP_MAIN_POST_LTM0_MEM | rpp_main | SRAM for local storage | Yes |
| RPP_MAIN_POST_LTM1_MEM | rpp_main | SRAM for local storage | Yes |
| RPP_MAIN_POST_LTM2_MEM | rpp_main | SRAM for local storage | Yes |
| RPP_MAIN_RGBDENOISE_MEM | rpp_main | SRAM for local storage | Yes |
| RPP_MAIN_POST_SHRPCNR_MEM | rpp_main | SRAM for local storage | Yes |
| RPP_HDR_MV_OUT_RAM_MEM | mv_out | SRAM for local storage | Yes |
| RPP_OUT_CONV_420_MEM | rpp_out | SRAM for local storage | Yes |
| axviff_spram_awchmem_128w48b | AXI FIFO | SRAM for AXI FIFO | No |
| axviff_spram_awchmem_128w256b | AXI FIFO | SRAM for AXI FIFO | No |
| axviff_spram_awchmem_128w6b | AXI FIFO | SRAM for AXI FIFO | No |

Table 54.67 EDC protected SRAM list in Channel Selector

| SRAM Name | Block | Category | EDC protected |
|---|---|---|---|
| CSI FIFO SRAM | Pixel Reconstructor | SRAM for FIFO of Pixel Reconstructor | Yes |
| Line Memory SRAM | RAW Scaler | Line Memory of RAM Scaler | Yes |

**3. Control Register**

**(a) Control Register for EDC for SRAM in the ISP core.**

There are no control register which selects enable/disable of this SM. This SM is always enabled.
By using ISPERR*n*_ENABLE register (*n*=0, 3), enable/disable selection for each error notification to ECM can be controlled.
If ISPERR*n*_ENABLE.EN*k* is set to 1, then error is notified to ECM.
If ISPERR*n*_ENABLE.EN*k* is set to 0, then error is not notified to ECM
In both case, status bit in ISPERRn_STATUS*n*.ST*k* is updated when error is occurred.
*k* is derived from the tables shown in the 54.4.7.3 for bit assignment of each enable bit in ISPERR*n*_ENABLE register.

**(b) Control Register for EDC for SRAM in the Channel Selector.**

There are no control register which selects enable/disable of this SM. This SM is always enabled.
By using ISPERR*n*_ENABLE register (*n*=1,2), enable/disable selection for each error notification to ECM can be controlled.
If ISPERR*n*_ENABLE.EN*k* is set to 1, then error is notified to ECM.
If ISPERR*n*_ENABLE.EN*k* is set to 0, then error is not notified to ECM
In both case, status bit in ISPERRn_STATUS*n*.ST*k* is updated when error is occurred.
*k* is derived from the tables shown in the 54.4.7.3 for bit assignment of each enable bit in ISPERR*n*_ENABLE register.

### 54.6.2 EDC with ID Checker for Channel Selector Data Path

**1. Overview**

EDC protection with ID is applied to pixel data in the data path of channel selector in order to detect fault in the filter processing (VC filter, line count filter, meta data filter, Horizontal/Vertical clipping and de-interleaving filter).

**2. Hardware Description**

**(a) EDC check of filter chain in Channel Selector (ID='B00)**

At the input of each filter chain in the Channel Selector, an EDC code is calculated for the pixel value with a 2-bit ID code (2'B00) added. At the output of the final stage (de-interleaving filter) of the pipeline, EDC checking is performed. If

EDC error issued, then ISPERR1_STATUS. CHANNEL_SELECT_CHn_ERROR (n is an index of each channel) is asserted. See the 54.4.7.3 for bit assignment of each error status in ISPERR1_STATUS register.

**(b) Control Register**

There are no control register which selects enable/disable of this SM. This SM is always enabled.

By using ISPERR1_ENABLE register, enable/disable selection for each error notification to ECM can be controlled.

If ISPERR1_ENABLE.EN*k* (*k*=8~19) is set to 0, then error is not notified to the ECM.

If ISPERR1_ENABLE.EN*k* (*k*=8~19) is set to 1, then error is notified to the ECM.

In both case, status bit in ISPERR1_STATUS.ST*k* (*k*=8~19) is updated when error is occurred.

See the 54.4.7.3 for bit assignment of each enable bit in ISPERR1_ENABLE register.

### 54.6.3 ECC check of CSI header of long packet and CRC check of CSI long packet payload

**1. Overview**

ECC (error check code) is applied to incoming CSI header of long packet. Correction feature of ECC is not supported, and only VC 0 to 3 is supported for checking. ECC checker must be disabled in case that VC 4 to 15 is used.
ECC check is used only for D-PHY mode.

CRC (cyclic redundancy check ) is applied to incoming CSI long packet payload.

Check timing of CRC is at the rising edge of Hsync on the next line.

The CRC Check timing of last line of the frame is carried over until the rising edge of Hsync in the next frame.

CRC check is not performed at the rising edge of Hsync immediately after starting ISP (ISPSTART_[c]).

**2. Hardware Description**

Channel Selector calculates ECC and CRC of incoming CSI long packets and compares it with its expected value transferred with pixel data from CSI-2 link. If calculated ECC/CRC and its expected value is mismatch, then ISPERR1.STATUS.ST4 (CSI_ECC_ERROR) or ISPERR1_STATUS.ST5 (CSI_CRC_ERROR) is updated.

**3. Control Register**

There are no control register which selects enable/disable of this SM. This SM is always enabled.
By using ISPERR1_ENABLE register, enable/disable for each error notification to ECM can be controlled

If ISPERR1_ENABLE.EN*k* (*k*=4,5) is set to 0, then error is not notified to the ECM.

If ISPERR1_ENABLE.EN*k* (*k*=4,5) is set to 1, then error is notified to the ECM.

In both case, status bit in ISPERR1_STATUS.ST*k* (*k*=4,5) is updated when error is occurred.

See the 54.4.7.3 for bit assignment of each enable bit in ISPERR1_ENABLE register.

### 54.6.4 Hardware Redundancy for Pixel Reconstructor in Channel Selector

**1. Overview**

Hardware redundancy is applied to the Pixel reconstructor in the Channel selector to detect fault of pixel reconstruction logic of incoming packed pixel from MIPI CSI-2 interface.

**2. Hardware Description**

Pixel reconstructor receives input pixel as packed pixel format, and outputs pixel stream as 1 pixel per clock format (unpacked pixel). There are 2 instances of pixel reconstructor. Pair of output pixel from each pixel reconstructor are always compared.
If pixel value form both pixel reconstructor does not match each other, then error status ISPERR2_STATUS.ST27 (CSI_LOCK_STEP_ERROR) is asserted.

There is no control register. This function is always enabled. User can control ON/OFF of notification to ECM for each error status bit by using ISPERR2_ENABLE register.

Note: For details of ISPERR2_STATUS and ISPERR2_ENABLE, see the 54.4.7.1.

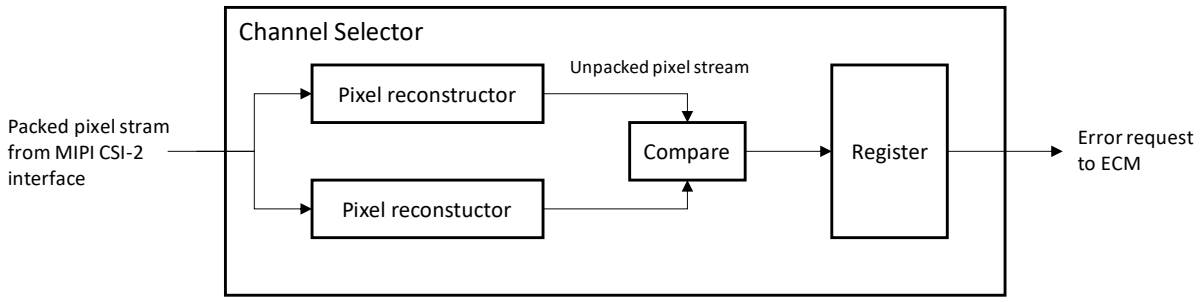



Figure 54.68 Block Diagram of Hardware redundancy for Pixel reconstructor

**3. Control Register**

There are no control register which selects enable/disable of the SM. This SM is always enabled.

By using ISPERR2_ENABLE register, enable/disable selection for each error notification to ECM can be controlled.
If ISPERR2_ENABLE.EN27 is set to 1, then error is not notified to ECM.
If ISPERR2_ENABLE.EN27 is set to 0, then error is not notified to ECM.
In both case, status bit in ISPERRn_STATUS2.ST27 is updated when error is occurred.

### 54.6.5 Pixel/Metadata Counter in Channel Selector

**1. Overview**

Pixel/Metadata counter is to detect fault of channel selector by counting number of pixels in one frame or in one line of incoming image. If number of pixels exceeds pre-defined maximum value or less than pre-defined minimum value, then error signal is asserted.

**2. Hardware Description**

If pixel count exceeds the value of ISP_PIXELCOUNT_MAX_CHn.MAX_PIXEL register (n=4~11), then error status ISPERR1_STATUS.CHANNEL_SELECT_CHn_PIXELCOUNT_MAX_ERROR (n=4~9) or ISPERR2_STATUS.CHANNEL_SELECT_CHn_PIXELCOUNT_MAX_ERROR (n=10,11) is asserted.

If pixel count is less than the value of ISP_PIXELCOUNT_MIN_CHn.MIN_PIXEL register (n=4~11), then error status ISPERR1_STATUS.CHANNEL_SELECT_CHn_PIXELCOUNT_MIN_ERROR (n=4~9) or ISPERR2_STATUS.CHANNEL_SELECT_CHn_PIXELCOUNT_MIN_ERROR (n=10,11) is asserted.

This function is always enabled. User can control ON/OFF of notification to ECM for each error status bit by using ISPERR1_ENABLE or ISPERR2_ENABLE register.

Note: For details of ISPERRn_STATUS and ISPERRn_ENABLE (n=1,2), see section the 54.4.7.3.

Principle of Pixel Count Checking

- Pixel/Metadata counter checker watches pixel stream which pass through each channel.
- It counts number of pixels in this pixel stream.
- Pixel count is cleared at horizontal blanking period or vertical blanking period.
- Pixel count does not depend on the length of horizontal/vertical blanking period.
- Some invalid cycles may exist in the pixel stream. VALID signal is low at the invalid cycle. Pixel count does not count up at the invalid cycles.

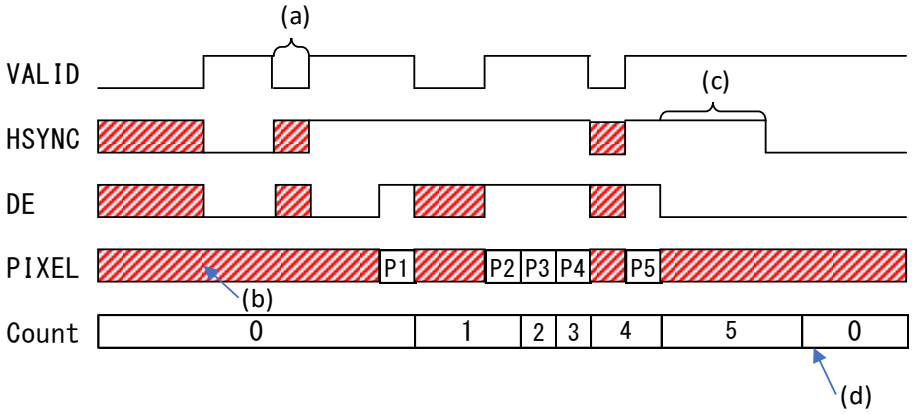



**Figure 54.69 Timing Chart of pixel count check**

Figure 54.69 is a timing chart of pixel counting. The important point is as below.

(a) When VALID==0, all corresponding signals (VSYNC, HSYNC, DE, PIXEL) are invalid
(b) When DE==0 or VALID==0, corresponding PIXEL signal is invalid. It means blanking period
(c) Period of HSYNC==1 && DE==0 is guaranteed by Pixel Reconstructor
(d) Pixel Count is cleared at the falling edge of HSYNC

When CLEAR_SYNC==1, HSYNC is not cleared at the falling edge of HSYNC, In this case, pixel count is cleared at the falling edge of VSYNC instead of falling edge of HSYNC.

### 54.6.6 Sync Interval Checker

#### 1. Overview

Sync Interval Checker detects “stuck at high” or “struck at low” by observing either the continuous high period or the continuous low period of the VSYNC, FIELD, HSYNC and DE (data enable) signal of the input image. This function is performed on the signal before the filtering process. However, this function observes the sticking period of the sync signal and it is not targeted to count the number of pixels in one frame. For counting the number of pixels, use the Pixel/Metadata Counter.

#### 2. Hardware Description

VSYNC, FIELD, HSYNC and DE (data enable) signals are checked with the Sync Interval Checker. The Sync Interval Checker probes these sync signals and monitors whether these signal falls into “stuck at high” or "stuck at low".

A block diagram is shown in Figure 54.70. Sync Interval Checker is executed by each virtual channel (VC). Error notification signal SYNC_ERROR_VCn (n=0,1,⋯15) is output individually for each VCs.

Regarding FIELD, VSYNC HSYNC and DE, the checker block denoted as “Sync Interval Checker (VCn) (n=0,1,⋯15)“ in Figure 54.70 probes these sync signals for each VC. That is, it observes the period that keeps taking a certain value. For FIELD and VSYNC, a signal exists for each VC. For DE, a signal masked by each VC is the target of observation.

Regarding HSYNC, the checker block observes both the period what keeps taking a certain value (HSYNC toggle interval check) and the number of rise edge of HSYNC (HSYNC rise count check) in one frame.

If the observed values of the above sync signals deviate from the range previously set in the register, then an error notification signal is asserted.

Error notification signal (Sync Error) is asserted when an error occurs in any of the 5 types of observations.

A point of note is, for FIELD, VSYNC and DE, the corresponding interval check block monitors them for each VC. On the other hand, the incoming HSYNC is a common signal between the VCs. Therefore, note that the observation result for the HSYNC signal is a common value for each VC.

The maximum and minimum value is defined as cycle in the ISP_SYNC_CHECK_*_MIN register and ISP_SYNC_CHECK_*_MAX register. The time for one cycle is calculated by the following formula.In this formula, *clock_frequency* denotes the frequency of S0D1_VIO $\phi$ clock.

*The time for one cycle* [us] = 1 / *clock_frequency* [MHz].

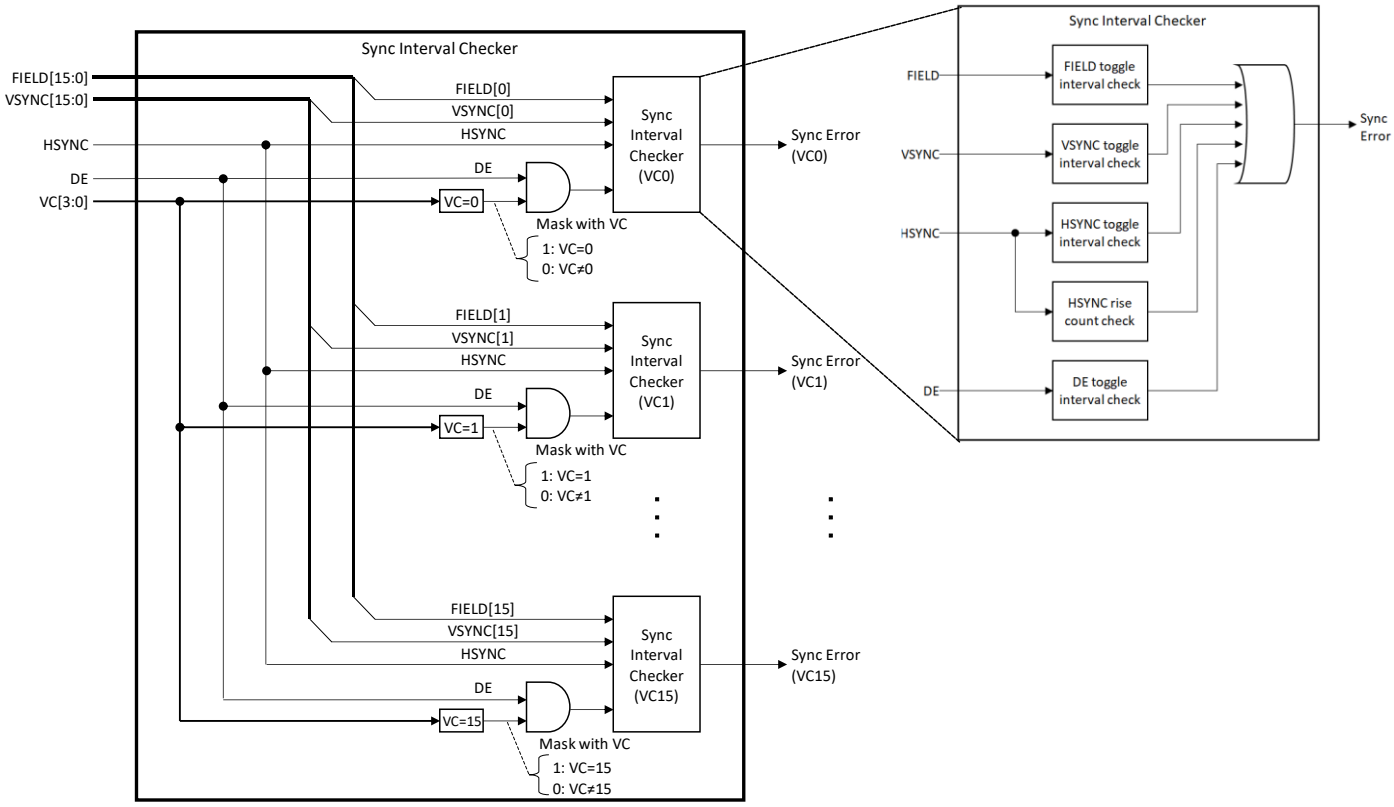



Figure 54.70 Block diagram of Sync Interval Checker

### 3. Control Register

#### (a) Enable/disable control.

This SM has enable/disable control register for each VC and type of sync signal. For VC that are not used, this SM shall be set to disabled.

Regarding FIELD toggle interval check, enable/disable is controlled by ISP_SYNC_CHECK_HSYNC_RISE_MAX. FIELD_CHECK_EN register.

Regarding VSYNC toggle interval check, enable/disable is controlled by ISP_SYNC_CHECK_HSYNC_RISE_MAX. VSYNC_CHECK_EN register.

Regarding HSYNC toggle interval check, enable/disable is controlled by ISP_SYNC_CHECK_HSYNC_RISE_MAX. HSYNC_TOGGLE_CHECK_EN register.

Regarding HSYNC rise count check, enable/disable is controlled by ISP_SYNC_CHECK_HSYNC_RISE_MAX. HSYNC_RISE_CHECK_EN register.

Regarding DE toggle interval check, enable/disable is controlled by ISP_SYNC_CHECK_HSYNC_RISE_MAX. DE_CHECK_EN register.

These enable/disable control register is 2-bit register. Setting value is shown as below (* is either FIELD, VSYNC, HSYNC_RISE, HSYNC_TOGGLE or DE).

- 2'B00 : Disable (default)
- 2'B01 : Checks only minimum value. Checks whether target period, i.e. period that keeps constant value, of observed sync signal gets less than the setting value of ISP_SYNC_CHECK_*_MIN register
- 2'B10 : Checks only maximum value. Checks whether target period, i.e. period that keeps constant value, of observed sync signal exceeds the setting value of ISP_SYNC_CHECK_*_MAX registers.
- 2'B11 : Checks both minimum value and maximum value simultaneously. Checks whether target period, i.e. period that keeps constant value, of observed signal is within the range from setting value of ISP_SYNC_CHECK_*_MIN register to the setting value of ISP_SYNC_CHECK_*_MAX register

Overwriting of these register during running is forbidden.

**(b) Notification of error**

When error is detected in the incoming image transferred on virtual channel 0 (VC0), then ISPERR2_STATUS.ST20 (SYNC_ERROR_VC0) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 1 (VC1), then ISPERR2_STATUS.ST21 (SYNC_ERROR_VC1) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 2 (VC2), then ISPERR2_STATUS.ST22 (SYNC_ERROR_VC2) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 3 (VC3), then ISPERR2_STATUS.ST23 (SYNC_ERROR_VC3) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 4 (VC4), then ISPERR2_STATUS.ST2 (SYNC_ERROR_VC4) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 5 (VC5), then ISPERR2_STATUS.ST3 (SYNC_ERROR_VC5) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 6 (VC6), then ISPERR2_STATUS.ST4 (SYNC_ERROR_VC6) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 7 (VC7), then ISPERR2_STATUS.ST5 (SYNC_ERROR_VC7) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 8 (VC8), then ISPERR2_STATUS.ST6 (SYNC_ERROR_VC8) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 9 (VC9), then ISPERR2_STATUS.ST7 (SYNC_ERROR_VC9) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 10 (VC10), then ISPERR2_STATUS.ST10 (SYNC_ERROR_VC10) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 11 (VC11), then ISPERR2_STATUS.ST11 (SYNC_ERROR_VC11) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 12 (VC12), then ISPERR2_STATUS.ST12 (SYNC_ERROR_VC12) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 13 (VC13), then ISPERR2_STATUS.ST13 (SYNC_ERROR_VC13) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 14 (VC14), then ISPERR2_STATUS.ST14 (SYNC_ERROR_VC14) is updated to 1.

When error is detected in the incoming image transferred on virtual channel 15 (VC15), then ISPERR2_STATUS.ST15 (SYNC_ERROR_VC15) is updated to 1.

By using ISPERR2_ENABLE register, enable/disable of notification to ECM for each error status bit can be controlled.

ISPERR2_ENABLE.EN*k* (*k*=2,3,4,5,6,7,10,11,12,13,14,15,20,21,22,23) is set 0, then error is not notified to ECM.

ISPERR2_ENABLE.EN*k* (*k*=2,3,4,5,6,7,10,11,12,13,14,15,20,21,22,23) is set 1, then error is notified to ECM.

Regardless of ISPERR2_ENABLE register value, status bit in ISPERR2_STATUS.ST*k* (*k*=2,3,4,5,6,7,10,11,12,13,14,15, 20,21,22,23) is updated when error is occurred.

### 54.6.7 Safety Mechanism of RAW Scaler in Channel Selector

#### 1. Overview

As a safety mechanism of RAW Scaler, EDC for SRAM and hardware redundancy is applied. EDC is used for line memory SRAM. Hardware redundancy is used for convolution logic to calculate pixels of converted RAW image.

#### 2. Hardware Description

RAW scaler has line memory SRAM. It is used for gathering surrounding pixels of the current pixel. This SRAM is protected by EDC.

As a safety mechanism of convolution operation, there are 2 instance of convolution circuits. Hardware redundancy is consisting of “Master convolution” and “Checker convolution”. Master convolution calculates pixels of converted RAW image. Checker convolution calculates pixels also by using same input (pixel and coefficients) as master convolution logics. Both results are compared mutually. If calculated value from between Master and Checker does not match each other, then error request (mismatch error) is asserted.

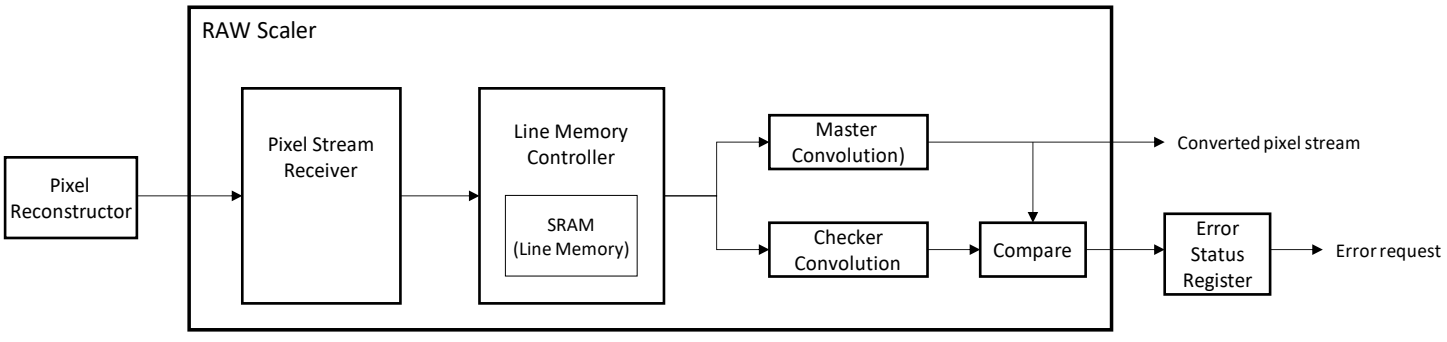



Figure 54.71 Block Diagram of Safety Mechanism of RAW Scaler

# 55. Video Input Module (VIN)

## 55.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

In addition, it includes safety mechanisms. For the detail, refer to functional safety documents.

The video input module (hereinafter abbreviated as VIN) is a video capture module that stores the YCbCr, RGB, RAW data from the MIPI Alliance CSI-2 interfaces in an external memory.

The module has up to eight video channels that independently controls capture of data into a capture area of up to 4096x4096 pixels.

VIN assumes that the input signal is an interlace video in which one frame consists of two fields. If the input signal is a progressive video, VIN generates a field identification signal by itself and captures it as a field image. Note that VIN captures in two frames at progressive input.

Input of data through the MIPI Alliance CSI-2 interfaces is possible.

Notes: 1. For detailed specifications of the MIPI CSI-2 interface, refer to section 53.
2. For detailed specifications of ISP/channel selector module, refer to section 54.
3. The maximum limits of the clipping size depend on each case of usage.
4. The number of video input channels for R-Car V4M is 16.

### 55.1.1 Features

#### (1) Video Channels 0 to 15

**Input Interface for Video Channels 0 to 15**

| Interface | Data Format | Data Type Number |
|---|---|---|
| MIPI CSI-2*1*2 | Embedded 8-bit non-Image Data | 0x12 |
| MIPI CSI-2*1*2 | YUV420 8-bit | 0x19, 0x1C(CSPS) |
| MIPI CSI-2*1*2 | YUV420 10-bit | 0x19, 0x1D(CSPS) |
| MIPI CSI-2*1*2 | YUV422 8-bit | 0x1E |
| MIPI CSI-2*1*2 | YUV422 10-bit | 0x1F |
| MIPI CSI-2*1*2 | YUV422, Y 12-bit/UV 8-bit | 0x1F (Transferred as YUV422 10-bit) |
| MIPI CSI-2*1*2 | RGB565 | 0x22 |
| MIPI CSI-2*1*2 | RGB888 | 0x24 |
| MIPI CSI-2*1*2 | RAW8 | 0x2A |
| MIPI CSI-2*1*2 | RAW10 | 0x2B |
| MIPI CSI-2*1*2 | RAW12 | 0x2C |
| MIPI CSI-2*1*2 | RAW14 | 0x2D |
| MIPI CSI-2*1*2 | RAW16 | 0x2E |
| MIPI CSI-2*1*2 | RAW20 | 0x2F |
| MIPI CSI-2*1*2 | RAW24 | 0x2C (Transferred as RAW12) |
| MIPI CSI-2*1*2 | User Defined 8-bit Data | From 0x30 to 0x37 |
| — | Unsupported | Others |

Notes: *1. Inputs are in the formats which conform to the standard for the MIPI CSI-2 interface.
*2. Receives MIPI CSI-2 interface via ISP (Channel Selector). The ISP (Channel Selector) setting is necessary.

### (2) Internal Sync Signal Generation

For the progressive video from CSI-2(via ISP Channel Selector), the field signal can be internally generated.

### (3) Capture Mode

The following four modes can be selected to capture the interlace images. In addition, single frame capture or continuous frame capture mode can be selected.

Triple-buffering control is provided in accordance with the captured field image and frame image to coordinate with the video capture mode of the display module.

- Odd-field capture mode
- Even-/odd-field capture mode
- Even-field capture mode
- Full interlace capture mode

### (4) Size Clipping

- Video channels 0 to 15

  The VIN module has one clipping circuits, which independently handle images with up to 4096 × 4096 pixels.

### (5) Image Data Format Conversion

The following data format conversions are available:

- YCbCr image data
  - Y/Cb/Cr 8-bit multiplex conversion
  - YC separation (separated into Y and CbCr components.)
  - Y component extraction
- RGB image data
  - RGB-888 → 32 bits/pixel conversion
  - RGB-888 → RGB-565 (16 bits/pixel) conversion
  - RGB-888 → ARGB-1555 (16 bits/pixel) conversion
  - RGB-888 → RGB-888 (32 bits/pixel) conversion
- RAW image data, Embedded 8-bit non Image Data, User Defined 8-bit Data
  - RAW8 → 8 bits/pixel conversion
  - Embedded 8-bit non Image Data, User Defined 8-bit Data → 8 bits/pixel conversion
  - RAW10 → 16 bits/pixel conversion
  - RAW12 → 16 bits/pixel conversion
  - RAW14 → 16 bits/pixel conversion
  - RAW16 → 16 bits/pixel conversion
  - RAW20 → 32 bits/pixel conversion
  - RAW24 → 32 bits/pixel conversion

### (6) Memory Output Data Format

Format-converted image data can be transferred to the memory. The following shows the available formats of the data stored.

- YCbCr image data
  - Y/Cb/Cr422, 8-bit multiplexed
  - Y/Cb/Cr, 10-bit multiplexed
  - YC separation, YCbCr422, Y, 8-bit Cb/Cr, 8-bit multiplexed
  - YC separation, YCbCr420, Y, 8-bit Cb/Cr, 8-bit multiplexed

  - YC separation, YCbCr422, Y, 10-bit Cb/Cr, 8-bit multiplexed
  - YC separation, YCbCr422, Y, 12-bit Cb/Cr, 8-bit multiplexed
  - YC separation, YCbCr422, Y, 10-bit Cb/Cr, 10-bit multiplexed
  - YC separation, Y data, 8-bit
  - YC separation, Y data, 10-bit
  - YC separation, Y data, 12-bit
  - YC separation, YCbCr420, Y, 10-bit Cb/Cr, 8-bit multiplexed
  - YC separation, YCbCr420, Y, 12-bit Cb/Cr, 8-bit multiplexed

- RGB image data
  - RGB-565 (16 bits/pixel)
  - ARGB-1555 (16 bits/pixel)
  - RGB-888 (32 bits/pixel)
  - ARGB-8888 (32 bits/pixel)

- RAW image data, Embedded 8-bit non Image Data, User Defined 8-bit Data
  - RAW8 (8 bits/pixel)
  - RAW8 (16 bits/pixel)
  - Embedded 8-bit non-Image Data, User Defined 8-bit Data (8 bits/pixel)
  - RAW10 (16 bits/pixel)
  - RAW12 (16 bits/pixel)
  - RAW14 (16 bits/pixel)
  - RAW16 (16 bits/pixel)
  - RAW20 (32 bits/pixel)
  - RAW24 (32 bits/pixel)

### 55.1.2 Block Diagram

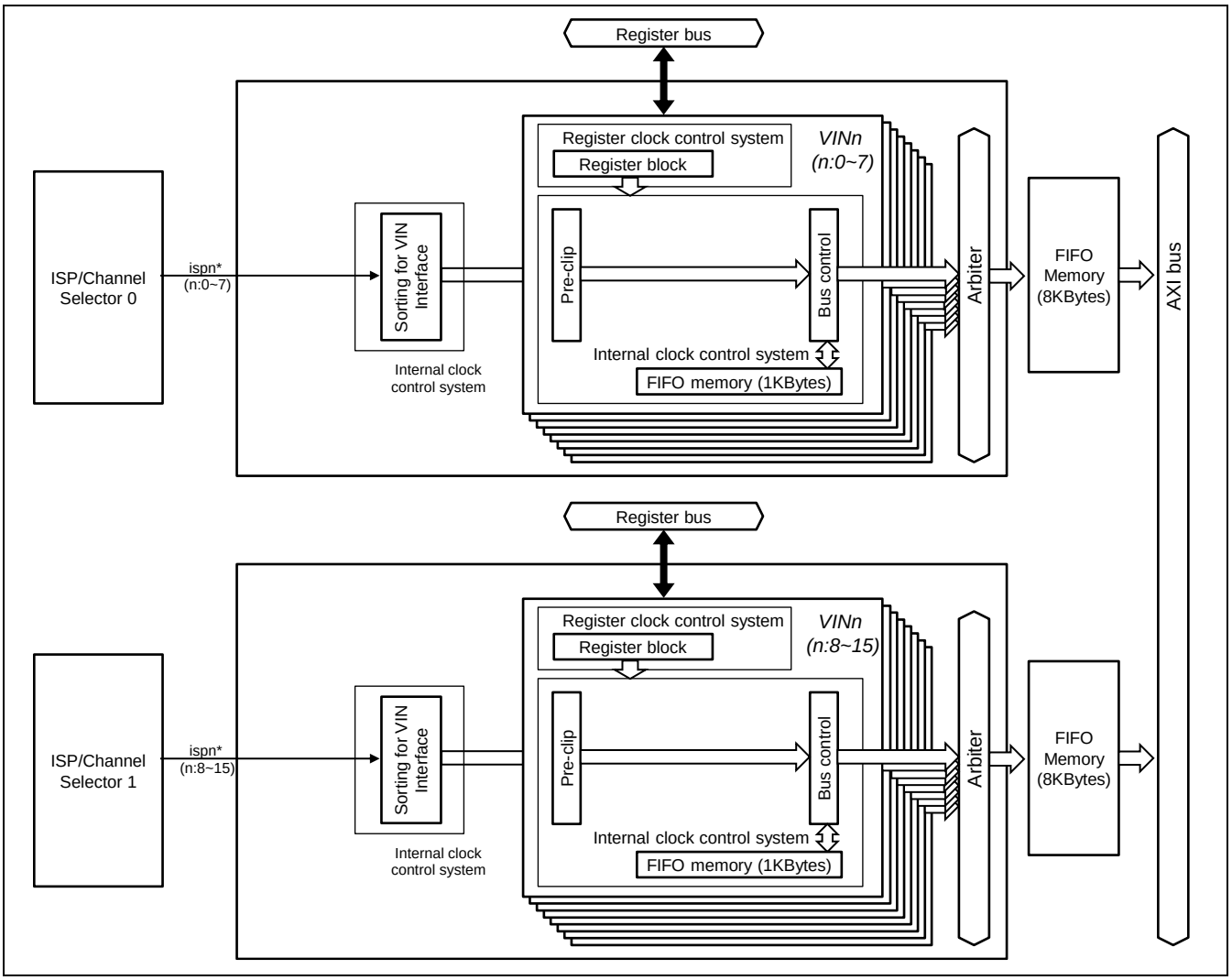



**Figure 55.1 Functional Block Diagram of Video Input Modules**

### 55.1.3 External Pins

VIN doesn't have external pins.

### 55.1.4 Register Configuration

Table 55.1 shows the VIN register configuration for each channel. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined. The access size is no supported except for 32 bits.

For the internal update mode, refer to the description of the VUP bit in the main control register (VnMC).

**Table 55.1 Base Address of Each VIN Channel**

| Base Address Name | Base Address |
|---|---|
| VIN0 | H'E6EF 0000 |
| VIN1 | H'E6EF 1000 |
| VIN2 | H'E6EF 2000 |
| VIN3 | H'E6EF 3000 |
| VIN4 | H'E6EF 4000 |
| VIN5 | H'E6EF 5000 |
| VIN6 | H'E6EF 6000 |
| VIN7 | H'E6EF 7000 |
| VIN8 | H'E6EF 8000 |
| VIN9 | H'E6EF 9000 |
| VIN10 | H'E6EF A000 |
| VIN11 | H'E6EF B000 |
| VIN12 | H'E6EF C000 |
| VIN13 | H'E6EF D000 |
| VIN14 | H'E6EF E000 |
| VIN15 | H'E6EF F000 |

**Table 55.2 Register Configuration of VIN (n = 0 to 15)**

| Module Name | Register Name | Symbol | R/W | Address n=0-15 | Initial Value | Access Size | Internal Update Mode Support* | Note |
|---|---|---|---|---|---|---|---|---|
| VIN | Video [n] Main Control Register(n=0-15) | V[n]MC | R/W | <VIN[n]> + H'000 | H'0000 0000 | 32 | FOC bit only supported | |
| VIN | Video [n] Module Status Register(n=0-15) | V[n]MS | R | <VIN[n]> + H'004 | H'0000 0018 | 32 | - | |
| VIN | Video [n] Frame Capture Register(n=0-15) | V[n]FC | R/W | <VIN[n]> + H'008 | H'0000 0000 | 32 | - | |
| VIN | Video [n] Start Line Pre-Clip Register(n=0-15) | V[n]SLPrC | R/W | <VIN[n]> + H'00C | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] End Line Pre-Clip Register(n=0-15) | V[n]ELPrC | R/W | <VIN[n]> + H'010 | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] Start Pixel Pre-Clip Register(n=0-15) | V[n]SPPrC | R/W | <VIN[n]> + H'014 | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] End Pixel Pre-Clip Register(n=0-15) | V[n]EPPrC | R/W | <VIN[n]> + H'018 | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] Image Stride Register(n=0-15) | V[n]IS | R/W | <VIN[n]> + H'02C | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] Memory Base 1 Register(n=0-15) | V[n]MB1 | R/W | <VIN[n]> + H'030 | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] Memory Base 2 Register(n=0-15) | V[n]MB2 | R/W | <VIN[n]> + H'034 | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] Memory Base 3 Register(n=0-15) | V[n]MB3 | R/W | <VIN[n]> + H'038 | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] Line Count Register(n=0-15) | V[n]LC | R | <VIN[n]> + H'03C | H'0000 0000 | 32 | - | |
| VIN | Video [n] Interrupt Enable Register(n=0-15) | V[n]IE | R/W | <VIN[n]> + H'040 | H'0000 0000 | 32 | - | |
| VIN | Video [n] Interrupt Status Register(n=0-15) | V[n]INTS | R/W | <VIN[n]> + H'044 | H'0000 0000 | 32 | - | |
| VIN | Video [n] Data Mode Register(n=0-15) | V[n]DMR | R/W | <VIN[n]> + H'058 | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] Data Mode Register 2(n=0-15) | V[n]DMR2 | R/W | <VIN[n]> + H'05C | H'0000 0000 | 32 | - | |
| VIN | Video [n] UV Address Offset Register(n=0-15) | V[n]UVAOF | R/W | <VIN[n]> + H'060 | H'0000 0000 | 32 | Supported | |
| VIN | Video [n] Write Line Count Register(n=0-15) | V[n]WLC | R | <VIN[n]> + H'0D4 | H'C000 C000 | 32 | - | |
| VIN | Video [n] IMR_VSPX Wake Up Channel Register(n=0-15) | V[n]IMWUCH | R/W | <VIN[n]> + H'0F0 | H'0000 0000 | 32 | - | |
| VIN | Video [n] IMR_VSPX Wake Up Line Register(n=0-15) | V[n]IMWUL | R/W | <VIN[n]> + H'0F4 | H'0000 0000 | 32 | - | |
| VIN | Video [n] IMR_VSPX Wake Up Line Register 2(n=0-15) | V[n]IMWUL2 | R/W | <VIN[n]> + H'0F8 | H'0000 0000 | 32 | - | |
| VIN | Video [n] Functional Safety Control Register(n=0-15) | V[n]FNSC | R/W | <VIN[n]> + H'320 | H'0000 0000 | 32 | - | |
| VIN | Video [n] FIELD Toggle Minimum Register(n=0-15) | V[n]FTMIN | R/W | <VIN[n]> + H'324 | H'0000 0000 | 32 | - | |

| Module Name | Register Name | Symbol | R/W | Address n=0-15 | Initial Value | Access Size | Internal Update Mode Support* | Note |
|---|---|---|---|---|---|---|---|---|
| VIN | Video [n] FIELD Toggle Maximum Register(n=0-15) | V[n]FTMAX | R/W | <VIN[n]> + H'328 | H'0000 0000 | 32 | - | |
| VIN | Video [n] HSYNC Toggle Minimum Register(n=0-15) | V[n]HSTMIN | R/W | <VIN[n]> + H'32C | H'0000 0000 | 32 | - | |
| VIN | Video [n] HSYNC Toggle Maximum Register(n=0-15) | V[n]HSTMAX | R/W | <VIN[n]> + H'330 | H'0000 0000 | 32 | - | |
| VIN | Video [n] VSYNC Toggle Minimum Register(n=0-15) | V[n]VSTMIN | R/W | <VIN[n]> + H'334 | H'0000 0000 | 32 | - | |
| VIN | Video [n] VSYNC Toggle Maximum Register(n=0-15) | V[n]VSTMAX | R/W | <VIN[n]> + H'338 | H'0000 0000 | 32 | - | |
| VIN | Video [n] Data Enable Term Register(n=0-15) | V[n]DET | R/W | <VIN[n]> + H'344 | H'0000 0000 | 32 | - | |
| VIN | Video [n] VCRC Code Register(n=0-15) | V[n]VCRC | R | <VIN[n]> + H'354 | H'0000 0000 | 32 | - | |
| VIN | Video [n] Hsync Rise Time Register(n=0-15) | V[n]HRT | R/W | <VIN[n]> + H'36C | H'0000 0000 | 32 | - | |
| VIN | Video [n] CRC Calculation Configuration Register(n=0-15) | V[n]CRCC | R/W | <VIN[n]> + H'380 | H'0000 0000 | 32 | - | |
| VIN | Video [n] CRC Polynomial Equation Coefficient Register(n=0-15) | V[n]CRPEC | R/W | <VIN[n]> + H'384 | H'0000 0000 | 32 | - | |
| VIN | Video [n] CRC Initial Remainder Register(n=0-15) | V[n]CRIR | R/W | <VIN[n]> + H'388 | H'0000 0000 | 32 | - | |
| VIN | Video [n] CRC XOR Coefficient Register(n=0-15) | V[n]CRXRC | R/W | <VIN[n]> + H'38C | H'0000 0000 | 32 | - | |
| VIN | Video [n] Error Enable Register(n=0-15) | V[n]EE | R/W | <VIN[n]> + H'3F0 | H'0000 0000 | 32 | - | |
| VIN | Video [n] Error Status Register(n=0-15) | V[n]ERS | R/W | <VIN[n]> + H'3F4 | H'0000 0000 | 32 | - | |

### 55.1.5 Connected Module

**Table 55.3 Connected Modules**

| Connected Modules | Function of the Connected Modules |
|---|---|
| ISP | Channel Selector: Demultiplexing MIPI/CSI-2 input data |
| CPG | Input reset and clock stop signal |
| INTTOP | Interrupt controller |
| ECM | Module related to Functional Safety |
| IMR | Output Wake Up signal |
| VSPX | Output Wake Up signal |
| VSPD | Output write buffer pointer for auto display function<br>For detaiis, refer to Section 60. |

## 55.2 Register Description

[Legend]

—: Reserved. The write value should always be 0.

Initial value: Register value after a reset

R/W: Readable/writable. The written value can be read.

R: Read-only. The write value should always be 0.

Vn: Video channel n

### 55.2.1 Video [n] Main Control Register(V[n]MC)(n=0-15)

Notes: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WLCEN | — | — | — | — | — | — | — | — | — | FOC | — | — | INF[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | EXINF[1:0] | | — | VUP | — | — | — | EN | — | IM[1:0] | | — | — | ME |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R/W | R | R | R | R/W | R | R/W | R/W | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | WLCEN | H'0 | R/W | Write Line Counter Enable<br>Always set 1 to this bit before capture start. |
| 30 to 22 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 21 | FOC | B'0 | R/W | Field Order Control<br>This bit controls the field order for full interlace capturing.<br>0: Top field = Odd field (field 1)<br>1: Top field = Even field (field 2)<br>Note: This bit supports the internal update mode. |
| 20, 19 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 18 to 16 | INF[2:0] | All 0 | R/W | Input Interface Format<br>These bits specify the image format input to the VIN.<br>[Input from Channel Selector]<br>000: Setting prohibited<br>001: YUV422 8-bit or YUV420 8-bit input*1<br>010: Setting prohibited<br>011: YUV422 10-bit or YUV420 10-bit or YUV422 12-bit (Y-12bit / UV 8-bit) input*1<br>100: Embedded 8-bit non Image Data or RAW8 input*1<br>101: Setting prohibited<br>110: RAW20 or RAW24 or RGB888 input*1<br>111: RAW8 (16 bits/pixel) or RAW10 or RAW12 or RAW14 or RAW16 or RGB565 input*1 |
| 15, 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13, 12 | EXINF[1:0] | H'0 | R/W | Extension Interface Select<br>These bits are used combined with the INF setting. |
| 11 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 10 | VUP | H'0 | R/W | VIN Register Update Control<br>This bit specifies the internal register update timing after writing register. See the list of registers for applicable registers.<br>0: The register contents are updated immediately after register writing.<br>1: The register contents are updated after a valid field is detected. |
| 9 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 | EN | H'0 | R/W | Endian Type<br>This bit specifies the endian type for data to be output to external memory.<br>0: Image data is packed and allocated in little endian.<br>1: Image data is packed and allocated in big endian.<br>Note: When allocating the YCbCr422 (UYVY format) data in big endian, be sure to set the BPSM bit in VnDMR to 1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 4, 3 | IM[1:0] | H'0 | R/W | Interlace Mode<br>These bits specify the capture mode. Do not modify this setting during capture operation.<br>00: Odd-field (field 1) capture mode<br>Handles only odd fields as frames and stores them in external memory.<br>01: Odd-/even-field capture mode<br>Handles odd and even fields as separate frames and stores them in external memory. This mode is available only in continuous frame capture mode.<br>10: Even-field (field 2) capture mode<br>Handles only even fields as frames and stores them in external memory.<br>11: Full interlace mode<br>Handles combinations of odd and even fields as single frames and stores them in external memory.<br>Note: Set 01 in case MFC is set to 1 |
| 2, 1 | — | H'0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | ME | H'0 | R/W | Module Enable<br>This is the enable bit for the VIN. Set this bit before setting the Frame Capture (VnFC) register.<br>0: The module operation is stopped.<br>1: The module operation is enabled. *2 |

Note 1: Table 55.4 shows the combinations of interfaces which can be set by the input interface format (the INF bits in VnMC) and extension interface select (the EXINF bit in VnMC). Do not make the other settings.

Note 2: To stop capturing operation, only set the ME bit to 0. Do not change the other bit settings.

### 55.2.2 Video [n] Module Status Register(V[n]MS)(n=0-15)

Notes: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | FBS[1:0] | | FS | AV | CA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4, 3 | FBS[1:0] | B'11 | R | Frame Buffer Status<br>These bits show the frame buffer status.<br>00: The latest valid frame buffer has the base address defined by the memory base 1 register.<br>01: The latest valid frame buffer has the base address defined by the memory base 2 register.<br>10: The latest valid frame buffer has the base address defined by the memory base 3 register.<br>11: There is no valid frame buffer.<br>Note: When video capture is operating, this bit should be read after the FIS bit in VnINTS is set to 1. |
| 2 | FS | H'0 | R | Field Status<br>This bit shows the type of the current capture field.<br>0: The current field is an odd field (field 1).<br>1: The current field is an even field (field 2). |
| 1 | AV | H'0 | R | Active Video Status<br>This bit shows whether the current field is in the active video area defined by the pre-clipping register.<br>0: The current field is not in the active video area.<br>1: The current field is in the active video area.<br>Note: This bit will be 0 if no input data is captured. |
| 0 | CA | H'0 | R | Video Capture Active Status<br>This bit shows the current video capture operation status. This bit is updated by the captured field signal.<br>0: Video capture is not operating.<br>1: Video capture is operating.<br>Note: In field capture mode, this bit is set to 1 even for the field that does not capture data. |

### 55.2.3 Video [n] Frame Capture Register(V[n]FC)(n=0-15)

Note: Availability of channels: n=0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CC | SC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | CC | H'0 | R/W | Continuous Frame Capture Mode<br>This bit specifies the continuous frame capture mode. In this mode, the first capture frame is written into the memory address specified by the memory base 1 (VnMB1) register. After that, the capture operation is repeated in the order of MB2, MB3, MB1, MB2, and such.<br>Writing 0 into this bit during continuous capture operation will immediately terminate the capture operation if the current frame is completed or it has not been captured.<br>0: The continuous frame capture mode is not set.<br>1: The continuous frame capture mode is set. |
| 0 | SC | H'0 | R/W | Single Frame Capture Mode<br>This bit specifies the single frame capture mode. In this mode, the capture frame is written into the memory address specified by the memory base 1 (VnMB1) register.<br>Immediately after this bit is set to 1, the frame buffer status (FBS) bits in MS are initialized and the SC bit is also cleared to 0.<br>0: The single frame capture mode is not set.<br>1: The single frame capture mode is set.<br>Note: Do not set this bit to 1 when the interlace mode (IM) bits in the main control register (VnMC) are set to 01 (odd-/even-field capture mode). |

Note 1: Do not specify the single frame capture mode and continuous frame capture mode at the same time.

### 55.2.4 Video [n] Start Line Pre-Clip Register(V[n]SLPrC)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SLPrC[11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | SLPrC[11:0] | H'000 | R/W | Start Line Pre-Clip<br>These bits specify the (pre-clipping start line - 1) value in line units. Specify a value in the range from 0 to 4094 so that the number of lines after pre-clipping will be 2 or more. (The value of 0 indicates the first valid line.) |

### 55.2.5 Video [n] End Line Pre-Clip Register(V[n]ELPrC)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ELPrC[11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | ELPrC[11:0] | H'000 | R/W | End Line Pre-Clip<br>These bits specify the (pre-clipping end line - 1) value in line units. Specify a value in the range from 1 to 4095 so that the number of lines after pre-clipping will be 2 or more.<br>Note: If the interlace images are input, the value here should be such that ELPrC - SLPrC is half of the value of the number of the original line. |

### 55.2.6 Video [n] Start Pixel Pre-Clip Register(V[n]SPPrC)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SPPrC[11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | SPPrC[11:0] | H'000 | R/W | Start Pixel Pre-Clip<br>These bits specify the (pre-clipping start pixel - 1) value in pixel units. Specify a value in the range from 0 to 4090 so that the number of pixels after pre-clipping will be an even number greater than or equal to 6.<br>Notes: 1. When SPPrC is set to 0x0, the first valid pixel is specified.<br>2. Specify an even number.<br>3. When Embedded 8-bit non-Image Data or RAW8 format are input, these bits should be even number. |

### 55.2.7 Video [n] End Pixel Pre-Clip Register(V[n]EPPrC)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | EPPrC[11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | EPPrC[11:0] | H'000 | R/W | End Pixel Pre-Clip<br>These bits specify the (pre-clipping end pixel - 1) value in pixel units. Specify a value in the range from 5 to 4095 so that the number of pixels after pre-clipping will be an even number greater than or equal to 6.<br>Notes: 1. Set this bit so that the (EPPrC - SPPrC) value is an odd number.<br>2. When Embedded 8-bit non-Image Data or RAW8 format are input, these bits should be odd number. |

### 55.2.8 Video [n] Image Stride Register(V[n]IS)(n=0-15)

Note: Availability of channels: n = 0 to 15

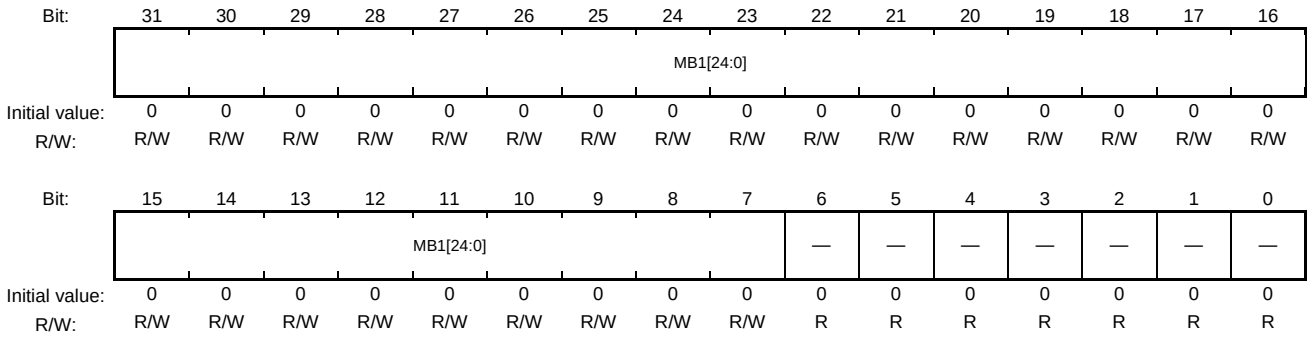
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IS[11:0] | | | | | | | | | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 4 | IS[11:0] | H'000 | R/W | Image Stride<br>These bits specify the width of the transfer destination memory in 16-pixel unit.<br>Specify a value no less than the pre-clipping width (EPPrC - SPPrC).<br>Note: When output format is other than RGB888 and ARGB8888, IS[0] bit shall be set to 0. |
| 3 to 0 | — | All 0 | R | Reserved bits that indicate the lower-order four bits of the image stride.<br>These bits are always read as 0. The write value should always be 0. |

### 55.2.9 Video [n] Memory Base 1 Register(V[n]MB1)(n=0-15)

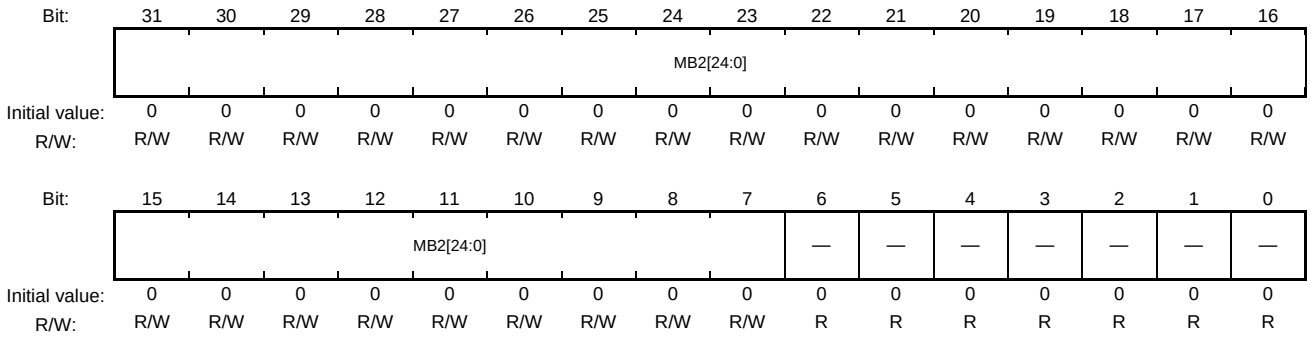
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 7 | MB1[24:0] | H'0000000 | R/W | Memory Base Address 1<br>These bits specify the transfer start address in frame buffer 1. Specify a value for physical address bits [31:7] in units of 128 bytes.<br>If the module is in continuous frame capture mode, this value is used as the MB1 address in the following capture sequence: MB1→ MB2 → MB3 → MB1 → MB2 → MB3.<br>In single frame capture mode, this value is used as the capture address. Specify a memory address taking into account the image size so that the image data does not exceed an area boundary on the address map. |
| 6 to 0 | — | All 0 | R | Reserved bits that indicate the lower-order seven bits of memory base address 1 (a multiple of 128 bytes).<br>These bits are always read as 0. The write value should always be 0. |

### 55.2.10 Video [n] Memory Base 2 Register(V[n]MB2)(n=0-15)

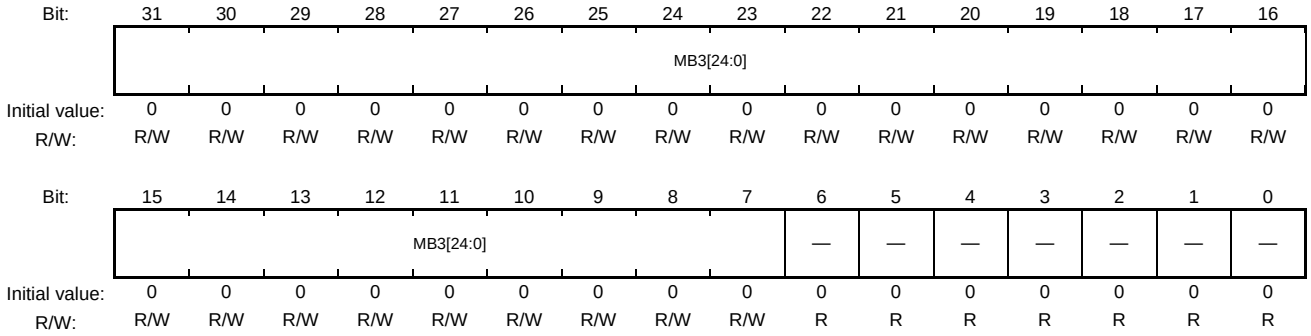
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 7 | MB2[24:0] | H'0000000 | R/W | Memory Base Address 2<br>These bits specify the transfer start address in frame buffer 2. Specify a value for physical address bits [31:7] in units of 128 bytes.<br>If the module is in continuous frame capture mode, this value is used as the MB2 address in the following capture sequence: MB1 → MB2 → MB3 → MB1 → MB2 → MB3.<br>Specify a memory address taking into account the image size so that the image data does not exceed an area boundary on the address map. |
| 6 to 0 | — | All 0 | R | Reserved bits that indicate the lower-order seven bits of memory base address 2 (a multiple of 128 bytes).<br>These bits are always read as 0. The write value should always be 0. |

### 55.2.11 Video [n] Memory Base 3 Register(V[n]MB3)(n=0-15)

Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 7 | MB3[24:0] | H'0000000 | R/W | Memory Base Address 3<br>These bits specify the transfer start address in frame buffer 3. Specify a value for physical address bits [31:7] in units of 128 bytes.<br>If the module is in continuous frame capture mode, this value is used as the MB3 address in the following capture sequence: MB1→ MB2 → MB3 → MB1 → MB2 → MB3.<br>Specify a memory address taking into account the image size so that the image data does not exceed an area boundary on the address map. |
| 6 to 0 | — | All 0 | R | Reserved bits that indicate the lower-order seven bits of memory base address 3 (a multiple of 128 bytes).<br>These bits are always read as 0. The write value should always be 0. |

### 55.2.12 Video [n] Line Count Register(V[n]LC)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | LC[11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | LC[11:0] | All 0 | R | Line Count<br>These bits show the line position in the current capture field.<br>The VnLC counts up at the rising edge of the next Hsync of captured line.<br>The VnLC stops counting up after the input of the line specified by the Line preclip register. |

### 55.2.13 Video [n] Interrupt Enable Register(V[n]IE)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FIE2 | — | — | — | — | — | — | WUE | — | — | — | — | — | — | VFE | VRE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | FIE | — | — | EFE | FOE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | FIE2 | H'0 | R/W | Field Interrupt Enable 2<br>This bit enables or disables INTC output for field interrupts.<br>Interrupt signals by this enable bit is asserted irrespective of whether capture is taking place.<br>0: Field interrupts are disabled.<br>1: Field interrupts are enabled. |
| 30 to 25 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 24 | WUE | H'0 | R/W | Wake-up Notification Interrupt Enable<br>This bit enables or disables INTC output for wake-up notification interrupts.<br>Interrupt signals by this enable bit is asserted irrespective of whether capture is taking place.<br>0: Wake-up notification interrupts are disabled.<br>1: Wake-up notification interrupts are enabled. |
| 23 to 18 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17 | VFE | H'0 | R/W | VSYNC Falling Edge Detect Interrupt Enable<br>This bit enables or disables VSYNC falling edge detect interrupts.<br>Interrupt signals by this enable bit is asserted irrespective of whether capture is taking place.<br>0: VSYNC falling edge detect interrupts are disabled.<br>1: VSYNC falling edge detect interrupts are enabled. |
| 16 | VRE | H'0 | R/W | VSYNC Rising Edge Detect Interrupt Enable<br>This bit enables or disables VSYNC rising edge detect interrupts. Interrupt signals by this enable bit is asserted irrespective of whether capture is taking place.<br>0: VSYNC rising edge detect interrupts are disabled.<br>1: VSYNC rising edge detect interrupts are enabled. |
| 15 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | FIE | H'0 | R/W | Field Interrupt Enable<br>This bit enables or disables field-switching interrupts. This interrupt enable setting is valid when the CA bit in VnMS is 1.<br>0: Field-switching interrupts are disabled.<br>1: Field-switching interrupts are enabled. |
| 3 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 2 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 1 | EFE | H'0 | R/W | End of Frame Interrupt Enable<br>This bit enables or disables end of frame interrupts. This interrupt enable setting is valid when the CA bit in VnMS is 1.<br>0: End of frame interrupts are disabled.<br>1: End of frame interrupts are enabled. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | FOE | H'0 | R/W | FIFO Overflow Interrupt Enable<br>This bit enables or disables FIFO overflow interrupts. This interrupt enable setting is valid when the CA bit in VnMS is 1.<br>0: FIFO overflow interrupts are disabled.<br>1: FIFO overflow interrupts are enabled. |

### 55.2.14 Video [n] Interrupt Status Register(V[n]INTS)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FIS2 | — | — | — | — | — | — | WUS | — | — | — | — | — | — | VFS | VRS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | FIS | — | — | EFS | FOS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | FIS2 | H'0 | R/W | Field Interrupt Status 2<br>This bit shows that the field has changed. This bit is set to 1 when a valid field is detected. After being set to 1, this bit is cleared to 0 by writing 1.<br>Note: This bit is set to 1 irrespective of whether or not capture is taking place. Be sure to clear this bit to 0 before using it. |
| 30 to 25 | — | All 0 | R | Reserved<br>Each bit is read as 0 or 1. The write value should always be 0. |
| 24 | WUS | H'0 | R/W | Wake-up Notification Interrupt Status<br>This bit is set to 1 when wake up notification event described in section 55.3.7 has occurred. After being set to 1, this bit is cleared to 0 by writing 1.<br>Note: This bit is set to 1 irrespective of whether or not capture is taking place. Be sure to clear this bit to 0 before using it. |
| 23 to 18 | — | All 0 | R | Reserved<br>These bits are always read as 0 or 1. The write value should always be 0. |
| 17 | VFS | H'0 | R/W | VSYNC Falling Edge Detect Interrupt Status<br>This bit shows that a VSYNC falling edge has been detected. After being set to 1, this bit is cleared to 0 by writing 1.<br>Note: This bit is set to 1 irrespective of whether or not capture is taking place. Be sure to clear this bit to 0 before using it. |
| 16 | VRS | H'0 | R/W | VSYNC Rising Edge Detect Interrupt Status<br>This bit shows that a VSYNC rising edge has been detected. After being set to 1, this bit is cleared to 0 by writing 1.<br>Note: This bit is set to 1 irrespective of whether or not capture is taking place. Be sure to clear this bit to 0 before using it. |
| 15 to 5 | — | All 0 | R | Reserved<br>Each bit is read as 0 or 1. The write value should always be 0. |
| 4 | FIS | H'0 | R/W | Field Interrupt Status<br>This bit shows that a field has been captured in the active capture operation.<br>This bit is set to 1 when a valid field is detected. After being set to 1, this bit is cleared to 0 by writing 1. |
| 3 | — | H'0 | R | Reserved<br>This bit is read as 0 or 1. The write value should always be 0. |
| 2 | — | H'0 | R | Reserved<br>This bit is read as 0 or 1. The write value should always be 0. |
| 1 | EFS | H'0 | R/W | End of Frame Interrupt Status<br>This bit shows that the last frame has been reached in the active capture operation. This bit is set to 1 at the end of even field (field 2). After being set to 1, this bit is cleared to 0 by writing 1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | FOS | H'0 | R/W | FIFO Overflow Interrupt Status<br>This bit shows that the FIFO has overflowed in the active capture operation. After being set to 1, this bit is cleared to 0 by writing 1.<br>If the FIFO overflows, the FIFO data is overwritten by the pixel data captured after the overflow, and the resultant data is sent to the frame buffer. |

### 55.2.15 Video [n] Data Mode Register(V[n]DMR)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A8BIT[7:0] | | | | | | | | — | FDS | RMODE[2:0] | | | — | — | EVA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | YMODE[2:0] | | | YC_THR | — | — | EXRGB | — | — | — | BPSM | — | ABIT | DTMD[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R | R | R/W | R | R | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | A8BIT[7:0] | H'00 | R/W | Alpha 8<br>These bits set the alpha value for the ARGB8888 format output. |
| 23 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | FDS | H'0 | R/W | Fill Data Select<br>This bit specifies selection of fill data for MSB bit when output data format is 8bits, 10 bits, 12 bits, 14 bits, 20bits, 24bits.<br>0: Fill data is 0<br>1: Fill data is sign data |
| 21 to 19 | RMODE[2:0] | H'0 | R/W | RAW Data Transfer Mode<br>These bits specify the bits of RAW data.<br>000: 8-bit RAW data, 16-bit RAW data, 24-bit RAW data<br>001: Setting prohibited<br>010: 10-bit RAW data<br>011: 12-bit RAW data<br>100: 14-bit RAW data<br>101: 20-bit RAW data<br>110: Setting prohibited<br>111: Setting prohibited |
| 18, 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | EVA | H'0 | R/W | Even Field Address Offset<br>This bit specifies the address offset of the even field (field 2) in memory in odd-/even-field capture mode.<br>0: Data are stored from the base address in external memory.<br>1: Data are stored from the base address plus the memory width in external memory. |
| 15 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 14 to 12 | YMODE[2:0] | H'0 | R/W | YC Data Transfer Mode<br>These bits specify the transfer method of Y/CbCr when the data conversion mode (DTMD) is 10 (YC separation).<br>000: Both Y and CbCr data are transferred to memory<br>001: Only Y data is transferred to memory as 8-bit data<br>010: 10-bit Y data is converted to 16-bit data and both Y and CbCr data are transferred to memory.<br>011: 10-bit Y data is converted to 16-bit data and only Y data is transferred to memory.<br>100: 12-bit Y data is converted to 16-bit data and both Y and CbCr data are transferred to memory.<br>101: 12-bit Y data is converted to 16-bit data and only Y data is transferred to memory.<br>110: Setting prohibited<br>111: Setting prohibited |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 | YC_THR | H'0 | R/W | YC Data Through Mode<br>Y and CbCr data are transferred to memory as 10-bit or 12-bit data according to the input format by setting this bit.<br>0: Y and CbCr data are transferred to memory according to YMODE[2:0] bits.<br>1: Y and CbCr data are transferred to memory as 10-bit or 12-bit data according to the input format. |
| 10, 9 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | EXRGB | H'0 | R/W | Extension RGB Conversion Mode<br>0: RGB data extension processing is not performed.<br>1: Data is extended to 32-bit RGB conversion when DTMD[1:0] is set to 00 or 01 as the data conversion mode. |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | BPSM | H'0 | R/W | Output Data Byte Swap Mode<br>0: Bytes are not swapped in output data.<br>1: Bytes are swapped in output data.<br>Note: When YCbCr-422 data is output in big endian, data is transferred in the YUYV format in most cases. To transfer data in the UYVY format, set this bit to 1. |
| 3 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 2 | ABIT | H'0 | R/W | Alpha Bit<br>This bit specifies the alpha value for data in ARGB-1555 output mode.<br>0: The alpha value is set to 0.<br>1: The alpha value is set to 1. |
| 1, 0 | DTMD[1:0] | H'0 | R/W | Data Conversion Mode<br>These bits set the format for storing RGB888 or YCbCr444 data in the external memory. *1<br>00: Data is not converted.<br>01: RGB is converted to ARGB before output.<br>10: YC is separated before output. *2<br>11: YC is separated before output (YCbCr420 format) |

Note 1: The data conversion modes that can be set are shown in Table 55.5. Do not set any other mode that is not listed in the table. RGB and YCbCr data should be set as shown in Table 55.5.

Note 2: Do not set for any other data format except for YCbCr422; set YC separation only for YCbCr data format.

### 55.2.16 Video [n] Data Mode Register 2(V[n]DMR2)(n=0-15)

Note: Availability of channels: n = 0 to 15

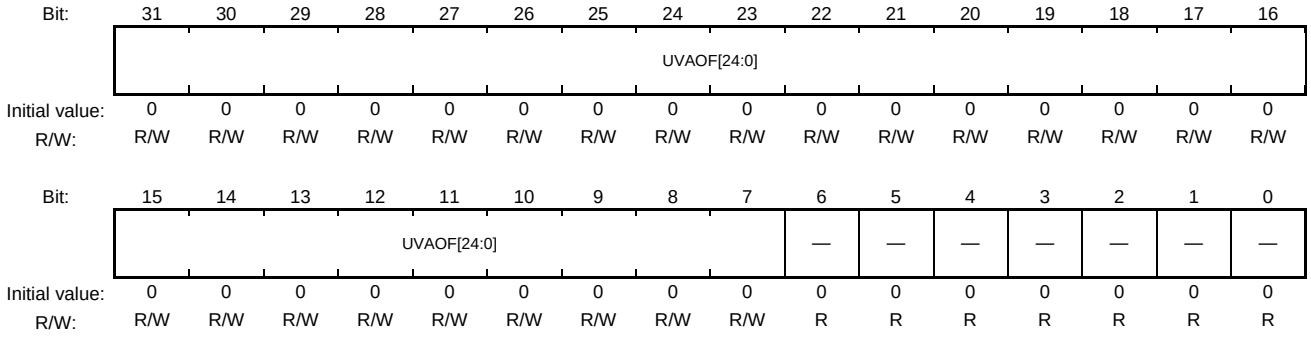
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | MFC | — | — | FTEV | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R/W | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VLV[3:0] | | | | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 20 | MFC | H'0 | R/W | Multi Frame Capture Mode<br>This is the enable bit for that the multi frame data is output to one frame buffer. When this bit is set to 1, it is necessary to set the VLV[3:0] bits.<br>0: Disables multi frame capture mode<br>1: Enables multi frame capture mode. In this mode, VIN captures from the first frame in case of progressive image.<br>Note: 1. Do not set both MFC and FTEV |
| 19, 18 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17 | FTEV | H'0 | R/W | VSYNC Field Toggle Mode Enable<br>The VSYNC field toggle mode changes the capture field signal level according to the count of input VSYNC signal assertion. As the VIN controls capture operation only when the input field signal level changes, select the VSYNC field toggle mode or multi frame capture mode when capturing progressive images.<br>0: The field toggle function according to the VSYNC count is disabled.<br>1: The field toggle function according to the VSYNC count is enabled. The period before a toggle should be specified in the VLV bits.<br>Note: Do not set both FTEV and MFC at the same time. |
| 16 | — | H'0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 12 | VLV[3:0] | H'0 | R/W | VSYNC Field Toggle Mode Transition Period<br>These bits specify the count of vertical sync signal input before the VSYNC field toggle mode is entered. After a transition to the VSYNC field toggle mode, the capture field signal is toggled every time VSYNC is input.<br>When a change in the input field signal is detected, the toggle mode is canceled.<br>H'0: The field signal is toggled at every VSYNC input.<br>H'1: Toggle mode is entered after VSYNC is input once.<br>H'2: Toggle mode is entered after VSYNC is input two times.<br>H'3: Toggle mode is entered after VSYNC is input three times.<br>:<br>H'E: Toggle mode is entered after VSYNC is input 14 times<br>H'F: Toggle mode is entered after VSYNC is input 15 times.<br>Note: If the field signal changes while the transition period is counted, the counter is initialized. |
| 11 to 0 | — | H'000 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 55.2.17 Video [n] UV Address Offset Register(V[n]UVAOF)(n=0-15)

Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 7 | UVAOF[24:0] | All 0 | R/W | UV Data Address Offset<br>These bits specify the transfer offset address for the YC separation YCbCr-422 UV and YCbCr-420 UV data.<br>Specify bits 31 to 7 of the physical address in 128-byte units.<br>Note: The specified address should be equal to or greater than the Y transfer size. Otherwise, the overwriting of Y data occurs. |
| 6 to 0 | — | All 0 | R | Reserved bits that indicate the 128-byte boundary of the UVAOF value. These bits are always read as 0. The write value should always be 0. |

### 55.2.18 Video [n] Write Line Count Register(V[n]WLC)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SWFBS[1:0] | | SWLC[13:0] | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WFBS[1:0] | | WLC[13:0] | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | SWFBS[1:0] | B'11 | R | Smallest Writing Frame Buffer Status<br>These bits show the smallest status buffer.<br>[VnIMWUCH.SVMD=0]<br>Video channels 0 to 7, 8 to 15:<br>Show the smallest frame buffer under writing among video channels 0 to 7, 8 to 15, respectively.<br><br>[VnIMWUCH.SVMD=1]<br>Video channels 0 to 2, 8 to 10:<br>Show the smallest frame buffer under writing among video channels 0 to 2, 8 to 10, respectively.<br>Video channels 3 to 5, 11 to 13:<br>Show the smallest frame buffer under writing among video channels 3 to 5, 11 to 13, respectively.<br>Video channels 6 and 7, 14 and 15:<br>Show the smallest frame buffer under writing among video channels 0 to 7, 8 to 15, respectively. |
| 29 to 16 | SWLC[13:0] | H'0000 | R | Smallest Write Line Count<br>These bits show the smallest write line count value.<br>[VnIMWUCH.SVMD=0]<br>Video channels 0 to 7, 8 to 15:<br>Show the smallest write line count value among video channels 0 to 7, 8 to 15, respectively.<br><br>[VnIMWUCH.SVMD=1]<br>Video channels 0 to 2, 8 to 10:<br>Show the smallest write line count value among video channels 0 to 2, 8 to 10, respectively.<br>Video channels 3 to 5, 11 to 13:<br>Show the smallest write line count value among video channels 3 to 5, 11 to 13, respectively.<br>Video channels 6 and 7, 14 and 15:<br>Show the smallest write line count value among video channels 0 to 7, 8 to 15, respectively. |
| 15, 14 | WFBS[1:0] | B'11 | R | Writing Frame Buffer Status<br>These bits show the status of frame buffer under writing.<br>00: Frame buffer under writing has the base address defined by the memory base 1 register.<br>01: Frame buffer under writing has the base address defined by the memory base 2 register.<br>10: Frame buffer under writing has the base address defined by the memory base 3 register.<br>11: There is no frame buffer under writing. |
| 13 to 0 | WLC[13:0] | H'0000 | R | Write Line Count<br>These bits show the line position which has been written to memory in the current capture field |

### 55.2.19 Video [n] IMR_VSPX Wake Up Channel Register(V[n]IMWUCH)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUMD | SVMD | — | — | — | — | — | — | — | — | — | — | — | WUE3 | WUE2 | WUE1 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | WUCH |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | WUMD | H'0 | R/W | Wake Up Mode<br>Specify 0 for using WLC. Selecting SWLC is prohibited.<br>This bit selects the counter value used for issuing IMR and VSPX wake up signals.<br>0: The WLC[13:0] bits value in the VnWLC<br>1: The SWLC[13:0] bits value in the VnWLC |
| 30 | SVMD | H'0 | R/W | Smallest Value Mode<br>This bit selects the SWLC[13:0] value.<br>0: The SWLC[13:0] bits value between video channel 0 to 7 is used<br>1: The SWLC[13:0] bits value between the below video channel is used<br>[The video channels 0 to 2, 8 to 10]<br>Between video channels 0 to 2, 8 to 10<br>[The video channels 3 to 5, 11 to 13]<br>Between video channels 3 to 5, 11 to 13<br>[The video channels 6, 7, 14, 15]<br>Between video channel 0 to 7, 8 to 15<br>Note: This bit is should be set to 0 in other than the video channels 0, 3, 8, 11. |
| 29 to 19 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 18 | WUE3 | H'0 | R/W | Wake Up Enable 3<br>This is the enable bit of the IMR/VSPX wake up notification 3 which is issued after all frame data is stored to memory.<br>0: Disables the issue of IMR/VSPX wake up notification 3<br>1: Enables the issue of IMR/VSPX wake up notification 3 |
| 17 | WUE2 | H'0 | R/W | Wake Up Enable 2<br>This is the enable bit of the IMR/VSPX wake up notification 2 which is issued by the WUL2[13:0] bits in the VnIMWUL.<br>0: Disables the issue of IMR/VSPX wake up notification 2<br>1: Enables the issue of IMR/VSPX wake up notification 2 |
| 16 | WUE1 | H'0 | R/W | Wake Up Enable 1<br>This is the enable bit of the IMR/VSPX wake up notification 1 which is issued by the WUL1[13:0] bits in the VnIMWUL.<br>0: Disables the issue of IMR/VSPX wake up notification 1<br>1: Enables the issue of IMR/VSPX wake up notification 1 |
| 15 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | WUCH | H'0 | R/W | Wake Up Enable<br>0: Disables wake up notification to IMRn and VSPXn<br>1: Enables wake up notification to IMRn and VSPXn |

Note 1: The setting of VnIMWUCH register shall not be changed while VIN captures.

### 55.2.20 Video [n] IMR_VSPX Wake Up Line Register(V[n]IMWUL)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | WUL2[13:0] | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | WUL1[13:0] | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29 to 16 | WUL2[13:0] | H'0000 | R/W | Wake Up Line 2<br>These bits specify a value to be compared with the WLC[13:0] bits or the SWLC[13:0] bits value in the VnWLC register in each field while the WUE2 bit in the VnIMWUCH register is set to 1.<br>When this value matches the WLC bits or the SWLC[13:0] bits value in the VnWLC register, wake up signal is notified to IMR/VSPX.<br>Note that when this value is set to 0, no wake up signal is notified even if WUE2 bit in VnIMWUCH is set to 1.<br>Whether to use the WLC[13:0] bits or the SWLC[13:0] bits are selected by the WUMD bit in the VnWUCH. |
| 15, 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 0 | WUL1[13:0] | H'0000 | R/W | Wake Up Line 1<br>These bits specify a value to be compared with the WLC[13:0] bits or the SWLC[13:0] bits value in the VnWLC register in each field while the WUE1 bit in the VnIMWUCH register is set to 1.<br>When this value matches the WLC bits or the SWLC[13:0] bits value in the VnWLC register, wake up signal is notified to IMR/VSPX.<br>Note that when this value is set to 0, no wake up signal is notified even if WUE1 bit in VnIMWUCH is set to 1.<br>Whether to use the WLC[13:0] bits or the SWLC[13:0] bits are selected by the WUMD bit in the VnWUCH. |

Note 1: The difference value between WUL2[13:0] bits and WUL1[13:0] shall be set to more than 2.

Note 2: The setting of VnIMWUL register shall not be changed while VIN captures.

### 55.2.21 Video [n] IMR_VSPX Wake Up Line Register 2(V[n]IMWUL2)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | WUL3[13:0] | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 0 | WUL3[13:0] | H'0000 | R/W | Wake Up Line 3<br>These bits specify a value to be compared with the WLC[13:0] bits or the SWLC[13:0] bits value in the VnWLC register in each field while the WUE3 bit in the VnIMWUCH register is set to 1.<br>When this value matches the WLC bits or the SWLC[13:0] bits value in the VnWLC register, wake up signal is notified to IMR/VSPX.<br>Note that when this value is set to 0, no wake up signal is notified even if WUE3 bit in VnIMWUCH is set to 1.<br>Whether to use the WLC[13:0] bits or the SWLC[13:0] bits are selected by the WUMD bit in the VnWUCH. |

Note 1: The difference value between WUL3[13:0] bits and WUL1[13:0] and WUL2[13:0] in the VnIMWUL shall be set to more than 2.

Note 2: The setting of VnIMWUL register shall not be changed while VIN captures.

### 55.2.22 Video [n] Functional Safety Control Register(V[n]FNSC)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HRCHK | — | — | — | HCEN[1:0] | | — | — | CRM | — | — | EDID[1:0] | | FED_EN | CED_EN | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R/W | R/W | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EDCDEI | DEI | — | — | DEER_DM | DECE[1:0] | | — | — | — | VSCE[1:0] | | HSCE[1:0] | | FLCE[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

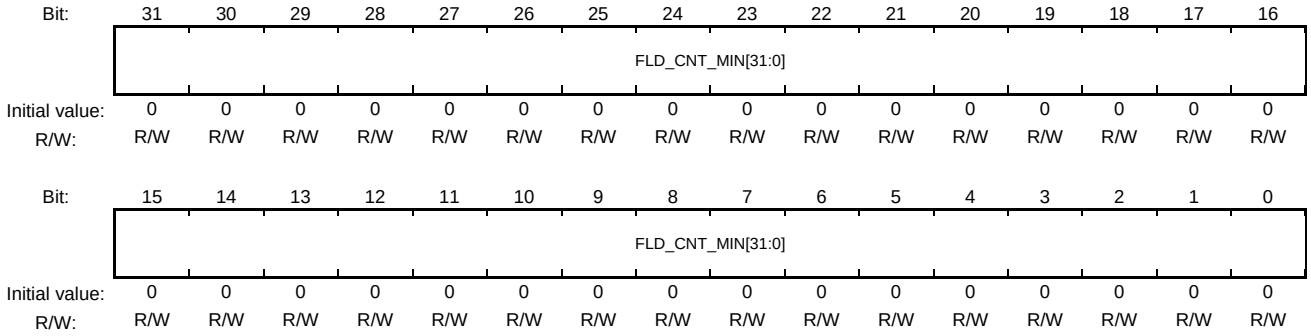
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | HRCHK | H'0 | R/W | HW Redundancy Error check<br>This bit enables or disables VIN HW Redundancy error detection.<br>0: Disable error of HW Redundancy detection<br>1: Enable error of HW Redundancy detection |
| 30 to 28 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27, 26 | HCEN[1:0] | H'0 | R/W | Enable Hsync Rise Time Measurement<br>These bits enable or disable to measure time of Hsync rise.<br>00: Disable Hsync rise time measurement<br>01: Enable minimum Hsync rise time measurement<br>10: Enable maximum Hsync rise time measurement<br>11: Enable minimum and maximum Hsync rise time measurement<br>Note: VnHRT setting are valid when this bit is 1. |
| 25, 24 | — | H'0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 | CRM | H'0 | R/W | CRC Calculation Mode<br>This bit specifies CRC calculation mode. When this bit is set to 1 and ERR bit in the FrameID is 0, if CRC calculation result is mismatched with the expected value, the error is detected.<br>0: Only calculate CRC<br>1: Input expected value and compare the CRC calculation result and expected value<br>Note: 1. This bit should be set to 0 in video channels 4 to 7, 12 to 15.<br>2. When ERR bit in the FrameID is 1, the error is not detected even if CRC calculation result is mismatched with expected value. |
| 22, 21 | — | All 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 20, 19 | EDID[1:0] | H'0 | R/W | EDC ID<br>These bits specify the ID number of Channel Selector EDC.<br>00: Channel Selector<br>01: Prohibit<br>10: Prohibit<br>11: Prohibit<br>Note: These bits are valid when the CDE_EN bit is set to 1. |
| 18 | FED_EN | H'0 | R/W | FIFO EDC Enable<br>This bit enables or disables EDC error of Write FIFO detection.<br>0: Disable EDC error of Write FIFO detection<br>1: Enable EDC error of Write FIFO detection |
| 17 | CED_EN | H'0 | R/W | Channel Selector EDC Enable<br>This bit enables or disables EDC error of channel selector detection.<br>0: Disable EDC error of channel selector detection<br>1: Enable EDC error of channel selector detection<br>Note: When this bit is set to 1, the EDID[1:0] bits should be set to 2'b00. |
| 16 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | EDCDEI | H'0 | R/W | EDC Dummy Error Injection<br>This bit issues EDC error (errreq_vin_vin_edc_p) for ECM module as dummy error.<br>0: Not issue dummy EDC error<br>1: Issue dummy EDC error<br>Note: n = 0 to 15 |
| 14 | DEI | H'0 | R/W | Dummy Error Injection<br>This bit issues error (errreq_vin_vin_p) for ECM module as dummy error.<br>0: Not issue dummy error<br>1: Issue dummy error<br>Note: n = 0 to 15 |
| 13, 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | DEER_DM | H'0 | R/W | Data Enable Assert Cycle Error Detection Mode<br>This bit specifies whether minimum error of Data enable assert cycle is detected when Data enable assert cycle is 0.<br>0: Data enable assert cycle error is not detected<br>1: Data enable assert cycle error is detected<br>Note: This bit setting is valid when the DECE[0] bit in VnFNSC is 1. |
| 10, 9 | DECE[1:0] | H'0 | R/W | Enable CLKENB Assert Cycle Measurement<br>This bit enables or disables to measure assert cycle of Data enable.<br>00: Disable Data enable assert cycle measurement<br>01: Enable minimum Data enable assert cycle measurement<br>10: Enable maximum Data enable assert cycle measurement<br>11: Enable minimum and maximum Data enable assert cycle measurement.<br>Note: VnDEMIN and VnDEMAX setting are valid when this bit is 1. |
| 8 to 6 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5, 4 | VSCE[1:0] | H'0 | R/W | Enable VSYNC Toggle Cycle Measurement<br>These bits enable or disable to measure toggle cycle of VSYNC.<br>00: Disable VSYNC toggle cycle measurement<br>01: Enable minimum VSYNC toggle cycle measurement<br>10: Enable maximum VSYNC toggle cycle measurement<br>11: Enable minimum and maximum VSYNC toggle cycle measurement<br>Note: VnVSTMIN and VnVSTMAX setting are valid when this bit is 1. |
| 3, 2 | HSCE[1:0] | H'0 | R/W | Enable HSYNC Toggle Cycle Measurement<br>These bits enable or disable to measure toggle cycle of HSYNC.<br>00: Disable HSYNC toggle cycle measurement<br>01: Enable minimum HSYNC toggle cycle measurement<br>10: Enable maximum HSYNC toggle cycle measurement<br>11: Enable minimum and maximum HSYNC toggle cycle measurement<br>Note: VnHSTMIN and VnHSTMAX setting are valid when this bit is 1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1, 0 | FLCE[1:0] | H'0 | R/W | Enable FIELD Toggle Cycle Measurement<br>These bits enable or disable to measure toggle cycle of FIELD.<br>00: Disable FIELD toggle cycle measurement<br>01: Enable minimum FIELD toggle cycle measurement<br>10: Enable maximum FIELD toggle cycle measurement<br>11: Enable minimum and maximum FIELD toggle cycle measurement<br>Note: VnFTMIN and VnFTMAX setting are valid when this bit is 1. |

### 55.2.23 Video [n] FIELD Toggle Minimum Register(V[n]FTMIN)(n=0-15)

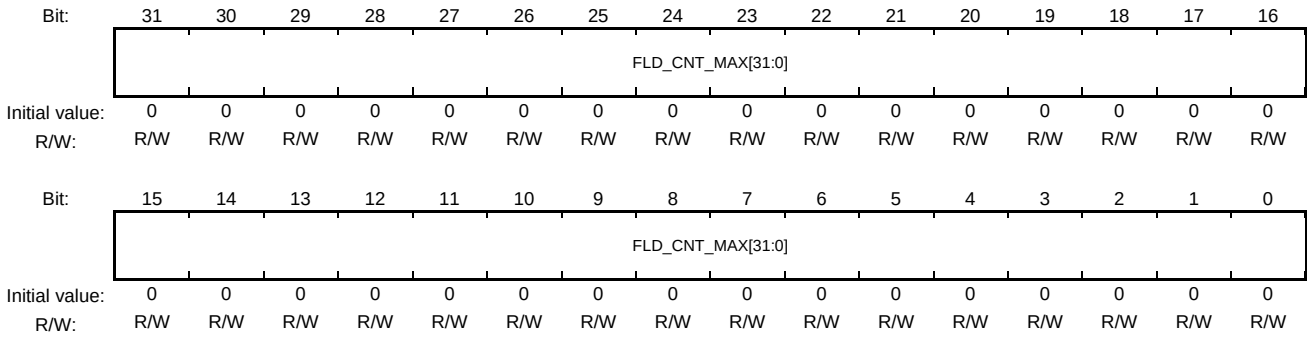
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | FLD_CNT_MIN[31:0] | H'0000 0000 | R/W | FIELD Toggle Cycle Minimum Value<br>These bits specify the minimum value of FIELD toggle cycle.<br>If FIELD toggle cycle is smaller than the value specified by these bits, Sync error is detected. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the FLCE[0] bit in VnFNSC is 1. |

### 55.2.24 Video [n] FIELD Toggle Maximum Register(V[n]FTMAX)(n=0-15)

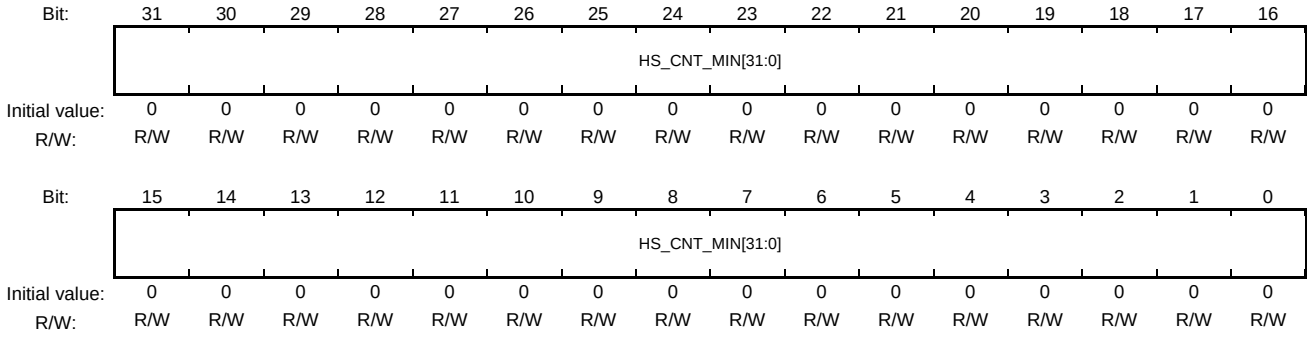
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | FLD_CNT_MAX[31:0] | H'0000 0000 | R/W | FIELD Toggle Cycle Maximum Value<br>These bits specify the maximum value of FIELD toggle cycle.<br>If FIELD toggle cycle is larger than the value specified by these bits, Sync error is detected. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the FLCE[1] bit in VnFNSC is 1. |

### 55.2.25 Video [n] HSYNC Toggle Minimum Register(V[n]HSTMIN)(n=0-15)

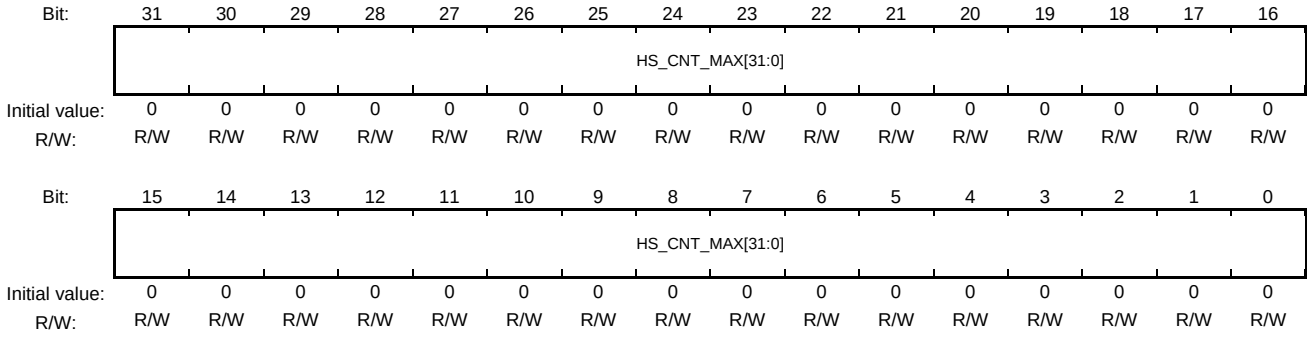
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | HS_CNT_MIN[31:0] | H'0000 0000 | R/W | HSYNC Toggle Cycle Minimum Value<br>These bits specify the minimum value of HSYNC toggle cycle.<br>If HSYNC toggle cycle is smaller than the value specified by these bits, Sync error is detected. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the HSCE[0] bit in VnFNSC is 1. |

### 55.2.26 Video [n] HSYNC Toggle Maximum Register(V[n]HSTMAX)(n=0-15)

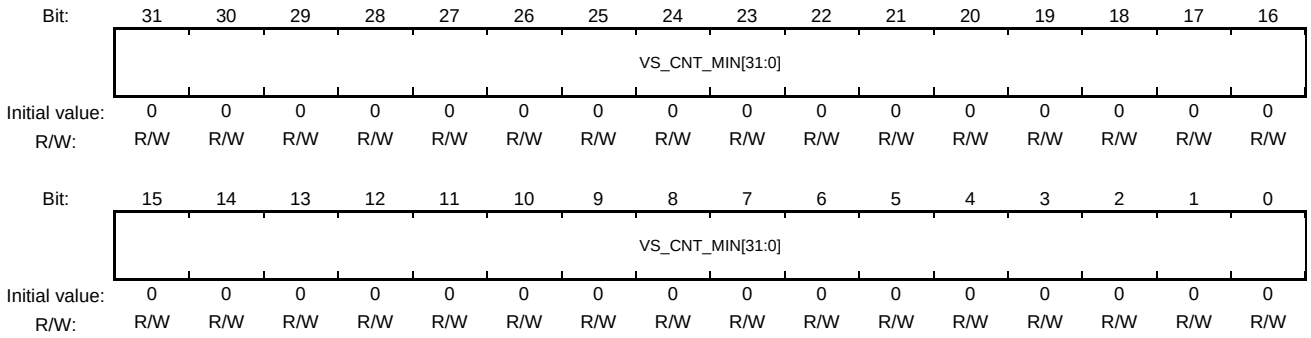
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | HS_CNT_MAX[31:0] | H'0000 0000 | R/W | HSYNC Toggle Cycle Maximum Value<br>These bits specify the maximum value of HSYNC toggle cycle.<br>If HSYNC toggle cycle is larger than the value specified by these bits, Sync error is detected. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the HSCE[1] bit in VnFNSC is 1. |

### 55.2.27 Video [n] VSYNC Toggle Minimum Register(V[n]VSTMIN)(n=0-15)

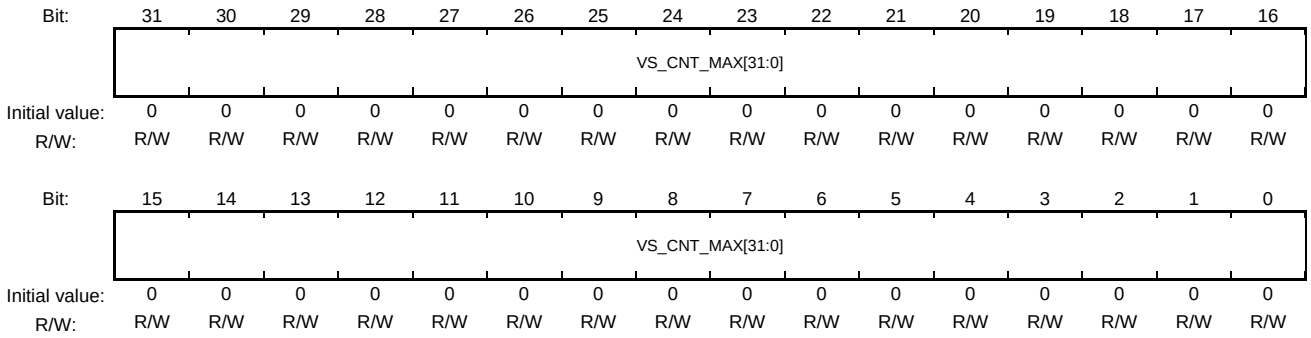
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | VS_CNT_MIN[31:0] | H'0000 0000 | R/W | VSYNC Toggle Cycle Minimum Value<br>These bits specify the minimum value of VSYNC toggle cycle.<br>If VSYNC toggle cycle is smaller than the value specified by these bits, Sync error is detected. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the VSCE[0] bit in VnFNSC is 1. |

### 55.2.28 Video [n] VSYNC Toggle Maximum Register(V[n]VSTMAX)(n=0-15)

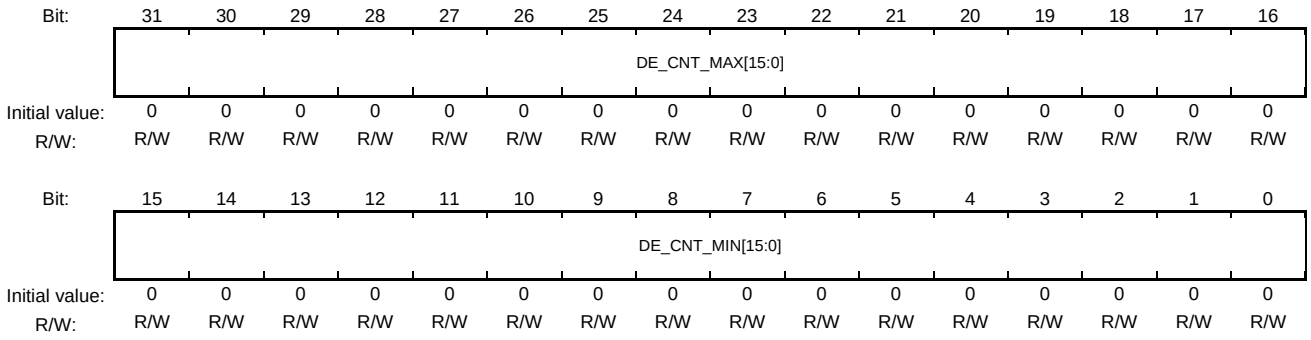
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | VS_CNT_MAX[31:0] | H'0000 0000 | R/W | VSYNC Toggle Cycle Maximum Value<br>These bits specify the maximum value of VSYNC toggle cycle.<br>If VSYNC toggle cycle is larger than the value specified by these bits, Sync error is detected. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the VSCE[1] bit in VnFNSC is 1. |

### 55.2.29 Video [n] Data Enable Term Register(V[n]DET)(n=0-15)

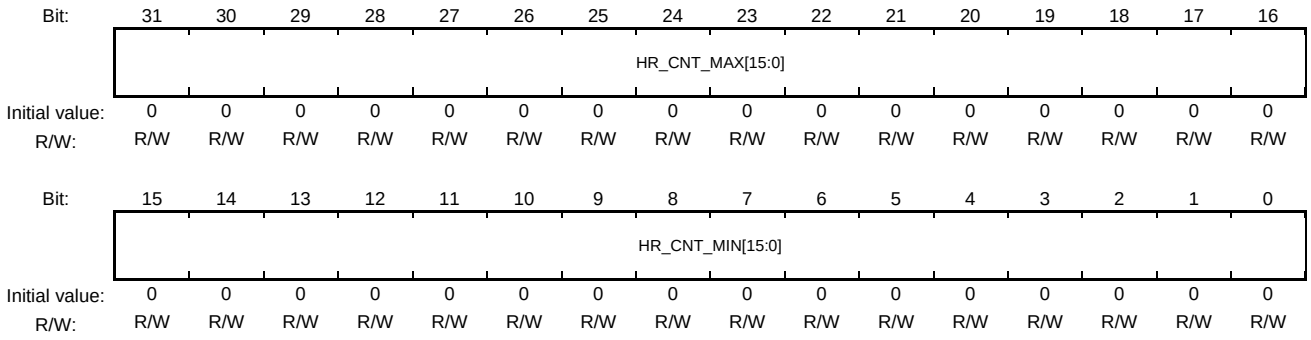
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | DE_CNT_MAX[15:0] | H'0000 | R/W | CLKENB Enable Cycle Maximum Value<br>These bits specify the maximum value of CLKENB enable cycle.<br>If CLKENB enable cycle is larger than the value specified by these bits, Sync error is detected. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the DECE[1] bit in VnFNSC is 1. |
| 15 to 0 | DE_CNT_MIN[15:0] | H'0000 | R/W | CLKENB Enable Cycle Minimum Value<br>These bits specify the minimum value of CLKENB enable cycle.<br>If CLKENB enable cycle is smaller than the value specified by these bits, Sync error is detected. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the DECE[0] bit in VnFNSC is 1. |

### 55.2.30 Video [n] Hsync Rise Time Register(V[n]HRT)(n=0-15)

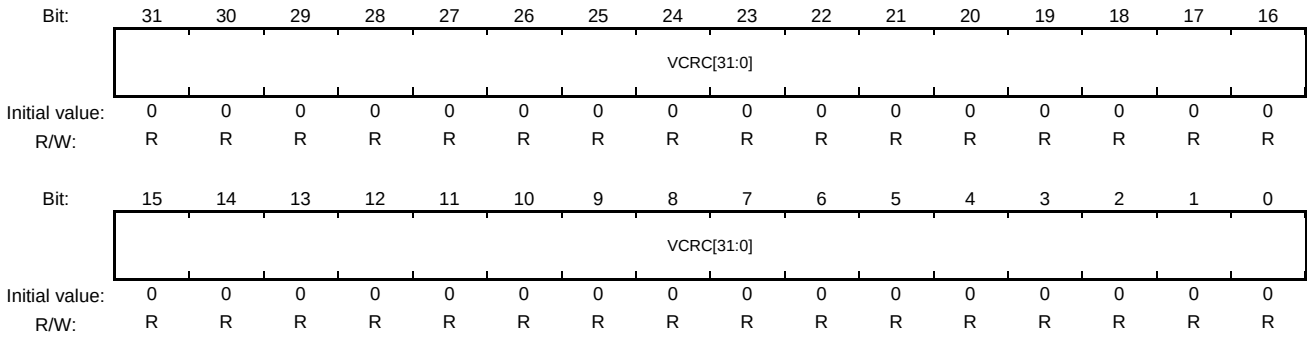
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | HR_CNT_MAX[15:0] | H'0000 | R/W | Hsync Rise Time Maximum Value<br>These bits specify the maximum number of rising edges of Hsync in one frame.<br>If the number of rising edges of Hsync is larger than the value specified by these bits, it is detected as Sync error. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the HCEN[1] bit in VnFNSC is 1. |
| 15 to 0 | HR_CNT_MIN[15:0] | H'0000 | R/W | Hsync Rise Time Minimum Value<br>These bits specify the minimum number of rising edges of Hsync in one frame.<br>If the number of rising edges of Hsync is smaller than the value specified by these bits, it is detected as Sync error. If Sync error is detected, VnINTS.SER bit is set to 1.<br>Note: These bits setting is valid when the HCEN[0] bit in VnFNSC is 1. |

### 55.2.31 Video [n] VCRC Code Register(V[n]VCRC)(n=0-15)

Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | VCRC[31:0] | H'0000 0000 | R | VCRC Code<br>These bits are show the results of the VCRC calculation. It is updated when VSYNC becomes to high (rise edge).<br>When the CRM bit in the VnFNSC is set to 1, the results of the VCRC calculation is compared with the expected value. |

### 55.2.32 Video [n] CRC Calculation Configuration Register(V[n]CRCC)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IDBOE[2:0] | | | — | — | — | Y420MD | INSEL | — | IDRE | — | — | — | — | IDBLE[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R | R | R | R/W | R/W | R | R/W | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IDBO[2:0] | | | PDBL[4:0] | | | | | ROR | IDR | — | — | — | — | IDBL[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W |

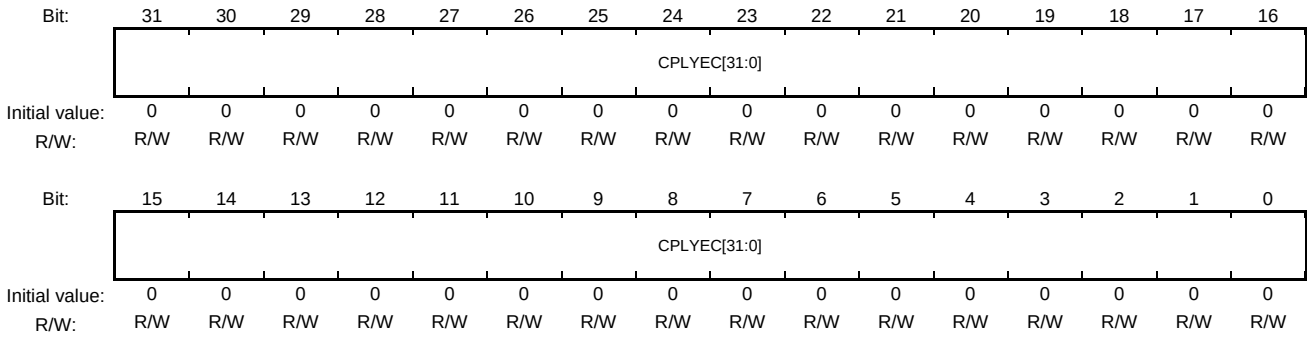
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | IDBOE[2:0] | H'0 | R/W | Set 0 to these bits. |
| 28 to 26 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 | Y420MD | H'0 | R/W | Set 0 to this bit. |
| 24 | INSEL | H'0 | R/W | Input Data Select<br>This bit is used to select the input data to CRC calculation.<br>0: Before image size clip<br>1: After image size clip |
| 23 | — | H'0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | IDRE | H'0 | R/W | Set 0 to this bit. |
| 21 to 18 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17, 16 | IDBLE[1:0] | H'0 | R/W | Set 0 to these bits. |
| 15 to 13 | IDBO[2:0] | H'0 | R/W | Input Data Byte Order<br>This bit specifies the byte swap setting for input data.<br>H'0: The bit order from MSB is bit 31 to 0<br>H'1: The bit order from MSB is bit 23 to 16, 31 to 24, 7 to 0, 15 to 8<br>H'2: The bit order from MSB is bit 15 to 0, 31 to 16<br>H'3: The bit order from MSB is bit 7to 0, 15 to 8, 23 to 16, 31 to 24<br>H'4: The bit order from MSB is bit 23 to 0, 31 to 24<br>H'5: The bit order from MSB is bit 15 to 8, 23 to16, 31 to 24, 7 to 0<br>H'6: Setting prohibited<br>H'7: Setting prohibited |
| 12 to 8 | PDBL[4:0] | H'00000 | R/W | Polynomial Equation Coefficient Data Bit Length<br>These bits specify the valid bit of coefficient which is specified by the VnCRPEC, VnCRIR, VnCRXRC.<br>H'00: 1-bit data (bit 0) is valid<br>H'01: 2-bit data (bit 1 and 0) are valid<br>H'02: 3-bit data (bit 2 to 0) are valid<br>H'03: 4-bit data (bit 3 to 0) are valid<br>~<br>H'0F: 16-bit data (bit 15 to 0) are valid<br>H'10: 17-bit data (bit 16 to 0) are valid<br>~<br>H'1E: 31-bit data (bit 30 to 0) are valid<br>H'1F: 32-bit data (bit 31 to 0) are valid |
| 7 | ROR | H'0 | R/W | Remainder Output Reflection<br>This bit specifies the reflection flag of remainder output. When this bit is set to 1, the bit order of remainder output data is reversed.<br>0: No reflection<br>1: Apply reflection |
| 6 | IDR | H'0 | R/W | Input Data Reflection<br>This bit specifies the reflection flag of input byte data. After applying byte order modification by IDBO[2:0] bits, the bit order of each byte of input data is reversed before CRC operation when this bit is set to 1.<br>0: No reflection<br>1: Apply reflection |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1, 0 | IDBL[1:0] | H'0 | R/W | Input Data Byte length<br>These bits specify the value of byte length of input data.<br>00: 1-byte data (bit 31 to 24) are valid<br>01: 2-byte data (bit 31 to 16) are valid<br>10: 3-byte data (bit 31 to 8) are valid<br>11: 4-byte data (bit 31 to 0) are valid<br>Note: This bit specifies the byte length of modified data by IDBO[2:0] and IDR bits. |

Note 1: Figure 55.2 shows the specifications of CRC calculation.
Figure 55.3 to Figure 55.7 show the specifications of each bit.

### 55.2.33 Video [n] CRC Polynomial Equation Coefficient Register(V[n]CRPEC)(n=0-15)

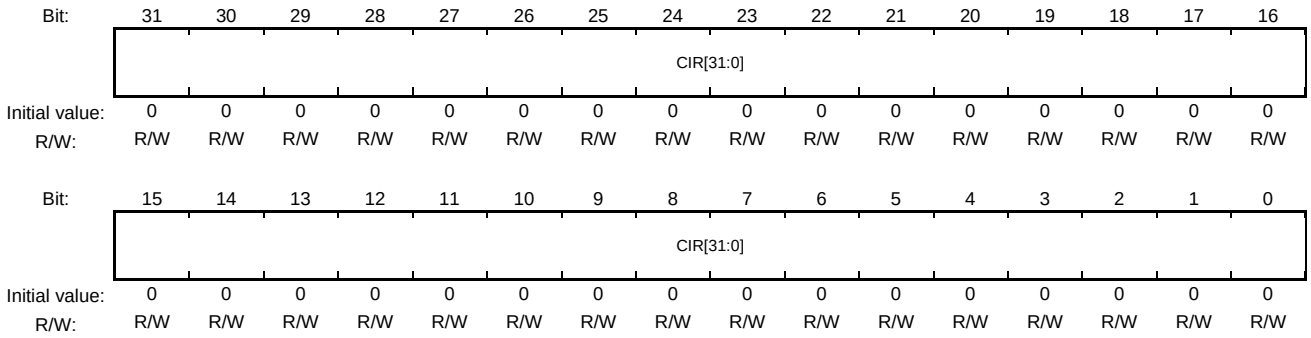
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CPLYEC[31:0] | H'0000 0000 | R/W | CRC Polynomial Equation Coefficient<br>These bits specify the coefficient of CRC polynomial equation. if polynomial is smaller than 32-bit, set to LSB side. |

### 55.2.34 Video [n] CRC Initial Remainder Register(V[n]CRIR)(n=0-15)

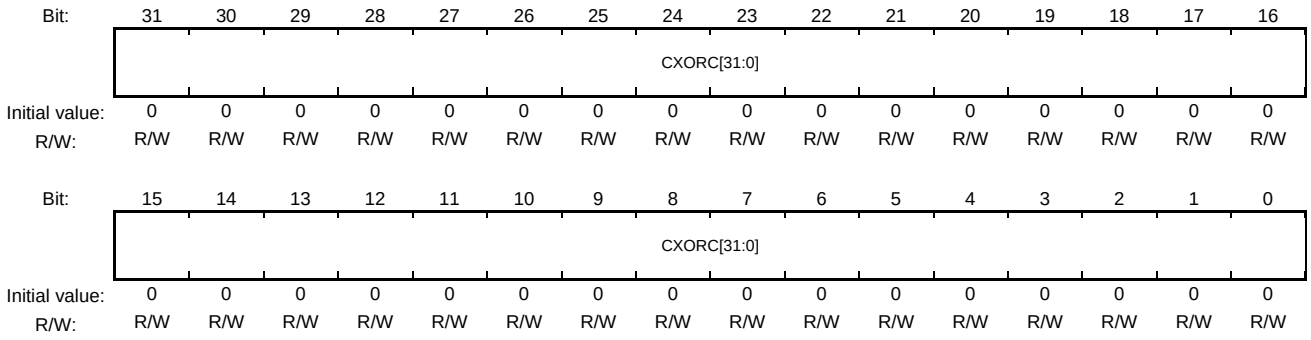
Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CIR[31:0] | H'0000 0000 | R/W | CRC Initial Remainder<br>These bits specify the initial value of reminder. if size of polynomial is smaller than 32-bit, set to LSB side. |

### 55.2.35 Video [n] CRC XOR Coefficient Register(V[n]CRXRC)(n=0-15)

Note: Availability of channels: n = 0 to 15




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CXORC[31:0] | H'0000 0000 | R/W | CRC Final XOR Coefficient<br>These bits specify the XOR operation coefficient to remainder. if size of polynomial is smaller than 32-bit, set to LSB side.<br>If no XOR operation is applied, set these bits to H'0000 0000. |

### 55.2.36 Video [n] Error Enable Register(V[n]EE)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | HREE[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | CCEE | EFEE | ECEE | — | SEE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 26 to 16 | HREE[10:0] | H'000 | R/W | HW Redundancy Error Detection Interrupt Enable<br>Each bit enables or disables HW redundancy logic error detection interrupt when HW redundancy logic error is detected. This interrupt enable setting is valid when the HRCHK bits in VnFNSC is 1.<br>0: HW redundancy logic error detection interrupts are disabled.<br>1: HW redundancy logic error detection interrupts are enabled.<br><br>Each bit specifies the below.<br>HREE[0]: AXI write request comparison error<br>HREE[1]: APB read data comparison error<br>HREE[2]: No operation<br>HREE[3]: Interrupt (For INTTOP module) comparison error<br>HREE[4]: Isack (For CPG module) comparison error<br>HREE[5]: Async or Sync reset (For CPG module) monitor comparison error<br>HREE[6]: Write Buffer (SRAM) access command comparison error<br>HREE[7]: IMR/VSPX Wake Up signal comparison error<br>HREE[8]: Frame Buffer ID (For VPS module) comparison error<br>HREE[9]: BUS(APB)-EDC output signals comparison error<br>HREE[10]: BUS(AXI)-EDC output signals comparison error |
| 15 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | CCEE | H'0 | R/W | ECRC Comparison Error Detection Interrupt Enable<br>This bit enables or disables ECRC comparison error detection interrupt when ECRC comparison error is detected. This interrupt enable setting is valid when the CRM bit in VnFNSC is 1.<br>0: ECRC comparison error detection interrupts are disabled.<br>1: ECRC comparison error detection interrupts are enabled. |
| 3 | EFEE | H'0 | R/W | EDC of FIFO Error Detection Interrupt Enable<br>This bit enables or disables EDC error of Write FIFO detection interrupt when EDC error of Write FIFO is detected. This interrupt enable setting is valid when the FED_EN bit in VnFNSC is 1.<br>0: EDC error of Write FIFO detection interrupts are disabled.<br>1: EDC error of Write FIFO detection interrupts are enabled. |
| 2 | ECEE | H'0 | R/W | EDC of Channel Selector Conversion Error Detection Interrupt Enable<br>This bit enables or disables EDC error of Channel Selector conversion logic detection interrupt when EDC error of Channel Selector conversion logic is detected. This interrupt enable setting is valid when the CED_EN bit in VnFNSC is 1.<br>0: EDC error of Channel Selector conversion logic detection interrupts are disabled.<br>1: EDC error of Channel Selector conversion logic detection interrupts are enabled. |
| 1 | — | B'0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | SEE | H'0 | R/W | Sync Interval Checker Error Detect Interrupt Enable<br>This bit enables or disables Sync Interval Checker error detect interrupts. This interrupt enable setting is valid when either of the DECE[1:0], VSCE[1:0], HSCE[1:0], FLCE[1:0] or HCEN[1:0] bits in VnFNSC are 1. Interrupt signals by this enable bit is asserted irrespective of whether capture is taking place.<br>0: Sync Interval Checker error detect interrupts are disabled.<br>1: Sync Interval Checker error detect interrupts are enabled. |

### 55.2.37 Video [n] Error Status Register(V[n]ERS)(n=0-15)

Note: Availability of channels: n = 0 to 15

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | HRER[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | CCER | EFER | ECER | — | SER |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 26 to 16 | HRER[10:0] | H'000 | R/W | HW Redundancy Error Detection Interrupt Status<br>Each bit shows that HW redundancy logic error has been detected.<br>Each bit is set to 1 when HW redundancy logic error is detected after being set to 1, this bit is cleared to 0 by writing 1.<br><br>Each bit specifies the below.<br>HRER[0]: AXI write request comparison error<br>HRER[1]: APB read data comparison error<br>HRER[2]: Removed feature<br>HRER[3]: Interrupt (For INTTOP module) comparison error<br>HRER[4]: Isack (For CPG module) comparison error<br>HRER[5]: Async or Sync reset (For CPG module) monitor comparison error<br>HRER[6]: Write Buffer (SRAM) access command comparison error<br>HRER[7]: IMR/VSPX Wake Up signal comparison error<br>HRER[8]: Frame Buffer ID (For VPS module) comparison error<br>HRER[9]: BUS(APB)-EDC output signals comparison error<br>HRER[10]: BUS(AXI)-EDC output signals comparison error |
| 15 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | CCER | H'0 | R/W | ECRC Comparison Error Detection Interrupt Status<br>This bit shows that ECRC comparison error has been detected.<br>This bit is set to 1 when ECRC comparison error is detected after being set to 1, this bit is cleared to 0 by writing 1. |
| 3 | EFER | H'0 | R/W | EDC of FIFO Error Detection Interrupt Status<br>This bit shows that EDC error of Write FIFO has been detected.<br>This bit is set to 1 when EDC error of Write FIFO is detected after being set to 1, this bit is cleared to 0 by writing 1. |
| 2 | ECER | H'0 | R/W | EDC of Channel Selector Conversion Error Detection Interrupt Status<br>This bit shows that EDC error of Channel Selector conversion logic has been detected.<br>This bit is set to 1 when EDC error of Channel Selector conversion logic is detected after being set to 1, this bit is cleared to 0 by writing 1. |
| 1 | — | B'0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | SER | H'0 | R/W | Sync Interval Checker Error Detect Interrupt Status<br>This bit shows that Sync error has been detected.<br>This bit is set to 1 when Sync Interval Checker error is detected after being set to 1, this bit is cleared to 0 by writing 1.<br>For detail of Sync Interval Checker error control, refer VnFNSC, VnFTMIN, VnFTMAX, VnHSTMIN, VnHSTMAX, VnVSTMIN, VnVSTMAX, VnVETMIN, VnVETMAX, VnDET and VnHRT registers. |

### 55.2.38 Tables and Figures

**Table 55.4 Capture Interface Settings (from Channel Selector)**

| Interface | VnMC/EXINF | VnMC/INF |
|---|---|---|
| YUV422 8-bit or YUV420 8-bit | 00 | 001 |
| YUV422 10-bit or YUV420 10-bit | 00 | 011 |
| YUV422 12-bit (Y-12bit / UV-8bit) | 11 | 011 |
| Embedded 8-bit non-Image Data or RAW8 | 01 | 100 |
| RAW8 (16 bits/pixel) or RAW10 or RAW12 or RAW14 or RAW16 or RGB565 | 00 | 111 |
| RAW20 or RAW24 or RGB888 | 00 | 110 |

**Table 55.5 Data Conversion Settings**

- RGB Data Conversion Modes

| Format of Data Stored in Memory | VnDMR/ YMODE[2:0] | VnDMR/ EXRGB | VnDMR/ YC_THR | VnDMR/ DTMD[1:0] | VnDMR/ RMODE[2:0] | Remarks |
|---|---|---|---|---|---|---|
| RGB-565 (16 bits/pixel) format | 000 | 0 | 0 | 00 | 000 | |
| RGB-888 (32 bits/pixel) format | 000 | 1 | 0 | 00 | 000 | |
| ARGB-1555 (16 bits/pixel) format | 000 | 0 | 0 | 01 | 000 | The alpha bit is set with the ABIT bit in VnDMR. |
| ARGB-8888 (32 bits/pixel) format | 000 | 1 | 0 | 01 | 000 | The alpha bit is set with the A8BIT bit in VnDMR. |

- YCbCr Data Conversion Modes

| Format of Data Stored in Memory | VnDMR/ YMODE[2:0] | VnDMR/ EXRGB | VnDMR/ YC_THR | VnDMR/ DTMD[1:0] | VnDMR/ RMODE[2:0] | Remarks |
|---|---|---|---|---|---|---|
| YCbCr-422 ( 8 bits) transfer | 000 | 0 | 0 | 00 | 000 | Set the BPSM bit in VnDMR to 1 to transfer in UYVY format in big endian.<br>Make these settings if the setting of the INF[2:0] bits in VnMC is B'100. |
| YCbCr-422 (10 bits) transfer* | 000 | 0 | 1 | 00 | 000 | |
| Y (8 bits)/CbCr separation transfer | 000 | 0 | 0 | 10/11 | 000 | CbCr transfer destination is determined by the VnUVAOF register setting. |
| Y (10 or 12 bits)/CbCr separation transfer* | 000 | 0 | 1 | 10 | 000 | Data selection of 10-bit or 12-bit is according to the input format. |
| Y (8 bits) transfer | 001 | 0 | 0 | 10 | 000 | |
| Y (16 bits)/CbCr separation transfer | 010/100 | 0 | 0 | 10 | 000 | CbCr transfer destination is determined by the VnUVAOF register setting. |

| Format of Data Stored in Memory | VnDMR/ YMODE[2:0] | VnDMR/ EXRGB | VnDMR/ YC_THR | VnDMR/ DTMD[1:0] | VnDMR/ RMODE[2:0] | Remarks |
|---|---|---|---|---|---|---|
| Y (16 bits)/CbCr separation transfer (YCbCr-420 format) | 010/100 | 0 | 0 | 11 | 000 | CbCr transfer destination is determined by the VnUVAOF register setting. |
| Y (16 bits) transfer | 011/101 | 0 | 0 | 10 | 000 | |

Notes: If any combination of values not listed above is specified, correct operation is not guaranteed.

- RAW Data Conversion Modes

| Format of Data Stored in Memory | VnDMR/ YMODE[2:0] | VnDMR/ EXRGB | VnDMR/ YC_THR | VnDMR/ DTMD[1:0] | VnDMR/ RMODE[2:0] | Remarks |
|---|---|---|---|---|---|---|
| RAW8 (8 bits/pixel) format | 000 | 0 | 0 | 00 | 000 | |
| RAW8 (16 bits/pixel) format | 000 | 0 | 0 | 00 | 000 | |
| RAW10 (16 bits/pixel) format | 000 | 0 | 0 | 00 | 010 | |
| RAW12 (16 bits/pixel) format | 000 | 0 | 0 | 00 | 011 | |
| RAW14 (16 bits/pixel) format | 000 | 0 | 0 | 00 | 100 | |
| RAW16 (16 bits/pixel) format | 000 | 0 | 0 | 00 | 000 | |
| RAW20 (32 bits/pixel) format | 000 | 1 | 0 | 00 | 101 | |
| RAW24 (32 bits/pixel) format | 000 | 1 | 0 | 00 | 000 | |

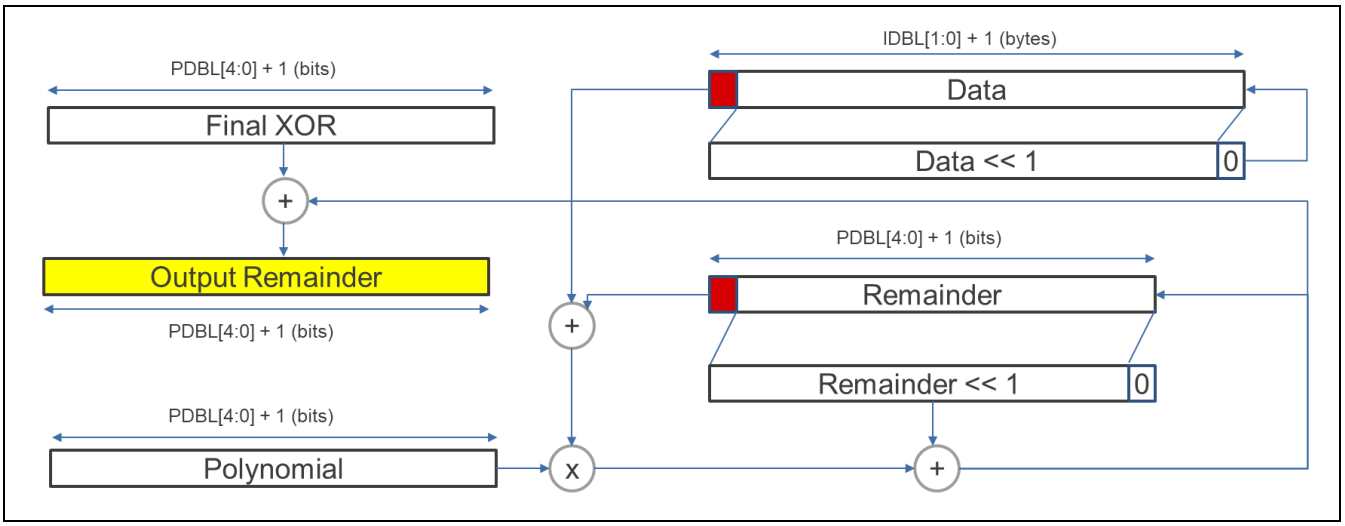



**Figure 55.2 Specifications of CRC calculation**

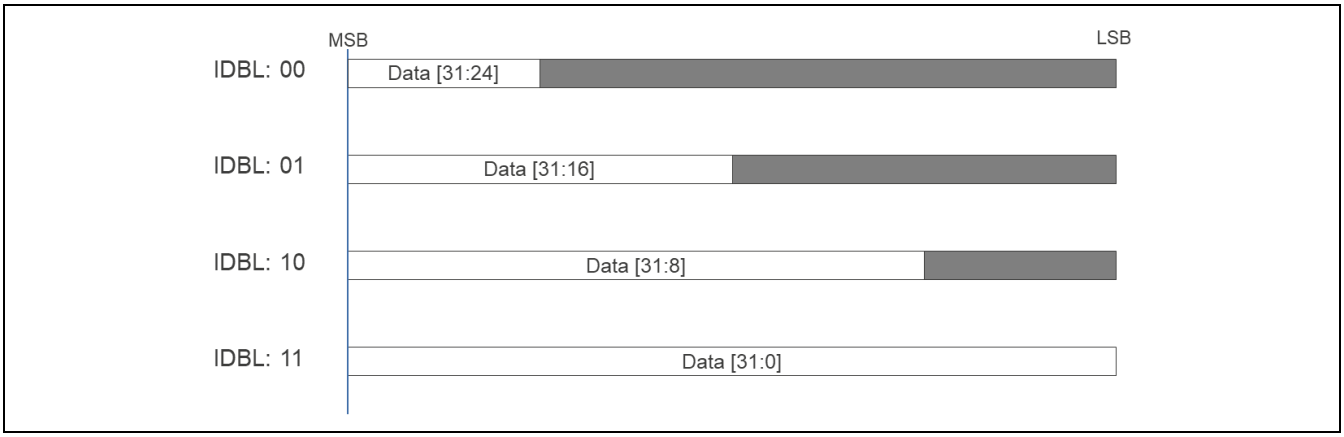



**Figure 55.3 Specifications of each bit (1)**

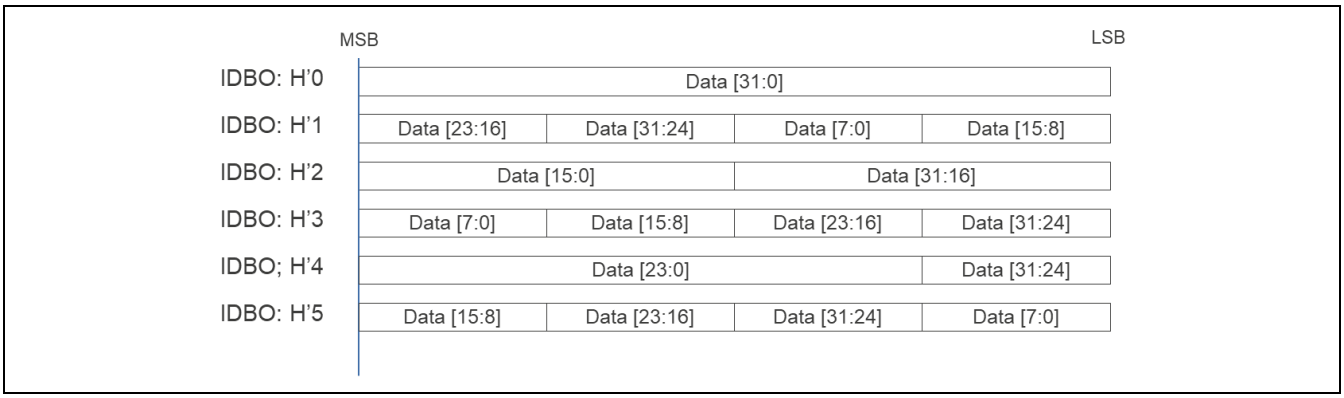



**Figure 55.4 Specifications of each bit (2)**

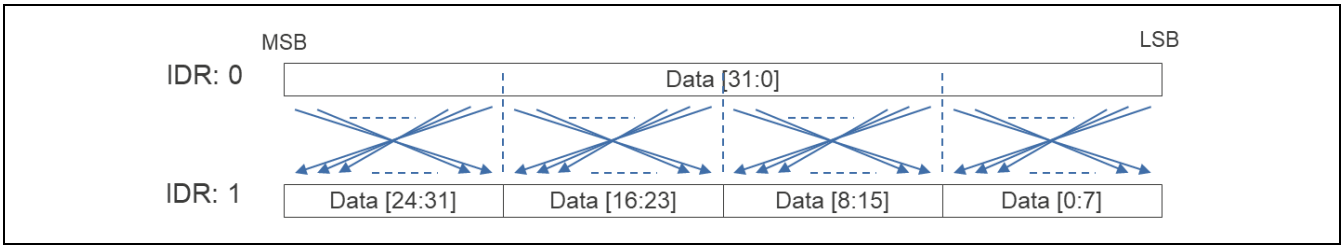



**Figure 55.5 Specifications of each bit (3)**

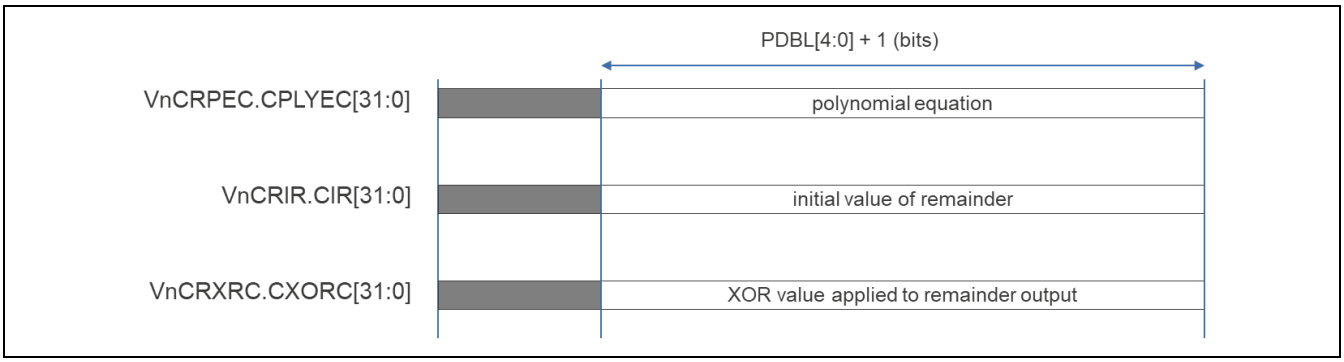



**Figure 55.6 Specifications of each bit (4)**

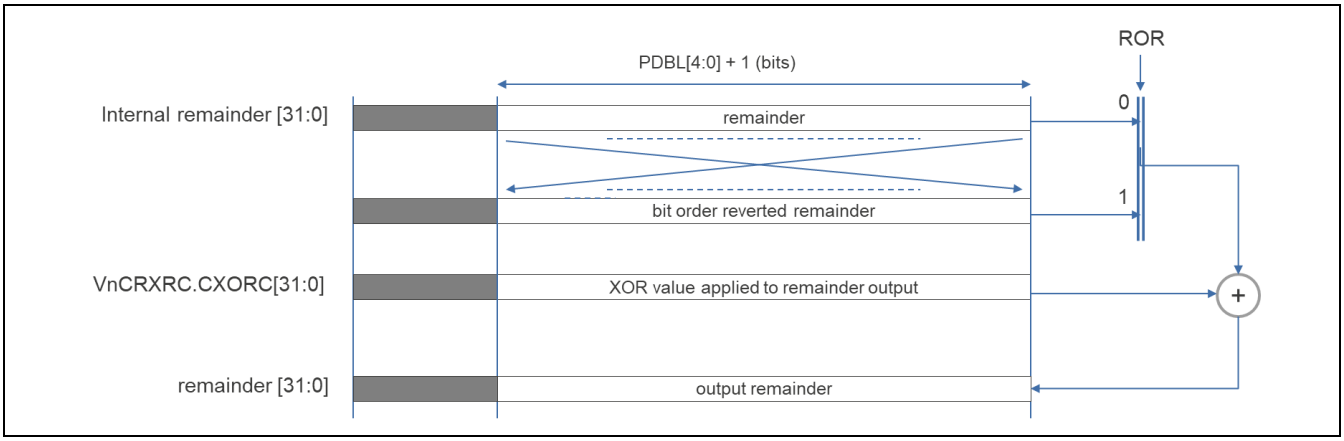



**Figure 55.7 Specifications of each bit (5)**

## 55.3 Operation

### 55.3.1 Input Interface

The following tables show the interface and data format supported in each product and its channels.

**Table 55.6 Video Channels and Supported Interfaces**

| | | | | Channel 0 and 8 | Channel 1 to 7, 9 to 15 |
|---|---|---|---|---|---|
| Input interface | Channel Selector | Embedded 8-bit non-Image Data | | Supported | Supported |
| | | YUV420 8-bit | | Supported | Supported |
| | | YUV420 10-bit | | Supported | Supported |
| | | YUV422 8-bit | | Supported | Supported |
| | | YUV422 10-bit | | Supported | Supported |
| | | YUV422, Y 12-bit/UV 8-bit | | Supported | Supported |
| | | RGB565 | | Supported | Supported |
| | | RGB888 | | Supported | Supported |
| | | RAW8 | | Supported | Supported |
| | | RAW10 | | Supported | Supported |
| | | RAW12 | | Supported | Supported |
| | | RAW14 | | Supported | Supported |
| | | RAW16 | | Supported | Supported |
| | | RAW20 | | Supported | Supported |
| | | RAW24 | | Supported | Supported |
| | | User Defined 8-bit Data | | Supported | Supported |
| Output interface | RGB output | RGB-565 | 16 bits | Supported | Supported |
| | | ARGB-1555 | 16 bits | Supported | Supported |
| | | RGB-888 | 32 bits/24 bits | Supported | Supported |
| | | ARGB-8888 | 32 bits | Supported | Supported |
| | YCbCr output | YCbCr-422 multiplexed | 8 bits | Supported | Supported |
| | | | 10bits | Supported | Supported |
| | | YCbCr-420 separated Y/CbCr | Y: 8 bits CbCr: 8 bits | Supported | Supported |
| | | | Y: 10 bits CbCr: 8 bits | Supported | Supported |
| | | | Y: 12 bits CbCr: 8 bits | Supported | Supported |
| | | YCbCr-422 separated Y/CbCr | Y: 8 bits CbCr: 8 bits | Supported | Supported |
| | | | Y: 10 bits CbCr: 8 bits | Supported | Supported |
| | | | Y: 10 bits CbCr: 10 bits | Supported | Supported |
| | | | Y: 12 bits CbCr: 8 bits | Supported | Supported |
| | RAW output | RAW8 | 8 bits | Supported | Supported |
| | | RAW10 | 10 bits | Supported | Supported |
| | | RAW12 | 12 bits | Supported | Supported |

| | | | Channel 0 and 8 | Channel 1 to 7, 9 to 15 |
|---|---|---|---|---|
| | RAW14 | 14 bits | Supported | Supported |
| | RAW16 | 16 bits | Supported | Supported |
| | RAW20 | 20 bits | Supported | Supported |
| | RAW24 | 24 bits | Supported | Supported |
| Non image data | Embedded 8-bit non-Image Data | 8 bits | Supported | Supported |
| | User Defined 8-bit Data | 8 bits | Supported | Supported |

### 55.3.2 Capture Mode

With the VIN, either of the single frame capture mode or continuous frame capture mode can be selected.

Specifying the capture field in IM bits of the main control (VnMC) register and then setting the SC bit in the frame capture (VnFC) register to 1 provides single frame capture mode. In this mode, the current frame is captured when the SC bit write timing (current scanline position) is smaller than the value of the start line pre-clip (VnSLPrC) register, or the next frame is captured in other cases. The capture data is transferred to the memory address that is set in the memory base 1 (VnMB1) register. For using single frame capture mode, necessary procedure of register setting is as followings.

1. Set 1 to the module enable (ME) bit in the main control register (VnMC).
2. Set 1 to the single frame capture mode (SC) bit in the frame capture register (VnFC).

Note: When using this setting, don't set 2'b01 to the interlace mode (IM) bits in the main control register (VnMC).

Specifying the capture field in IM bits of VnMC and then setting the CC bit of VnFC to 1 provides continuous frame capture mode, in which capture data is sequentially transferred to the addresses that are set in MB1 to MB3. In this case, the latest captured frame ID is shown in the FBS bits in the module status (VnMS) register. For using continuous frame capture mode, necessary procedure of register setting is as followings.

1. Set 1 to the module enable (ME) bit in the main control register (VnMC).
2. Set 1 to the continuous frame capture mode (CC) bit in the frame capture register (VnFC).

Note: Don't set 1 to the single frame capture mode (SC) bit and the continuous frame capture mode (CC) bit at the same time.

When the IM bits in VnMC are set to the full interlace mode, data in the capture start field is stored in every other line in the memory as the odd field (field 1) data and then data in the next field is stored as the even field (field 2) between the written lines so that the top and bottom field lines alternate with each other for interlace composition. The capture start field (top field) can be changed through the FOC bit in VnMC.

The following is a schematic diagram of capturing in full interlace mode when the odd field (field 1) is selected as the top field, the memory width is set to H'200, and VnMB1 is set to H'0000.

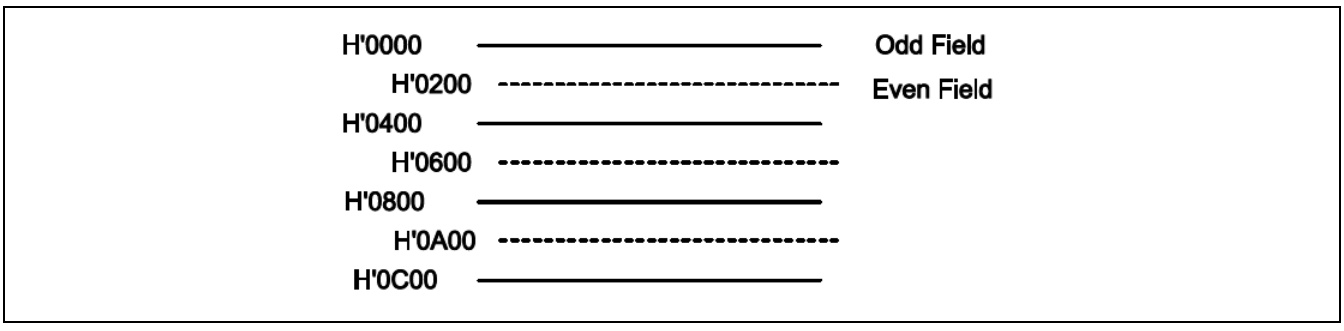



**Figure 55.8 Example of Capturing Full Interlace**

The VIN starts capture operation after detecting a frame signal switching. When capturing progressive data, in which the field signal does not change, use the internal field signal generation function to toggle the internal field signal or use multi frame capture mode. For details, see VnDMR2 register description.

### 55.3.3 Size Clipping

Image data that has been captured is pre-clipped according to the settings of the following registers: start line pre-clip (VnSLPrC), end line pre-clip (VnELPrC), start pixel pre-clip (VnSPPrC), and end pixel pre-clip (VnEPPrC).

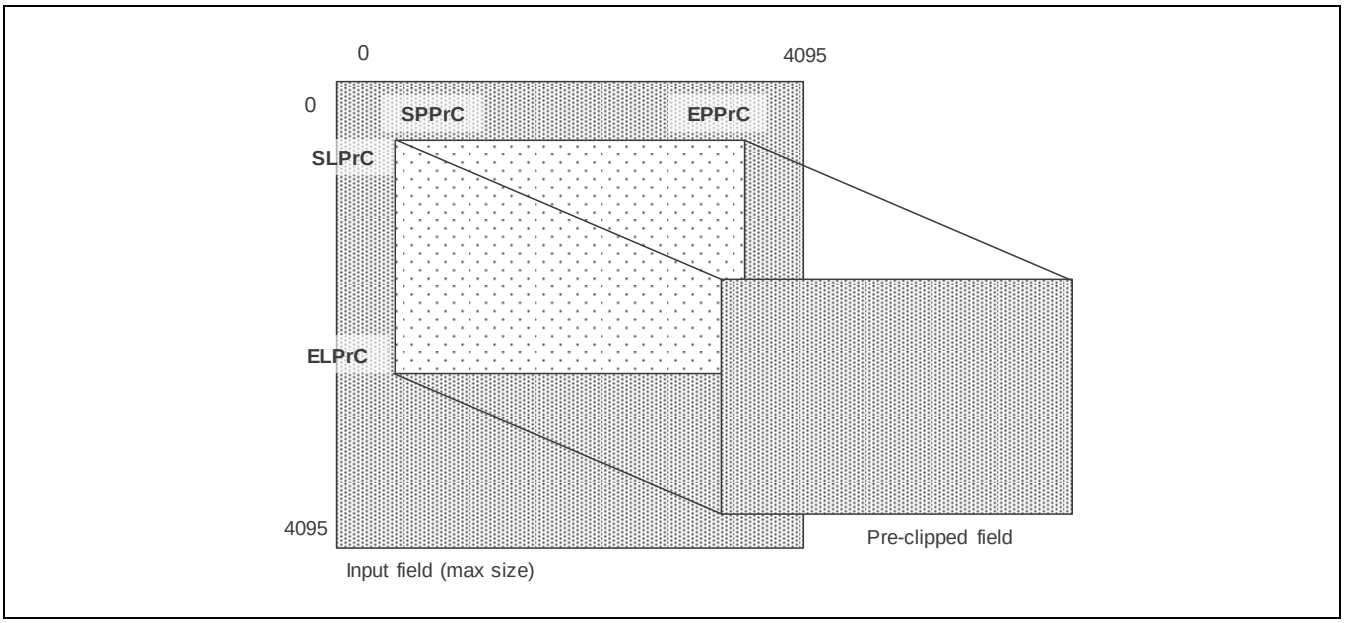



**Figure 55.9 Example of Clipping**

For all the post-clipped lines, the lengths of individual lines written into the memory are defined by the image stride (VnIS) register. The setting can be larger than the post-clipped frame width but cannot be smaller than the width. The VnIS register must be filled with a value larger than the horizontal post-clipping width. The input field in the above figure shows an effective image area from the video decoder; the VIN does not allow anything exceeding the image area to be captured.

Note: Each of the following registers specifies a distance from the starting point in the effective image area: start line pre-clip (VnSLPrC), end line pre-clip (VnELPrC), start pixel pre-clip (VnSPPrC), and end pixel pre-clip (VnEPPrC).

### 55.3.4 Internal Field Signal Generation

As the video input module controls capture of data in interlaced mode, correct capture control is not achieved if the external field signal level does not change.

Through the internal field signal generation function, the VIN can control the capture field signal even when the input field signal does not change, such as during progressive data capturing. The following settings can be made for the internal field generation function the FTEV in the data mode register 2 (VnDMR2).

- VSYNC field toggle mode (FTEV = 1 in VnDMR2)
  When this setting is made, the VSYNC field toggle mode is entered for capture field signal control if the input field signal does not change for the VSYNC cycles specified by the VLV bits in VnDMR2. The toggle mode is canceled when a change in the external field signal level is detected (the capture operation is controlled according to the input field signal).

Notes: Immediately after cancellation of the toggle mode, capture control is skipped for one VSYNC cycle in some cases depending on the input field signal state.

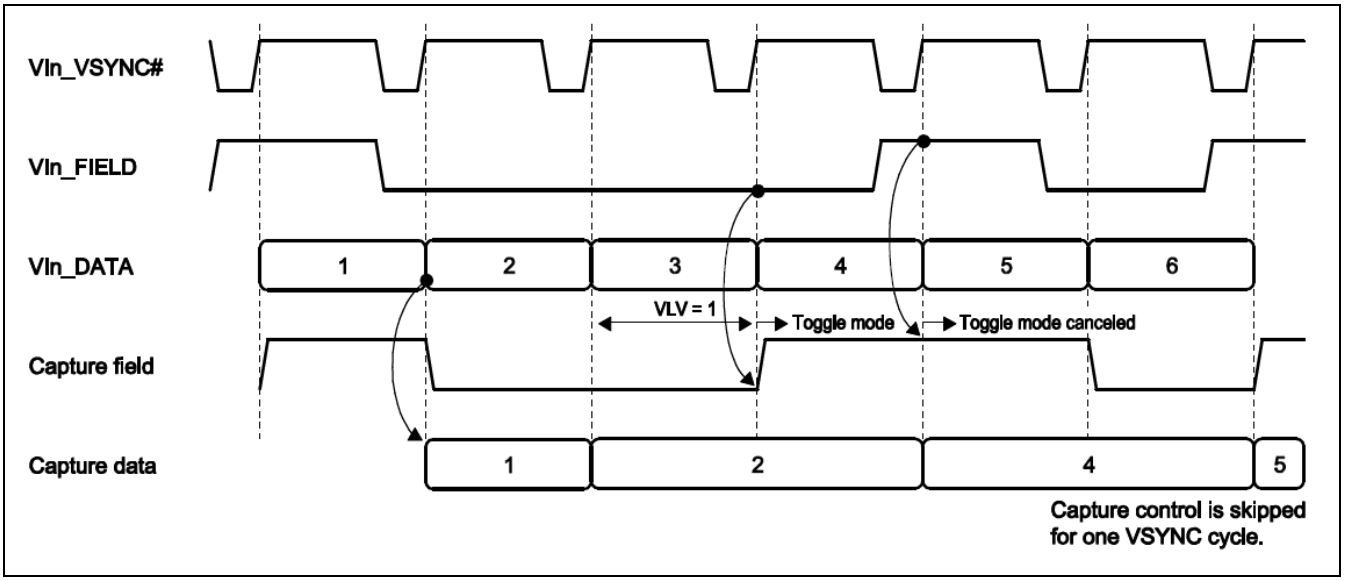



**Figure 55.10 Overview and Notes of VSYNC Field Toggle Mode**

### 55.3.5 Write Line Counter and Writing Frame Buffer Status

When the WLCEN bit in the VnMC is set 1, the write line counter and writing frame buffer status is available.

The initial value of write line counter is 0. When all data of line was stored to memory, the write line counter is counted by 1. When all data of frame was stored to memory, the write line counter is reset to 0. The write line counter is shown as the WLC[13:0] bits in the VnWLC. The maximum value of WLC[13:0] bits are (ELPrC – SLPrC + 1).

The initial value of writing frame buffer status is B'11. When all data of frame was stored to memory, the writing frame buffer status is updated. Each status shows the below.

B'00: The writing frame buffer has the base address defined by the memory base 1 register.

B'01: The writing frame buffer has the base address defined by the memory base 2 register.

B'10: The writing frame buffer has the base address defined by the memory base 3 register.

B'11: There is no writing frame buffer.

The writing frame buffer status updates as the below order.

B'11 (Initial) -> B'00 -> B'01 -> B'10 -> B'00 -> B'01 -> …

The writing frame buffer status is shown as the WFBS[1:0] bits in the VnWLC.

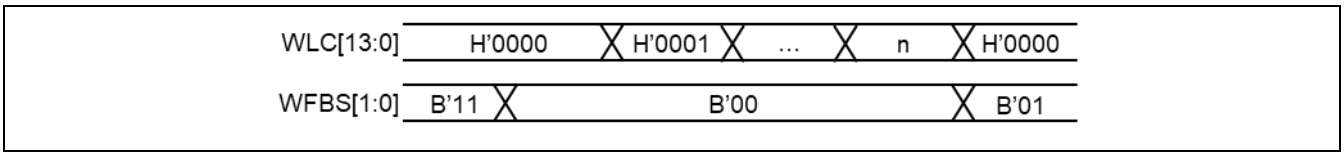



**Figure 55.11 Transition of write line counter and writing frame buffer status**

The smallest write line counter and smallest writing frame buffer status are also managed as SWLC[13:0] bits and SWFBS[1:0] bits in the VnWLC respectively. The SWLC[13:0] bits and SWFBS[1:0] bits in the VnWLC are available when channel combinations are channel 0 to 1, 0 to 2, 0 to 3, 0 to 4, 0 to 5, 0 to 6, and 0 to 7.

### 55.3.6 Multi Frame Capture Function

When the MFC bit in the VnDMR2 is set 1, the multiple frames data are stored to one frame buffer. The number of frames to be stored is controlled by the VLV[3:0] bits in the VnDMR2. The number of frames to be stored is (VLV[3:0] + 1). The number of frames can be specified from 1 (VLV[3:0] is H'0) to 16 (VLV[3:0] is H'F).

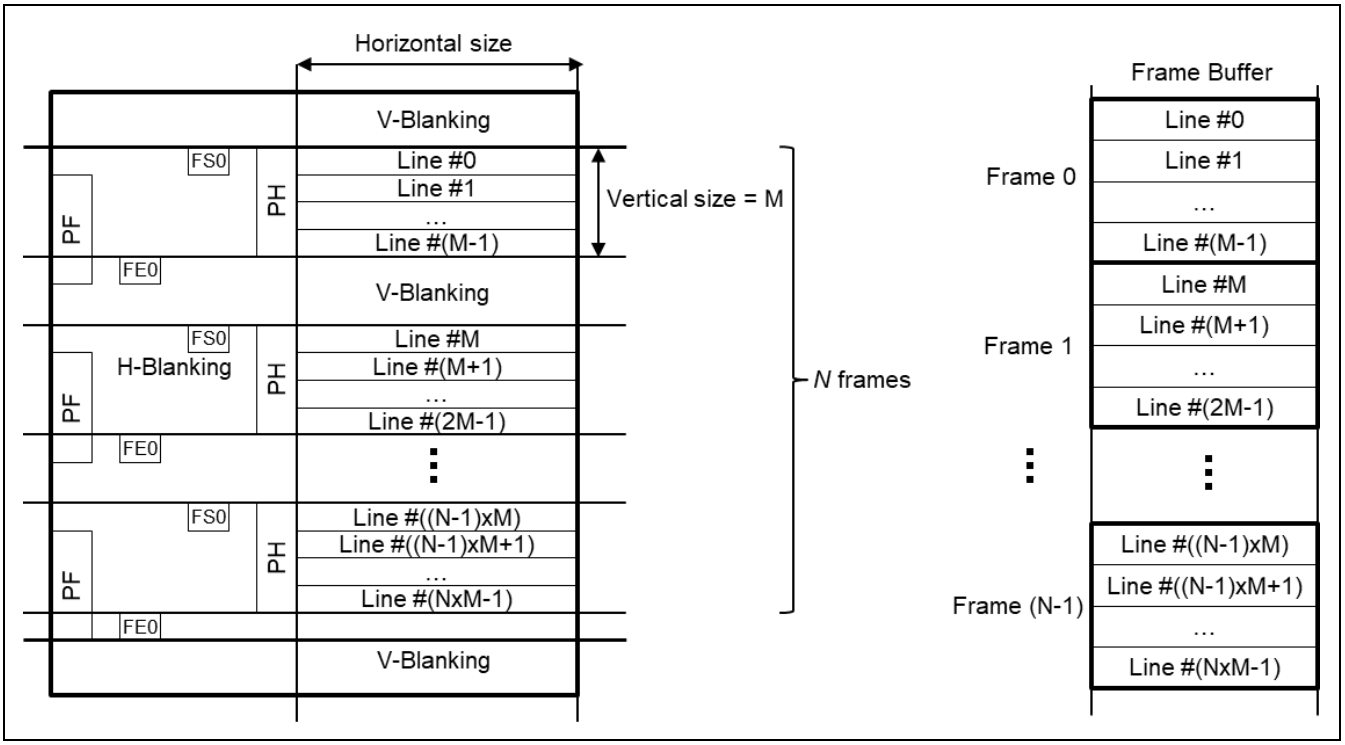



**Figure 55.12 Multi frame capture function**

When the multi frame capture function is used, there are below limitations.

- It is necessary that the start setting of video input module is finished before inputting image data
- The interlace image inputting is not available
- It is prohibited that the MFC and FTEV bits in the VnDMR2 are set to 1 at the same time

### 55.3.7 Wake Up Signals Generation

The video input module can output wake up signals to IMR and VSPX module. It is necessary that the WLCEN bit in the VnMC is set 1 to use this function.

Each video channel in the video input module have a wake up signal and contolled by the WUCH bit in the VnIMWUCH. A common wake up signal is connected to IMR and VSPX. The IMR and VSPX select which channel's wake up signal to use. When 1 is set to Wake-up Notification Interrupt Enable (VnIE.WUE), Wake-up Notification Interrupt Status (VnINTS.WUS) is set to 1 at the timing of wake up signal asserted.

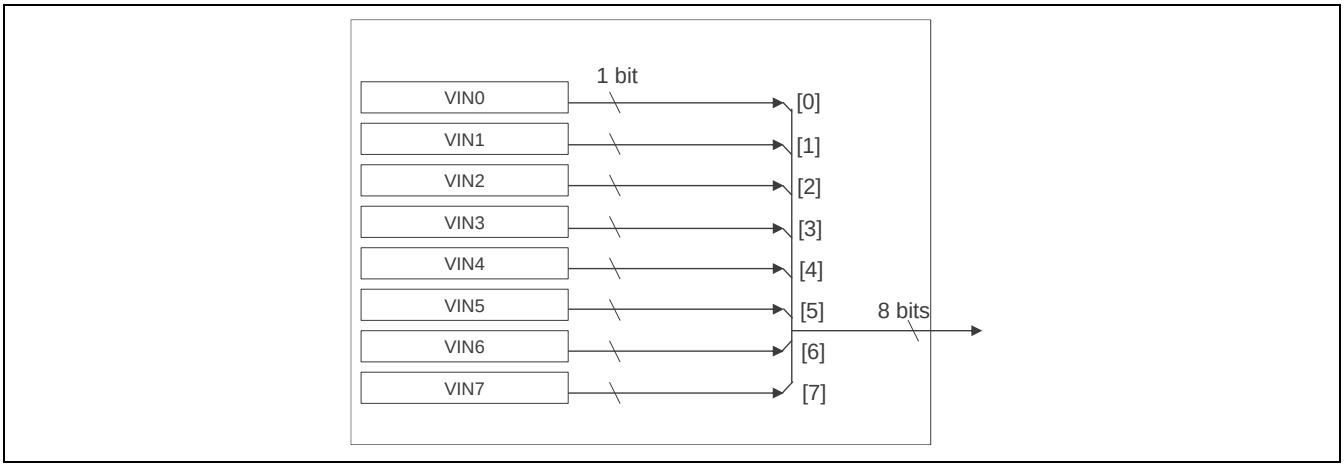



**Figure 55.13 Configuration of wake up signals**

The wake up signal is output at the below timing.

- After line data of specified vertical position (up to 2 positions) was stored to memory.
- After all data of frame was stored to memory.

The vertical position can be specified by the WUL1[13:0] and WUL2[13:0] bits in the VnIMWUL. In addition, the wake up signals of each timing can be controlled to output or not by WUE1, WUE2 and WUE3 bits in the VnIMWUCH respectively.

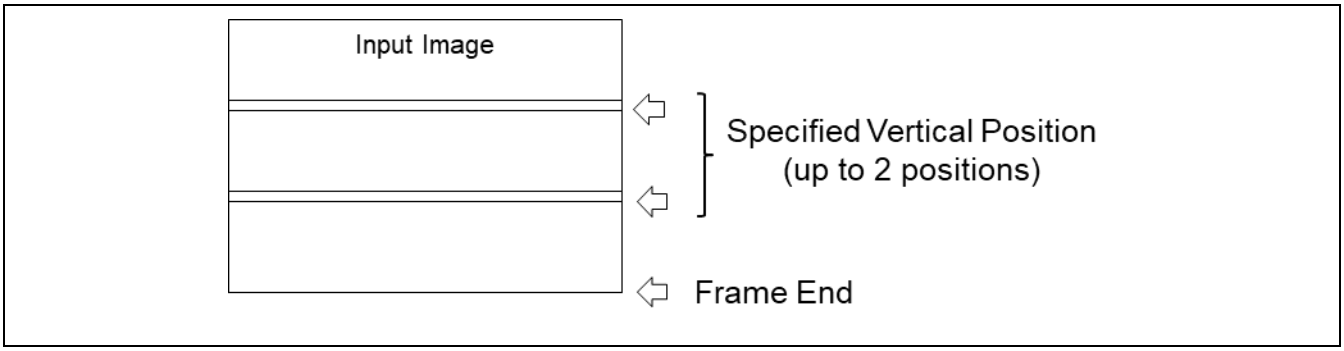



**Figure 55.14 Wake up signals output timing**

### 55.3.8 Output Data Format

The VIN can output image data in the following formats. The figures in this section assume that data is stored in unified memory in little endian.

**(1) YC: YcbCr-422, 8 Bits**

The 8-bit YUV image data in the YC (YcbCr) = 4:2:2 format is shown below. YC data can be switched between the UYVY format and YUYV format through the BPSM bit in VnDMR.

- BPSM = 0 in VnDMR: UYVY format

8-bit YcbCr-422 data (UYVY format)

| | | | | |
|---|---|---|---|---|
| D63 to D48 | 63 | 56 | 55 | 48 |
| Image data 3 and 4 | Y3[7:0] | | Cr2[7:0] | |
| D47 to D32 | 47 | 40 | 39 | 32 |
| Image data 3 and 4 | Y2[7:0] | | Cb2[7:0] | |
| D31 to D16 | 31 | 24 | 23 | 16 |
| Image data 1 and 2 | Y1[7:0] | | Cr0[7:0] | |
| D15 to D0 | 15 | 8 | 7 | 0 |
| Image data 1 and 2 | Y0[7:0] | | Cb0[7:0] | |

- BPSM = 1 in VnDMR: YUYV format

8-bit YcbCr-422 data (YUYV format)

| | | | | |
|---|---|---|---|---|
| D63 to D48 | 63 | 56 | 55 | 48 |
| Image data 3 and 4 | Cr2[7:0] | | Y3[7:0] | |
| D47 to D32 | 47 | 40 | 39 | 32 |
| Image data 3 and 4 | Cb2[7:0] | | Y2[7:0] | |
| D31 to D16 | 31 | 24 | 23 | 16 |
| Image data 1 and 2 | Cr0[7:0] | | Y1[7:0] | |
| D15 to D0 | 15 | 8 | 7 | 0 |
| Image data 1 and 2 | Cb0[7:0] | | Y0[7:0] | |

### (2) YC: YcbCr-422, 10 Bits

The 10-bit YUV image data in the YC (YcbCr) = 4:2:2 format is shown below.

Notes: 1. This format needs to set “1” to VnDMR/YC_THR.
2. A word address is output for YCbCr data according to the VnIS register setting.

10-bit YcbCr-422 data (UYVY format)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| D127 to D112 | 127 | | | | | 122 | 121 | 112 |
| Image data 8 | * | * | * | * | * | * | Y3[9:0] | |
| D111 to D96 | 111 | | | | | 106 | 105 | 96 |
| Image data 7 | * | * | * | * | * | * | Cr2[9:0] | |
| D95 to D80 | 95 | | | | | 90 | 89 | 80 |
| Image data 6 | * | * | * | * | * | * | Y2[9:0] | |
| D79 to D64 | 79 | | | | | 74 | 73 | 64 |
| Image data 5 | * | * | * | * | * | * | Cb2[9:0] | |
| D63 to D48 | 63 | | | | | 58 | 57 | 48 |
| Image data 4 | * | * | * | * | * | * | Y1[9:0] | |
| D47 to D32 | 47 | | | | | 42 | 41 | 32 |
| Image data 3 | * | * | * | * | * | * | Cr0[9:0] | |
| D31 to D16 | 31 | | | | | 26 | 25 | 16 |
| Image data 2 | * | * | * | * | * | * | Y0[9:0] | |
| D15 to D0 | 15 | | | | | 10 | 9 | 0 |
| Image data 1 | * | * | * | * | * | * | Cb0[9:0] | |

Note: * This bit is transferred to memory as 0 or sign data (Yn[9], Cbn[9], Crn[9]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

### (3) YC: YC Separation YcbCr-422, 8 Bits

This is 8-bit YUV image data in the YC separated YC (YcbCr) = 4:2:2 format. UV data is supported by the NV16 format only.

If the YMODE[2:0] bits in the data mode register (VnDMR) are set to B'000 or B'001, setting the DTMD[1:0] bits in VnDMR to B'10 changes the format to 4:2:2 and transfers the UV data to an address which is specified by the addition of the value set in the UV address offset register (VnUVAOF) to the Memory Base. If the YMODE[2:0] bits in VnDMR are set to B'001, only Y data can be transferred and UV data cannot be transferred.

Y data

| D31 to D16 | 31 – 24 | 23 – 16 |
|---|---|---|
| Image data 3 and 4 | Y3[7:0] | Y2[7:0] |
| D15 to D0 | 15 – 8 | 7 – 0 |
| Image data 1 and 2 | Y1[7:0] | Y0[7:0] |

Cb, Cr data

| D31 to D16 | 31 – 24 | 23 – 16 |
|---|---|---|
| Image data 3 and 4 | Cr2[7:0] | Cb2[7:0] |
| D15 to D0 | 15 – 8 | 7 – 0 |
| Image data 1 and 2 | Cr0[7:0] | Cb0[7:0] |

### (4) YC: YC Separation YcbCr-420, 8 Bits

This is 8-bit YUV image data in the YC separated YC (YcbCr) = 4:2:0 format. UV data is supported by the NV12 format only.

If the YMODE[2:0] bits in the data mode register (VnDMR) are set to B'000, setting the DTMD[1:0] bits in VnDMR to B'11 changes the format to 4:2:0 and transfers the UV data to an address which is specified by the addition of the value set in the UV address offset register (VnUVAOF) to the Memory Base. The UV data are transferred to memory every even line.

Y data

| D31 to D16 | 31 to 24 | 23 to 16 |
|---|---|---|
| Image data 3 and 4 | Y3[7:0] | Y2[7:0] |
| D15 to D0 | 15 to 8 | 7 to 0 |
| Image data 1 and 2 | Y1[7:0] | Y0[7:0] |

Cb, Cr data

| D31 to D16 | 31 to 24 | 23 to 16 |
|---|---|---|
| Image data 3 and 4 | Cr02[7:0] | Cb02[7:0] |
| D15 to D0 | 15 to 8 | 7 to 0 |
| Image data 1 and 2 | Cr00[7:0] | Cb00[7:0] |

### (5) YC: YC Separation YcbCr-422, Y(10 Bits) / C(10 Bits)

This is 10-bit YUV image data in the YC separated YC (YcbCr) = 4:2:2 format. UV data is supported by the NV12 format only.

If the YMODE[2:0] bits in the data mode register (VnDMR) are set to B'000 or B'001, setting the DTMD[1:0] bits in VnDMR to B'10 changes the format to 4:2:2 and transfers the UV data to an address which is specified by the addition of the value set in the UV address offset register (VnUVAOF) to the Memory Base. If the YMODE[2:0] bits in VnDMR are set to B'001, only Y data can be transferred and UV data cannot be transferred.

Notes: 1. This format needs to set "1" to VnDMR/YC_THR.
2. A word address is output for YCbCr data according to the VnIS register setting.

Y data (10 bits)

| | 63 | | | | | 58 | 57 | 48 |
|---|---|---|---|---|---|---|---|---|
| D63 to D48 | | | | | | | | |
| Image data 4 | * | * | * | * | * | * | Y3[9:0] | |
| D47 to D32 | 47 | | | | | 42 | 41 | 32 |
| Image data 3 | * | * | * | * | * | * | Y2[9:0] | |
| D31 to D16 | 31 | | | | | 26 | 25 | 16 |
| Image data 2 | * | * | * | * | * | * | Y1[9:0] | |
| D15 to D0 | 15 | | | | | 10 | 9 | 0 |
| Image data 1 | * | * | * | * | * | * | Y0[9:0] | |

Note: * This bit is transferred to memory as 0 or sign data (Yn[9]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

Cb, Cr data (10 bits)

| | 63 | | | | | 58 | 57 | 48 |
|---|---|---|---|---|---|---|---|---|
| D63 to D48 | | | | | | | | |
| Image data 4 | * | * | * | * | * | * | Cr2[9:0] | |
| D47 to D32 | 47 | | | | | 42 | 41 | 32 |
| Image data 3 | * | * | * | * | * | * | Cb2[9:0] | |
| D31 to D16 | 31 | | | | | 26 | 25 | 16 |
| Image data 2 | * | * | * | * | * | * | Cr0[9:0] | |
| D15 to D0 | 15 | | | | | 10 | 9 | 0 |
| Image data 1 | * | * | * | * | * | * | Cb0[9:0] | |

Note: * This bit is transferred to memory as 0 or sign data (Cbn[9], Crn[9]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

### (6) YC: YC Separation YcbCr-422, 10 Bits

This is 10-bit YUV image data in the YC separated YC (YcbCr) = 4:2:2 format. UV data is supported by the NV16 format only.

If the YMODE[2:0] bits in the data mode register (VnDMR) are set to B'010 or B'011, setting the DTMD[1:0] bits in VnDMR to B'10 converts 10-bit Y data to 16-bit and changes the format to 4:2:2. UV data is transferred to the address obtained by adding the value set in the UV address offset register (VnUVAOF) to Memory Base. If the YMODE[2:0] bits in VnDMR are set to B'011, only Y data can be transferred and UV data cannot be transferred.

Notes: 1. A word address is output for Y data according to the VnIS register setting.
2. A byte address is output for CbCr data according to the VnIS register setting.

Y data

| D31 to D16 | 31 | | | | | 26 | 25 | | | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 3 and 4 | * | * | * | * | * | * | Y1[9:0] | | | | | | | | | |
| D15 to D0 | 15 | | | | | 10 | 9 | | | | | | | | | 0 |
| Image data 1 and 2 | * | * | * | * | * | * | Y0[9:0] | | | | | | | | | |

Cb, Cr data

| D31 to D16 | 31 | | | | | | | 24 | 23 | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 3 and 4 | Cr2[7:0] | | | | | | | | Cb2[7:0] | | | | | | | |
| D15 to D0 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
| Image data 1 and 2 | Cr0[7:0] | | | | | | | | Cb0[7:0] | | | | | | | |

Note: * This bit is transferred to memory as 0 or sign data (Yn[9]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

### (7) YC: YC Separation YcbCr-422, 12 Bits

This is 12-bit YUV image data in the YC separated YC (YcbCr) = 4:2:2 format. UV data is supported by the NV16 format only.

If the YMODE[2:0] bits in the data mode register (VnDMR) are set to B'100 or B'101, setting the DTMD[1:0] bits in VnDMR to B'10 converts 12-bit Y data to 16-bit and changes the format to 4:2:2. UV data is transferred to the address obtained by adding the value set in the UV address offset register (VnUVAOF) to Memory Base. If the YMODE[2:0] bits in VnDMR are set to B'101, only Y data can be transferred and UV data cannot be transferred.

Notes: 1. A word address is output for Y data according to the VnIS register setting.
2. A byte address is output for CbCr data according to the VnIS register setting.

Y data

| D31 to D16 | 31 | | | | 28 | 27 | | | | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 3 and 4 | * | * | * | * | Y1[11:0] | | | | | | | | | | | |
| D15 to D0 | 15 | | | | 12 | 11 | | | | | | | | | | 0 |
| Image data 1 and 2 | * | * | * | * | Y0[11:0] | | | | | | | | | | | |

Note: * This bit is transferred to memory as 0 or sign data (Yn[11]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

Cb, Cr data

| D31 to D16 | 31 | | | | | | | 24 | 23 | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 3 and 4 | Cr2[7:0] | | | | | | | | Cb2[7:0] | | | | | | | |
| D15 to D0 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
| Image data 1 and 2 | Cr0[7:0] | | | | | | | | Cb0[7:0] | | | | | | | |

### (8) 16 Bits/Pixel: RGB-565

The RGB levels are expressed through 5 bits for R, 6 bits for G, and 5 bits for B.

16 bits/pixel data (RGB data) format

| D15 to D0 | 15 | | | | | 11 | 10 | | | | | 5 | 4 | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data | R[4:0] | | | | | | G[5:0] | | | | | | B[4:0] | | | | | |

### (9) 16 Bits/Pixel: ARGB-1555

The ARGB levels are expressed through 1 bit for A, 5 bits for R, 5 bits for G, and 5 bits for B. For data conversion to ARGB-1555, the lowest bit of the G data in RGB-565 data is truncated, and the A value specified through the register is added.

Set the DTMD[1:0] bits in the data mode register (VnDMR) to B'01 to specify conversion to ARGB-1555, and specify the A value in the ABIT bit in VnDMR.

16 bits/pixel data (ARGB data) format

| D15 to D0 | 15 | 14 | | | | 10 | 9 | | | | | 5 | 4 | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data | A | R[4:0] | | | | | G[4:0] | | | | | | B[4:0] | | | | | |

### (10) 32 Bits/Pixel: RGB-888

The RGB levels are expressed through 8 bits for R, 8 bits for G, and 8 bits for B. Bits 31 to 24 are fixed to 0.

32 bits/pixel data (RGB data) format

| D31 to D16 | 31 | | | | | | | 24 | 23 | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data | * | * | * | * | * | * | * | * | Rn[7:0] | | | | | | | |
| D15 to D0 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
| Image data | Gn[7:0] | | | | | | | | Bn[7:0] | | | | | | | |

Note: * This bit is transferred to memory as 0 or sign data (Rn[7]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

### (11) 32 Bits/Pixel: ARGB-8888

The ARGB levels are expressed through 8 bits for A, 8 bits for R, 8 bits for G, and 8 bits for B. The A value specified by the A8BIT[7:0] bits in VnDMR is set in bits 31 to 24.

32 bits/pixel data (ARGB data) format

| D31 to D16 | 31 | | | | | | | 24 | 23 | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data | A[7:0] | | | | | | | | R[7:0] | | | | | | | |
| D15 to D0 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
| Image data | G[7:0] | | | | | | | | B[7:0] | | | | | | | |

### (12) RAW: 8 Bits and Embedded 8-Bit Non-Image Data, User Defined 8-bit Data

The 8-bit RAW image data format, Embedded 8-bit non image data and User Defined 8-bit data format are shown below.

Notes: 1. These data are transferred by 2-Byte unit.
2. A word address is output for RAW data and 8-bit non-image data according to the VnIS register setting.

- When even pixels are input

8-bit data format

| D15 to D0 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 1 and 2 | P1[7:0] | | | | | | | | P0[7:0] | | | | | | | |

- When odd pixels are input (8-bit RAW data format)

8-bit data format

| D15 to D0 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 1 | * | * | * | * | * | * | * | * | P0[7:0] | | | | | | | |

Note: * This bit is transferred to memory as 0 or sign data (Pn[7]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

- When odd pixels are input (8-bit non-image data format)

8-bit data format

| D15 to D0 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P0[7:0] | | | | | | | |

### (13) RAW: 10 Bits

The 10-bit RAW image data format is shown below.

Notes: 1. These data are transferred by 2-Byte unit.
2. A word address is output for RAW data according to the VnIS register setting.

10-bit RAW data format

| D31 to D16 | 31 | | | | | | 26 | 25 | | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 2 | * | * | * | * | * | * | P1[9:0] | | | | | | | | | |
| D15 to D0 | 15 | | | | | | 10 | 9 | | | | | | | | 0 |
| Image data 1 | * | * | * | * | * | * | P0[9:0] | | | | | | | | | |

Note: * This bit is transferred to memory as 0 or sign data (Pn[9]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

### (14) RAW: 12 Bits

The 12-bit RAW image data format is shown below.

Notes: 1. These data are transferred by 2-Byte unit.
2. A word address is output for RAW data according to the VnIS register setting.

12-bit RAW data format

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D127 to D112 | 127 | | | | | | | | | | | 124 | 123 | | 112 |
| Image data 8 | * | * | * | * | P7[11:0] | | | | | | | | | | |
| D111 to D96 | 111 | | | | | | | | | | | 108 | 107 | | 96 |
| Image data 7 | * | * | * | * | P6[11:0] | | | | | | | | | | |
| D95 to D80 | 95 | | | | | | | | | | | 92 | 91 | | 80 |
| Image data 6 | * | * | * | * | P5[11:0] | | | | | | | | | | |
| D79 to D64 | 79 | | | | | | | | | | | 76 | 75 | | 64 |
| Image data 5 | * | * | * | * | P4[11:0] | | | | | | | | | | |
| D63 to D48 | 63 | | | | | | | | | | | 60 | 59 | | 48 |
| Image data 4 | * | * | * | * | P3[11:0] | | | | | | | | | | |
| D47 to D32 | 47 | | | | | | | | | | | 44 | 43 | | 32 |
| Image data 3 | * | * | * | * | P2[11:0] | | | | | | | | | | |
| D31 to D16 | 31 | | | | | | | | | | | 28 | 27 | | 16 |
| Image data 2 | * | * | * | * | P1[11:0] | | | | | | | | | | |
| D15 to D0 | 15 | | | | | | | | | | | 12 | 11 | | 0 |
| Image data 1 | * | * | * | * | P0[11:0] | | | | | | | | | | |

Note: * This bit is transferred to memory as 0 or sign data (Pn[11]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

**(15) RAW: 14 Bits**

The 14-bit RAW image data format is shown below.

Notes: 1. These data are transferred by 2-Byte unit.
2. A word address is output for RAW data according to the VnIS register setting.

14-bit RAW data format

| D31 to D16 | 31 | 30 | 29 ... 16 |
|---|---|---|---|
| Image data 2 | * | * | P1[13:0] |
| **D15 to D0** | **15** | **14** | **13 ... 0** |
| Image data 1 | * | * | P0[13:0] |

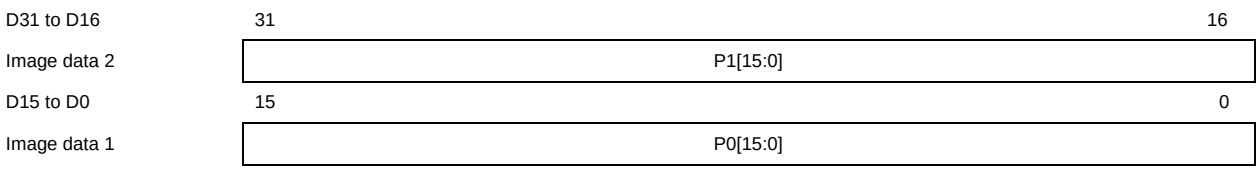
Note: * This bit is transferred to memory as 0 or sign data (Pn[13]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

### (16) RAW: 16 Bits

The 16-bit RAW image data format is shown below.

Notes: 1. These data are transferred by 2-Byte unit.
2. A word address is output for RAW data according to the VnIS register setting.

16-bit RAW data format



### (17) RAW: 20 Bits

The 20-bit RAW image data format is shown below.

Notes: 1. These data are transferred by 4-Byte unit.
2. A word address is output for RAW data according to the VnIS register setting.

24-bit RAW data format

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D31 to D16 | 63 | | | | | | | | | | | 52 | | 48 |
| Image data | * | * | * | * | * | * | * | * | * | * | * | * | P1[19:16] | |
| D15 to D0 | 47 | | | | | | | | | | | | | 32 |
| Image data | P1[15:0] | | | | | | | | | | | | | |
| D31 to D16 | 31 | | | | | | | | | | | 20 | 19 | 16 |
| Image data | * | * | * | * | * | * | * | * | * | * | * | * | P0[19:16] | |
| D15 to D0 | 15 | | | | | | | | | | | | | 0 |
| Image data | P0[15:0] | | | | | | | | | | | | | |

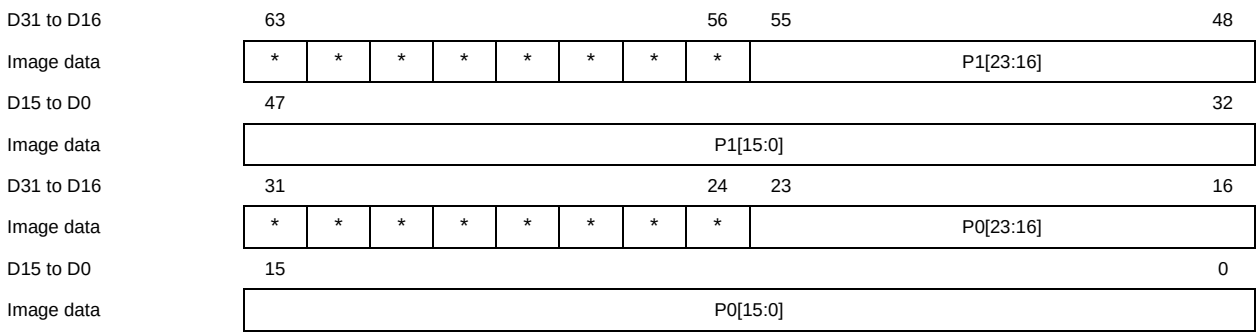
Note: * This bit is transferred to memory as 0 or sign data (Pn[19]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

### (18) RAW: 24 Bits

The 24-bit RAW image data format is shown below.

Notes: 1. These data are transferred by 4-Byte unit.
2. A word address is output for RAW data according to the VnIS register setting.

24-bit RAW data format




Note: * This bit is transferred to memory as 0 or sign data (Pn[23]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

**(19) YC: YC Separation YcbCr-420, 10 Bits**

This is 10-bit YUV image data in the YC separated YC (YcbCr) = 4:2:0 format. UV data is supported by the NV16 format only.

If the YMODE[2:0] bits in the data mode register (VnDMR) are set to B'010, setting the DTMD[1:0] bits in VnDMR to B'11 converts 10-bit Y data to 16-bit and changes the format to 4:2:0. UV data is transferred to the address obtained by adding the value set in the UV address offset register (VnUVAOF) to Memory Base. The UV data are transferred to memory every even line.

Notes: 1. A word address is output for Y data according to the VnIS register setting.
2. A byte address is output for CbCr data according to the VnIS register setting.

Y data

| D31 to D16 | 31 | | | | | 26 | 25 | | | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 3 and 4 | * | * | * | * | * | * | Y1[9:0] | | | | | | | | | |
| D15 to D0 | 15 | | | | | 10 | 9 | | | | | | | | | 0 |
| Image data 1 and 2 | * | * | * | * | * | * | Y0[9:0] | | | | | | | | | |

Cb, Cr data

| D31 to D16 | 31 | | | | | | | 24 | 23 | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image data 3 and 4 | Cr2[7:0] | | | | | | | | Cb2[7:0] | | | | | | | |
| D15 to D0 | 15 | | | | | | | 8 | 7 | | | | | | | 0 |
| Image data 1 and 2 | Cr0[7:0] | | | | | | | | Cb0[7:0] | | | | | | | |

Note: * This bit is transferred to memory as 0 or sign data (Yn[9]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

**(20) YC: YC Separation YcbCr-420, 12 Bits**

This is 12-bit YUV image data in the YC separated YC (YcbCr) = 4:2:0 format. UV data is supported by the NV16 format only.

If the YMODE[2:0] bits in the data mode register (VnDMR) are set to B'100, setting the DTMD[1:0] bits in VnDMR to B'11 converts 12-bit Y data to 16-bit and changes the format to 4:2:0. UV data is transferred to the address obtained by adding the value set in the UV address offset register (VnUVAOF) to Memory Base. The UV data are transferred to memory every even line.

Notes: 1. A word address is output for Y data according to the VnIS register setting.
2. A byte address is output for CbCr data according to the VnIS register setting.

Y data

| D31 to D16 | 31 | | | 28 | 27 | 16 |
|---|---|---|---|---|---|---|
| Image data 3 and 4 | * | * | * | * | Y1[11:0] | |
| D15 to D0 | 15 | | | 12 | 11 | 0 |
| Image data 1 and 2 | * | * | * | * | Y0[11:0] | |

Note: * This bit is transferred to memory as 0 or sign data (Yn[11]).
When the FDS bit in the VnDMR is set to 1, this bit is filled with sign data.

Cb, Cr data

| D31 to D16 | 31 | 24 | 23 | 16 |
|---|---|---|---|---|
| Image data 3 and 4 | Cr2[7:0] | | Cb2[7:0] | |
| D15 to D0 | 15 | 8 | 7 | 0 |
| Image data 1 and 2 | Cr0[7:0] | | Cb0[7:0] | |

### (21) RAW: 8 Bits (16 Bits/Pixel) Data

The 8-bit RAW (16 bits/pixel) image data format is shown below.

Notes: 1. These data are transferred by 2-Byte unit.
2. A word address is output for RAW data and 8-bit non-image data according to the VnIS register setting.

8-bit data format

| D15 to D0 | 15 | | | | | | | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Image data 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P0[7:0] | |

### 55.3.9 Endian Conversion

The VIN stores captured data in memory in little endian with the initial settings. Set the EN bit in the main control register (VnMC) to 1 to convert data into big endian before storing in memory.

Endian conversion in word units is controlled through the EN bit in VnMC, and swapping in byte units is controlled through the BPSM bit in the data mode register (VnDMR). For conversion to big endian, specify the BPSM bit as shown in the following table according to the data format specified through the DTMD bits in VnDMR.

**Table 55.7 Endian Conversion Unit**

| Data Format | DTMD[1:0] in VnDMR | EXRGB in VnDMR | BPSM in VnDMR | Endian Conversion Unit |
|---|---|---|---|---|
| RGB-565 | 00 | 0 | 0 | Word units |
| RGB-888 | 00 | 1 | 0 | Longword units |
| YcbCr-422 | 00 | 0 | 1 | Byte units |
| ARGB-1555 | 01 | 0 | 0 | Word units |
| ARGB-8888 | 01 | 1 | 0 | Longword units |
| YC | 10 | 0 | 0 | Byte units |

The following figures show endian conversions in byte, word, and longword units.

Endian conversion in byte units:

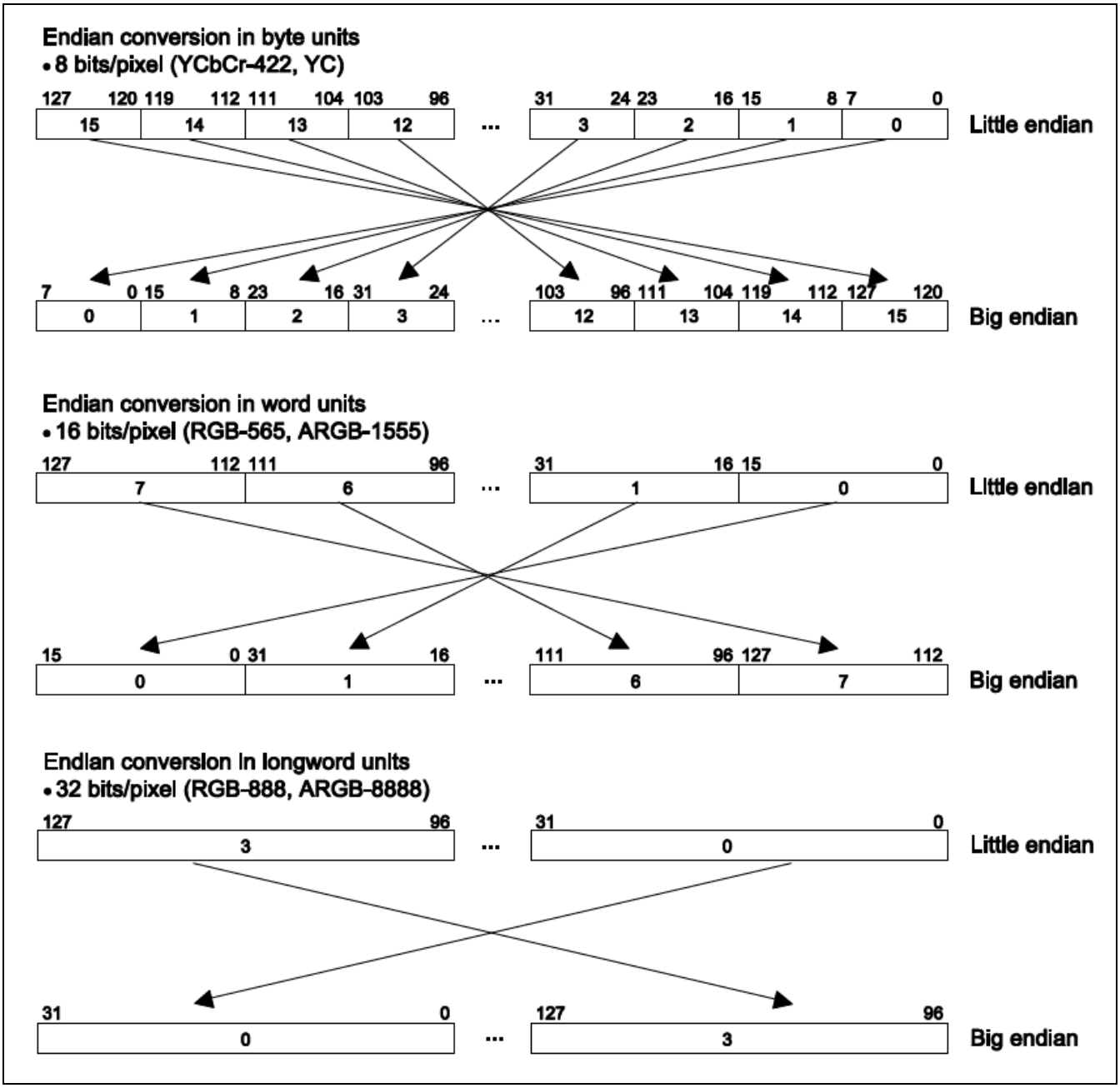



**Figure 55.15 Data Alignment Conversion from Little Endian to Big Endian**

### 55.3.10 Module Standby Mode

The LSI supports module standby mode in which clock supply to the VIN is stopped. The video input module should not be accessed during module standby mode.

### 55.3.11 Transition to Module Standby Mode

1. Execute capture stop procedure described in section 55.3.15 if VIN is in capture operation.
2. Enable Module Stop in the Module Stop Control Register.
3. Check the bit that the target module of MSTPSRn is 1.

For detail of registers of Module Standby, refer to section 9.

### 55.3.12 Cancellation of Module Standby Mode

1. Disable Module Stop in the Module Stop Control Register.
2. Check the bit that the target module of MSTPSRn is 0.
3. Apply Software Reset in the Software Reset Register.
4. Clear Software Reset in the Software Reset Clearing Register.

For detail of registers of Module Standby , refer to section 9.

### 55.3.13 Interrupts

The Video Input Module can generate 100 interrupts. For details, refer to section 15 and 16.

**Table 55.8 Interrupt Assignment**

| Interrupt Name | Cause of interrupt |
|---|---|
| VIN.chX | FIS2 or VFS or VRS or FIS or CES or SIS or EFS or FOS bit in VnINTS |
| errreq_vin_viX_p | * |
| errreq_vin_viX_edc_p | * |
| errreq_vin_vin0_clm_p | * |
| errreq_vin_vin1_clm_p | * |
| errreq_vin_vin2_clm_p | * |
| errreq_vin_vin3_clm_p | * |

Notes: X = 0 to 15
*: This interrupt is functional safety use only.

### 55.3.14 Initialize Procedure

The initialization and capture start procedure of VIN is as followings.

1. Set VUP bit in VnMC to 0.
2. Set other configuration bits in VnMC and keep this bit during the following sequence.
3. Set VUP bit in VnMC to the desired value.
4. Set VIN registers.
5. Set the VSYNC field toggle mode enable (FTEV) bit in the Video n data mode register 2 (VnDMR2) to 1. The VSYNC field toggle mode transition period (VLV[3:0]) bits in VnDMR2 shall be set to 4'h0.
6. Set the module enable (ME) bit in the main control register (VnMC) to 1 to start the video input module.
7. Set the continuous frame capture (CC) bit or the single frame capture (SC) bit in the frame capture register (VnFC) to 1.
8. Set the ISP module registers*1.
9. Set the Link and PHY of CSI-2 module registers*2.
10. Start camera device.
11. Confirm PHY of CSI-2 module starts.

Notes: *1. For detail, refer section 54.
*2. For detail, refer section 53.

### 55.3.15 Capture Stop procedure

The capture stop procedure of VIN is as followings.

1. Clear the module enable (ME) bit in the main control register (VnMC) and the continuous frame capture (CC) bit and the single frame capture (SC) bit in the frame capture register (VnFC) to 0 to stop the video input module.
2. Confirm that the capture active (CA) bit in the module status register (VnMS) is cleared to 0.
3. Reset the ISP module*1.
4. Reset the Link and PHY of CSI-2 module*2.
5. Stop camera device of MIPI-CSI-2 side.

Notes: *1. For detail, refer section 54.
*2. For detail, refer section 53.

### 55.3.16 Capture Restarting procedure

The capture restarting procedure of VIN is same as the initialize procedure described in section 55.3.14.

### 55.3.17 Reset Procedure

There are two reset sequences.

**(1) Power-on Reset**

The VIN is in a module-stopped state after power is turned on. The procedure of module stop release described in section 55.3.12 includes the VIN reset process.

**(2) Reset when VIN gets Hang Up**

If VIN gets hang up such as:

- The capture active (CA) bit in the module status register (VnMS) is not set to 1 after executing the initialize procedure in section 55.3.14 or the capture restarting procedure in section 55.3.16.
- The capture active (CA) bit in the module status register (VnMS) is not cleared to 0 after executing the capture stop procedure described in section 55.3.15.
- The end of frame interrupt status (EFS) bit or field interrupt status (FIS) bit is not set to 1 after more than one V-period of input video sequence.
- Or when it is judged that the VIN is not working,

VIN must be reset as follows.

1. Apply Software Reset in the Software Reset Register.
2. Clear Software Reset in the Software Reset Clearing Register.

For detail of registers of Software Reset, refer to section 9.

## 55.4 Usage Notes

### 55.4.1 Specification

The following shows the specification.

**Table 55.9 Specification**

| Item | Description |
|---|---|
| Limitation on size clipping | The upper limits on size clipping depends on the use case. |
| Capture in progressive mode | Use the internal field signal generation function or multi frame capture mode with an input interface in progressive mode. For details, see VnDMR2 register description. |
| Limitation on register update | If a register is updated during capture, data captured immediately after the register update cannot be guaranteed. To update a register which supports the internal update mode as shown in Table 55.2. VIN Registers, specify an internal update for VIN registers by using the MC/VUP bit. To update other registers, stop capture operation and then update them. |
| Field capture mode image quality | Images of odd-numbered, odd/even-numbered and even-numbered fields, captured by VnMC/IM interlace mode bit settings, contain every other line from the input interlace images. Therefore, note that the horizontal resolution for video display is in units of fields. |
| Full interlace mode image quality | In full interlace composition mode, horizontal stripe noise (such as combing noise) is generated in composite images as fields based on different timelines are combined in memory due to the interlacing method. |
| Interrupt event timing | Interrupt event asserted by this module indicates the time when an interrupt event occurs in VIN, not the time when the transfer of capture data to memory is completed. |
| YC Restrictions on YC separation function | Set the offset register that stores UV data (VnUVAOF) such that the storage areas of Y and UV data are not the same. If the same storage area is set for both data, Y data will be overwritten. |
| Video display operation | This module performs capture control based on the external input synchronization signal. Note that it is not synchronous to the timing of frame update set in the Display Unit and thus video display in sync with the capture frame is disabled. |
| Clip size setting | Pre-Clip size of Odd and Even field is set to the common by VnSLPrC, VnELPrC, VnSPPrC and VnEPPrC. |
| Limitation on interlace input | When the interlace image is input, be sure to supply the field signal to VIN. When the interlace image is input from the MIPI CSI-2 interface, it is necessary to create the field signal in CSI-2 module. For the detail of the field signal creation in CSI-2, see section 53. |
| Horizontal clipping specification | Refer to the 55.2.6 Video n Start Pixel Pre-Clip Register (VnSPPrC) and 55.2.7 Video n End Pixel Pre-Clip Register (VnEPPrC). |
| Vertical clipping specification | When YcbCr420 format is output, the vertical clipping size shall be set to even lines. |

### 55.4.2 Limitations on Usage

The VIN does not operate correctly in some cases depending on the usage. The following shows cases that require attention.

**Table 55.10 Limitations on Usage**

| Item | Description |
|---|---|
| Image Stride setting | When outputting the NV16 format and NV12 format, set value of 32pixel unit to the image stride (IS) bit in the image stride register (VnIS). |

### 55.4.3 Clock to Use

VIN uses following clocks.

S0D1_VIOφ 800MHz
S0D4_VIOφ 200MHz

## 55.5 Safety Mechanism

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 55.5.1 EDC for VIN

#### 55.5.1.1 Overview

VIN has two kinds of EDC protection function; first is EDC protection for write data FIFO, second is EDC checking data from ISP.

#### 55.5.1.2 Hardware Description

Figure 55.16 shows EDC protection for write data FIFO. FIFO controller generates 16-bit EDC code at the timing of write access to FIFO. Then it concatenates the EDC code and the write data (256 bits) and stores in FIFO. FIFO controller also checks whether the EDC code is consistent with the read data at the read access timing.

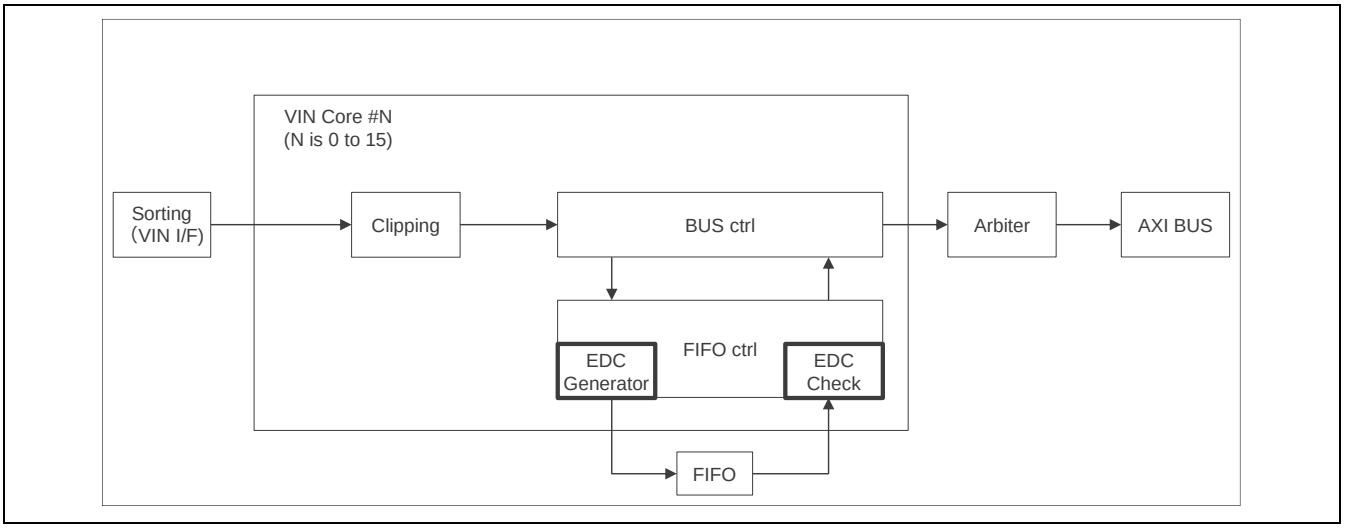



**Figure 55.16 Overview of EDC for Write FIFO**

Figure 55.17 shows EDC checking data from ISP. This EDC value is determined from 2-bit source ID and 24-bit data. The source ID is used to confirm that the data transfer path is correct. The value of the source ID used by the EDC checker on the VIN side is set by a register.

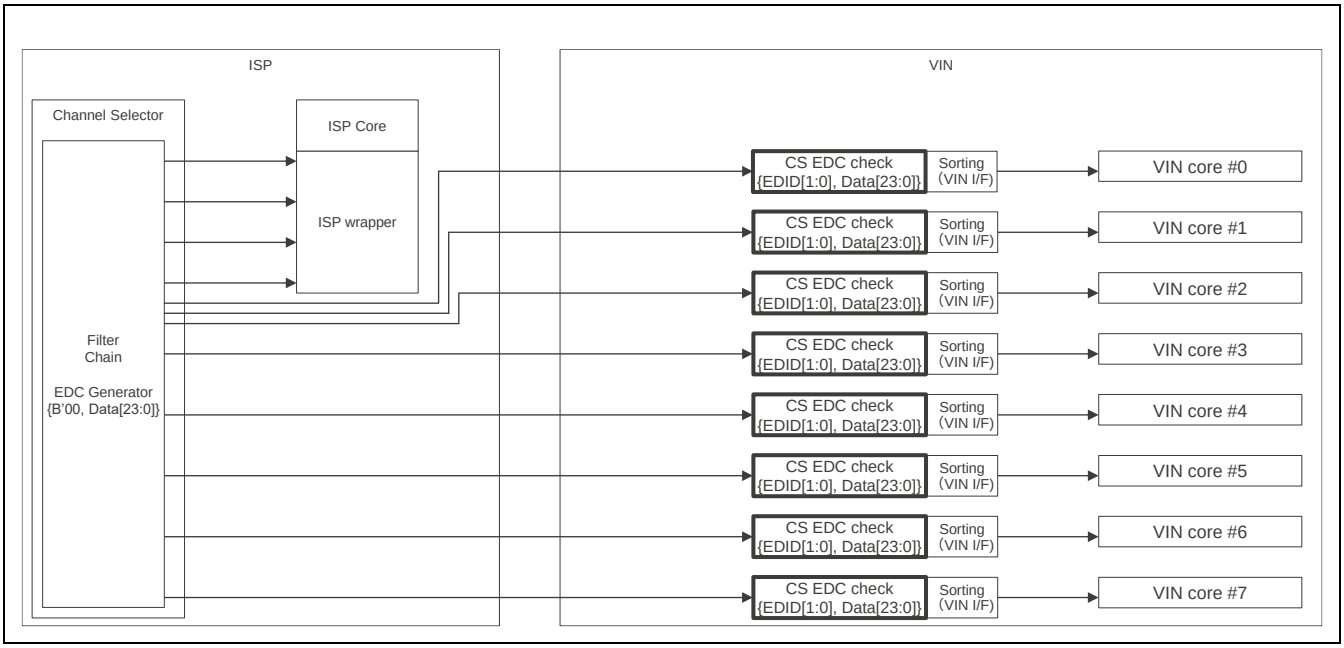



**Figure 55.17 Overview of EDC for Data from ISP**

Below table shows the registers related to EDC function.

**Table 55.11 Registers for EDC function**

| Name | Symbol |
|---|---|
| Video n Functional Safety Control Register | VnFNSC |
| Video n Error Enable Register | VnEE |
| Video n Error Status Register | VnERS |

If VnFNSC.FED_EN is 1, EDC protection for write data FIFO is enabled. When EDC error of write FIFO is detected, EFER bit in VnERS is set to 1.

If VnFNSC.CED_EN is 1, EDC checking for input data from channel selector is enabled. In this case VnFNSC.EDID[1:0] should be set to B'00. When EDC error of input data from channel selector is detected, ECER bit in VnERS is set to 1.

Below table shows an error interrupt signal related to EDC function.

**Table 55.12 Error Interrupt Signals and Register Mapping of EDC function (*n* = 0 to 15)**

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi*n*_edc_p | EFER, ECER | V*n*ERS | Video Input Channel-*n* Module Error |

Below shows connected module related to EDC function.

**Table 55.13 Connected Module**

| Module Name | Connected Module Name | Function of Connected Module |
|---|---|---|
| VIN | ISP (channel selector) | Input of Image Data |
| | ECM | Error Interruption Controller |

Below table shows an error interrupt signal and related ECM setting register.

**Table 55.14 Error Interrupt Signals and ECM Register Mapping of EDC function (*n* = 0 to 15)**

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi0_edc_p | 30 | ECMERRxxxR36 | Video Input Channel-0 Module Combined EDC Error |
| errreq_vin_vi1_edc_p | 28 | ECMERRxxxR36 | Video Input Channel-1 Module Combined EDC Error |
| errreq_vin_vi2_edc_p | 26 | ECMERRxxxR36 | Video Input Channel-2 Module Combined EDC Error |
| errreq_vin_vi3_edc_p | 24 | ECMERRxxxR36 | Video Input Channel-3 Module Combined EDC Error |
| errreq_vin_vi4_edc_p | 22 | ECMERRxxxR36 | Video Input Channel-4 Module Combined EDC Error |
| errreq_vin_vi5_edc_p | 20 | ECMERRxxxR36 | Video Input Channel-5 Module Combined EDC Error |
| errreq_vin_vi6_edc_p | 18 | ECMERRxxxR36 | Video Input Channel-6 Module Combined EDC Error |
| errreq_vin_vi7_edc_p | 16 | ECMERRxxxR36 | Video Input Channel-7 Module Combined EDC Error |
| errreq_vin_vi8_edc_p | 14 | ECMERRxxxR36 | Video Input Channel-8 Module Combined EDC Error |
| errreq_vin_vi9_edc_p | 12 | ECMERRxxxR36 | Video Input Channel-9 Module Combined EDC Error |
| errreq_vin_vi10_edc_p | 10 | ECMERRxxxR36 | Video Input Channel-10 Module Combined EDC Error |
| errreq_vin_vi11_edc_p | 8 | ECMERRxxxR36 | Video Input Channel-11 Module Combined EDC Error |
| errreq_vin_vi12_edc_p | 6 | ECMERRxxxR36 | Video Input Channel-12 Module Combined EDC Error |
| errreq_vin_vi13_edc_p | 4 | ECMERRxxxR36 | Video Input Channel-13 Module Combined EDC Error |
| errreq_vin_vi14_edc_p | 2 | ECMERRxxxR36 | Video Input Channel-14 Module Combined EDC Error |
| errreq_vin_vi15_edc_p | 0 | ECMERRxxxR36 | Video Input Channel-15 Module Combined EDC Error |

### 55.5.2 Hardware Redundancy for VIN

#### 55.5.2.1 Overview

VIN has redundancy logic to detect a temporary failure.

#### 55.5.2.2 Hardware Description

Below figure shows overview of hardware redundancy for VIN. All outputs of VIN logic marked as master core are checked. The redundancy target logic are the followings.

- Interface logic from ISP
- Sync interval checker
- Each video channel logic (excluding internal write FIFO (SRAM))
- Arbiter
- Conversion logic from an internal data bus to AXI-IF (s2X)
- Conversion logic from APB-IF to an internal register interface (BUS I/F of APB BUS)

The non-redundancy logic are the followings.

- Internal write buffer (SRAM)
- Synchronization logic

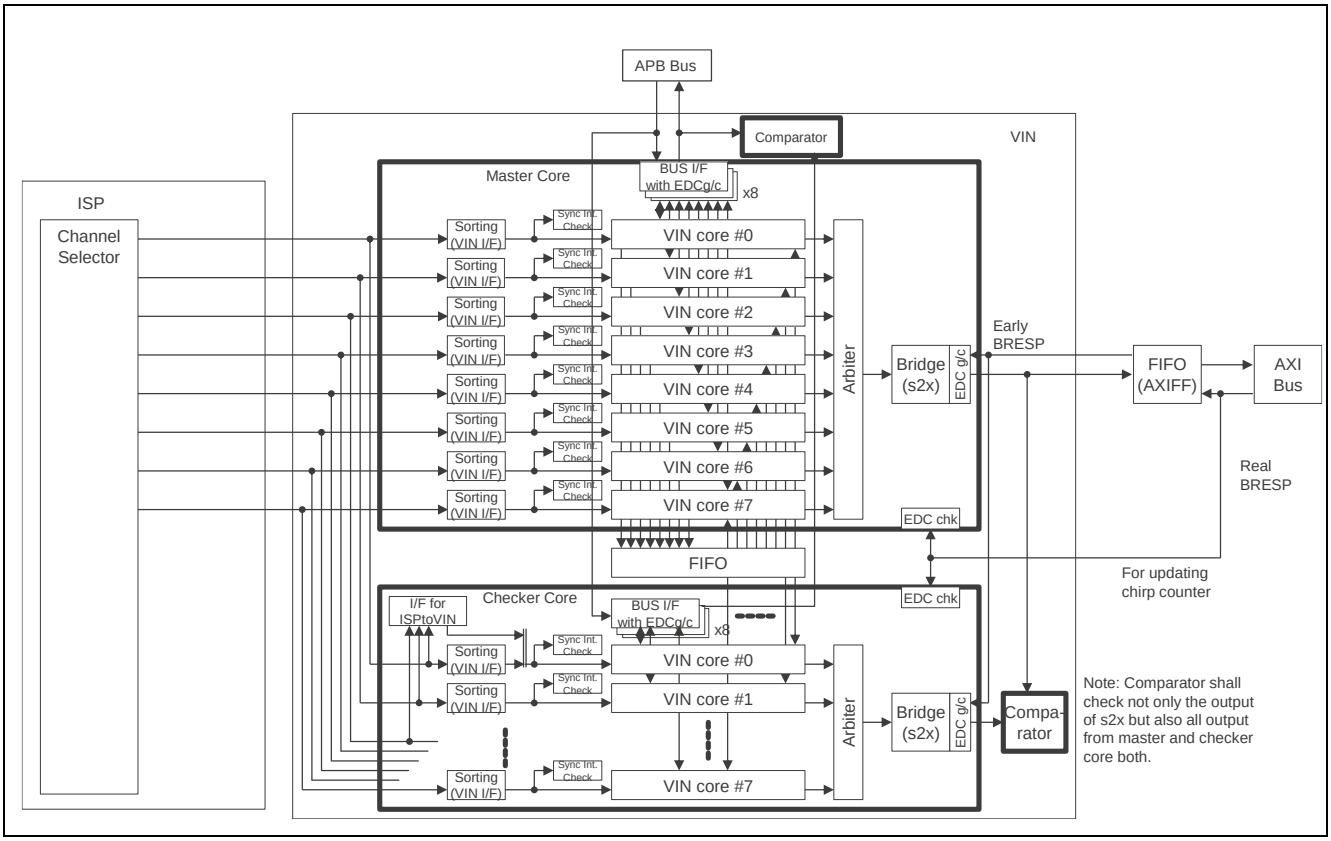



**Figure 55.18 Overview of Hardware Redundancy**

Below table shows the registers related to EDC function.

**Table 55.15 Registers for Hardware Redundancy**

| Name | Symbol |
|---|---|
| Video n Functional Safety Control Register | VnFNSC |
| Video n Error Enable Register | VnEE |
| Video n Error Status Register | VnERS |

If VnFNSC.HRCHK is set to1, VIN check all output signals of master core. They are classified into eleven. If a mismatched error occurs in the signal of each classification, HRER[10:0] bits in VnERS is set to 1 according signal category.

Below table shows an error interrupt signal related to hardware redundancy.

**Table 55.16 Error Interrupt Signals and Register Mapping of Hardware Redundancy (*n = 0 to 15*)**

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi*n*_p | HRER[10:0] | V*n*ERS | Video Input Channel-*n* Module Error |

Below shows connected module related to Sync Interval Checker.

**Table 55.17 Connected Module**

| Module Name | Connected Module Name | Function of Connected Module |
|---|---|---|
| VIN | ISP (channel selector) | Input of Image Data |
| | ECM | Error Interruption Controller |

Below table shows an error interrupt signal and related ECM setting register.

**Table 55.18 Error Interrupt Signals and ECM Register Mapping of Hardware Redundancy (*n* = 0 to 15)**

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi0_p | 31 | ECMERRxxxR36 | Video Input Channel-0 Module Combined Safety Error |
| errreq_vin_vi1_p | 29 | ECMERRxxxR36 | Video Input Channel-1 Module Combined Safety Error |
| errreq_vin_vi2_p | 27 | ECMERRxxxR36 | Video Input Channel-2 Module Combined Safety Error |
| errreq_vin_vi3_p | 25 | ECMERRxxxR36 | Video Input Channel-3 Module Combined Safety Error |
| errreq_vin_vi4_p | 23 | ECMERRxxxR36 | Video Input Channel-4 Module Combined Safety Error |
| errreq_vin_vi5_p | 21 | ECMERRxxxR36 | Video Input Channel-5 Module Combined Safety Error |
| errreq_vin_vi6_p | 19 | ECMERRxxxR36 | Video Input Channel-6 Module Combined Safety Error |
| errreq_vin_vi7_p | 17 | ECMERRxxxR36 | Video Input Channel-7 Module Combined Safety Error |
| errreq_vin_vi8_p | 15 | ECMERRxxxR36 | Video Input Channel-8 Module Combined Safety Error |
| errreq_vin_vi9_p | 13 | ECMERRxxxR36 | Video Input Channel-9 Module Combined Safety Error |
| errreq_vin_vi10_p | 11 | ECMERRxxxR36 | Video Input Channel-10 Module Combined Safety Error |
| errreq_vin_vi11_p | 9 | ECMERRxxxR36 | Video Input Channel-11 Module Combined Safety Error |
| errreq_vin_vi12_p | 7 | ECMERRxxxR36 | Video Input Channel-12 Module Combined Safety Error |
| errreq_vin_vi13_p | 5 | ECMERRxxxR36 | Video Input Channel-13 Module Combined Safety Error |
| errreq_vin_vi14_p | 3 | ECMERRxxxR36 | Video Input Channel-14 Module Combined Safety Error |
| errreq_vin_vi15_p | 1 | ECMERRxxxR36 | Video Input Channel-15 Module Combined Safety Error |

### 55.5.3 Sync Interval Checker

#### 55.5.3.1 Overview

This safety mechanism checks the interval of sync signals to detect whether the sync signal from ISP is stuck.

#### 55.5.3.2 Hardware Description

VSYNC, HSYNC, and FIELD signals can be checked with sync interval timers. Interval timer checks both the signal high state interval and low state interval. To check signal high state interval, the timer starts when the signal changes from low to high and ends when signal changes from high to low. To check signal low state interval, the timer starts when the signal changes from high to low and ends when signal changes from low to high. In both case the interval time shall be less than the maximum interval time and greater than the minimum interval time. HSYNC checker doesn't check the interval if VSYNC is low (during V-blanking period). In addition, HSYNC checker also has the rise edge counter and can check HSYNC rise count by each frame. CLKENB (Data Enable) signal can be checked for each line by pixel counter. The expected range of each sync interval can be configured by software.

Sync Interval Checker is implemented for each VIN channel. Figure 55.19 shows the location where Sync Interval Checker is implemented.

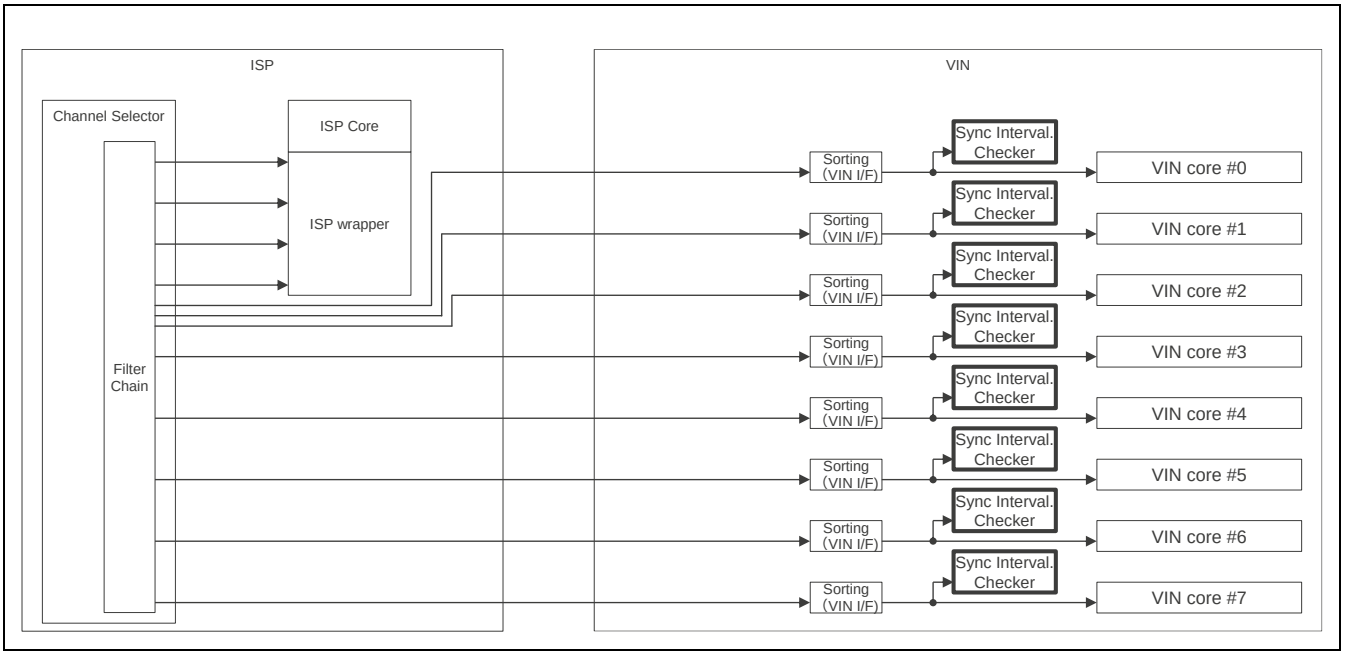



**Figure 55.19 Location the Sync Interval Checker is implemented**

Figure 55.20 shows block diagram of Sync Interval Checker. It captures four signals and counts cycle and edge number. Note that the clock frequency of VIN is 800MHz. The maximum and minimum interval time are approximated by the setting value in each register multiplied by 1/(800,000,000) second.

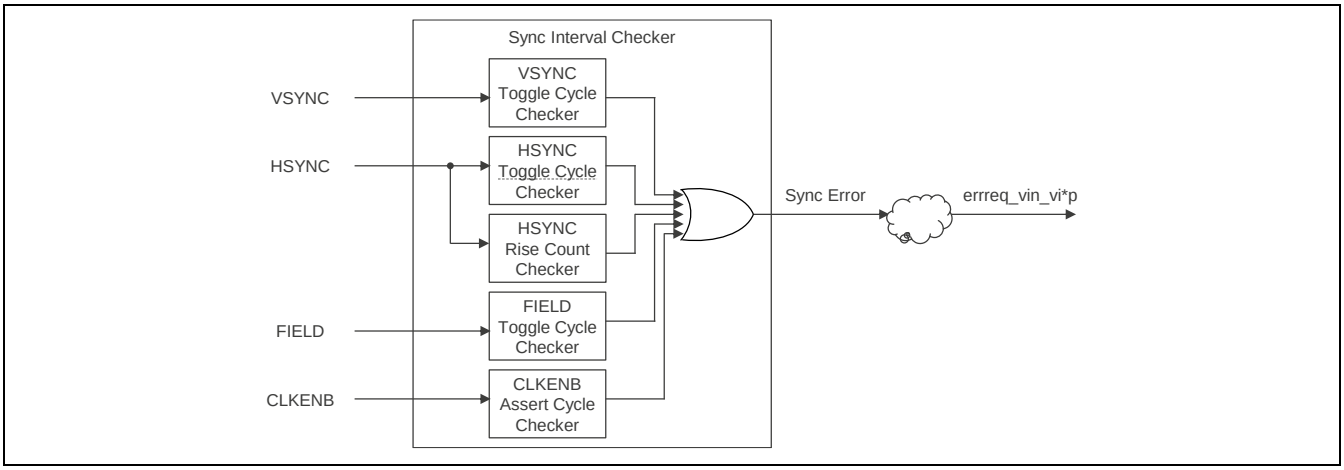



**Figure 55.20 Block Diagram of Sync Interval Checker**

Below table shows the registers related to the sync interval checker.

**Table 55.19 Registers for Sync Interval Checker**

| Name | Symbol |
|---|---|
| Video n Functional Safety Control Register | VnFNSC |
| Video n Error Enable Register | VnEE |
| Video n Error Status Register | VnERS |
| Video n Field Toggle Minimum Register | VnFTMIN |
| Video n Field Toggle Maximum Register | VnFTMAX |
| Video n Hsync Toggle Minimum Register | VnHSTMIN |
| Video n Hsync Toggle Maximum Register | VnHSTMAX |
| Video n Vsync Toggle Minimum Register | VnVSTMIN |
| Video n Vsync Toggle Maximum Register | VnVSTMAX |
| Video n Data Enable Term Register | VnDET |
| Video n Hsync Rise Time Register | VnHRT |

When the toggle term of VSYNC, HSYNC, FIELD is less than minimum expected value or more than maximum expected value, the error is detected. When the assert term of CLKENB is less than minimum expected value or more than maximum expected value, the error is detected. When the error was detected, the SER bit in the VnERS is set to 1.

The minimum expected value and the maximum expected value can be set by the VnFTMIN, VnFTMAX, VnHSTMIN, VnHSTMAX, VnVSTMIN, VnVSTMAX, VnDET and VnHRT.

The error detection is controllable by FLCE[1:0], HSCE[1:0], VSCE[1:0], DECE[1:0] and HCEN[1:0] bits in the VnFNSC. When the bit[0] is set to 1, error of minimum can be checked. When the bit[1] is set to 1, error of maximum can be checked.

When VSCE is set to enabled, the VSYNC toggle cycle checker becomes effective after the next VSYNC toggle. When FLCE, HSCE, and HCEN are set to enabled, the FIELD toggle cycle checker, the HSYNC toggle cycle checker, and HSYNC rise count checker become effective after the next VSYNC toggle, too. When HCEN is set to enabled, CLKENB assert cycle checker becomes effective after the next HSYNC fall toggle.

When each checker is enabled, all checkers must be disabled before VIN stops the capture process.

The DEER_DM bit in the VnFNSC controls the error detection mode of CLKENB. When DEER_DM bit is set to 0, if CLKENB pixel counter is 0, the error of minimum can’t be detected. When DEER_DM bit is set to 1, the error of minimum can be detected even if CLKENB pixel counter is 0.

- **Notes on HSYNC Rise Count Checker at Sync Interval Checker**

Regarding the behavior of the filter function described in 54.4.2.1 in section 54. Image Signal Processor (ISP)/Channel Selector (CS), the Channel Selector sets an invalid value (low) to the Data Enable signal (named as CLKENB) at the line or pixel blocked by the filter. In the other hand, the HSYNC signal is passed through even if the corresponding line or pixel are blocked. That is, the HSYNC signal is propagated from Channel Selector to VIN regardless of whether it is blocked or passed through.

For example, as shown in Figure 55.21, this example shows that 5 lines (a), (b), (c), (d), and (e) are input to the filter. Line (a), (c) and (e) represent one line which are sent as a long packet belonging to VC0. Line (b) and (d) represent one line which are sent as a long packet belonging to VC1. And Figure 55.21 shows an example where only the lines corresponding to VC0 is passed through and transferred from Channnel Selector to VIN.

At this time, when blocking the line (b) and (d), the Channel Selector controls the CLKENB signal on lines (b) and (d) to the invalid side (low), but the HSYNC signal during the same period remains valid value (high).

This example shows the result of filter processing by the VC Filter, but the same applies to other filters. That is, even filters other than the VC Filter control the CLKENB signal, and the HSYNC signal is passed through regardless of the filter processing result.

Considering above mentioned behavior, when counting the rise edge of HSYNC using the Sync Interval Checker, the number of rise edge of HSYNC is counted at the timing of line (b) and (d) even if these lines are blocked by the filter.

Therefore, it is necessary to set the number of HSYNC rise time (VnHRT) at the HSYNC Rise Time Measuremnt considering the number of long packets (*) transmitted from CSI during the observation period (active period of VSYNC).

(* The number of long packets is equal to the rise count of HSYNC)

Similarly, the Max (VnHSTMAX) and Min (VnHSTMIN) registers of the Hsync Toggle Cycle Measurement shall also be set based on the above specification.

If 5 lines (a), (b), (c), (d), and (e) are input to CSI, as shown in Figure 55.21, the maximum number of rise edge of HSYNC should be set as 5, even if the image size (number of line) to be captured by VIN is 3 lines. In particular, when using multiple sensors, the maximum number of rise edge of HSYNC should be set to the total number of lines transmitted from each sensor within the observation period.

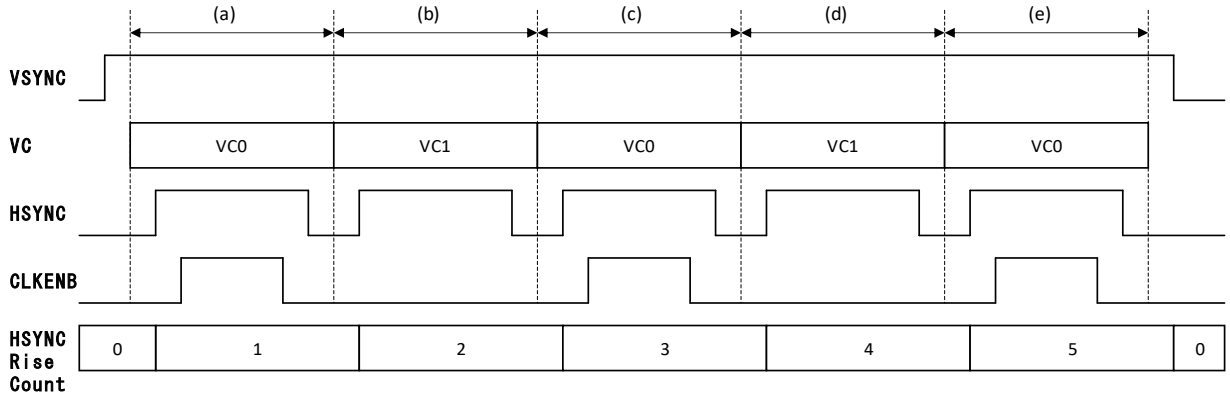



**Figure 55.21** Timing example of VSYNC, HSYNC and CLKENB from Channel Selector (CS)

Below table shows an error interrupt signal related to Sync Interval Checker.

**Table 55.20 Error Interrupt Signals and Register Mapping of Sync Interval Checker (*n = 0 to 15*)**

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi*n*_p | SER | V*n*ERS | Video Input Channel-*n* Module Error |

Below shows connected module related to Sync Interval Checker.

**Table 55.21 Connected Module**

| Module Name | Connected Module Name | Function of Connected Module |
|---|---|---|
| VIN | ISP | Input of Image Data |
| | ECM | Error Interruption Controller |

Below table shows an error interrupt signal and related ECM setting register.

**Table 55.22 Error Interrupt Signals and ECM Register Mapping of Sync Interval Checker (*n* = 0 to 15)**

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi0_p | 31 | ECMERRxxxR36 | Video Input Channel-0 Module Combined Safety Error |
| errreq_vin_vi1_p | 29 | ECMERRxxxR36 | Video Input Channel-1 Module Combined Safety Error |
| errreq_vin_vi2_p | 27 | ECMERRxxxR36 | Video Input Channel-2 Module Combined Safety Error |
| errreq_vin_vi3_p | 25 | ECMERRxxxR36 | Video Input Channel-3 Module Combined Safety Error |
| errreq_vin_vi4_p | 23 | ECMERRxxxR36 | Video Input Channel-4 Module Combined Safety Error |
| errreq_vin_vi5_p | 21 | ECMERRxxxR36 | Video Input Channel-5 Module Combined Safety Error |
| errreq_vin_vi6_p | 19 | ECMERRxxxR36 | Video Input Channel-6 Module Combined Safety Error |
| errreq_vin_vi7_p | 17 | ECMERRxxxR36 | Video Input Channel-7 Module Combined Safety Error |
| errreq_vin_vi8_p | 15 | ECMERRxxxR36 | Video Input Channel-8 Module Combined Safety Error |
| errreq_vin_vi9_p | 13 | ECMERRxxxR36 | Video Input Channel-9 Module Combined Safety Error |
| errreq_vin_vi10_p | 11 | ECMERRxxxR36 | Video Input Channel-10 Module Combined Safety Error |
| errreq_vin_vi11_p | 9 | ECMERRxxxR36 | Video Input Channel-11 Module Combined Safety Error |
| errreq_vin_vi12_p | 7 | ECMERRxxxR36 | Video Input Channel-12 Module Combined Safety Error |
| errreq_vin_vi13_p | 5 | ECMERRxxxR36 | Video Input Channel-13 Module Combined Safety Error |
| errreq_vin_vi14_p | 3 | ECMERRxxxR36 | Video Input Channel-14 Module Combined Safety Error |
| errreq_vin_vi15_p | 1 | ECMERRxxxR36 | Video Input Channel-15 Module Combined Safety Error |

### 55.5.4 Frame CRC Calculation

#### 55.5.4.1 Overview

This safety mechanism is used in two cases. First is detecting frozen image by comparing the CRC value of a frame and previous frame. Second is checking the integrity of the data sent from another R-Car V4M.

#### 55.5.4.2 Hardware Description

VIN has an operator capable of CRC of any polynomial and there are two CRC operation mode.

First mode is only calculation CRC. In this mode, each channels of VIN can calculate CRC of the input data. VIN can select either before or after clipping data to input to CRC. The result of CRC is reflected to register at the rising edge of VSYNC signal.

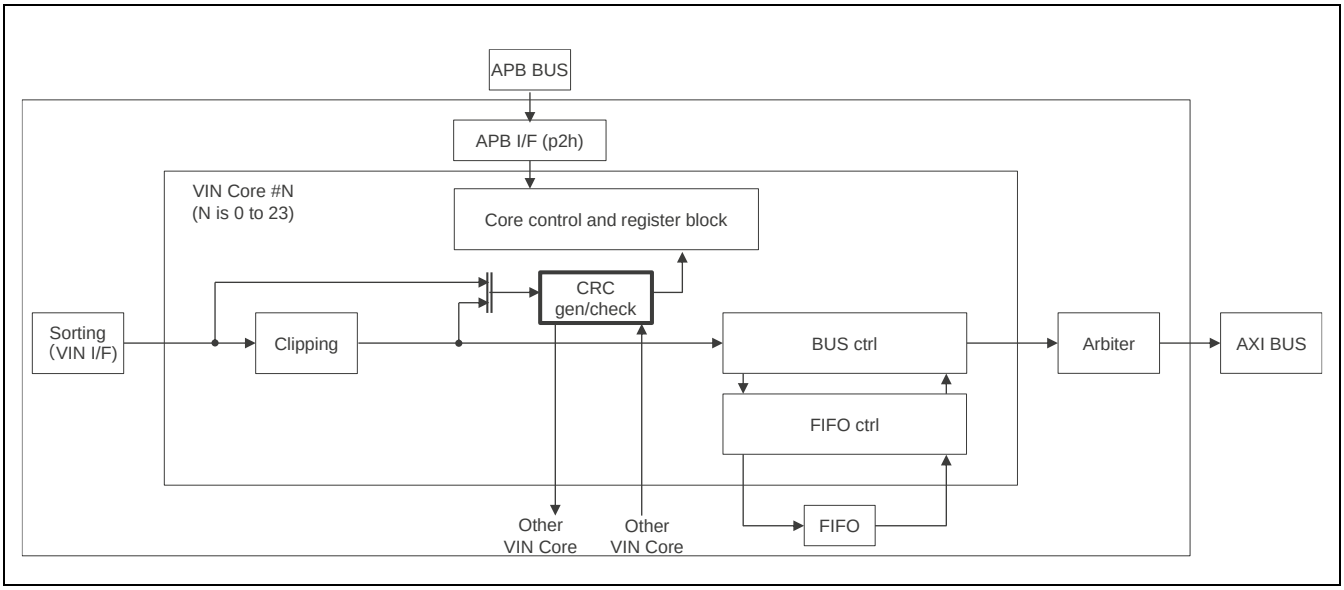



**Figure 55.22 Overview of CRC operation**

Second mode, VIN calculates CRC and compares with expection value. This mode is used for checking the integrity of the data sent via inter-chip connection using MIPI/CSI-2. Figure 55.23 shows a concept of end-to-end protection by CRC. VSPD of R-Car V4M #0 generates CRC of data sent and information linked to the data called “FrameID”. They are sent in MIPI/CSI-2 embedded 8-bit non image data type. On the receiving side R-Car V4M, ISP/ChannelSelector extracts the data and CRC/FrameID, and sends to VIN. VIN calculates CRC of input data and compares with CRC sent from VSPD to check the integrity of the data.

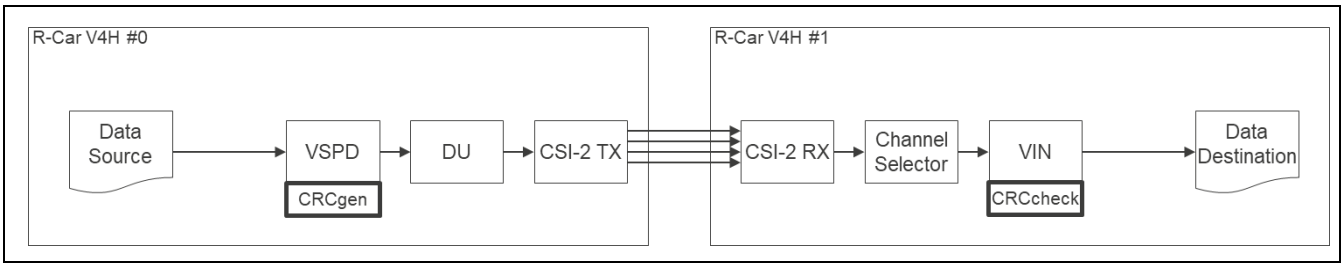



**Figure 55.23 Concept of End-to-End Protection by CRC**

In this mode, the data is input to channel 0 to 3 of each VIN instance (channel 0 to 3, 8 to 11), and these channels calculate CRC of the input data. The CRC/FrameID is input to channel 4 to 7 of each VIN instance (channel 4 to 7, 12 to 15), and these channels only send CRC as expected to channel 0 to 3 of each VIN instance. Below figure shows flow of CRC/FrameID.

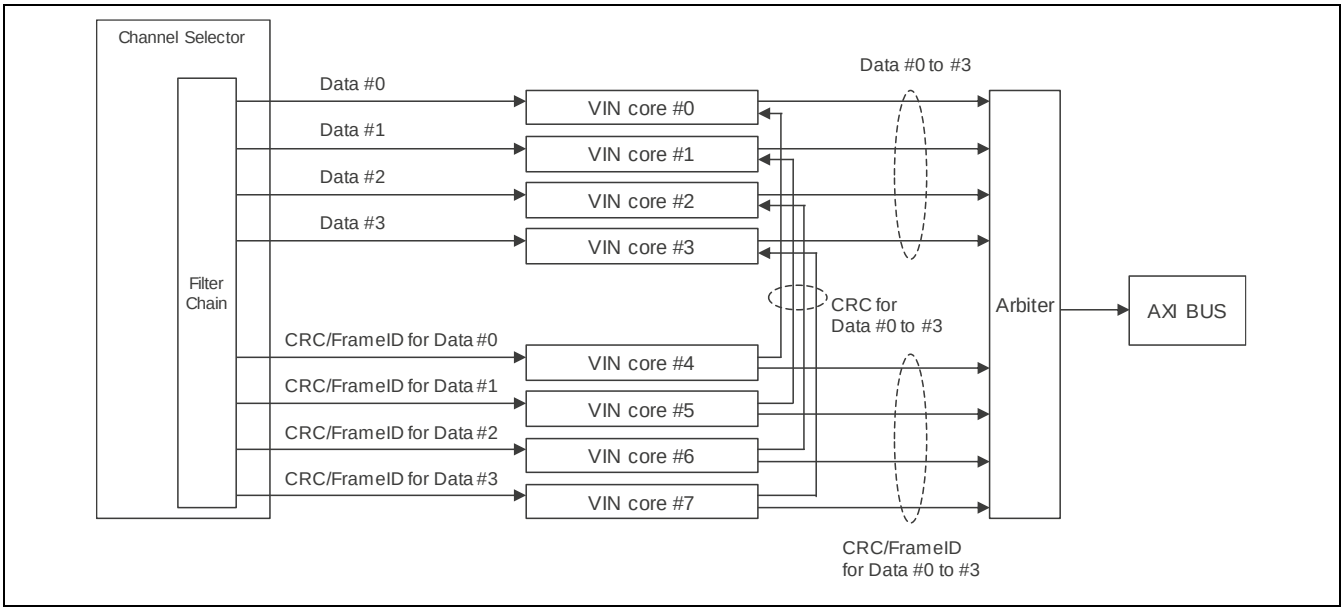



**Figure 55.24 Flow of CRC and FrameID for each VIN instance**

FrameID is 16-bit value and consists of the following three fields.

Error Flag:
1-bit error flag of CRC data, “0” means the CRC is valid and “1” means the CRC is not valid. If Error Flag is 1, channel 0 to 3 doesn’t compare CRC and expected.

Unique ID:
3-bit fixed number indicates source VSPD instance number. It is used for checking routing from VSPD to VIN.

Frame Num:
12-bit increment value for each frame. It is used for checking whether frame data dropping is occurred.

Below is a format of CRC/FrameID.

Byte 0: CRC [31:24]
Byte 1: CRC [23:16]
Byte 2: CRC [15:8]
Byte 3: CRC [7:0]
Byte 4: {Error Flag [0], Unique ID [2:0], Frame Num [11:8]}
Byte 5: Frame Num [7:0]
Byte 6: 8’h00
Byte 7: 8’h00

The channel 4 to 7 not only send CRC data to channel 0 to 3 but also store both CRC and FrameID to DRAM.

Below table shows the registers related to the frame CRC.

**Table 55.23 Registers for Frame CRC Calculation**

| Name | Symbol |
|---|---|
| Video n Functional Safety Control Register | VnFNSC |
| Video n Error Enable Register | VnEE |
| Video n Error Status Register | VnERS |

| Name | Symbol |
|---|---|
| Video n CRC Code Register | VnVCRC |
| Video n CRC Calculation Configuration Register | VnCRCC |
| Video n CRC Polynomial Equation Coefficient Register | VnCRPEC |
| Video n CRC Initial Remainder Register | VnCRIR |
| Video n CRC XOR Coefficient Register | VnCRXRC |

CRC operation mode is specified by V*n*FNSC.CRM bit. Only CRM bit of channel 0 to 3 in each VIN instance can be set to 1. If VnFNSC.CRM is 0:

- The channel-n calculate CRC and set to VnVCRC.VCRC[31:0] at the rising edge of VSYNC signal.

Othrewise (VnFNSC.CRM is 1):

- The channel-n should calculate CRC of entire frame (VnCRCC.INSEL is 0). The result is set to VnVCRC.VCRC[31:0] at the rising edge of VSYNC signal.
- The channel-n compares the calculation result and the expectation value when Error Flag in FrameID from the channel-(n+4) is 0. If they are mismatched, VnERS.CCER bit is set to 1. If the Error Flag is 1, the channel-n doesn't compare with the expected value.
- VCRC[31:0] for channel-(n+4) is not available.
- The channel-(n+4) outputs the expected CRC value and FrameID to DRAM.

VIN can calculate CRC without storing data to DRAM if VnCRCC.INSEL is 0. In other words, the CRC operation of the entire frame becomes valid when data is input.

Below table shows an error interrupt signal related to CRC checking if VnFNSC.CRM is 1.

**Table 55.24 Error Interrupt Signals and Register Mapping of CRC checking (*n = 0 to 15*)**

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi*n*_p | CCER | V*n*ERS | Video Input Channel-*n* Module Error |

Below table shows connected module related to CRC checking if VnFNSC.CRM is 1.

**Table 55.25 Connected Module**

| Module Name | Connected Module Name | Function of Connected Module |
|---|---|---|
| VIN | ISP | Input of Image Data |
| | ECM | Error Interruption Controller |

Below table shows an error interrupt signal and related ECM setting register.

**Table 55.26 Error Interrupt Signals and ECM Register Mapping of CRC checking (*n* = 0 to 15)**

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi0_p | 31 | ECMERRxxxR36 | Video Input Channel-0 Module Combined Safety Error |
| errreq_vin_vi1_p | 29 | ECMERRxxxR36 | Video Input Channel-1 Module Combined Safety Error |
| errreq_vin_vi2_p | 27 | ECMERRxxxR36 | Video Input Channel-2 Module Combined Safety Error |
| errreq_vin_vi3_p | 25 | ECMERRxxxR36 | Video Input Channel-3 Module Combined Safety Error |
| errreq_vin_vi4_p | 23 | ECMERRxxxR36 | Video Input Channel-4 Module Combined Safety Error |
| errreq_vin_vi5_p | 21 | ECMERRxxxR36 | Video Input Channel-5 Module Combined Safety Error |
| errreq_vin_vi6_p | 19 | ECMERRxxxR36 | Video Input Channel-6 Module Combined Safety Error |

| Error Interrupt Name | Bit | Register | Error Content in ECM |
|---|---|---|---|
| errreq_vin_vi7_p | 17 | ECMERRxxxR36 | Video Input Channel-7 Module Combined Safety Error |
| errreq_vin_vi8_p | 15 | ECMERRxxxR36 | Video Input Channel-8 Module Combined Safety Error |
| errreq_vin_vi9_p | 13 | ECMERRxxxR36 | Video Input Channel-9 Module Combined Safety Error |
| errreq_vin_vi10_p | 11 | ECMERRxxxR36 | Video Input Channel-10 Module Combined Safety Error |
| errreq_vin_vi11_p | 9 | ECMERRxxxR36 | Video Input Channel-11 Module Combined Safety Error |
| errreq_vin_vi12_p | 7 | ECMERRxxxR36 | Video Input Channel-12 Module Combined Safety Error |
| errreq_vin_vi13_p | 5 | ECMERRxxxR36 | Video Input Channel-13 Module Combined Safety Error |
| errreq_vin_vi14_p | 3 | ECMERRxxxR36 | Video Input Channel-14 Module Combined Safety Error |
| errreq_vin_vi15_p | 1 | ECMERRxxxR36 | Video Input Channel-15 Module Combined Safety Error |

The operator can calculate CRC of any polynomial by VnCRCC, VnCRPEC, VnCRIR and VnCRXRC. Following table shows setting examples.

**Table 55.27 Setting Examples of CRC**

| CRC Type | Register | Setting Value |
|---|---|---|
| CRC-32 IEEE 802.3 | VnCRPEC.CPLYEC[31:0] | H'04C1 1DB7 |
| | VnCRCC.PDBL[4:0] | 31 |
| | VnCRCC.IDR, VnCRCC.IDRE | 1 |
| | VnCRCC.ROR | 1 |
| | VnCRIR.CIR | H'FFFF FFFF |
| | VnCRXRC.CXORC | H'FFFF FFFF |
| CRC-32 AUTOSAR 4.2.2 | VnCRPEC.CPLYEC[31:0] | H'F4AC FB13 |
| | VnCRCC.PDBL[4:0] | 31 |
| | VnCRCC.IDR, VnCRCC.IDRE | 1 |
| | VnCRCC.ROR | 1 |
| | VnCRIR.CIR | H'FFFF FFFF |
| | VnCRXRC.CXORC | H'FFFF FFFF |
| CRC-16 CCITT-FALSE | VnCRPEC.CPLYEC[31:0] | H'0000 1021 |
| | VnCRCC.PDBL[4:0] | 15 |
| | VnCRCC.IDR, VnCRCC.IDRE | 0 |
| | VnCRCC.ROR | 0 |
| | VnCRIR.CIR | H'0000 FFFF |
| | VnCRXRC.CXORC | H'0000 0000 |
| CRC-8 SAE J1850 | VnCRPEC.CPLYEC[31:0] | H'0000 001D |
| | VnCRCC.PDBL[4:0] | 7 |
| | VnCRCC.IDR, VnCRCC.IDRE | 0 |
| | VnCRCC.ROR | 0 |
| | VnCRIR.CIR | H'0000 00FF |
| | VnCRXRC.CXORC | H'0000 00FF |
| CRC-21 0x102899 | VnCRPEC.CPLYEC[31:0] | H'0010 2899 |
| | VnCRCC.PDBL[4:0] | 20 |
| | VnCRCC.IDR, VnCRCC.IDRE | 1 |
| | VnCRCC.ROR | 1 |
| | VnCRIR.CIR | H'001F FFFF |

| CRC Type | Register | Setting Value |
|---|---|---|
| | VnCRXRC.CXORC | H'001F FFFF |

Below table shows CRC supported formats. VnCRCC register should be set for each input data format shown in below table.

**Table 55.28 Setting Examples of VnCRCC**

| Input Data Format | Register | Setting Value If VnCRCC.IDR = 0 | Setting Value If VnCRCC.IDR = 1 |
|---|---|---|---|
| RAW8, Embedded 8-bit, User Defined 8-bit | IDBO[2:0] | B'011 | B'011 |
| | IDBL[1:0] | B'00 | B'00 |
| | Y420MD | 0 | 0 |
| RAW10 | IDBO[2:0] | B'010 | B'011 |
| | IDBL[1:0] | B'01 | B'01 |
| | Y420MD | 0 | 0 |
| RAW12 | IDBO[2:0] | B'010 | B'011 |
| | IDBL[1:0] | B'01 | B'01 |
| | Y420MD | 0 | 0 |
| RAW14 | IDBO[2:0] | B'010 | B'011 |
| | IDBL[1:0] | B'01 | B'01 |
| | Y420MD | 0 | 0 |
| RAW16, YCbCr-422 8-bit, RGB-565 | IDBO[2:0] | B'010 | B'011 |
| | IDBL[1:0] | B'01 | B'01 |
| | Y420MD | 0 | 0 |
| RAW20, YCbCr-422 10-bit, YCbCr-422 12-bit (Y12CbCr8) | IDBO[2:0] | B'100 | B'011 |
| | IDBL[1:0] | B'10 | B'10 |
| | Y420MD | 0 | 0 |
| RAW24, RGB-888 | IDBO[2:0] | B'100 | B'011 |
| | IDBL[1:0] | B'10 | B'10 |
| | Y420MD | 0 | 0 |

# 56. IMR-LX6

## 56.1 Overview

The image renderer light extended 6 (IMR-LX6*), or the distortion correction engine, is a drawing processor with a simple instruction system capable of referencing data on an external memory as two-dimensional texture data and performing texture mapping and drawing with respect to any shape that is split into triangular objects. By modifying the shape of an image, the IMR-LX6 can be used in such applications as surroundings monitoring and backward monitoring. The IMR-LX6 also allows setting the values for correcting luminance/hue of each vertex, which enables correction of luminance/hue difference within a camera lens, and that among multiple lenses.

Note: The actual modules are referred to as the IMR-LX6-V4M in the R-Car V4M.

The R-Car V4M has four IMR-LX6-V4M modules (Four channels). IMR channel 0/1 is also specified by IMS0/1. IMR channel2/3 is also specified by IMR0/1

**Table 56.2 Base Addresses and Connected Clock Signals of IMR-LX6-V4M**

| Channel | Base Address | Rotator & Scaler Availability | Clock |
|---|---|---|---|
| 0 (IMS0) | H'FE86 0000 | — | VCBUSD2ϕ (for access to the registers), VCBUSϕ (for operations other than access to registers) |
| 1 (IMS1) | H'FE87 0000 | — | |
| 2 (IMR0) | H'FE88 0000 | √ | |
| 3 (IMR1) | H'FE89 0000 | √ | |

### 56.1.1 Features

- Pixel performance:
  - 533 Mpixel/sec (peak, @VCBUSϕ)
- Display list (DL*1) system
- Drawing functions
  - Texture mapping
  - Bilinear filtering
  - Filtering using pyramidal images
  - Automatic coordinate generation
  - Luminance/hue correction
  - Luminance conversion using lookup table (LUT)
  - Rotation and scaling-down*6
- Input image data formats
  - Luminance: Y 8 bpp, 10 bpp, 12 bpp, 14bpp, 16bpp; Color difference: UV*2 8 bpp, 10 bpp, 12 bpp, 14 bpp, 16 bpp; Data: 16 bpp; Data: 32 bpp
  - Double luminance plane(DYP mode) *9: Y 8 bpp, 10 bpp, 12 bpp, 14bpp, 16bpp
  - YUV422 interleaved mode
  - YUV422 semi-planar mode
  - YUV420 semi-planer mode
  - Linear and tile addressing modes
- Output image data formats*3
  - Luminance: Y 8 bpp, 10 bpp, 12 bpp, 14bpp, 16bpp; Color difference: UV*2 8 bpp, 10 bpp, 12 bpp, 14 bpp, 16 bpp; Data: 16 bpp, Data: 32 bpp
  - Double luminance plane(DYP mode) *9: Y 8 bpp, 10 bpp, 12 bpp, 14bpp, 16bpp
  - YUV422 interleaved mode
  - YUV422 semi-planar mode
  - YUV420 semi-planer mode
  - Linear and tile addressing modes
- Instruction set
  - Single draw instruction: TRI for drawing a triangle strip.
  - Fourteen control instructions: TRAP, INT, NOP, SYNCM, WTL, WTL2, WTS, GOSUB, RET, WUP, SLP, STRECT*6, OUTNOR*6, and OUTROT*6
- Drawing space
  - Destination coordinates (X, Y) are 16bit, drawing space (XMAX, YMAX) : 0 ≤ XMAX ≤ 2,047, 0 ≤ YMAX ≤ 2,047

    The size of the circumscribed rectangle of a triangle must be less than 128 pixels for both X-direction and Y-direction
  - Source*4 coordinates (u, v)*2: 0 ≤ u*2 < 2,048, 0 ≤ v*2 < 2,048
- Endian and address mode
  - Little endian and 32-bit address mode are supported for DL and pixel (source and destination) data.
- Destination cache
  - Includes a dedicated destination cache (4 Kbytes x 2 = 64 lines × 64 bytes/line x 2)*5
- Source cache
  - Includes a dedicated source cache (16 Kbytes x 2)*7
  - Double source cache mode (32 Kbytes x 2)*8
  - Non-blocking Cache Mode

Notes: 1. This manual uses DL as the abbreviation of the display list (a drawing/control instructions).

2. This manual uses uppercase letters U and V to express the color difference in the YUV data format, and uses lowercase letters u and v to express the axes and coordinates in the source coordinate space.

   In this section, YCbCr is written as YUV.

3. When the EFPE bit in the CMRCR register is set to 0, the output bit precision should be equal to or lower than the input bit precision. If the output bit precision is lower than the input bit precision, the output data is truncated from LSB to adjust the input bit precision.

   When the EFPE bit in the CMRCR register is set to 1 and the SHFE bit in the TRIMR register is set to 0, the output bit precision should be equal to the input bit precision.

   When the EFPE bit in the CMRCR register and the SHFE bit in the TRIMR register are both set to 1, specify the output precision in bits according to the combinations shown in Table 56.19 and Table 56.20.

4. The terms “texture” and “source” in this section indicate the two-dimensional data that are referred to for texture mapping during the drawing of triangles.

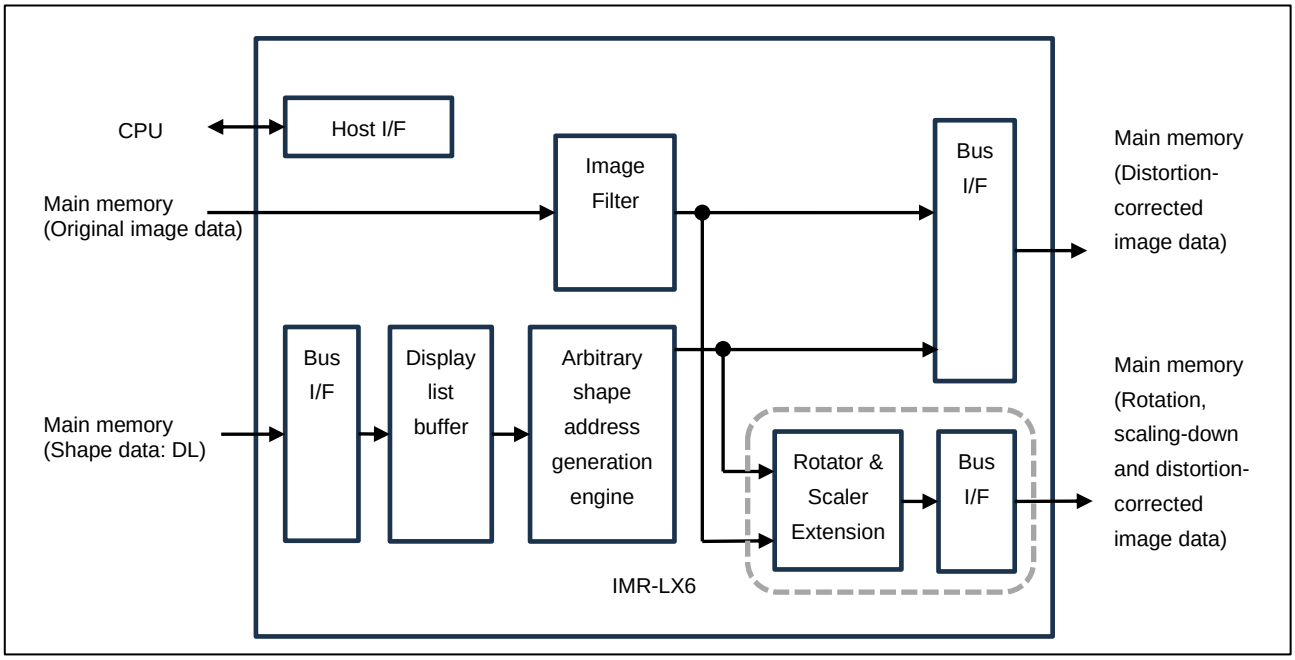
5. When the YISM bit in the CMRCR register and the YUV422E or YUVSEMI bit in the CMRCR2 register are both set to 1,
   8 Kbytes of the destination cache are used, with 4 Kbytes for luminance (Y) and 4 Kbytes for color difference (UV) data. With other settings, only 4 Kbytes of the destination cache are used.

6. STRECT, OUTNOR, and OUTROT are only applicable for those channels that support rotation and scaling down. See Table 56.2regarding the availability of rotation and scaling down.

7. In the linear addressing mode, the source cache is configured as 64 lines × 256 bytes/line x 2. The configuration in tile addressing mode is 256 lines × 64 bytes/line x 2. When the TFE bit in the TRIMR register is set to 1, 64 Kbytes of the source cache are used.

8. When pyramidal images filtering is not used, 64 Kbytes of the source cache can be used as dedicated source cache.

9. Use same data format for input and output when using double luminance plane. And use same bit depth for double (both) Y plane.




Note: The functions indicated in the broken-outlined box are only present in those channels where rotation and scaling-down are available. See Table 56.2 regarding the availability of rotation and scaling-down.

**Figure 56.1 Block Diagram**

### 56.1.2 Register Configuration (1)

Do not write to any address other than those listed in the tables. Otherwise, the operation is not guaranteed. Reading a non-listed address returns an undefined value.

**Table 56.3 Register Configuration (1) - Control Registers**

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| Control Register | CR | R/W | H'0008 | 32 |
| Status Register | SR | R | H'000C | 32 |
| Status Clear Register | SRCR | —/W | H'0010 | 32 |
| Interrupt Control Register | ICR | R/W | H'0014 | 32 |
| Interrupt Mask Register | IMR | R/W | H'0018 | 32 |
| DL Stack Pointer Register | DLSP | R | H'001C | 32 |
| DL Status Register | DLPR | R | H'0020 | 32 |
| Executed DL Status Register | EDLR | R/W | H'0028 | 32 |

**Table 56.4 Register Configuration (1) - Memory Control Registers**

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| DL Start Address Register | DLSAR | R/W | H'0030 | 32 |
| Destination Start Address Register | DSAR | R/W | H'0034 | 32 |
| Source Start Address Register | SSAR | R/W | H'0038 | 32 |
| Destination Stride Register | DSTR | R/W | H'003C | 32 |
| Source Stride Register | SSTR | R/W | H'0040 | 32 |
| Destination Start Offset Address Register | DSOR | R/W | H'0050 | 32 |
| Source Start Address Offset Register | SSAOR | R/W | H’01A0 | 32 |
| Source Start Address Offset Register1 | SSAOR1 | R/W | H’01A4 | 32 |
| Source Start Address Offset Register2 | SSAOR2 | R/W | H’01A8 | 32 |
| Source Start Address Offset Register3 | SSAOR3 | R/W | H’01AC | 32 |
| Source Start Address Offset Register4 | SSAOR4 | R/W | H’01B0 | 32 |
| UV Source Start Address Offset Register | UVSSAOR | R/W | H’01B4 | 32 |
| UV Source Start Address Offset Register1 | UVSSAOR1 | R/W | H’01B8 | 32 |
| UV Source Start Address Offset Register2 | UVSSAOR2 | R/W | H’01BC | 32 |
| UV Source Start Address Offset Register3 | UVSSAOR3 | R/W | H’01C0 | 32 |
| UV Source Start Address Offset Register4 | UVSSAOR4 | R/W | H’01C4 | 32 |
| Destination Start Address Offset Register | DSAOR | R/W | H’01C8 | 32 |
| X/Y Offset Register | XYOFSR | R/W | H’01CC | 32 |
| u/v Offset Register | UVOFSR | R/W | H’01D0 | 32 |
| Source Start Address Register for Map 1 | SSAR1 | R/W | H'0B00 | 32 |
| Source Stride Register for Map 1 | SSTR1 | R/W | H'0B04 | 32 |
| Source Start Address Register for Map 2 | SSAR2 | R/W | H'0B08 | 32 |
| Source Stride Register for Map 2 | SSTR2 | R/W | H'0B0C | 32 |
| Source Start Address Register for Map 3 | SSAR3 | R/W | H'0B10 | 32 |
| Source Stride Register for Map 3 | SSTR3 | R/W | H'0B14 | 32 |
| Source Start Address Register for Map 4 | SSAR4 | R/W | H'0B18 | 32 |
| Source Stride Register for Map 4 | SSTR4 | R/W | H'0B1C | 32 |
| UV Source Start Address Register for Map | UVSSAR | R/W | H'0B20 | 32 |
| UV Source Stride Register for Map | UVSSTR | R/W | H'0B24 | 32 |
| UV Source Start Address Register for Map 1 | UVSSAR1 | R/W | H'0B28 | 32 |
| UV Source Stride Register for Map 1 | UVSSTR1 | R/W | H'0B2C | 32 |
| UV Source Start Address Register for Map 2 | UVSSAR2 | R/W | H'0B30 | 32 |

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| UV Source Stride Register for Map 2 | UVSSTR2 | R/W | H'0B34 | 32 |
| UV Source Start Address Register for Map 3 | UVSSAR3 | R/W | H'0B38 | 32 |
| UV Source Stride Register for Map 3 | UVSSTR3 | R/W | H'0B3C | 32 |
| UV Source Start Address Register for Map 4 | UVSSAR4 | R/W | H'0B40 | 32 |
| UV Source Stride Register for Map 4 | UVSSTR4 | R/W | H'0B44 | 32 |
| UV Destination Stride Register | UVDSTR | R/W | H'0B48 | 32 |

**Table 56.5 Register Configuration (1) - Rendering Control Registers**

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| Rendering Mode Register | CMRCR | R | H'0054 | 32 |
| Rendering Mode Set Register | CMRCSR | —/W | H'0058 | 32 |
| Rendering Mode Clear Register | CMRCCR | —/W | H'005C | 32 |
| Triangle Mode Register | TRIMR | R | H'0060 | 32 |
| Triangle Mode Set Register | TRIMSR | —/W | H'0064 | 32 |
| Triangle Mode Clear Register | TRIMCR | —/W | H'0068 | 32 |
| Triangle Color Register | TRICR | R/W | H'006C | 32 |
| Source and Destination Coordinate Decimal Point Register | UVDPOR | R/W | H'0070 | 32 |
| Source Width Register | SUSR | R/W | H'0074 | 32 |
| Source Height Register | SVSR | R/W | H'0078 | 32 |
| MIN Clipping X Register | XMINR | R/W | H'0080 | 32 |
| MIN Clipping Y Register | YMINR | R/W | H'0084 | 32 |
| MAX Clipping X Register | XMAXR | R/W | H'0088 | 32 |
| MAX Clipping Y Register | YMAXR | R/W | H'008C | 32 |
| Mesh Generation X Size Register | AMXSR | R/W | H'0090 | 32 |
| Mesh Generation Y Size Register | AMYSR | R/W | H'0094 | 32 |
| Mesh Generation X Start Register | AMXOR | R/W | H'0098 | 32 |
| Mesh Generation Y Start Register | AMYOR | R/W | H'009C | 32 |
| Triangle Color Register 2 | TRICR2 | R/W | H'00A0 | 32 |
| Triangle Mode Register 2 | TRIMR2 | R | H'00A4 | 32 |
| Triangle Mode Set Register 2 | TRIMSR2 | —/W | H'00A8 | 32 |
| Triangle Mode Clear Register 2 | TRIMCR2 | —/W | H'00AC | 32 |
| Minimum Luminance Correction Y Register | YLMINR | R/W | H'00B0 | 32 |
| Minimum Hue Correction U Register | UBMINR | R/W | H'00B4 | 32 |
| Minimum Hue Correction V Register | VRMINR | R/W | H'00B8 | 32 |
| Maximum Luminance Correction Y Register | YLMAXR | R/W | H'00BC | 32 |
| Maximum Hue Correction U Register | UBMAXR | R/W | H'00C0 | 32 |
| Maximum Hue Correction V Register | VRMAXR | R/W | H'00C4 | 32 |
| Correction Decimal Point Register | CPDPOR | R/W | H'00D0 | 32 |
| Luminance Correction Parameter Y Register | YLCPR | R/W | H'00D4 | 32 |
| Hue Correction Parameter U Register | UBCPR | R/W | H'00D8 | 32 |
| Hue Correction Parameter V Register | VRCPR | R/W | H'00DC | 32 |
| Triangle color register 3 | TRICR3 | R/W | H’00E0 | 32 |
| Rendering Mode Register 2 | CMRCR2 | R | H'00E4 | 32 |
| Rendering Mode Set Register 2 | CMRCSR2 | —/W | H'00E8 | 32 |
| Rendering Mode Clear Register 2 | CMRCCR2 | —/W | H'00EC | 32 |
| Non-blocking Cache Mode Register | NCMR | R/W | H’00F0 | 32 |
| Destination Cache Control Register | DCCR | R/W | H’0114 | 32 |

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| Source Cache Control Register | SCCR | R/W | H’0118 | 32 |
| Performance Counter Control register | PEFCCR | R/W | H’0120 | 32 |
| Total Cycle Count Register | PEFCTCR | R | H’0124 | 32 |
| Total Cache Miss Cycle Count Register | PEFCTMR | R | H’0128 | 32 |
| Maximum Cache Miss Penalty Register | PEFCMAXPR | R | H’012C | 32 |
| Minimum Cache Miss Penalty Register | PEFCMINPR | R | H’0130 | 32 |
| Number of Destination Cache Access Register | PEFCDCAR | R | H’0134 | 32 |
| Number of Destination Cache Miss Register | PEFCDCMR | R | H’0138 | 32 |
| Number of Source Cache Access Register | PEFCSCAR | R | H’0144 | 32 |
| Number of Source Cache Miss Register | PEFCSCMR | R | H’0148 | 32 |

**Table 56.6 Register Configuration (1) – Lookup Table Data Registers**

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| Lookup Table Data Register n | LUTDRn | —/W | H'1000 – H'1FFC | 32 |

**Table 56.7 Register Configuration (1) – Synchronization Control Registers**

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| Synchronization Control Register 0 | SYNCCR0 | R/W | H’02A0 | 32 |
| Synchronization Control Register 1 | SYNCCR1 | R/W | H’02A4 | 32 |
| Synchronization Control Register 2 | SYNCCR2 | R/W | H’02A8 | 32 |
| Synchronization Control Register 3 | SYNCCR3 | R/W | H’02AC | 32 |

### 56.1.3 Register Configuration (2)

Writing to some registers in the IMR-LX6 while it is drawing is possible by using a WTS, WTL, or WTL2 instruction in the DL or through access from a CPU. In the following tables, √ indicates that the register can be rewritten by using the WTS, WTL, or WTL2 instruction or from a CPU during drawing operations. H: CPG reset, S: Software reset by SWRST bit of CR register.

**Table 56.8 Register Configuration (2) - Control Registers**

| Name | Symbol | WTL/WTL2/WTS | CPU Write Access during Drawing | Reset H/SH |
|---|---|---|---|---|
| Control Register | CR | — | — | H |
| Status Register | SR | — | — | H/SH |
| Status Clear Register | SRCR | — | √ | — |
| Interrupt Control Register | ICR | √ | — | H/SH |
| Interrupt Mask Register | IMR | √ | — | H/SH |
| DL Stack Pointer Register | DLSP | — | — | — |
| DL Status Register | DLPR | — | — | — |
| Executed DL Status Register | EDLR | √ | — | H/SH |

**Table 56.9 Register Configuration (2) - Memory Control Registers**

| Name | Symbol | WTL/WTL2/WTS | CPU Write Access during Drawing |
|---|---|---|---|
| DL Start Address Register | DLSAR | — | √ |
| Destination Start Address Register | DSAR | √ | — |
| Source Start Address Register | SSAR | √ | — |
| Destination Stride Register | DSTR | √ | — |
| Source Stride Register | SSTR | √ | — |
| Destination Start Offset Address Register | DSOR | √ | — |
| Source Start Address Offset Register | SSAOR | √ | — |
| Source Start Address Offset Register for Map 1 | SSAOR1 | √ | — |
| Source Start Address Offset Register for Map 2 | SSAOR2 | √ | — |
| Source Start Address Offset Register for Map 3 | SSAOR3 | √ | — |
| Source Start Address Offset Register for Map 4 | SSAOR4 | √ | — |
| UV Source Start Address Offset Register | UVSSAOR | √ | — |
| UV Source Start Address Offset Register 1 | UVSSAOR1 | √ | — |
| UV Source Start Address Offset Register 2 | UVSSAOR2 | √ | — |
| UV Source Start Address Offset Register 3 | UVSSAOR3 | √ | — |
| UV Source Start Address Offset Register 4 | UVSSAOR4 | √ | — |
| Destination Start Address Offset Register | DSAOR | √ | — |
| X/Y Offset Register | XYOFSR | √ | — |
| u/v Offset Register | UVOFSR | √ | — |
| Source Start Address Register for Map 1 | SSAR1 | √ | — |
| Source Stride Register for Map 1 | SSTR1 | √ | — |
| Source Start Address Register for Map 2 | SSAR2 | √ | — |
| Source Stride Register for Map 2 | SSTR2 | √ | — |
| Source Start Address Register for Map 3 | SSAR3 | √ | — |
| Source Stride Register for Map 3 | SSTR3 | √ | — |
| Source Start Address Register for Map 4 | SSAR4 | √ | — |
| Source Stride Register for Map 4 | SSTR4 | √ | — |
| UV Source Start Address Register | UVSSAR | √ | — |
| UV Source Stride Register | UVSSTR | √ | — |
| UV Source Start Address Register for Map 1 | UVSSAR1 | √ | — |
| UV Source Stride Register for Map 1 | UVSSTR1 | √ | — |

| Name | Symbol | WTL/WTL2/WTS | CPU Write Access during Drawing |
|---|---|---|---|
| UV Source Start Address Register for Map 2 | UVSSAR2 | √ | — |
| UV Source Stride Register for Map 2 | UVSSTR2 | √ | — |
| UV Source Start Address Register for Map 3 | UVSSAR3 | √ | — |
| UV Source Stride Register for Map 3 | UVSSTR3 | √ | — |
| UV Source Start Address Register for Map 4 | UVSSAR4 | √ | — |
| UV Source Stride Register for Map 4 | UVSSTR4 | √ | — |
| UV Destination Stride Register | UVDSTR | √ | — |

**Table 56.10 Register Configuration (2) - Rendering Control Registers**

| Name | Symbol | WTL/WTL2/WTS | CPU Write Access during Drawing |
|---|---|---|---|
| Rendering Mode Register | CMRCR | — | — |
| Rendering Mode Set Register | CMRCSR | √ | — |
| Rendering Mode Clear Register | CMRCCR | √ | — |
| Triangle Mode Register | TRIMR | — | — |
| Triangle Mode Set Register | TRIMSR | √ | — |
| Triangle Mode Clear Register | TRIMCR | √ | — |
| Triangle Color Register | TRICR | √ | — |
| Source and Destination Coordinate Decimal Point Register | UVDPOR | √ | — |
| Source Width Register | SUSR | √ | — |
| Source Height Register | SVSR | √ | — |
| MIN Clipping X Register | XMINR | √ | — |
| MIN Clipping Y Register | YMINR | √ | — |
| MAX Clipping X Register | XMAXR | √ | — |
| MAX Clipping Y Register | YMAXR | √ | — |
| Mesh Generation X Size Register | AMXSR | √ | — |
| Mesh Generation Y Size Register | AMYSR | √ | — |
| Mesh Generation X Start Register | AMXOR | √ | — |
| Mesh Generation Y Start Register | AMYOR | √ | — |
| Triangle Color Register 2 | TRICR2 | √ | — |
| Triangle Mode Register 2 | TRIMR2 | — | — |
| Triangle Mode Set Register 2 | TRIMSR2 | √ | — |
| Triangle Mode Clear Register 2 | TRIMCR2 | √ | — |

| Name | Symbol | WTL/WTL2/WTS | CPU Write Access during Drawing |
|---|---|---|---|
| Minimum Luminance Correction Y Register | YLMINR | √ | — |
| Minimum Hue Correction U Register | UBMINR | √ | — |
| Minimum Hue Correction V Register | VRMINR | √ | — |
| Maximum Luminance Correction Y Register | YLMAXR | √ | — |
| Maximum Hue Correction U Register | UBMAXR | √ | — |
| Maximum Hue Correction V Register | VRMAXR | √ | — |
| Correction Decimal Point Register | CPDPOR | √ | — |
| Luminance Correction Parameter Y Register | YLCPR | √ | — |
| Hue Correction Parameter U Register | UBCPR | √ | — |
| Hue Correction Parameter V Register | VRCPR | √ | — |
| Triangle Color Register 3 | TRICR3 | √ | — |
| Rendering Mode Register 2 | CMRCR2 | — | — |
| Rendering Mode Set Register 2 | CMRCSR2 | √ | — |
| Rendering Mode Clear Register 2 | CMRCCR2 | √ | — |
| Non-blocking Cache Mode Register | NCMR | — | — |
| Destination Cache Control Register | DCCR | √ | — |
| Source Cache Control Register | SCCR | √ | — |
| Performance Counter Control register | PEFCCR | √ | — |
| Total Cycle Count Register | PEFCTCR | — | — |
| Total Cache Miss Cycle Count Register | PEFCTMR | — | — |
| Maximum Cache Miss Penalty Register | PEFCMAXPR | — | — |
| Minimum Cache Miss Penalty Register | PEFCMINPR | — | — |
| Number of Destination Cache Access Register | PEFCDCAR | — | — |
| Number of Destination Cache Miss Register | PEFCDCMR | — | — |
| Number of Source Cache Access Register | PEFCSCAR | — | — |
| Number of Source Cache Miss Register | PEFCSCMR | — | — |

**Table 56.11 Register Configuration (2) - Lookup Table Data Registers**

| Name | Symbol | WTL/WTL2/ WTS | CPU Write Access during Drawing |
|---|---|---|---|
| Lookup Table Data Register n | LUTDRn | √* | — |

Note: Writing to this register by using the WTL or WTS instruction is not possible.

**Table 56.12 Register Configuration (2) - Synchronization Control Registers**

| Name | Symbol | WTL/WTL2/ WTS | CPU Write Access during Drawing |
|---|---|---|---|
| Synchronization Control Register 0 | SYNCCR0 | — | — |
| Synchronization Control Register 1 | SYNCCR1 | — | — |
| Synchronization Control Register 2 | SYNCCR2 | — | — |
| Synchronization Control Register 3 | SYNCCR3 | — | — |

## 56.2 Register Description

**[Legend]**

Initial value: Register value after a power-on reset
—: Undefined value
R/W: Readable/writable. The written value can be read.
R: Read-only. The write value should be the same as the read value. Only modify the target bit (Read Modify Write) because the initial value should not be changed.
-/W: Write-only. The read value is undefined.

### 56.2.1 Control Register (CR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SWRST | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 | SWRST | 0 | R/W | Software Reset<br>Writing 1 to this bit applies a software reset and writing 0 cancels a software reset.<br>0: Cancels a software reset for this module.<br>1: Applies a software reset to this module.<br>The following registers are not initialized by a software reset:<br>CR, DSAR, SSAR, DSTR, SSTR, DSOR, SSAR1, SSTR1, SSAR2, SSTR2, SSAR3, SSTR3, SSAR4, SSTR4, DCCR, SCCR, SSAOR, SSAORn, UVSSAOR, UVSSAORn, DSAOR, UVSSTR, UVSSARn, UVSSTRn, UVDSTR . |
| 14 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 0 | RS | 0 | R/W | Rendering Start<br>Writing 1 to this bit reads the DL starting at the address in DLSAR and starts execution. This bit is always read as 0.<br>0: Does not start rendering.<br>1: Starts rendering. Writing 1 is prohibited until the TRA bit in SR is set. |

Note: This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.2 Status Register (SR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | WERR | WOVF | — | — | — | — | REN | — | — | INT | IER | TRA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 11 | WERR | 0 | R | Indicates that this module has received the WUP signal from another module while it is not executing a display list. This bit is cleared to 0 by writing 1 to the WERRCLR bit in SRCR.<br>0: This module has not received the WUP signal while it is not executing a display list.<br>1: This module has received the WUP signal while it is not executing a display list.<br>Note: This bit is not set to 1 when the setting of the WERRENB bit in ICR is 0.<br>This bit is only set to 1 when the WERRENB bit in ICR is set to 1 and the above condition is satisfied. |
| 10 | WOVF | 0 | R | Indicates that the number of WUP signals this module has received but not processed has exceeded the maximum number allowed. The maximum number in this product is 1. This bit is cleared to 0 by writing 1 to the WOVFCLR bit in SRCR.<br>0: The number of WUP signals that have not been processed has not exceeded the maximum number allowed.<br>1: The number of WUP signals that have not been processed has exceeded the maximum number allowed.<br>Note: This bit is not set to 1 when the setting of the WOVFENB bit in ICR is 0.<br>This bit is only set to 1 when the WOVFENB bit in ICR is set to 1 and the above condition is satisfied. |
| 9 to 6 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | REN | 0 | R | Rendering-in-Progress Flag<br>Indicates whether this IP is currently rendering data. This bit is set to 1 after the rendering start function is activated and is cleared to 0 when the TRA bit is set to 1 or an illegal instruction has occurred. When the INT instruction is executed, the setting of this bit remains 0 until the INT bit in the SR register is cleared, after which this bit is set to 1.<br>The value of this REN bit is not guaranteed when IER or INT interrupt occurs by enabling the IERENB or INTENB bit of Interrupt Control Register (ICR) at the same time.<br>0: Rendering is not in progress.<br>1: Rendering is in progress. |
| 4, 3 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 2 | INT | 0 | R | INT Instruction Decode<br>Indicates whether the INT instruction has been decoded. This bit is cleared by writing 1 to the INTCLR bit in SRCR.<br>0: The INT instruction in the DL has not been decoded.<br>1: The INT instruction in the DL has been decoded.<br>Note: When INTENB bit in ICR is 0, this bit is not set to 1 even if the INT instruction is decoded.<br>When INTENB bit in ICR is 1, this bit is set to 1 if the INT instruction is decoded. |
| 1 | IER | 0 | R | Illegal Instruction Decode<br>Indicates whether any illegal instruction has been decoded. This bit is cleared by writing 1 to the IERCLR bit in SRCR. Whether or not a given instruction is an illegal instruction is determined using the OP CODE field assigned to bits 31 to 24 of each instruction; the N field assigned to bits 15 to 0 of the TRI, WTL, and NOP instructions; and the N field assigned to bits 9 to 0 of the WTL2 instruction. Tests on illegal instructions are not performed in terms of the address or write data of the WTL/WTL2 instructions or the vertex data of the TRI instruction.<br>0: No illegal instruction has been decoded in the DL.<br>1: An illegal instruction has been decoded in the DL.<br>Notes: 1. These OP CODE values are not detected as illegal instructions: OP CODE described in section 56.4, Display List and a reserved value (H'84, H'85, H'90, and H'91)<br>2. When IERENB bit in ICR is 0, this bit is not set to 1 even if illegal instructions are decoded.<br>When IERENB bit in ICR is 1, this bit is set to 1 if illegal instructions are decoded.<br>3. The WUP and SLP instructions are not detected as illegal in those channels where executing these instructions is prohibited. See Table 56.21 for the applicable channels.<br>4. The instructions listed in section 56.5.4 are detected as illegal in those channels that do not have the rotator & scaler extension. See Table 56.2 for the applicable channels. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | TRA | 0 | R | Trap<br>Indicates whether rendering operation has been completed. Specifically, this bit is set to 1 after the TRAP instruction is decoded, rendering is completed, and all the dirty lines in the destination cache are written to the memory.<br>0: Rendering has not been started or is in progress.<br>1: The TRAP instruction has been decoded and rendering has been completed.<br>Note: When TRAENB bit in ICR is 0, this bit is not set to 1 even if the TRAP instruction is decoded.<br>When TRAENB bit in ICR is 1, this bit is set to 1 if the TRAP instruction is decoded. |

### 56.2.3 Status Clear Register (SRCR)

Writing 1 to a bit in SRCR clears the corresponding status bit in the status register (SR).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | WERR CLR | WOVF CLR | — | — | — | — | — | — | — | INTCLR | IERCL R | TRACL R |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | —/W | —/W | R | R | R | R | R | R | R | —/W | —/W | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 11 | WERRCLR | 0 | —/W | Writing 1 to this bit clears the WERR bit in SR.<br>This bit is always read as 0. |
| 10 | WOVFCLR | 0 | —/W | Writing 1 to this bit clears the WOVF bit in SR.<br>This bit is always read as 0. |
| 9 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 2 | INTCLR | 0 | —/W | Writing 1 to this bit clears the INT bit in SR.<br>This bit is always read as 0. |
| 1 | IERCLR | 0 | —/W | Writing 1 to this bit clears the IER bit in SR.<br>This bit is always read as 0. |
| 0 | TRACLR | 0 | —/W | Writing 1 to this bit clears the TRA bit in SR.<br>This bit is always read as 0. |

### 56.2.4 Interrupt Control Register (ICR)

The effective bits in the ICR are used to enable setting of the corresponding bit in the status register (SR) in response to the corresponding interrupt sources. To allow the generation of an interrupt, this register should be set to enable setting of the corresponding bit in SR and the interrupt mask register (IMR) should also be set for non-masking of the corresponding interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | WERRENB | WOVFENB | — | — | — | — | — | — | — | INTENB | IERENB | TRAENB |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 11 | WERRENB | 1 | R/W | 0: Disables the generation of an interrupt when this module receives the WUP signal from another module while it is not executing a display list.<br>1: Enables the generation of an interrupt when this module receives the WUP signal from another module while it is not executing a display list. |
| 10 | WOVFENB | 1 | R/W | 0: Disables the generation of an interrupt when the number of WUP signals this module has received but not processed exceeds the maximum number allowed.<br>1: Enables the generation of an interrupt when the number of WUP signals this module has received but not processed exceeds the maximum number allowed. |
| 9 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 4, 3 | — | All 1 | R | Reserved<br>These bits are always read as 1.<br>The write value should always be 1. Otherwise, the operation cannot be guaranteed. |
| 2 | INTENB | 1 | R/W | 0: Disables the generation of INT interrupts.<br>1: Enables the generation of an interrupt when an INT instruction is decoded from the DL unless interrupt generation is masked with the INM bit in IMR. |
| 1 | IERENB | 1 | R/W | 0: Disables the generation of IER interrupts generation.<br>1: Enables the generation of an interrupt when an illegal instruction is decoded unless interrupt generation is masked with the IEM bit in IMR. |
| 0 | TRAENB | 1 | R/W | 0: Disables the generation of TRAP interrupts.<br>1: Enables the generation of an interrupt when a TRAP instruction is decoded unless interrupt generation is masked with the TRAM bit in IMR. |

### 56.2.5 Interrupt Mask Register (IMR)

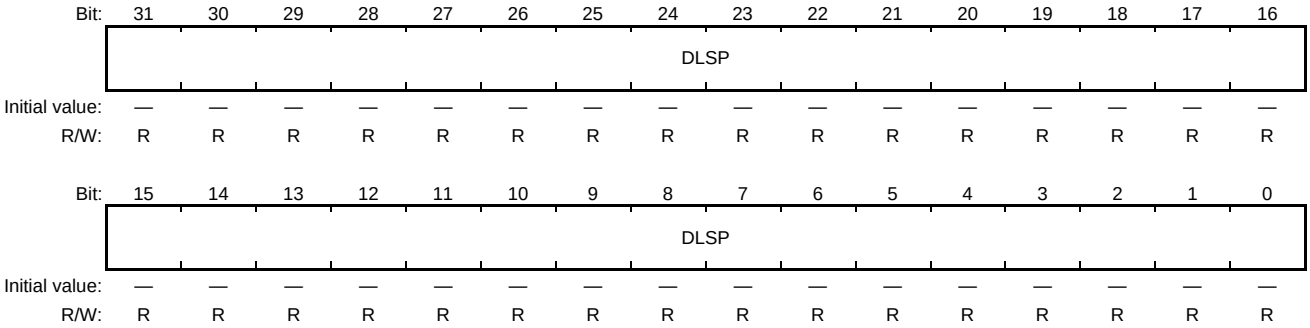
The effective bits in the IMR are used for masking or non-masking of the output of the corresponding interrupt to the interrupt controller when the corresponding interrupt source bits are set in the SR. To allow interrupt generation, this register should be set for non-masking of the corresponding interrupt output and the interrupt control register (ICR) should be set to enable reflection of the corresponding interrupt source in the corresponding bit in SR.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | WERRM | WOVFM | — | — | — | — | — | — | — | INM | IEM | TRAM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R/W | R/W | R/W |

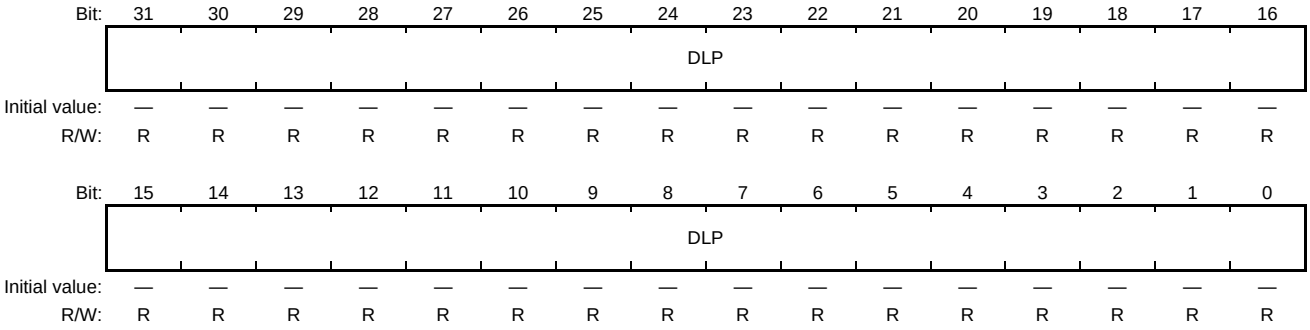
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 11 | WERRM | 0 | R/W | 0: The generation of interrupts when this module receives the WUP signal from another module while it is not executing a display list is not masked.<br>1: The generation of interrupts when this module receives the WUP signal from another module while it is not executing a display list is masked. |
| 10 | WOVFM | 0 | R/W | 0: The generation of interrupts when the number of WUP signals this module has received but not processed exceeds the maximum number allowed is not masked.<br>1: The generation of interrupts when the number of WUP signals this module has received but not processed exceeds the maximum number allowed is masked. |
| 9 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 4, 3 | — | All 1 | R | Reserved<br>These bits are always read as 1.<br>The write value should always be 1. Otherwise, the operation cannot be guaranteed. |
| 2 | INM | 1 | R/W | 0: The generation of INT interrupts is not masked.<br>1: The generation of INT interrupts is masked. |
| 1 | IEM | 1 | R/W | 0: The generation of IER interrupts is not masked.<br>1: The generation of IER interrupts is masked. |
| 0 | TRAM | 1 | R/W | 0: The generation of TRAP interrupts is not masked.<br>1: The generation of TRAP interrupts is masked. |

### 56.2.6 DL Stack Pointer Register (DLSP)



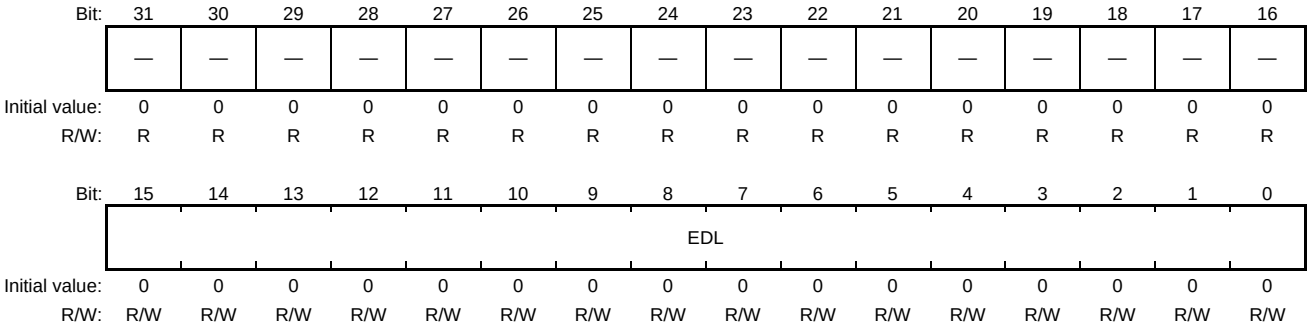

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DLSP | Undefined | R | DL Stack Pointer<br>The return address to be used upon RET instruction execution, which is saved upon GOSUB instruction execution, can be read out. |

### 56.2.7 DL Status Register (DLPR)



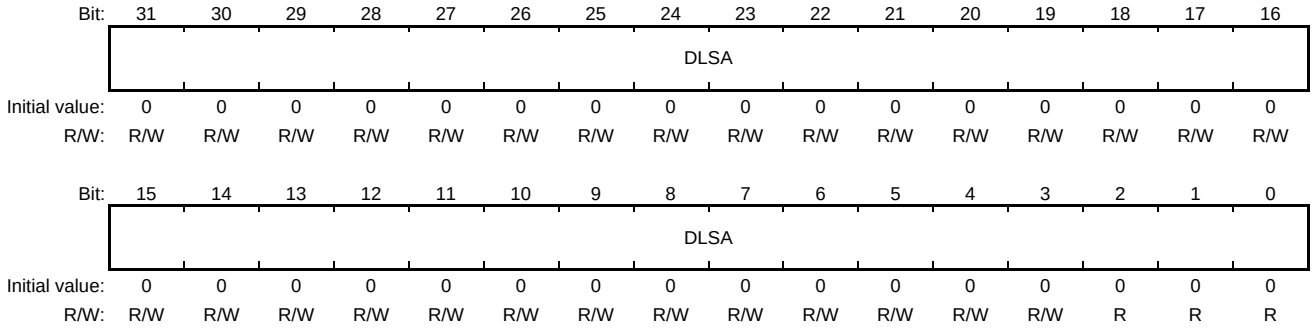

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DLP | Undefined | R | DL Pointer<br>The address of the DL immediately following the currently executed DL can be read out. |

### 56.2.8 Executed DL Status Register (EDLR)



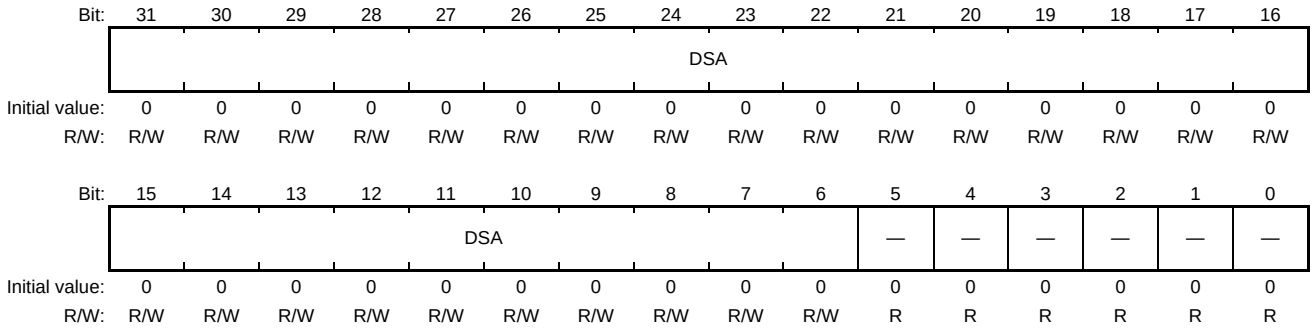

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 0 | EDL | H'0000 | R/W | Executed DL Status<br>Any value can be set to these bits from the DL using the WTL, WTL2, or WTS instruction. The set value can be read by the CPU.<br>These bits can be conveniently used to determine the executed DL by the CPU. |

### 56.2.9 DL Start Address Register (DLSAR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DLSA | H'0000 0000 | [31:3]:R/W<br>[2:0]:R | DL Start Address<br>Specifies the DL start address in 8-byte alignment. The least significant three bits must always be 0. |

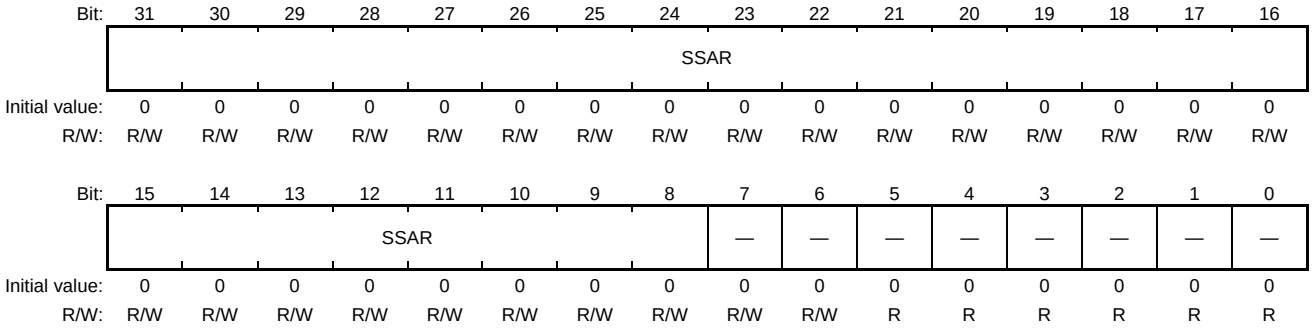
### 56.2.10 Destination Start Address Register (DSAR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | DSA | H'000 0000 | R/W | Destination Start Address<br>Specifies the start address of the destination area in 64-byte alignment. To modify the value during execution, execute the SYNCM instruction before modification. |
| 5 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Note: This register is not initialized by the SWRST bit in the control register (CR).

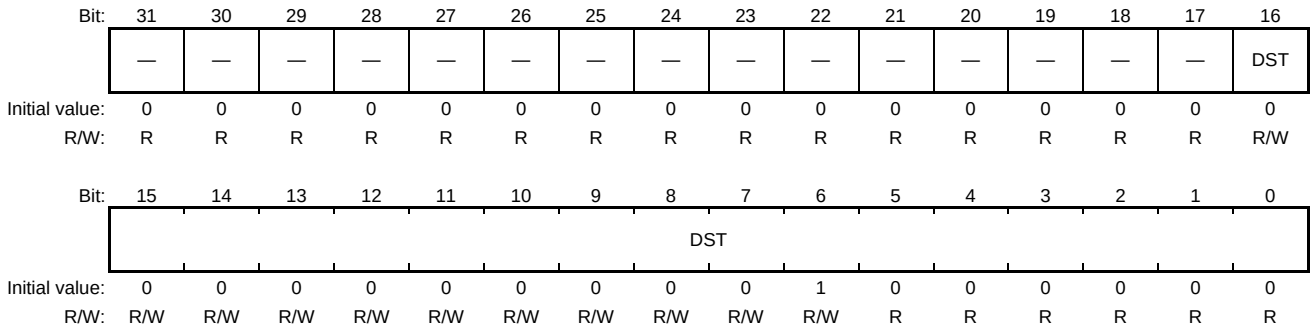
### 56.2.11 Source Start Address Register (SSAR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | SSAR | H'000 0000 | R/W | SRC Start Address<br>Specifies the start address of the source area in 256-byte alignment when reading texture data from the external memory. To modify the value during execution, execute the SYNCM instruction before modification. |
| 7, 6 | — | All 0 | R/W | Reserved<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 5 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Note: This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.12 Destination Stride Register (DSTR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DST | H'00040 | [16:6]: R/W<br>[5:0]: R | Specifies the memory width of the DST area in bytes within a range from 64 to 65,536 bytes in 64-byte alignment. In tile addressing mode, quadruple the value of the setting is used as the memory width. To modify the value during execution, execute the SYNCM instruction before modification. The least significant 6 bits must always be 0. |

Notes: 1. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.13 Source Stride Register (SSTR)

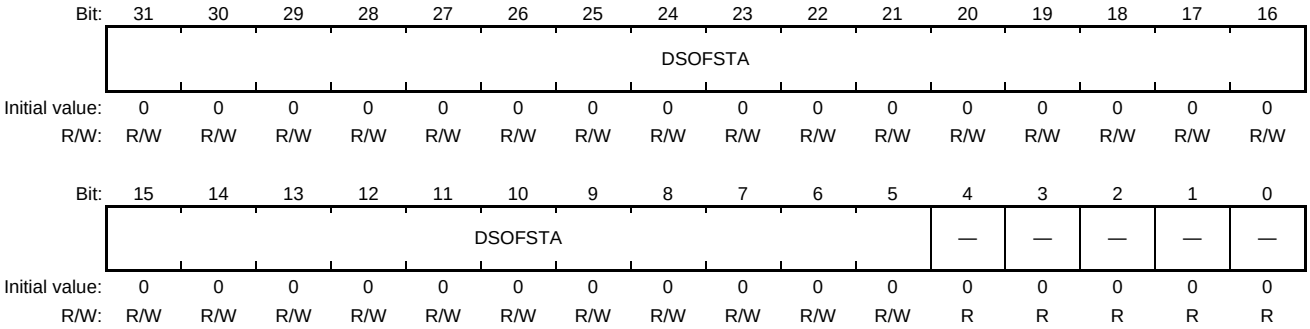
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSTR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | SSTR | H'00100 | [16:6]: R/W<br>[5:0]: R | Specifies the memory width of the SRC area in bytes within a range from 256 to 65,536 bytes in 256-byte alignment in the linear addressing mode when reading texture data from the external memory. In tile addressing mode, quadruple the value of the setting is used as the memory width. To modify the value during execution, execute the SYNCM instruction before modification. The least significant 6 bits must always be 0. |

Notes: 1. This register is not initialized by the SWRST bit in the control register (CR).

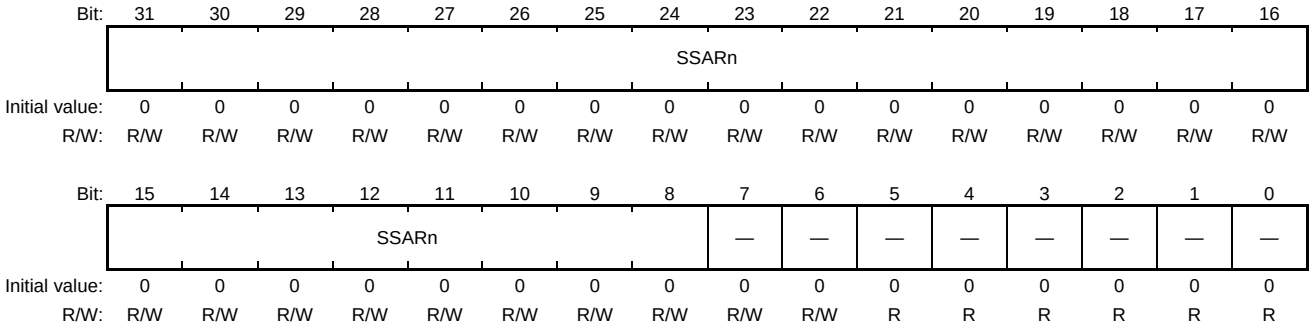
### 56.2.14 Destination Start Offset Address Register (DSOR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | DSOFSTA | H'000 0000 | R/W | In Y/UV separate output mode, set an offset value for the UV output destination address that corresponds to the output destination address of Y. The offset must be a signed 32-bit value. Specify the value in 64-byte alignment.<br>The write value to bit 5 should always be 0.<br>To modify the value during execution, execute the SYNCM instruction before modification. |
| 4 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Note: This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.15 Source Start Address Register for Map n (SSARn)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | SSARn | H'000 0000 | R/W | SRC Start Address for Map n<br>Specifies the address where the source area for the n-th map starts on a 256-byte boundary when reading texture data from the external memory. When modifying the value during execution of a DL, execute the SYNCM instruction before doing so. |
| 7, 6 | — | All 0 | R/W | Reserved<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 5 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Notes: n = 1 to 4.
This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.16 Source Stride Register for Map n (SSTRn)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SSTRn |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSTRn | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | SSTRn | H'00100 | [16:6]: R/W<br>[5:0]: R | Specifies the memory width of the source area for the n-th map in bytes as an integer multiple of 256 within the range from 256 to 65,536 bytes in the linear addressing mode when reading texture data from the external memory. In tile addressing mode, quadruple the value of this setting is used as the memory width. When modifying the value during execution of a DL, execute the SYNCM instruction before doing so. The less significant 6 bits must always be 0. |

Notes: 1. n = 1 to 4.
2. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.17 Source Start Address Offset Register (SSAOR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSAOR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSAOR | | | | | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | SSAOR | H'000 0000 | R/W | SRC Start Address Offset<br>Specifies the offset address of the source area in 256-byte alignment when reading texture data from the external memory. To modify the value during execution, execute the SYNCM instruction before modification. |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Note: 1. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.18 Source Start Address Offset Register n(SSAORn)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSAORn | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSAORn | | | | | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | SSAORn | H'000 0000 | R/W | SRC Offset Address for Map n<br>Specifies the offset address where the source area for the n-th map starts on a 256-byte boundary when reading texture data from the external memory. When modifying the value during execution of a DL, execute the SYNCM instruction before doing so. |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Notes: 1. n = 1 to 4.
2. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.19 UV Source Start Address Register (UVSSAR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSAR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSAR | | | | | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | UVSSAR | H'00 0000 | R/W | SRC Start Address for UV plane<br>Sets the start address in 256-byte unit when reading the UV plane in Semi-planar format as texture data from external memory. To modify the value during execution, execute the SYNCM instruction before modification. |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Note: 1. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.20 UV Source Stride Register (UVSSTR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | UVSSTR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSTR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | UVSSTR | H'00100 | [16:6]: R/W<br>[5:0]: R | Sets the memory width of texture data read from the external memory of UV plane in Semi-planar format to a range from 256 bytes to 65,536 bytes and in 256-byte unit. In Linear Addressing mode, the value set in this register is used as-is, while in Tile addressing mode, the value is quadrupled and used as the memory width. To modify the value during execution, execute the SYNCM instruction before modification. The least significant 6 bits must always be 0. |

Notes: 1. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.21 UV Source Start Address Register n (UVSSARn)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSARn | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSARn | | | | | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | UVSSARn | H'00 0000 | R/W | SRC Start Address for UV plane for Map n<br>Sets the start address in 256-byte unit when reading the UV plane for the n-th Map in Semi-planar format as texture data from external memory. To modify the value during execution, execute the SYNCM instruction before modification. |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Note: 1. n = 1 to 4.
2. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.22 UV Source Stride Register n (UVSSTRn)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | UVSSTRn |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSTRn | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | UVSSTRn | H'0100 | [16:6]: R/W<br>[5:0]: R | Sets the memory width of texture data read from the external memory of UV plane for the n-th Map in Semi-planar format to a range from 256 bytes to 65,536 bytes and in 256-byte unit. In Linear Addressing mode, the value set in this register is used as-is, while in Tile addressing mode, the value is quadrupled and used as the memory width. To modify the value during execution, execute the SYNCM instruction before modification. The least significant 6 bits must always be 0. |

Notes: 1. n = 1 to 4.
2. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.23 UV Destination Stride Register (UVDSTR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | UVDSTR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVDSTR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | UVDSTR | H'0040 | [16:6]: R/W<br>[5:0]: R | Sets the memory width of destination data of UV plane in Semi-planar format to a range from 64 bytes to 65,536 bytes and in 64-byte unit. If the UVDSTSEL bit of DCCR register is 1, the setting of this register is used as memory stride, while if the bit is 0, the setting of DSTR register is used as memory stride. In Linear Addressing mode, the value set in this register is used as-is, while in Tile addressing mode, the value is quadrupled and used as the memory width. To modify the value during execution, execute the SYNCM instruction before modification. The least significant 6 bits must always be 0. |

Notes: 1. This register is not initialized by the SWRST bit in the control register (CR)

### 56.2.24 Destination Start Address Offset Register (DSAOR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSAO | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSAO | | | | | | | | | | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | DSAO | H'000 0000 | R/W | Destination Offset Address<br>Specifies the offset address of the destination area in 64-byte alignment. To modify the value during execution, execute the SYNCM instruction before modification. |
| 5 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Note: 1. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.25 UV Source Start Address Offset Register (UVSSAOR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSAOR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSAOR | | | | | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | UVSSAOR | H'00 0000 | R/W | SRC Start Address Offset for UV plane<br>Sets the offset of start address in 256-byte unit when reading the UV plane in Semi-planar format as texture data from external memory. To modify the value during execution, execute the SYNCM instruction before modification. |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Note: 1. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.26 UV Source Start Address Offset Register for Map n (UVSSAORn)

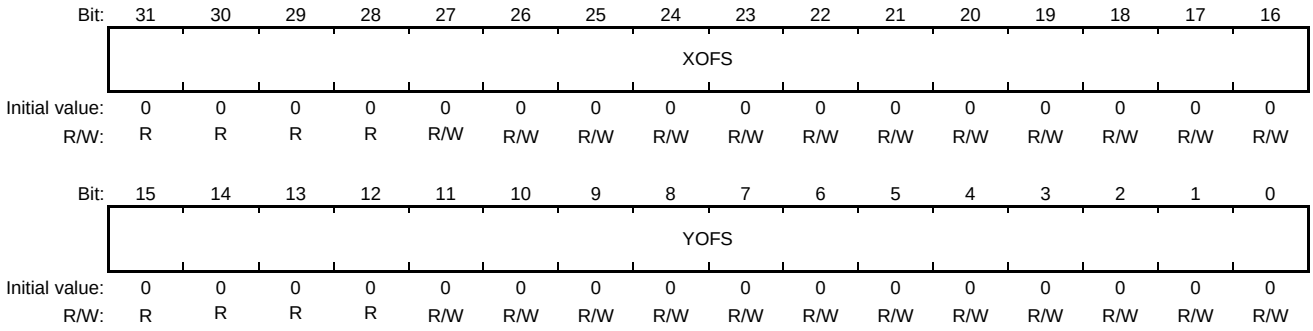
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSAORn | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSSAORn | | | | | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | UVSSAORn | H'00 0000 | R/W | SRC Start Address Offset for UV plane for Map n<br>Sets the offset of start address in 256-byte unit when reading the UV plane for the n-th Map in Semi-planar format as texture data from external memory. To modify the value during execution, execute the SYNCM instruction before modification. |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Notes: 1. n = 1 to 4.
2. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.27 X/Y Offset Register (XYOFSR)



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | H'0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be same as bit 27 extended. Otherwise, the operation cannot be guaranteed. |
| 27 to 16 | XOFS | H'000 | R/W | The value, obtained by adding the offset specified in this field to the X coordinate automatically generated or specified by DL, is used for processing as X coordinate.<br>Specify the offset in signed 12-bit format.<br>When the DDP bit of the UVDPOR register is 1, the lower 2 bits are treated as fractional part, while the remaining ones are treated as sign bit and integer part. |
| 15 to 12 | — | H'0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be same as bit 11 extended. Otherwise, the operation cannot be guaranteed. |
| 11 to 0 | YOFS | H'000 | R/W | The value, obtained by adding the offset specified in this field to the Y coordinate automatically generated or specified by DL, is used for processing as Y coordinate.<br>Specify the offset in signed 12-bit format.<br>When the DDP bit of the UVDPOR register is 1, the lower 2 bits are treated as fractional part, while the remaining ones are treated as sign bit and integer part. |

### 56.2.28 u/v Offset Register (UVOFSR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UOFS | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VOFS | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | UOFS | H'0000 | R/W | The value, obtained by adding the offset specified in this field to the u coordinate automatically generated or specified by DL, is used for processing as u coordinate.<br>Specify the offset in signed 16-bit format.<br>For the digits specified by the value in UVDPO field of UVDPOR register, the lower bits are treated as fractional part, while the remaining ones are treated as sign bit and integer part. |
| 15 to 0 | VOFS | H'0000 | R/W | The value, obtained by adding the offset specified in this field to the v coordinate automatically generated or specified by DL, is used for processing as v coordinate.<br>Specify the offset in signed 16-bit format.<br>For the digits specified by the value in UVDPO field of UVDPOR register, the lower bits are treated as fractional part, while the remaining ones are treated as sign bit and integer part. |

### 56.2.29 Rendering Mode Register (CMRCR)

CMRCR indicates the output mode, bit precision, and correction mode for the drawing results. To set each bit in this register to 1, write 1 to the corresponding bit in CMRCSR. To clear each bit to 0, write 1 to the corresponding bit in CMRCCR.

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

Note: CMRCR must be initialized (cleared) by using CMRCCR after the software reset.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFPE | CP32E | SUV1416 | | DUV1416 | | SY1416 | | DY1416 | | — | — | CLSM | CLOM | LUSM | LUOM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CP16E | YCM | UVS | SY12 | SY10 | YOM | Y12 | Y10 | YISM | SUV | | DUV | | CLCE | LUCE | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EFPE | 0 | R | Enables or disables the extended filtering pipeline.<br>0: Disabled<br>1: Enabled<br>The settings of bits SUV1416, DUV1416, SY1416, and DY1416 of this register and bits SHFVAL, SHFE, and RDE bits in TRIMR are only effective when this bit is set to 1.<br>The results of processing may not be the same with each setting (enabled or disabled), even though the same pixel format is specified.<br>This bit should be set to 1 when using DYP mode.<br>Setting this bit to 1 is prohibited when bilinear filtering is not in use (the BFE bit in TRIMR is 0). Also, setting this bit to 1 is prohibited when conversion by the LUT is in use (the LUTE in CMRCR2 is 1) and copying 16-bit data mode is enabled (the CP16E in CMRCR is 1). |
| 30 | CP32E | 0 | R | Copy 32-bit mode<br>Enables copying 32-bit data as it is without complementing the data.<br>When this bit is set to 1 and Texture mapping is used:<br>set the TME bit in the TRIMR register to 1;<br>set all the bits except the CP32E bit in the CMRCR register to 0; set all the bits in the CMRCR2 and TRIMR2 registers to 0;<br>set all the bits except the TME, TCM, AUTOSG and AUTODG bits in the TRIMR register to 0; and<br>set RSCR.RSE bit to 0, Otherwise operation is not guaranteed.<br>When this bit is set to 1 and 32-bits single color is used:<br>Set all the bits (TME = 0 to use single color mode), except the TCM, AUTOSG and AUTODG in the TRIMR register to 0;<br>Set all the bits except the CP32E bit in the CMRCR register to 0.<br>Set all the bits in the CMRCR2 and TRIMR2 registers to 0;<br>Set RSE bit in the RSCR register to 0;<br>Set TRICR2 as Single-Color data.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Disables copying 32-bit data.<br>1: Enables copying 32-bit data. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 29, 28 | SUV1416 | 00 | R | Specifies the pixel format of the hue of the source data as 14 or 16 bpp or makes neither specification.<br>00: Does not specify the format as 14 or 16 bpp.<br>10: Specifies the format as 14 bpp.<br>11: Specifies the format as 16 bpp.<br>Others: Setting prohibited<br>When the setting of this bit field is 00, specify the pixel format in the SUV bit field. When the setting of these bits is not 00, set the SUV bit field to 00.<br>This setting is only effective when the EFPE bit is set to 1. When the setting of the EFPE bit is 0, setting this bit field to a value other than 00 is prohibited.<br>Make the same setting in this bit field and in the DUV1416 bit field when the setting of the SHFE bit in TRIMR is 0. When the setting of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>When reading texture data from the external memory, 14-bpp precision destination data should be 16-bit data in which data is LSB-justified with 0 in the upper 2 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the source data and the color difference of the source data must have the same precision. |
| 27, 26 | DUV1416 | 00 | R | Specifies the pixel format of the hue of the destination data as 14 or 16 bpp or makes neither specification.<br>00: Does not specify the format as 14 or 16 bpp.<br>10: Specifies the format as 14 bpp.<br>11: Specifies the format as 16 bpp.<br>Others: Setting prohibited<br>When the setting of this bit field is 00, specify the pixel format in the DUV bit field. When the setting of these bits is not 00, set the DUV bit field to 00.<br>This setting is only effective when the EFPE bit is set to 1. When the setting of the EFPE bit is 0, setting this bit field to a value other than 00 is prohibited.<br>Make the same setting in this bit field and in the SUV1416 bit field when the setting of the SHFE bit in TRIMR is 0. When the setting of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>In 14-bpp precision output, 16-bit data is output in which U and V are each LSB-justified with 0 in the upper 2 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the destination data and the color difference of the destination data must have the same precision. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 25, 24 | SY1416 | 00 | R | Specifies the pixel format of the luminance of the source data as 14 or 16 bpp or makes neither specification.<br>00: Does not specify the format as 14 or 16 bpp.<br>10: Specifies the format as 14 bpp.<br>11: Specifies the format as 16 bpp.<br>Others: Setting prohibited<br>When the setting of this bit field is 00, specify the pixel format in the SY10 and SY12 bits. When the setting of these bits is not 00, set the SY10 and SY12 bits to 0.<br>This setting is only effective when the EFPE bit is set to 1. When the setting of the EFPE bit is 0, setting this bit field to a value other than 00 is prohibited.<br>Make the same setting in this bit field and in the DY1416 bit field when the setting of the SHFE bit in TRIMR is 0. When the setting of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>When reading texture data from the external memory, 14-bpp precision source data should be 16-bit data in which data is LSB-justified with 0 in the upper 2 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the source data and the color difference of the source data must have the same precision. |
| 23, 22 | DY1416 | 00 | R | Specifies the pixel format of the luminance of the destination data as 14 or 16 bpp or makes neither specification.<br>00: Does not specify the format as 14 or 16 bpp.<br>10: Specifies the format as 14 bpp.<br>11: Specifies the format as 16 bpp.<br>Others: Setting prohibited<br>When the setting of this bit field is 00, specify the pixel format in the Y10 and Y12 bits. When the setting of these bits is not 00, set the Y10 and Y12 bits to 0.<br>This setting is only effective when the EFPE bit is set to 1. When the setting of the EFPE bit is 0, setting this bit field to a value other than 00 is prohibited.<br>Make the same setting in this bit field and in the SY1416 bit field when the setting of the SHFE bit in TRIMR is 0. When the setting of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>In 14-bpp precision output, 16-bit data is output in which data is LSB-justified with 0 in the upper 2 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the destination data and the color difference of the destination data must have the same precision. |
| 21, 20 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 19 | CLSM | 0 | R | Hue Correction Scale Parameter Register Specification Mode<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Uses the hue correction scale parameters specified in the DL.<br>1: Uses the hue correction scale parameters specified by register settings (UBCPR and VRCPR registers). |
| 18 | CLOM | 0 | R | Hue Correction Offset Parameter Register Specification Mode<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Uses the hue correction offset parameters specified in the DL.<br>1: Uses the hue correction offset parameters specified by register settings (UBCPR and VRCPR registers). |
| 17 | LUSM | 0 | R | Luminance Correction Scale Parameter Register Specification Mode<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Uses the luminance correction scale parameters specified in the DL.<br>1: Uses the luminance correction scale parameter specified by register settings (YLCPR register). |
| 16 | LUOM | 0 | R | Luminance Correction Offset Parameter Register Specification Mode<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Uses the luminance correction offset parameters specified in the DL.<br>1: Uses the luminance correction offset parameter specified by register settings (YLCPR register). |
| 15 | CP16E | 0 | R | Copy 16-bit mode<br>Enables copying 16-bit data as it is without complementing the data.<br>When this bit is set to 1, set the TME bit in the TRIMR register to 1; set all the bits except the CP16E bit in the CMRCR register to 0; set all the bits in the CMRCR2 register to 0; set all the bits except the TME, TCM, AUTOSG and AUTODG bits in the TRIMR register to 0; set RSCR.RSE bit to 0; and setting in TRIMR2 register cannot be usable, otherwise operation is not guaranteed.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Disables copying 16-bit data.<br>1: Enables copying 16-bit data. |
| 14 | YCM | 0 | R | YC Mode<br>Selects either Y or UV data to be processed when reading texture data from the external memory.<br>Set this bit to 0 when the setting of the YUV422E or YUVSEMI or DYP bit in the CMRCR2 register is 1.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Processes Y.<br>1: Processes UV. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 13 | UVS | 0 | R | Specifies whether the input UV plane is in YUV420 format.<br>0: The input UV plane is not in YUV420 format<br>1: The input UV plane is in YUV420 format |
| 12 | SY12 | 0 | R | Specifies the precision of luminance processing of source data. When this bit is set to 1, set the SY10 bit to 0.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: 8-bpp precision or 10-bpp precision (10-bpp precision when SY10 is 1)<br>1: 12-bpp precision<br>When the setting of the EFPE bit in this register is 0, specify the same precision as specified by the Y12 bit being set or a higher precision. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 0, specify the same setting in this bit and in the Y12 bit. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20.<br>When reading texture data from the external memory, 12-bpp precision source data should be 16-bit data in which data is LSB-justified with 0 in the upper 4 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the source data and the color difference of the source data must have the same precision. |
| 11 | SY10 | 0 | R | Specifies the precision of luminance processing of source data.<br>When this bit is set to 1, set the SY12 bit to 0.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: 8-bpp precision or 12-bpp precision (12-bpp precision when SY12 is 1)<br>1: 10-bpp precision<br>When the setting of the EFPE bit in this register is 0, specify the same precision as specified by the Y10 bit being set or a higher precision. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 0, specify the same setting in this bit and in the Y10 bit. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20.<br>When reading texture data from the external memory, 10-bpp precision source data should be 16-bit data in which data is LSB-justified with 0 in the upper 6 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the source data and the color difference of the source data must have the same precision. |
| 10 | YOM | 0 | R | Output only Y data and discard U/V data.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | Y12 | 0 | R | Set this bit when Y data is output in 12-bpp precision.<br>When this bit is set to 1, set the Y10 bit to 0.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Outputs Y data in 8- or 10-bpp precision.<br>(10-bpp precision when Y10 is 1)<br>1: Outputs Y data in 12-bpp precision.<br>In 12-bpp precision output, 16-bit data is output in which data is LSB-justified with 0 in the upper 4 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the destination data and the color difference of the destination data must have the same precision.<br>When SY12 bit is 0 and EFPE bit is 0, setting 1 is prohibited.<br>When the setting of the EFPE bit in this register is 0, specify the same precision as specified by the SY12 bit being set or a lower precision.<br>When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 0, specify the same setting in this bit and in the SY12 bit. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20. |
| 8 | Y10 | 0 | R | Set this bit when Y data is output in 10-bpp precision.<br>When this bit is set to 1, set the Y12 bit to 0.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Outputs Y data in 8- or 12-bpp precision.<br>(12-bpp precision when Y12 is 1).<br>1: Outputs Y data in 10-bpp precision.<br>In 10-bpp precision output, 16-bit data is output in which data is LSB-justified with 0 in the upper 6 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the destination data and the color difference of the destination data must have the same precision.<br>When SY12, SY10 and EFPE bits are 0, setting 1 is prohibited.<br>When the setting of the EFPE bit in this register is 0, specify the same precision as specified by the SY10 bit being set or a lower precision.<br>When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 0, specify the same setting in this bit and in the SY10 bit. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | YISM | 0 | R | Selects the output format for YUV data.<br>This bit is valid when the setting of the YUV422E bit or YUVSEMI bit in the CMRCR2 register is 1. Set this bit to 0 when the setting of the YUV422E bit and YUVSEMI bit in the CMRCR2 register is 0. Operation cannot be guaranteed if this bit is set to 1 while the setting of the YUV422E bit or YUVSEMI bit in the CMRCR2 register is 0. When the YUV422E bit and YUVSEMI bit in the CMRCR2 register and this bit are set to 0 at the same time, either Y or UV values are output as selected by the setting of the YCM bit in the CMRCR register.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Produces the interleave output of YUV data.<br>1: Produces the separate output of Y/UV data.<br>This bit should be set to 1 when using DYP mode. |
| 6, 5 | SUV | 00 | R | Specifies the precision of color difference processing of source data.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>00: 8-bpp precision<br>01: 10-bpp precision<br>10: 12-bpp precision<br>11: Setting prohibited<br>When the setting of the EFPE bit in this register is 0, specify the same precision as specified by the DUV field being set or a higher precision. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 0, specify the same setting in this bit and in the DUV field. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20.<br>When reading texture data from the external memory, 12-bpp precision source data should be 16-bit data in which U and V are each LSB-justified with 0 in the upper 4 bits; and 10-bpp precision source data should be 16-bit data in which U and V are each LSB-justified with 0 in the upper 6 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the source data and the color difference of the source data must have the same precision. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4, 3 | DUV | 00 | R | Specifies the precision of color difference of output data.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>00: Outputs UV data in 8-bpp precision.<br>01: Outputs UV data in 10-bpp precision.<br>10: Outputs UV data in 12-bpp precision.<br>11: Setting prohibited.<br>When the setting of the EFPE bit in this register is 0, specify the same precision as specified by the SUV field being set or a lower precision. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 0, specify the same setting in this bit and in the SUV field. When the setting of the EFPE bit in this register is 1 and that of the SHFE bit in TRIMR is 1, set this bit field according to the combinations shown in Table 56.19 and Table 56.20.<br>In 12-bpp precision output, 16-bit data is output in which U and V are each LSB-justified with 0 in the upper 4 bits; and in 10-bpp precision output, 16-bit data is output in which U and V are each LSB-justified with 0 in the upper 6 bits.<br>In YUV422 interleaved mode (YUV422E in the CMRCR2 register is set to 1), the luminance of the destination data and the color difference of the destination data must have the same precision. |
| 2 | CLCE | 0 | R | Hue Correction Enable<br>Enables hue correction using scale and offset values.<br>0: Disables hue correction.<br>1: Enables hue correction.<br>If Y data is processed (YCM bit in CMRCR = 0 and YUV422E bit in CMRCR2 = 0 or YOM bit is 1), this bit must be set to 0. |
| 1 | LUCE | 0 | R | Luminance Correction Enable<br>Enables luminance correction using scale and offset values.<br>0: Disables luminance correction.<br>1: Enables luminance correction.<br>If UV data is processed (YCM bit in CMRCR = 1 and YUV422E bit in CMRCR2 = 0), this bit must be set to 0. |
| 0 | — | 1 | R | Reserved<br>This bit is always read as 1.<br>The write value should always be 1. Otherwise, the operation cannot be guaranteed. |

### 56.2.30 Rendering Mode Set Register (CMRCSR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFPES | CP32ES | SUV1416S | | DUV1416S | | SY1416S | | DY1416S | | — | — | CLSMS | CLOMS | LUSMS | LUOMS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/W | -/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | R | R | —/W | —/W | —/W | —/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CP16ES | YCMS | UVSS | SY12S | SY10S | YOMS | Y12S | Y10S | YISMS | SUVS | | DUVS | | CLCES | LUCES | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EFPES | 0 | —/W | 0: The EFPE bit in CMRCR is not set to 1.<br>1: The EFPE bit in CMRCR is set to 1. |
| 30 | CP32ES | 0 | —/W | 0: The CP32E bit in CMRCR is not set to 1.<br>1: The CP32E bit in CMRCR is set to 1. |
| 29, 28 | SUV1416S | 0 | —/W | 00: Neither of the SUV1416 bits in CMRCR is set to 1.<br>01: Sets bit 0 of the SUV1416 bits in CMRCR to 1.<br>10: Sets bit 1 of the SUV1416 bits in CMRCR to 1.<br>11: Sets bit 0 and bit 1 of the SUV1416 bits in CMRCR to 11. |
| 27, 26 | DUV1416S | 0 | —/W | 00: Neither of the DUV1416 bits in CMRCR is set to 1.<br>01: Sets bit 0 of the DUV1416 bits in CMRCR to 1.<br>10: Sets bit 1 of the DUV1416 bits in CMRCR to 1.<br>11: Sets bit 0 and bit 1 of the DUV1416 bits in CMRCR to 11. |
| 25, 24 | SY1416S | 0 | —/W | 00: Neither of the SY1416 bits in CMRCR is set to 1.<br>01: Sets bit 0 of the SY1416 bits in CMRCR to 1.<br>10: Sets bit 1 of the SY1416 bits in CMRCR to 1.<br>11: Sets bit 0 and bit 1 of the SY1416 bits in CMRCR to 11. |
| 23, 22 | DY1416S | 0 | —/W | 00: Neither of the DY1416 bits in CMRCR is set to 1.<br>01: Sets bit 0 of the DY1416 bits in CMRCR to 1.<br>10: Sets bit 1 of the DY1416 bits in CMRCR to 1.<br>11: Sets bit 0 and bit 1 of the DY1416 bits in CMRCR to 11. |
| 21, 20 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 19 | CLSMS | 0 | —/W | 0: The CLSM bit in CMRCR is not set to 1.<br>1: The CLSM bit in CMRCR is set to 1. |
| 18 | CLOMS | 0 | —/W | 0: The CLOM bit in CMRCR is not set to 1.<br>1: The CLOM bit in CMRCR is set to 1. |
| 17 | LUSMS | 0 | —/W | 0: The LUSM bit in CMRCR is not set to 1.<br>1: The LUSM bit in CMRCR is set to 1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | LUOMS | 0 | —/W | 0: The LUOM bit in CMRCR is not set to 1.<br>1: The LUOM bit in CMRCR is set to 1 |
| 15 | CP16ES | 0 | —/W | 0: The CP16E bit in CMRCR is not set to 1.<br>1: The CP16E bit in CMRCR is set to 1. |
| 14 | YCMS | 0 | —/W | 0: The YCM bit in CMRCR is not set to 1.<br>1: The YCM to bit in CMRCR is set to 1. |
| 13 | UVSS | 0 | —/W | 0: The UVS bit in CMRCR is not set to 1.<br>1: The UVS bit in CMRCR is set to 1. |
| 12 | SY12S | 0 | —/W | 0: The SY12 bit in CMRCR is not set to 1.<br>1: The SY12 bit in CMRCR is set to 1. |
| 11 | SY10S | 0 | —/W | 0: The SY10 bit in CMRCR is not set to 1.<br>1: The SY10 bit in CMRCR is set to 1. |
| 10 | YOMS | 0 | —/W | 0: The YOM bit in CMRCR is not set to 1.<br>1: The YOM bit in CMRCR is set to 1. |
| 9 | Y12S | 0 | —/W | 0: The Y12 bit in CMRCR is not set to 1.<br>1: The Y12 bit in CMRCR is set to 1. |
| 8 | Y10S | 0 | —/W | 0: The Y10 bit in CMRCR is not set to 1.<br>1: The Y10 bit in CMRCR is set to 1. |
| 7 | YISMS | 0 | —/W | 0: The YISM bit in CMRCR is not set to 1.<br>1: The YISM bit in CMRCR is set to 1. |
| 6, 5 | SUVS | 00 | —/W | 00: Neither of the SUV bits in CMRCR is set to 1.<br>01: Sets bit 0 of the SUV bits in CMRCR to 1.<br>10: Sets bit 1 of the SUV bits in CMRCR to 1.<br>11: Setting prohibited. |
| 4, 3 | DUVS | 00 | —/W | 00: Neither of the DUV bits in CMRCR is set to 1.<br>01: Sets bit 0 of the DUV bits in CMRCR to 1.<br>10: Sets bit 1 of the DUV bits in CMRCR to 1.<br>11: Setting prohibited. |
| 2 | CLCES | 0 | —W | 0: The CLCE bit in CMRCR is not set to 1.<br>1: The CLCE bit in CMRCR is set to 1. |
| 1 | LUCES | 0 | —/W | 0: The LUCE bit in CMRCR is not set to 1.<br>1: The LUCE bit in CMRCR is set to 1. |
| 0 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

### 56.2.31 Rendering Mode Clear Register (CMRCCR)

CMRCCR is used to clear the corresponding bits of the rendering mode register (CMRCR). To disable the modes and functions selected by the bits in CMRCR, write 1 to the corresponding bits in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFPEC | CP32EC | SUV1416C | | DUV1416C | | SY1416C | | DY1416C | | — | — | CLSMC | CLOMC | LUSMC | LUOMC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/W | -/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | R | R | —/W | —/W | —/W | —/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CP16EC | YCMC | UVSC | SY12C | SY10C | YOMC | Y12C | Y10C | YISMC | SUVC | | DUVC | | CLCEC | LUCEC | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EFPEC | 0 | —/W | 0: Does not clear the EFPE bit in CMRCR to 0.<br>1: Clears the EFPE bit in CMRCR to 0. |
| 30 | CP32EC | 0 | —/W | 0: Does not clear the CP32E bit in CMRCR to 0.<br>1: Clears the CP32E bit in CMRCR to 0. |
| 29, 28 | SUV1416C | 0 | —/W | 00: Does not clear either of the SUV1416 bits in CMRCR to 0.<br>01: Clears bit 0 of the SUV1416 bits in CMRCR to 0.<br>10: Clears bit 1 of the SUV1416 bit in CMRCR to 0.<br>11: Clears bit 0 and bit 1 of the SUV1416 bits in CMRCR to 00. |
| 27, 26 | DUV1416C | 0 | —/W | 00: Does not clear either of the DUV1416 bits in CMRCR to 0.<br>01: Clears bit 0 of the DUV1416 bits in CMRCR to 0.<br>10: Clears bit 1 of the DUV1416 bit in CMRCR to 0.<br>11: Clears bit 0 and bit 1 of the DUV1416 bits in CMRCR to 00. |
| 25, 24 | SY1416C | 0 | —/W | 00: Does not clear either of the SY1416 bits in CMRCR to 0.<br>01: Clears bit 0 of the SY1416 bits in CMRCR to 0.<br>10: Clears bit 1 of the SY1416 bit in CMRCR to 0.<br>11: Clears bit 0 and bit 1 of the SY1416 bits in CMRCR to 00. |
| 23, 22 | DY1416C | 0 | —/W | 00: Does not clear either of the DY1416 bits in CMRCR to 0.<br>01: Clears bit 0 of the DY1416 bits in CMRCR to 0.<br>10: Clears bit 1 of the DY1416 bit in CMRCR to 0.<br>11: Clears bit 0 and bit 1of the DY1416 bits in CMRCR to 00. |
| 21, 20 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 19 | CLSMC | 0 | —/W | 0: Does not clear the CLSM bit in CMRCR to 0.<br>1: Clears the CLSM bit in CMRCR to 0. |
| 18 | CLOMC | 0 | —/W | 0: Does not clear the CLOM bit in CMRCR to 0.<br>1: Clears the CLOM bit in CMRCR to 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | LUSMC | 0 | —/W | 0: Does not clear the LUSM bit in CMRCR to 0.<br>1: Clears the LUSM bit in CMRCR to 0. |
| 16 | LUOMC | 0 | —/W | 0: Does not clear the LUOM bit in CMRCR to 0.<br>1: Clears the LUOM bit in CMRCR to 0. |
| 15 | CP16EC | 0 | —/W | 0: Does not clear the CP16E bit in CMRCR to 0.<br>1: Clears the CP16E bit in CMRCR to 0. |
| 14 | YCMC | 0 | —/W | 0: Does not clear the YCM bit in CMRCR to 0.<br>1: Clears the YCM bit in CMRCR to 0. |
| 13 | UVSC | 0 | —/W | 0: Does not clear the UVS bit in CMRCR to 0.<br>1: Clears the UVS bit in CMRCR to 0. |
| 12 | SY12C | 0 | —/W | 0: Does not clear the SY12 bit in CMRCR to 0.<br>1: Clears the SY12 bit in CMRCR to 0. |
| 11 | SY10C | 0 | —/W | 0: Does not clear the SY10 bit in CMRCR to 0.<br>1: Clears the SY10 bit in CMRCR to 0. |
| 10 | YOMC | 0 | —/W | 0: Does not clear the YOM bit in CMRCR to 0.<br>1: Clears the YOM bit in CMRCR to 0. |
| 9 | Y12C | 0 | —/W | 0: Does not clear the Y12 bit in CMRCR to 0.<br>1: Clears the Y12 bit in CMRCR to 0. |
| 8 | Y10C | 0 | —/W | 0: Does not clear the Y10 bit in CMRCR to 0.<br>1: Clears the Y10 bit in CMRCR to 0. |
| 7 | YISMC | 0 | —/W | 0: Does not clear the YISM bit in CMRCR to 0.<br>1: Clears the YISM bit in CMRCR to 0. |
| 6, 5 | SUVC | 00 | —/W | 00: Does not clear either of the SUV bits in CMRCR to 0.<br>01: Clears bit 0 of the SUV bits in CMRCR to 0.<br>10: Clears bit 1 of the SUV bit in CMRCR to 0.<br>11: Setting prohibited. |
| 4, 3 | DUVC | 00 | —/W | 00: Does not clear either of the DUV bit in CMRCR to 0.<br>01: Clears bit 0 of the DUV bits in CMRCR to 0.<br>10: Clears bit 1 of the DUV bits in CMRCR to 0.<br>11: Setting prohibited. |
| 2 | CLCEC | 0 | —/W | 0: Does not clear the CLCE bit in CMRCR to 0.<br>1: Clears the CLCE bit in CMRCR to 0. |
| 1 | LUCEC | 0 | —/W | 0: Does not clear the LUCE bit in CMRCR to 0.<br>1: Clears the LUCE bit in CMRCR to 0. |
| 0 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

### 56.2.32 Rendering Mode Register 2 (CMRCR2)

CMRCR2 indicates the output mode for the drawing results. To set each bit in this register to 1, write 1 to the corresponding bit in CMRCSR2. To clear each bit to 0, write 1 to the corresponding bit in CMRCCR2.

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

Note. CMRCR2 must be initialized (cleared) by using CMRCCR2 after the software reset.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCTE | — | — | TCTE | — | — | — | — | — | UVFORM | YUV422FORM | DYP | YUVSEMI | YUV422E | UVC | LUTE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 | DCTE | 0 | R | Tile Addressing Mode Enable for Destination Data<br>0: Linear addressing mode<br>1: Tile addressing mode |
| 14, 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 | TCTE | 0 | R | Tile Addressing Mode Enable for Texture Data<br>0: Linear addressing mode<br>1: Tile addressing mode |
| 11, 10 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 9 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | UVFORM | 0 | R | Swap the order of adjacent 'U' and 'V' on the YUV422 interleaved/ YUV422 semi-planar/ YUV420 semi-planar texture data.<br>0: Keeps the original order.<br>1: Swaps the order.<br>Set this bit to 0 when both YUVSEMI and TCTE bits are 1. |
| 5 | YUV422FORM | 0 | R | Swap the order of adjacent 'Y' and 'U' or 'V' on the YUV422 interleaved texture data.<br>0: Keeps the original order.<br>1: Swaps the order. |
| 4 | DYP | 0 | R | DYP: Double luminance plane mode<br>0: Not use double luminance plane function.<br>1: Use double luminance plane.<br>When this bit is 1, the data of first Y plane is read from the space set by SSAR and SSTR. The data of second Y plane is read from the space set by UVSSAR and UVSSTR.<br>When this bit is 1, setting of the YUVSEMI and YUV422E bit shall be 0.<br>Set this bit to 0 when the setting of the RSE bit in the RSCR is 1. Operation cannot be guaranteed if this bit is set to 1 while the setting of the RSE bit in the RSCR is 1.<br>Refer to the Table 56.13 for possible combination. |
| 3 | YUVSEMI | 0 | R | YUV Semi-planar mode<br>0: The input image format is not YUV Semi-planar<br>1: The input image format is YUV Semi-planar<br>When this bit is 1, the data of Y plane is read from the space set by SSAR, SSARn, SSTR, and SSTRn. The data of UV plane is read from the space set by UVSSAR, UVSSARn, UVSSTR, UVSSTRn.<br>When this bit is 1, setting of the DYP and YUV422E bit shall be 0.<br>Refer to the Table 56.13 for possible combination. |
| 2 | YUV422E | 0 | R | YUV422 Interleaved Mode Enable<br>0: Processes Y only or UV only plane.<br>1: Processes YUV422 interleaved plane.<br>Set this bit to 0 when the setting of the RSE bit in the RSCR is 1. Operation cannot be guaranteed if this bit is set to 1 while the setting of the RSE bit in the RSCR is 1.<br>When this bit is 1, setting of the DYP and YUVSEMI bit shall be 0.<br>Refer to the Table 56.13 for possible combination. |
| 1 | UVC | 0 | R | UV plane format conversion<br>When this bit is set, if the input UV plane is in YUV422 Semi-planar format, it will be converted to YUV420 Semi-planar format; if the input UV plane is in YUV420 Semi-Planar format, it will be converted to YUV422 Semi-planar format. Do not set this bit to 1 if the input UV plane is in another format.<br>0: Does not convert the format of UV plane<br>1: Converts the format of UV plane<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | LUTE | 0 | R | Lookup Table Enable<br>When using the LUT to convert the output data from 10 bpp or 12 bpp to 8 bpp, set this bit to 1.<br>In 12 to 8 bpp conversion, the upper 10 bits are converted to 8 bpp using the LUT. Only luminance can be converted. It is prohibited to convert color difference data.<br>When YUV output precision is not 8 bpp, set this bit to 0.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change.<br>0: Does not convert data using the LUT.<br>1: Converts data using the LUT. |

### 56.2.33 Rendering Mode Set Register 2 (CMRCSR2)

CMRCSR2 is used to set the corresponding bits in the rendering mode register 2 (CMRCR2). To enable the modes and functions selected by the bits in CMRCR2, write 1 to the corresponding bits in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCTES | — | — | TCTES | — | — | — | — | — | UVFORMS | YUV422FORMS | DYPS | YUVSEMIS | YUV422ES | UVCS | LUTES |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/W | R | R | —/W | R | R | R | R | R | —/W | —/W | —/W | —/W | —/W | —/W | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 | DCTES | 0 | —/W | 0: The DCTE bit in CMRCR2 is not set to 1.<br>1: The DCTE bit in CMRCR2 is set to 1. |
| 14, 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 | TCTES | 0 | —/W | 0: The TCTE bit in CMRCR2 is not set to 1.<br>1: The TCTE bit in CMRCR2 is set to 1. |
| 11, 10 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 9 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 6 | UVFORMS | 0 | —/W | 0: The UVFORM bit in CMRCR2 is not set to 1.<br>1: The UVFORM bit in CMRCR2 is set to 1. |
| 5 | YUV422FORMS | 0 | —/W | 0: The YUV422FORM bit in CMRCR2 is not set to 1.<br>1: The YUV422FORM bit in CMRCR2 is set to 1. |
| 4 | DYPS | 0 | —/W | 0: The DYP bit in CMRCR2 is not set to 1<br>1: The DYP bit in CMRCR2 is set to 1 |
| 3 | YUVSEMIS | 0 | —/W | 0: The YUVSEMI bit in CMRCR2 is not set to 1.<br>1: The YUVSEMI bit in CMRCR2 is set to 1. |
| 2 | YUV422ES | 0 | —/W | 0: The YUV422E bit in CMRCR2 is not set to 1.<br>1: The YUV422E bit in CMRCR2 is set to 1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | UVCS | 0 | —/W | 0: The UVC bit in CMRCR2 is not set to 1.<br>1: The UVC bit in CMRCR2 is set to 1. |
| 0 | LUTES | 0 | —/W | 0: The LUTE bit in CMRCR2 is not set to 1.<br>1: The LUTE bit in CMRCR2 is set to 1. |

### 56.2.34 Rendering Mode Clear Register 2 (CMRCCR2)

CMRCCR2 is used to clear the corresponding bits of the rendering mode register 2 (CMRCR2). To disable the modes and functions selected by the bits in CMRCR2, write 1 to the corresponding bits in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCTEC | — | — | TCTEC | — | — | — | — | — | UVFORMC | YUV422FORMC | DYPC | YUVSEMIC | YUV422EC | UVCC | LUTEC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/W | R | R | —/W | R | R | R | R | R | —/W | —/W | —/W | —/W | —/W | —/W | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 | DCTEC | 0 | —/W | 0: Does not clear the DCTE bit in CMRCR2 to 0.<br>1: Clears the DCTE bit in CMRCR2 to 0. |
| 14, 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 | TCTEC | 0 | —/W | 0: Does not clear the TCTE bit in CMRCR2 to 0.<br>1: Clears the TCTE bit in CMRCR2 to 0. |
| 11, 10 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 9 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 6 | UVFORMC | 0 | —/W | 0: Does not clear the UVFORM bit in CMRCR2 to 0.<br>1: Clears the UVFORM bit in CMRCR2 to 0. |
| 5 | YUV422FORMC | 0 | —/W | 0: Does not clear the YUV422FORM bit in CMRCR2 to 0.<br>1: Clears the YUV422FORM bit in CMRCR2 to 0. |
| 4 | DYPC | 0 | —/W | 0: The DYP bit in CMRCR2 is not clear to 0.<br>1: The DYP bit in CMRCR2 is clear to 0. |
| 3 | YUVSEMIC | 0 | —/W | 0: Does not clear the YUVSEMI bit in CMRCR2 to 0.<br>1: Clears the YUVSEMI bit in CMRCR2 to 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | YUV422EC | 0 | —/W | 0: Does not clear the YUV422E bit in CMRCR2 to 0.<br>1: Clears the YUV422E bit in CMRCR2 to 0. |
| 1 | UVCC | 0 | —/W | 0: Does not clear the UVC bit in CMRCR2 to 0.<br>1: Clears the UVC bit in CMRCR2 to 0. |
| 0 | LUTEC | 0 | —/W | 0: Does not clear the LUTE bit in CMRCR to 0.<br>1: Clears the LUTE bit in CMRCR to 0. |

### 56.2.35 Triangle Mode Register (TRIMR)

TRIMR is used to select the various triangle drawing modes including coordinate generation modes and filter types. To set each bit in this register to 1, write 1 to the corresponding bit in TRIMSR. To clear each bit to 0, write 1 to the corresponding bit in TRIMCR.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DIDL | — | — | — | UVSHFVAL | — | — | — | SHFVAL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSHFE | SHFE | RDE | — | TFE | — | — | — | — | TCM | — | — | AUTOSG | AUTODG | BFE | TME |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 24 | DIDL | 0 | R | Disable input information of CLCE or LUCE mode in DL.<br>0: Enable input information of CLCE or LUCE mode in DL.<br>1: Disable input information of CLCE or LUCE mode in DL.<br>In case of use CLCE to 1, need to set bit CLSM and CLOM.<br>In case of use LUCE to 1, need to set bit LUSM and LUOM. |
| 23 to 21 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 20 | UVSHFVAL | 0 | R | Specifies one bit of logical right-shifting after filtering when the extended filtering pipeline is in use. This setting is applied to the UV plane as described in Table 56.13. Otherwise, SHFE and SHFVAL settings are used. This setting is only possible when the EFPE bit in CMRCR and the UVSHFE bit in this register are both set to 1. Set this bit to 0 when the EFPE in CMRCR and the SHFE bit in this register are not both set to 1. Set this bit in accord with the combinations shown in Table 56.19 and Table 56.20.<br>Set this bit to 0 when using DYP.<br>When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this bit without executing the SYNCM instruction. |
| 19 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 | SHFVAL | 0 | R | Specifies one bit of logical right-shifting after filtering when the extended filtering pipeline is in use. Sets the UV plane as described in Table 56.13. This setting is only possible when the EFPE bit in CMRCR and the SHFE bit in this register are both set to 1. Set this bit to 0 when the EFPE in CMRCR and the SHFE bit in this register are not both set to 1. Set this bit in accord with the combinations shown in Table 56.19 and Table 56.20.<br>Set this bit to 0 when using DYP.<br>When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this bit without executing the SYNCM instruction. |
| 15 | UVSHFE | 0 | R | Specifies whether to truncate the fractional part or to retain the specified fractional digits following filtering by the extended filtering pipeline. This setting is applied to the UV plane as described in Table 56.13. Otherwise, SHFE and SHFVAL settings are used.<br>0: The fractional part is truncated after filtering.<br>1: Some of the digits of the fractional part after filtering are retained.<br>When the setting of this bit is 1 and the input pixel format is N bpp, data are output in the (N + 3)-bpp format. The number of fractional bits to be retained is 2 in the case of bilinear filtering, selected when the BFE bit in TRIMR is 1 and the TFE bit in TRIMR is 0 (the lowest-order bit is always 0).<br>When filtering using pyramidal images is enabled (the BFE bit in TRIMR is 1 and the TFE bit in TRIMR is 1), the number of fractional bits to be retained is 3.<br>This setting is only enabled when the EFPE bit in CMRCR is 1. When the setting of the EFPE bit is 0, set this bit to 0.<br>Set this bit to 0 when using DYP.<br>When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this bit without executing the SYNCM instruction. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 14 | SHFE | 0 | R | Specifies whether to truncate the fractional part or to retain the specified fractional digits following filtering by the extended filtering pipeline. Set the UV plane as described in Table 56.13.<br>0: The fractional part is truncated after filtering.<br>1: Some of the digits of the fractional part after filtering are retained.<br>When the setting of this bit is 1 and the input pixel format is N bpp, data are output in the (N + 3)-bpp format. The number of fractional bits to be retained is 2 in the case of bilinear filtering, selected when the BFE bit in TRIMR is 1 and the TFE bit in TRIMR is 0 (the lowest-order bit is always 0).<br>When filtering using pyramidal images is enabled (the BFE bit in TRIMR is 1 and the TFE bit in TRIMR is 1), the number of fractional bits to be retained is 3.<br>This setting is only enabled when the EFPE bit in CMRCR is 1. When the setting of the EFPE bit is 0, set this bit to 0.<br>Set this bit to 0 when using DYP.<br>When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this bit without executing the SYNCM instruction. |
| 13 | RDE | 0 | R | Specifies how to round the fractional part in filtering.<br>0: Truncation<br>1: Rounding up or down<br>This setting is only enabled when the EFPE bit in CMRCR is 1. When the setting of the EFPE bit is 0, set this bit to 0.<br>When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this bit without executing the SYNCM instruction. |
| 12 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 11 | TFE | 1 | R | Pyramidal Image Filtering Enable<br>0: Disables the filtering between two adjacent pyramidal images.<br>1: Enables the filtering between two adjacent pyramidal images.<br>When setting this bit to 1, set the BFE bit to 1 as well.<br>When this bit is 1, SCCR.TCMENB must be set to 0.<br>When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this bit without executing the SYNCM instruction. |
| 10 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 6 | TCM | 0 | R | Triangle Clockwise Mode<br>Specifies the order of the first three vertexes of the strip-type triangle with N vertexes that is drawn by the TRI instruction. Any TRI instruction not consistent with the order specified in this bit is not executed.<br>0: Specifies triangle vertexes counterclockwise.<br>1: Specifies triangle vertexes clockwise. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5, 4 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 3 | AUTOSG | 0 | R | Automatic Source Coordinate Generation Mode<br>In automatic source coordinate generation mode, the rectangular source coordinates are internally generated according to the mesh generation X size and Y size register settings and mesh generation X and Y start register settings.<br>0: Disables the automatic source coordinate generation function.<br>1: Enables the automatic source coordinate generation function.<br>When this bit is set to 1 to enable the automatic source coordinate generation mode, source coordinates cannot be input to the DL. |
| 2 | AUTODG | 0 | R | Automatic Destination Coordinate Generation Mode<br>In automatic destination coordinate generation mode, the rectangular destination coordinates are internally generated according to the mesh generation X size and Y size register settings and mesh generation X and Y start register settings.<br>0: Disables the automatic destination coordinate generation function.<br>1: Enables the automatic destination coordinate generation function.<br>When this bit is set to 1 to enable the automatic destination coordinate generation mode, destination coordinates cannot be input to the DL. |
| 1 | BFE | 1 | R | Bilinear Filter Enable<br>Indicates whether bilinear filtering is used for texture mapping.<br>0: Bilinear filtering is not used.<br>1: Bilinear filtering is used.<br>When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this bit without executing the SYNCM instruction. |
| 0 | TME | 1 | R | Texture Mapping Enable<br>Indicates whether texture mapping is used or the single color specified by TRICR, TRICR2, and TRICR3 is used for triangle drawing.<br>Set the same format on the input image data format and the output image format if this bit is clear to 0.<br>0: The single color specified by TRICR, TRICR2, and TRICR3 is used.<br>1: Texture mapping is used.<br>When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this bit without executing the SYNCM instruction. |

### 56.2.36 Triangle Mode Set Register (TRIMSR)

TRIMSR is used to set the corresponding bits of the triangle mode register (TRIMR). To enable the modes and functions selected by the bits in TRIMR, write 1 to the corresponding bits in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DIDLS | — | — | — | UVSHFVALS | — | — | — | SHFVALS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | —/W | R | R | R | —/W | R | R | R | —/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSHFES | SHFES | RDES | — | TFES | — | — | — | — | TCMS | — | — | AUTOSGS | AUTODGS | BFES | TMES |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/W | —/W | —/W | R | —/W | R | R | R | R | —/W | R | R | —/W | —/W | —/W | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 24 | DIDLS | 0 | —/W | 0: The DIDL bit is not set to 1.<br>1: The DIDL bit is not set to 1. |
| 23 to 21 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 20 | UVSHFVALS | 0 | —/W | 0: The UVSHFVAL bit is not set to 1.<br>1: The UVSHFVAL bit is set to 1. |
| 19 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 | SHFVALS | 0 | —/W | 0: The SHFVAL bit is not set to 1.<br>1: The SHFVAL bit is set to 1. |
| 15 | UVSHFVALS | 0 | —/W | 0: The UVSHFVAL bit is not set to 1.<br>1: The UVSHFVAL bit is set to 1. |
| 14 | SHFES | 0 | —/W | 0: The SHFE bit is not set to 1.<br>1: The SHFE bit is set to 1. |
| 13 | RDES | 0 | —/W | 0: The RDE bit is not set to 1.<br>1: The RDE bit is set to 1. |
| 12 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 | TFES | 0 | —/W | 0: The TFE bit is not set to 1.<br>1: The TFE bit is set to 1. |
| 10 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 6 | TCMS | 0 | —/W | 0: The TCM bit is not set to 1.<br>1: The TCM bit is set to 1. |
| 5, 4 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 3 | AUTOSGS | 0 | —/W | 0: The AUTOSG bit is not set to 1.<br>1: The AUTOSG bit is set to 1. |
| 2 | AUTODGS | 0 | —/W | 0: The AUTODG bit is not set to 1.<br>1: The AUTODG bit is set to 1. |
| 1 | BFES | 0 | —/W | 0: The BFE bit is not set to 1.<br>1: The BFE bit is set to 1. |
| 0 | TMES | 0 | —/W | 0: The TME bit is not set to 1.<br>1: The TME bit is set to 1. |

### 56.2.37 Triangle Mode Clear Register (TRIMCR)

TRIMCR is used to clear the corresponding bits of the triangle mode register (TRIMR). To disable the modes and functions selected by the bits in TRIMR, write 1 to the corresponding bits in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DIDLC | — | — | — | UVSHFVALC | — | — | — | SHFVALC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | —/W | R | R | R | —/W | R | R | R | —/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVSHFEC | SHFEC | RDEC | — | TFEC | — | — | — | — | TCMC | — | — | AUTOSGC | AUTODGC | BFEC | TMEC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/W | —/W | —/W | R | —/W | R | R | R | R | —/W | R | R | —/W | —/W | —/W | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 24 | DIDLC | 0 | —/W | Set bit DIDL in TRIMR<br>0: Bit DIDL in TRIMR is not clear.<br>1: Bit DIDL in TRIMR is clear to 0. |
| 23 to 21 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 20 | UVSHFVALC | 0 | —/W | 0: Does not clear the UVSHFVAL bit to 0.<br>1: Clears the UVSHFVAL bit to 0. |
| 19 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 | SHFVALC | 0 | —/W | 0: Does not clear the SHFVAL bit to 0.<br>1: Clears the SHFVAL bit to 0. |
| 15 | UVSHFVALC | 0 | —/W | 0: Does not clear the UVSHFVAL bit to 0.<br>1: Clears the UVSHFVAL bit to 0. |
| 14 | SHFEC | 0 | —/W | 0: Does not clear the SHFE bit to 0.<br>1: Clears the SHFE bit to 0. |
| 13 | RDEC | 0 | —/W | 0: Does not clear the RDE bit to 0.<br>1: Clears the RDE bit to 0. |
| 12 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 | TFEC | 0 | —/W | 0: Does not clear the TFE bit to 0.<br>1: Clears the TFE bit to 0. |
| 10 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 6 | TCMC | 0 | —/W | 0: Does not clear the TCM bit to 0.<br>1: Clears the TCM bit to 0. |
| 5, 4 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 3 | AUTOSGC | 0 | —/W | 0: Does not clear the AUTOSG bit to 0.<br>1: Clears the AUTOSG bit to 0. |
| 2 | AUTODGC | 0 | —/W | 0: Does not clear the AUTODG bit to 0.<br>1: Clears the AUTODG bit to 0. |
| 1 | BFEC | 0 | —/W | 0: Does not clear the BFE bit to 0.<br>1: Clears the BFE bit to 0. |
| 0 | TMEC | 0 | —/W | 0: Does not clear the TME bit to 0.<br>1: Clears the TME bit to 0. |

### 56.2.38 Triangle Color Register (TRICR)

TRICR is used to specify the color for single-color drawing with a TRI instruction when single-color drawing is selected with the TME bit in TRIMR.

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | YCFORM | — | — | — | TCY3 | | TCY2 | | TCV | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCU | | | | | | | | TCY | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

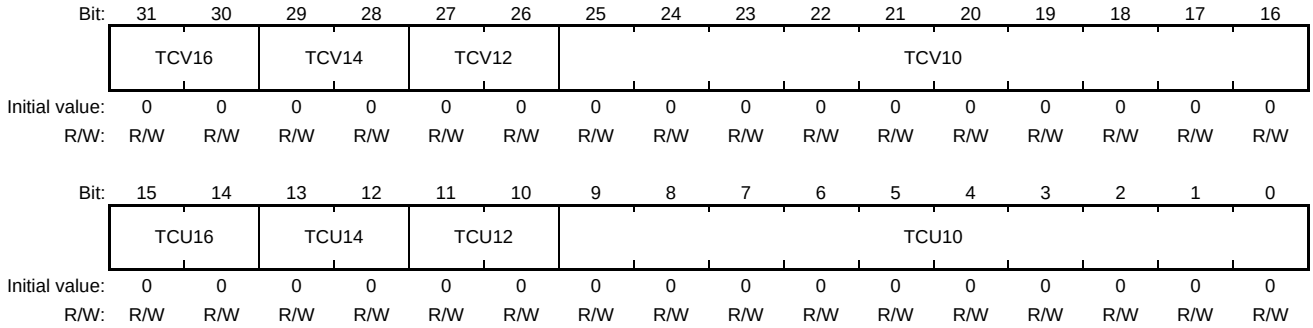
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | YCFORM | 0 | R/W | For interleave output (for which the YISM bit in CMRCR is 0 and the YUV422E bit in CMRCR2 is 1), changes the order of Y and U/V. Normally, Y and U/V are in a {U/V, Y} order; this bit, however, can reverse the order to {Y, U/V}.<br>Setting this bit to 1 is prohibited for operations other than interleave output.<br>0: For interleave output, does not change the order of Y and U/V.<br>1: For interleave output, changes the order of Y and U/V.<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change. |
| 30 to 28 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 27, 26 | TCY3 | 00 | R/W | When Y is to be output in 12 bpp, these bits specify bits 11 and 10 of color Y for single-color drawing with the TRI instruction. |
| 25, 24 | TCY2 | 00 | R/W | When Y is to be output in 10/12 bpp, these bits specify bits 9 and 8 of color Y for single-color drawing with the TRI instruction. |
| 23 to 16 | TCV | H'00 | R/W | When V is to be output in 8 bits, these bits specify color V for single-color drawing with the TRI instruction. When outputting V in 10 bits/12 bits, the V value in the TRICR2 register is used. |
| 15 to 8 | TCU | H'00 | R/W | When U is to be output in 8 bits, these bits specify color U for single-color drawing with the TRI instruction. When outputting U in 10 bits/12 bits, the U value in the TRICR2 register is used. |
| 7 to 0 | TCY | H'00 | R/W | Specifies color Y for single-color drawing with the TRI instruction. |

### 56.2.39 Triangle Color Register 2 (TRICR2)

TRICR2 is used to specify the color for single-color drawing with a TRI instruction when single-color drawing is selected with the TME bit in TRIMR. This register is used when UV is output in 10 bpp/12 bpp.

When 32-bits single color mode is selected by the setting of CP32E in CMRCR and TME bit in TRIMR, this register is used as single-Color data.

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | TCV16 | 00 | R/W | When V is to be output in 16 bpp, these bits specify bits 15 and 14 of color V for single-color drawing with the TRI instruction. |
| 29, 28 | TCV14 | 00 | R/W | When V is to be output in 14/16 bpp, these bits specify bits 13 and 12 of color V for single-color drawing with the TRI instruction. |
| 27, 26 | TCV12 | 00 | R/W | When V is to be output in 12/14/16 bpp, these bits specify bits 11 and 10 of color V for single-color drawing with the TRI instruction. |
| 25 to 16 | TCV10 | H'000 | R/W | When V is to be output in 10/12/14/16 bpp, these bits specify bits 9 to 0 of color V for single-color drawing with the TRI instruction. |
| 15, 14 | TCU16 | 00 | R/W | When U is to be output in 16 bpp, these bits specify bits 15 and 14 of color U for single-color drawing with the TRI instruction. |
| 13, 12 | TCU14 | 00 | R/W | When U is to be output in 14/16 bpp, these bits specify bits 13 and 12 of color U for single-color drawing with the TRI instruction. |
| 11, 10 | TCU12 | 00 | R/W | When U is to be output in 12/14/16 bpp, these bits specify bits 11 and 10 of color U for single-color drawing with the TRI instruction. |
| 9 to 0 | TCU10 | H'000 | R/W | When U is to be output in 10/12/14/16 bpp, these bits specify bits 9 to 0 of color U for single-color drawing with the TRI instruction. |

### 56.2.40 Triangle Mode Register 2 (TRIMR2)

TRIMR2 is used to select the maximum level of detail.

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MAXLODU | | | — | — | — | MAXLODV | | | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 11 to 9 | MAXLODU | 000 | R | The maximum level of detail for pyramidal images to be used in the u direction. The setting should be from 0 to 4. Other settings are prohibited. To modify the value during execution, execute the SYNCM instruction before modification. When this bit is not 0, SCCR.TCMENB must be set to 0. |
| 8 to 6 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 5 to 3 | MAXLODV | 000 | R | The maximum level of detail for pyramidal images to be used in the v direction. The setting should be from 0 to 4. Other settings are prohibited. To modify the value during execution, execute the SYNCM instruction before modification. When this bit is not 0, SCCR.TCMENB must be set to 0. |
| 2 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

### 56.2.41 Triangle Mode Set Register 2 (TRIMSR2)

TRIMSR2 is used to set the corresponding bits of the triangle mode register (TRIMR2). To enable the modes and functions selected by the bits in TRIMR2, write 1 to the corresponding bits in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MAXLODUS | | | — | — | — | MAXLODVS | | | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | —/W | —/W | —/W | R | R | R | —/W | —/W | —/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 11 to 9 | MAXLODUS | 000 | —/W | Sets the corresponding MAXLODU bits in the TRIMR2 register. |
| 8 to 6 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 5 to 3 | MAXLODVS | 000 | —/W | Sets the corresponding MAXLODV bits in the TRIMR2 register. |
| 2 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

### 56.2.42 Triangle Mode Clear Register 2 (TRIMCR2)

TRIMCR2 is used to clear the corresponding bits of the triangle mode register (TRIMR2). To disable the modes and functions selected by the bits in TRIMR2, write 1 to the corresponding bits in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MAXLODUC | | | — | — | — | MAXLODVC | | | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | —/W | —/W | —/W | R | R | R | —/W | —/W | —/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 11 to 9 | MAXLODUC | 000 | —/W | Clears the corresponding MAXLODU bits in the TRIMR2 register. |
| 8 to 6 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 5 to 3 | MAXLODVC | 000 | —/W | Clears the corresponding MAXLODV bits in the TRIMR2 register. |
| 2 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

### 56.2.43 Source and Destination Coordinate Decimal Point Register (UVDPOR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DDP | — | — | — | — | — | UVDPO | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 8 | DDP | 0 | R/W | Specifies the destination coordinates described in the DL and the registers related to the setting of destination coordinates.<br>0: Specifies destination coordinates in integer.<br>1: Specifies destination coordinates in fixed-point number (the lower 2 bits are below the decimal point). (Destination sub-pixel mode)<br>When changing the setting of this bit by the DL, be sure to execute the SYNCM instruction before making the change. |
| 7 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 2 to 0 | UVDPO | 000 | R/W | Source Coordinate Decimal Point<br>Specifies the number of decimal places of the source coordinates described with the DL within a range from 0 to 5. When 0 is specified, source coordinates are represented in integers. When 5 is specified, the coordinates specified with the DL are considered to have an integer part and a 5-bit decimal part. |

### 56.2.44 Source Width Register (SUSR)

SUSR is used to specify the width (horizontal size) of the source. The width value depends on the source data format.

- When using UV data as source, specify the “UV source width − 2” in bits [26:16] and the “UV source width − 1” in bits [10:0]. The UV source width is a sum of the U and V pixels. Specify an even value as the UV source width; therefore, specify an even value in bits [26:16] and an odd value in bits [10:0].
- When using Y data as source, specify the “luminance Y source width − 1” in bits [10:0].
- When using YUV422 interleaved data as source, specify the “source width - 2” in bits [26:16] and the “source width - 1” in bits [10:0]. The source width is a sum of YU and YV pixels. Specify an even value as the source width; therefore, specify an even value in bits [26:16] and an odd value in bits [10:0].
- When using (YUV422 or YUV420) semi-planar data as source, specify the “source width - 2” in bits [26:16] and the “source width - 1” in bits [10:0]. The source width is width of Y pixels.
- When setting 1 to the CP16E bit in the CMRCR register, specify the “source width − 1” in bits [10:0].
- When setting 1 to the CP32E bit in the CMRCR register, specify the “source width - 2” in bits [26:16] and “source width − 1” in bits [10:0].

For example, for the VGA system (width = 640), specify 638 in bits [26:16] and 639 in bits [10:0]. When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SUW | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SVW | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 26 to 16 | SUW | H'4FE | R/W | Specify “source width − 2”. The maximum value is 2,046. |
| 15 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 10 to 0 | SVW | H'4FF | R/W | Specify “source width − 1”. The maximum value is 2,047. |

### 56.2.45 Source Height Register (SVSR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this field without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SVS | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 10 to 0 | SVS | H'3FF | R/W | Specifies the height (vertical size) of the source.<br>Specify (height – 1) in [10:0].<br>When processing the UV plane in YUV420 format, perform the setting based on the number of lines of UV plane data rather than the height of the corresponding Y plane. Refer to the Table 56.13 input format UV plane only. |

### 56.2.46 MIN Clipping X Register (XMINR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | XMIN | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 to 0 | XMIN | H'0000 | R/W | X Clip MIN<br>Specify the minimum value of the X coordinate of the clipping area in an integer less than or equal to 2,047. Nothing is drawn at an X coordinate less than this X coordinate. If the destination is in sub-pixel mode, the lower 2 bits will be below the fixed decimal point, in which case 0 must always be assigned to the lower 2 bits. If the destination is not in sub-pixel mode, always assign 0 to the upper 2 bits. |

### 56.2.47 MIN Clipping Y Register (YMINR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | YMIN | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 to 0 | YMIN | H'0000 | R/W | Y Clip MIN<br>Specify the minimum value of the Y coordinate of the clipping area in an integer less than or equal to 2,047. Nothing is drawn at a Y coordinate less than this Y coordinate. If the destination is in sub-pixel mode, the lower 2 bits will be below the fixed decimal point, in which case 0 must always be assigned to the lower 2 bits. If the destination is not in sub-pixel mode, always assign 0 to the upper 2 bits. |

### 56.2.48 MAX Clipping X Register (XMAXR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | XMAX | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 to 0 | XMAX | H'0000 | R/W | X Clip MAX<br>Specify the maximum value of the X coordinate of the clipping area in an integer less than or equal to 2,047. Nothing is drawn at an X coordinate greater than this X coordinate. If the destination is in sub-pixel mode, the lower 2 bits will be below the fixed decimal point, in which case 0 must always be assigned to the lower 2 bits. If the destination is not in sub-pixel mode, always assign 0 to the upper 2 bits. |

### 56.2.49 MAX Clipping Y Register (YMAXR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | YMAX | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 to 0 | YMAX | H'0000 | R/W | Y Clip MAX<br>Specify the maximum value of the Y coordinate of the clipping area in an integer less than or equal to 2,047. Nothing is drawn at a Y coordinate greater than this Y coordinate. If the destination is in sub-pixel mode, the lower 2 bits will be below the fixed decimal point, in which case 0 must always be assigned to the lower 2 bits. If the destination is not in sub-pixel mode, always assign 0 to the upper 2 bits. |

### 56.2.50 Mesh Generation X Size Register (AMXSR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | AMXS | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 to 0 | AMXS | H'0001 | R/W | Automatic Mesh X Size<br>Specify the X-direction width of the mesh when destination or source coordinates are to be automatically generated. (Setting 0 is prohibited.) If destination coordinates are generated automatically and the destination is in sub-pixel mode, the lower 2 bits will be below the decimal point.<br>If source coordinates are generated automatically and Source Coordinate Decimal Point (UVDPO) in UVDPOR is set to n (range of n is from 0 to 5), the lower n bits will be below the decimal point.<br>When AUTODG in TRIMR is 1, this register must be less than 128 pixels. |

### 56.2.51 Mesh Generation Y Size Register (AMYSR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | AMYS | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 to 0 | AMYS | H'0001 | R/W | Automatic Mesh Y Size<br>Specify the Y-direction width of the mesh when destination or source coordinates are to be automatically generated. (Setting 0 is prohibited.) If destination coordinates are generated automatically and the destination is in sub-pixel mode, the lower 2 bits will be below the decimal point.<br>If source coordinates are generated automatically and Source Coordinate Decimal Point (UVDPO) in UVDPOR is set to n (range of n is from 0 to 5), the lower n bits will be below the decimal point.<br>When AUTODG in TRIMR is 1, this register must be less than 128 pixels. |

### 56.2.52 Mesh Generation X Start Register (AMXOR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | AMXO | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 to 0 | AMXO | H'0000 | R/W | Automatic Mesh X Origin<br>Specify the X coordinate of the origin when destination or source coordinates are to be automatically generated. (In case of generating source coordinates, specify the u coordinate.) If destination coordinates are generated automatically and the destination is in sub-pixel mode, the lower 2 bits will be below the decimal point.<br>If source coordinates are generated automatically and Source Coordinate Decimal Point (UVDPO) in UVDPOR is set to n (range of n is from 0 to 5), the lower n bits will be below the decimal point. |

### 56.2.53 Mesh Generation Y Start Register (AMYOR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | AMYO | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 12 to 0 | AMYO | H'0000 | R/W | Automatic Mesh Y Origin<br>Specify the Y coordinate of the origin when destination or source coordinates are to be automatically generated. (In case of generating source coordinates, specify the v coordinate.) If destination coordinates are generated automatically and the destination is in sub-pixel mode, the lower 2 bits will be below the decimal point.<br>If source coordinates are generated automatically and Source Coordinate Decimal Point (UVDPO) in UVDPOR is set to n (range of n is from 0 to 5), the lower n bits will be below the decimal point. |

### 56.2.54 Minimum Luminance Correction Y Register (YLMINR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | YLMIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 0 | YLMIN | H'0000 | R/W | Specifies the minimum luminance value when luminance correction is applied. If after correction the luminance value is lower than the value specified in this register, the luminance value is clamped at the value specified by these bits.<br>When the EFPE bit in the CMRCR register is 1 and the DY1416 is B'11, bits 15 to 0 are valid.<br>When the EFPE bit in the CMRCR register is 1 and the DY1416 is B'10, bits 13 to 0 are valid. In this case, bit [15:14] must be set to 0.<br>When the destination data precision of Y is not 16 bit or 14 bit and the Y12 bit in the CMRCR register is 1, bits 11 to 0 are valid. In this case, bit [15:12] must be set to 0.<br>When the destination data precision of Y is not 16bit or 14 bit and the Y10 bit in the CMRCR register is 1, bits 9 to 0 are valid. In this case, bit [15:10] must be set to 0.<br>In the other cases, only bits 7 to 0 are valid. In this case, bit [15:8] must be set to 0.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.55 Minimum Hue Correction U Register (UBMINR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UBMIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 0 | UBMIN | H'0000 | R/W | Specifies the minimum U value when hue correction is applied. If after correction the U value is lower than the value specified in this register, the U value is clamped at the value specified by these bits.<br>When the EFPE bit in the CMRCR register is 1 and the DUV1416 is B’11, bits 15 to 0 are valid.<br>When the EFPE bit in the CMRCR register is 1 and the DUV1416 is B’10, bits 13 to 0 are valid. In this case, bit [15:14] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'10, bits 11 to 0 are valid. In this case, bit [15:12] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'01, bits 9 to 0 are valid. In this case, bit [15:10] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'00, only bits 7 to 0 are valid. In this case, bit [15:8] must be set to 0.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.56 Minimum Hue Correction V Register (VRMINR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VRMIN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 0 | VRMIN | H'0000 | R/W | Specifies the minimum V value when hue correction is applied. If after correction the V value is lower than the value specified in this register, the V value is clamped at the value specified by these bits.<br>When the EFPE bit in the CMRCR register is 1 and the DUV1416 is B’11, bits 15 to 0 are valid.<br>When the EFPE bit in the CMRCR register is 1 and the DUV1416 is B’10, bits 13 to 0 are valid. In this case, bit [15:14] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'10, bits 11 to 0 are valid. In this case, bit [15:12] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'01, bits 9 to 0 are valid. In this case, bit [15:10] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'00, only bits 7 to 0 are valid. In this case, bit [15:8] must be set to 0.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.57 Maximum Luminance Correction Y Register (YLMAXR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | YLMAX | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 0 | YLMAX | H'00FF | R/W | Specifies the maximum luminance value when luminance correction is applied. If after correction the luminance value is higher than the value specified in this register, the luminance value is clamped at the value specified by these bits.<br>When the EFPE bit in the CMRCR register is 1 and the DY1416 is B'11, bits 15 to 0 are valid.<br>When the EFPE bit in the CMRCR register is 1 and the DY1416 is B'10, bits 13 to 0 are valid. In this case, bit [15:14] must be set to 0.<br>When the destination data precision of Y is not 16 bit or 14 bit and the Y12 bit in the CMRCR register is 1, bits 11 to 0 are valid. In this case, bit [15:12] must be set to 0.<br>When the destination data precision of Y is not 16bit or 14 bit and the Y10 bit in the CMRCR register is 1, bits 9 to 0 are valid. In this case, bit [15:10] must be set to 0.<br>In the other cases, only bits 7 to 0 are valid. In this case, bit [15:8] must be set to 0.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.58 Maximum Hue Correction U Register (UBMAXR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UBMAX | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 0 | UBMAX | H'00FF | R/W | Specifies the maximum U value when hue correction is applied. If after correction the U value is higher than the value specified in this register, the U value is clamped at the value specified by these bits.<br>When the EFPE bit in the CMRCR register is 1 and the DUV1416 is B’11, bits 15 to 0 are valid.<br>When the EFPE bit in the CMRCR register is 1 and the DUV1416 is B’10, bits 13 to 0 are valid. In this case, bit [15:14] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'10, bits 11 to 0 are valid. In this case, bit [15:12] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'01, bits 9 to 0 are valid. In this case, bit [15:10] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'00, only bits 7 to 0 are valid. In this case, bit [15:8] must be set to 0.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.59 Maximum Hue Correction V Register (VRMAXR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VRMAX | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 0 | VRMAX | H'00FF | R/W | Specifies the maximum V value when hue correction is applied. If after correction the V value is higher than the value specified in this register, the V value is clamped at the value specified by these bits.<br>When the EFPE bit in the CMRCR register is 1 and the DUV1416 is B’11, bits 15 to 0 are valid.<br>When the EFPE bit in the CMRCR register is 1 and the DUV1416 is B’10, bits 13 to 0 are valid. In this case, bit [15:14] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'10, bits 11 to 0 are valid. In this case, bit [15:12] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'01, bits 9 to 0 are valid. In this case, bit [15:10] must be set to 0.<br>When the destination data precision of UV is not 16 bit or 14 bit and the DUV bits in the CMRCR register are B'00, only bits 7 to 0 are valid. In this case, bit [15:8] must be set to 0.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.60 Correction Decimal Point Register (CPDPOR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

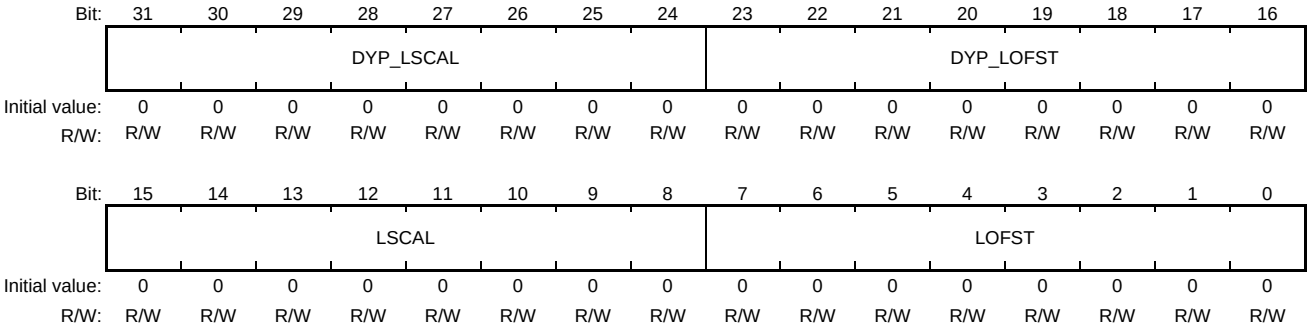
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | YLDPO | | | — | UBDPO | | | — | VRDPO | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 10 to 8 | YLDPO | 000 | R/W | Specifies the number of bits after the decimal point for the value specified as the luminance correction scale value.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |
| 7 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 6 to 4 | UBDPO | 000 | R/W | Specifies the number of bits after the decimal point for the value specified as the hue correction U value scale value.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |
| 3 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 2 to 0 | VRDPO | 000 | R/W | Specifies the number of bits after the decimal point for the value specified as the hue correction V value scale value.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.61 Luminance Correction Parameter Y Register (YLCPR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | DYP_LSCAL | H'00 | R/W | Specifies the scale parameter of second Y plane in DYP mode if the luminance correction scale parameter is specified by a register setting (LUSM = 1 in CMRCR).<br>The setting is passed as unsigned fixed-point data, and the number of decimal places is specified by YLDPO in CPDPOR. Specify the value of these bits as unsigned 8-bit data.<br>Always execute the SYNCM instruction immediately before updating this register by DL |
| 23 to 16 | DYP_LOFST | H'00 | R/W | Specifies the offset parameter of second Y plane in DYP mode<br>if the luminance correction offset parameter is specified by a register setting (LUOM = 1 in CMRCR). Specify the value of these bits as signed 8-bit data.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |
| 15 to 8 | LSCAL | H'00 | R/W | Specifies the scale parameter when the luminance correction scale parameter is specified by a register setting (LUSM = 1 in CMRCR). The setting is passed as unsigned fixed-point data, and the number of decimal places is specified by YLDPO in CPDPOR. Specify the value of these bits as unsigned 8-bit data.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |
| 7 to 0 | LOFST | H'00 | R/W | Specifies the offset parameter when the luminance correction offset parameter is specified by a register setting (LUOM = 1 in CMRCR). Specify the value of these bits as signed 8-bit data.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.62 Hue Correction Parameter U Register (UBCPR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UBSCL | | | | | | | | UBOFS | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 8 | UBSCL | H'00 | R/W | Specifies the U value scale parameter when the hue correction scale parameters are specified by register settings (CLSM = 1 in CMRCR). The setting is passed as fixed-point data, and the number of decimal places is specified by UBDPO in CPDPOR. Specify the value of these bits as unsigned 8-bit data.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |
| 7 to 0 | UBOFS | H'00 | R/W | Specifies the U value offset parameter when the hue correction offset parameters are specified by register settings (CLOM = 1 in CMRCR). Specify the value of these bits as signed 8-bit data.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.63 Hue Correction Parameter V Register (VRCPR)

When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VRSCL | | | | | | | | VROFS | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 to 8 | VRSCL | H'00 | R/W | Specifies the V value scale parameter when the hue correction scale parameters are specified by register settings (CLSM = 1 in CMRCR). The setting is passed as fixed-point data, and the number of decimal places is specified by VRDPO in CPDPOR. Specify the value of these bits as unsigned 8-bit data.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |
| 7 to 0 | VROFS | H'00 | R/W | Specifies the V value offset parameter when the hue correction offset parameters are specified by register settings (CLOM = 1 in CMRCR). Specify the value of these bits as signed 8-bit data.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

### 56.2.64 Triangle Color Register 3 (TRICR3)

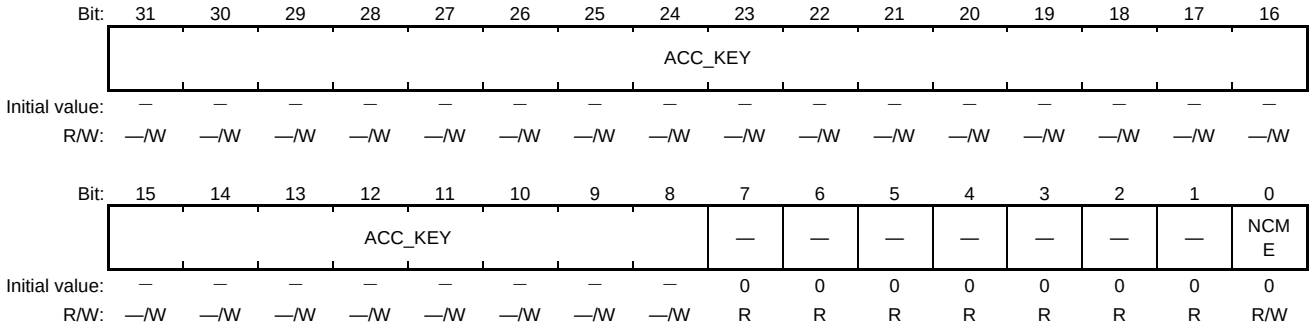
When the NCME bit in NCMR is 1 for non-blocking cache mode, do not change this register without executing the SYNCM instruction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DYP_TCY16 | | DYP_TCY14 | | DYP_TCY12 | | DYP_TCY10 | | DYP_TCY8 | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCY16 | | TCY14 | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | DYP_TCY16 | All 0 | R/W | When 2nd Y in DYP mode is to be output in 16 bpp, these bits specify bits 15 and 14 of color 2nd Y for single-color drawing with the TRI instruction. |
| 29 to 28 | DYP_TCY14 | All 0 | R/W | When 2nd Y in DYP mode is to be output in 14 bpp or 16 bpp, these bits specify bits 13 and 12 of color 2nd Y for single-color drawing with the TRI instruction. |
| 27 to 26 | DYP_TCY12 | All 0 | R/W | When 2nd Y in DYP mode is to be output in 12bpp or 14 bpp or 16 bpp, these bits specify bits 11 and 10 of color 2nd Y for single-color drawing with the TRI instruction. |
| 25 to 24 | DYP_TCY10 | All 0 | R/W | When 2nd Y in DYP mode is to be output in 10bpp or 12bpp or 14 bpp or 16 bpp, these bits specify bits 9 and 8 of color 2nd Y for single-color drawing with the TRI instruction. |
| 23 to 16 | DYP_TCY8 | All 0 | R/W | In DYP mode, these bits specify bits 7 to 0 of color 2nd Y for single-color drawing with the TRI instruction. |
| 15 to 14 | TCY16 | H'0 | R/W | When Y is to be output in 16 bpp, these bits specify bits 15 and 14 of color Y for single-color drawing with the TRI instruction. |
| 13 to 0 | TCY14 | H'0000 | R/W | When Y is to be output in 14 bpp or 16 bpp, these bits specify bits 13 to 0 of color Y for single-color drawing with the TRI instruction. |

### 56.2.65 Non-blocking Cache Mode Register (NCMR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | ACC_KEY | — | —/W | Access Key to change NCME bit of this register.<br>It is required write “0xACCE55” to this field to change the value of the bit. |
| 7 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 0 | NCME | 0 | R/W | Non-blocking cache mode enable bit for source data.<br>Allow to issue multiple outstanding requests for fetching the source data if this bit is set to 1.<br>If this bit is clear to 0, no request for fetching the source data is issued if there is a leading on-the-fly request.<br>Please refer to ACC_KEY field of this register to update this bit. |

### 56.2.66 Lookup Table Data Register n (LUTDRn)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

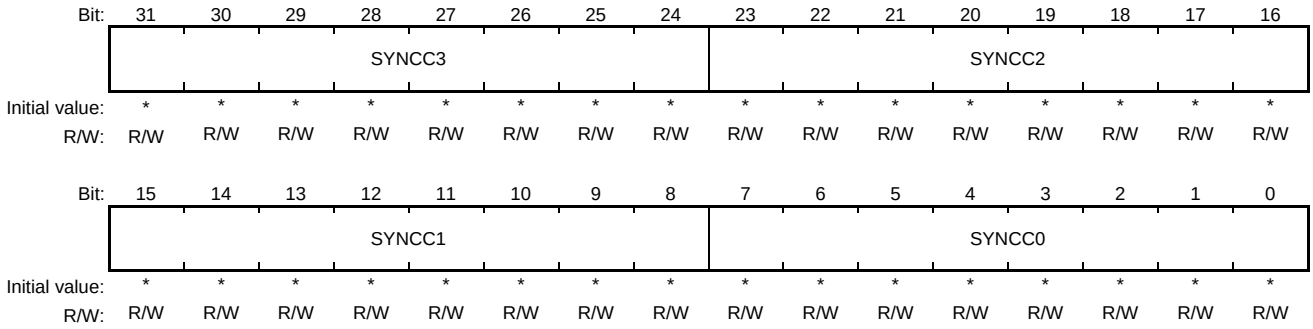
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LUTDRn | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | —/W | —/W | —/W | —/W | —/W | —/W | —/W | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 7 to 0 | LUTDRn | Undefined | —/W | These bits are used when the LUTE bit in the CMRCR2 register is 1.<br>Set 1,024 entries for the LUT to convert 10- or 12-bit output data to 8-bit data.<br>These bits can be set from the CPU and DL; when DL is used for setting, use the WTL2 instruction.<br>Always execute the SYNCM instruction immediately before updating this register by DL. |

Note: $0 \leq n \leq 1023$

### 56.2.67 Synchronization Control Register 0 (SYNCCR0)

This register specifies the module corresponding to bits 0 to 3 in the SYNCC enable field of the WUP and SLP instructions. Set the number for specifying the module in this register. See **Table 56.22** for the numbers for specifying the module. Do not set a number that is not listed in **Table 56.22**. Do not modify the setting of this register until the value of the TRA bit in SR become 1 and after 1 was written to the RS bit in CR.

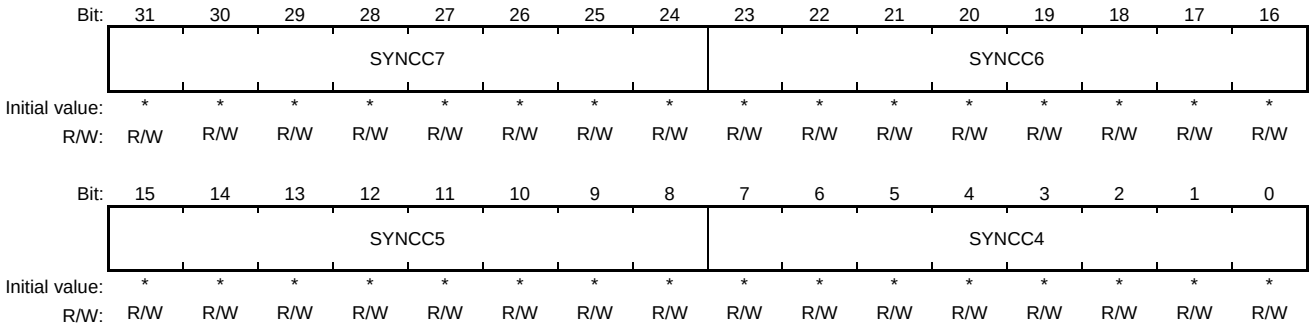



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC3 | * | R/W | Specifies the module corresponding to bit 3 of the SYNCC enable field of the WUP and SLP instructions. |
| 23 to 16 | SYNCC2 | * | R/W | Specifies the module corresponding to bit 2 of the SYNCC enable field of the WUP and SLP instructions. |
| 15 to 8 | SYNCC1 | * | R/W | Specifies the module corresponding to bit 1 of the SYNCC enable field of the WUP and SLP instructions. |
| 7 to 0 | SYNCC0 | * | R/W | Specifies the module corresponding to bit 0 of the SYNCC enable field of the WUP and SLP instructions. |

Initial value is 0xFFFFFFFF.

### 56.2.68 Synchronization Control Register 1 (SYNCCR1)

This register specifies the module corresponding to bits 4 to 7 in the SYNCC enable field of the WUP and SLP instructions. Set the number for specifying the module in this register. See **Table 56.22** for the numbers for specifying the module. Do not set a number that is not listed in **Table 56.22**. Do not modify the setting of this register until the value of the TRA bit in SR become 1 and after 1 was written to the RS bit in CR.

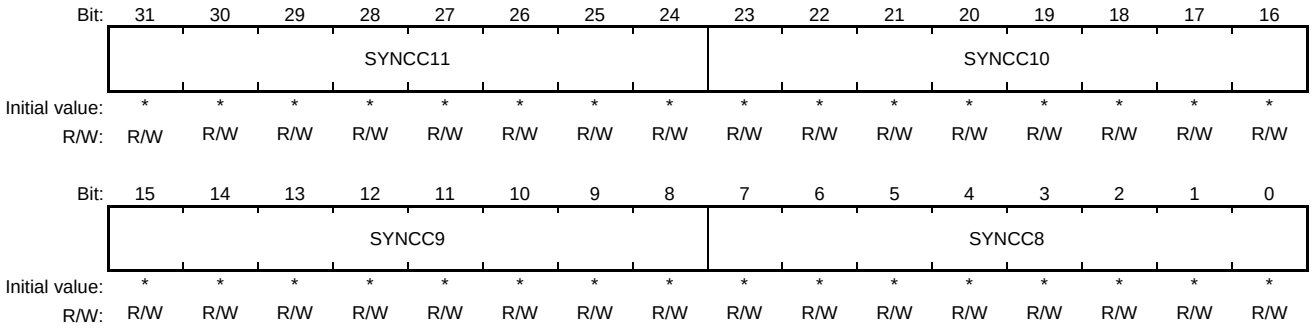



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC7 | * | R/W | Specifies the module corresponding to bit 7 of the SYNCC enable field of the WUP and SLP instructions. |
| 23 to 16 | SYNCC6 | * | R/W | Specifies the module corresponding to bit 6 of the SYNCC enable field of the WUP and SLP instructions. |
| 15 to 8 | SYNCC5 | * | R/W | Specifies the module corresponding to bit 5 of the SYNCC enable field of the WUP and SLP instructions. |
| 7 to 0 | SYNCC4 | * | R/W | Specifies the module corresponding to bit 4 of the SYNCC enable field of the WUP and SLP instructions. |

Initial value is 0xFFFFFFFF.

### 56.2.69 Synchronization Control Register 2 (SYNCCR2)

This register specifies the module corresponding to bits 8 to 11 in the SYNCC enable field of the WUP and SLP instructions. Set the number for specifying the module in this register. See **Table 56.22** for the numbers for specifying the module. Do not set a number that is not listed in **Table 56.22**. Do not modify the setting of this register until the value of the TRA bit in SR become 1 and after 1 was written to the RS bit in CR.

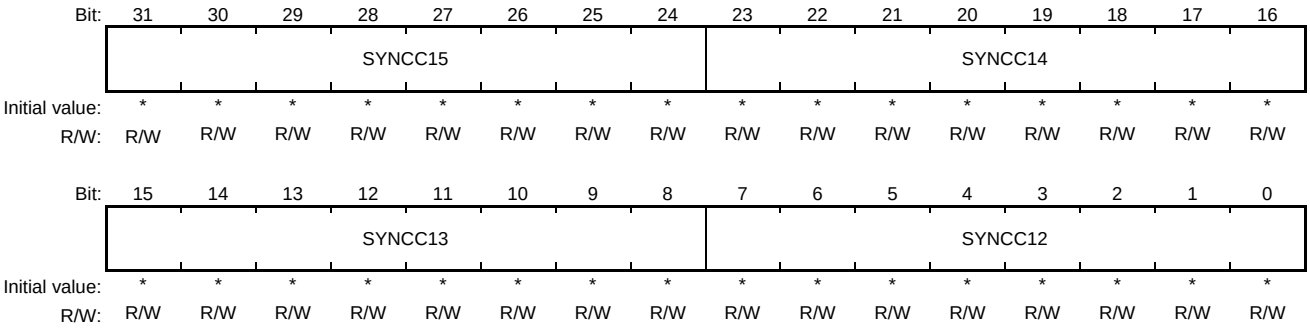



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC11 | * | R/W | Specifies the module corresponding to bit 11 of the SYNCC enable field of the WUP and SLP instructions. |
| 23 to 16 | SYNCC10 | * | R/W | Specifies the module corresponding to bit 10 of the SYNCC enable field of the WUP and SLP instructions. |
| 15 to 8 | SYNCC9 | * | R/W | Specifies the module corresponding to bit 9 of the SYNCC enable field of the WUP and SLP instructions. |
| 7 to 0 | SYNCC8 | * | R/W | Specifies the module corresponding to bit 8 of the SYNCC enable field of the WUP and SLP instructions. |

Initial value is 0xFFFFFFFF.

### 56.2.70 Synchronization Control Register 3 (SYNCCR3)

This register specifies the module corresponding to bits 12 to 15 in the SYNCC enable field of the WUP and SLP instructions. Set the number for specifying the module in this register. See **Table 56.22** for the numbers for specifying the module. Do not set a number that is not listed in **Table 56.22**. Do not modify the setting of this register until the value of the TRA bit in SR become 1 and after 1 was written to the RS bit in CR.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC15 | * | R/W | Specifies the module corresponding to bit 15 of the SYNCC enable field of the WUP and SLP instructions. |
| 23 to 16 | SYNCC14 | * | R/W | Specifies the module corresponding to bit 14of the SYNCC enable field of the WUP and SLP instructions. |
| 15 to 8 | SYNCC13 | * | R/W | Specifies the module corresponding to bit 13 of the SYNCC enable field of the WUP and SLP instructions. |
| 7 to 0 | SYNCC12 | * | R/W | Specifies the module corresponding to bit 12 of the SYNCC enable field of the WUP and SLP instructions. |

### 56.2.71 Destination Cache Control Register (DCCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UVDSTSEL | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 15 | UVDSTSEL | 0 | R/W | Sets the register that sets the memory stride of UV plane when outputting YUV422/420 semi-planar.<br>0: Uses the setting value of DSTR register as memory stride<br>1: Uses the setting value of UVDSTR register as memory stride |
| 14 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Notes: 1. When changing the setting of this register by the DL, be sure to execute the SYNCM instruction before making the change.
2. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.72 Source Cache Control Register (SCCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TCMENB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 | TCMENB | 0 | R/W | Select the size of the source cache when pyramidal images filtering not supported. When this bit is 1, set TRIMR.TFE, TRIMR2.MAXLODU and TRIMR2.MAXLODV to 0.<br>0: 32 Kbytes<br>1: 64 Kbytes |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

Notes:

1. When changing the setting of this register by the DL, be sure to execute the SYNCM instruction before making the change.
2. This register is not initialized by the SWRST bit in the control register (CR).

### 56.2.73 Performance Counter Control Register (PEFCCR)

This register controls the following performance-counter-related registers: PEFCTCR, PEFCTMR, PERFCMAXPR, PEFCMINPR, PEFCDCAR, PEFCDCMR, PEFCSCAR, PEFCSCMR. However, counted value of the performance counter is not guaranteed.

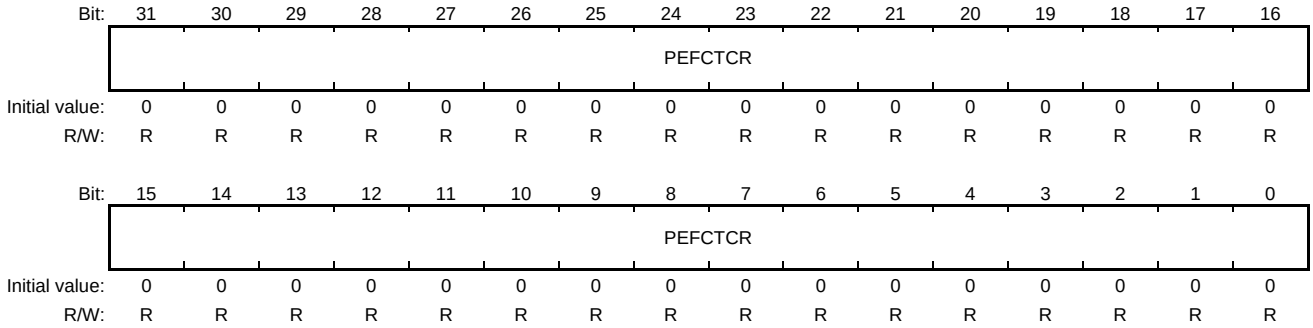
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLR | EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 1 | CLR | 0 | R/W | Clear all the performance counter related registers. |
| 0 | EN | 0 | R/W | Enable all performance-counter-related registers.<br>0: Counting-up by the performance-counter-related registers is disabled.<br>1: Counting-up by the performance-counter-related registers is enabled. |

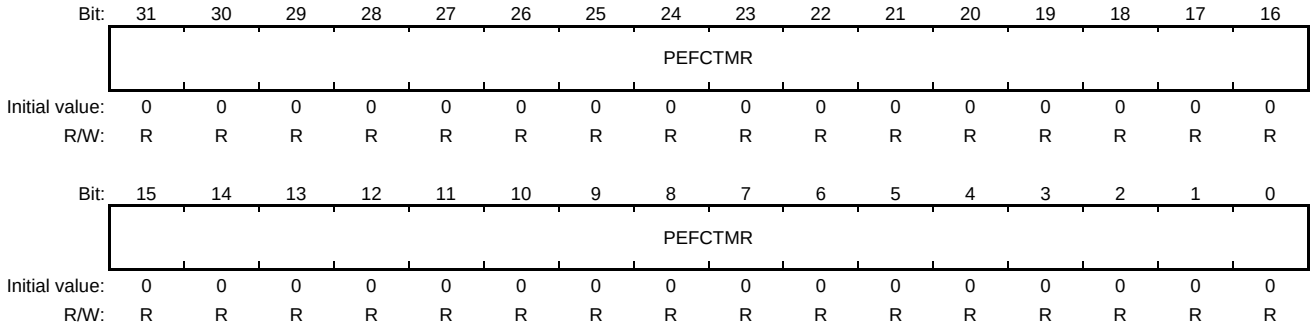
When changing the setting of this register by the DL, be sure to execute the SYNCM instruction before making the change

### 56.2.74 Total Cycle Count Register (PEFCTCR)



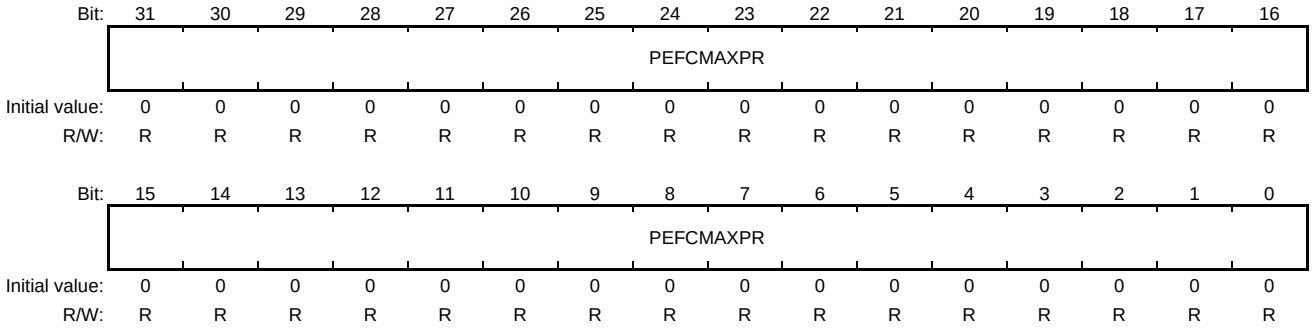

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PEFCTCR | H'0000 0000 | R | Performance counter to count the number of cycles when the setting of the EN bit of PEFCCR is 1.<br>Performance counter to count the total number of cycles in rendering by this module. |

### 56.2.75 Total Cache Miss Cycle Count Register (PEFCTMR)



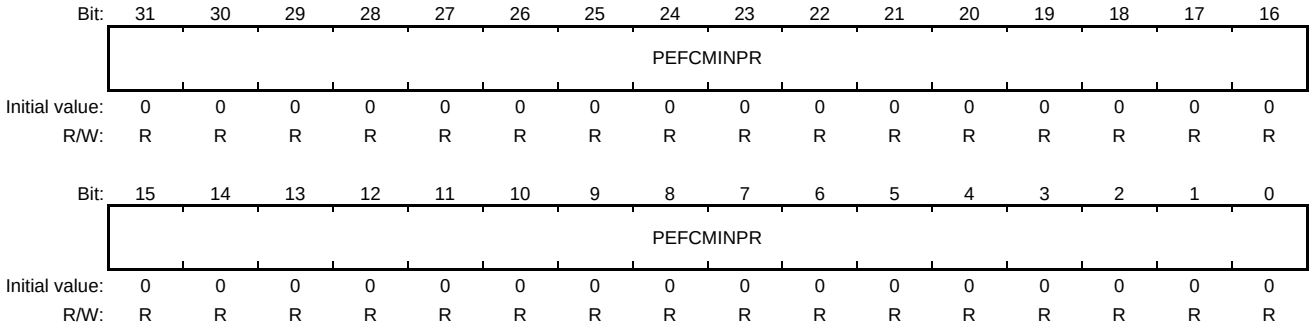

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PEFCTMR | H'0000 0000 | R | Performance counter to count the total number of cycles among those counted by PEFCTCR in which the cache is missed.<br>This counter shows the number of cycles of stalling because the cache is missed. |

### 56.2.76 Maximum Cache Miss Penalty Register (PEFCMAXPR)



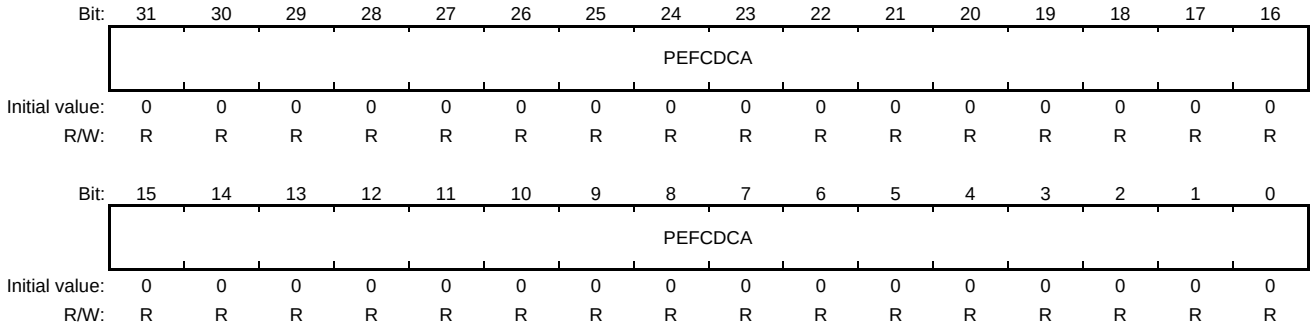

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PEFCMAXPR | H'0000 0000 | R | Performance counter to count the maximum number of cycles necessitated by a cache miss.<br>This register does not show the correct value when the non-blocking cache mode is enabled. |

### 56.2.77 Minimum Cache Miss Penalty Register (PEFCMINPR)



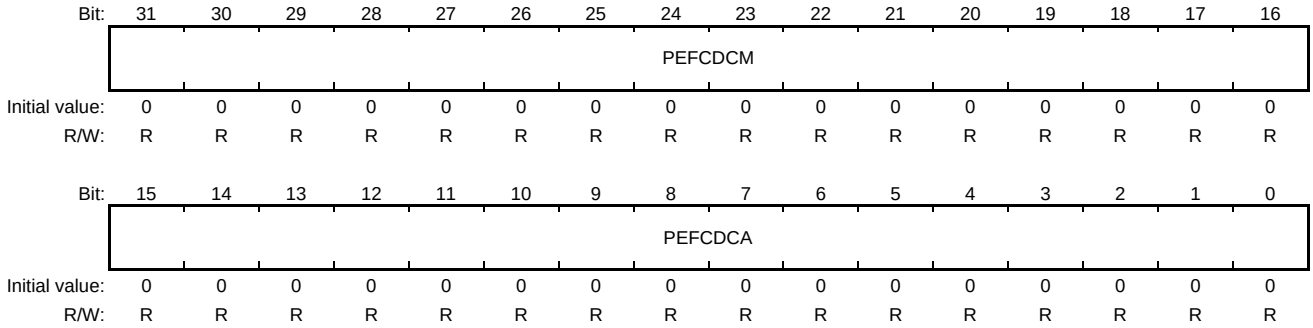

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PEFCMINPR | H'0000 0000 | R | Performance counter to count the minimum number of cycles necessitated by a cache miss.<br>This register does not show the correct value when the non-blocking cache mode is enabled. |

### 56.2.78 Number of Destination Cache Access Register (PEFCDCAR)



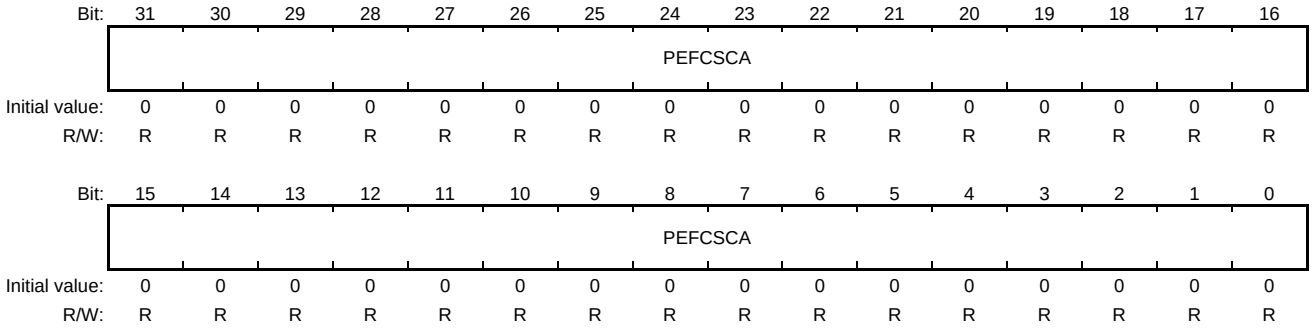

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PEFCDCA | H'0000 0000 | R | Performance counter to count the number of times the destination cache is accessed.<br>This register does not show the correct value when the non-blocking cache mode is enabled. |

### 56.2.79 Number of Destination Cache Miss Register (PEFCDCMR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PEFCDCM | H'0000 0000 | R | Performance counter to count the number of times the destination cache is missed.<br>This register does not show the correct value when the non-blocking cache mode is enabled. |

### 56.2.80 Number of Source Cache Access Register (PEFCSCAR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PEFCSCA | H'0000 0000 | R | Performance counter to count the number of times the source cache is accessed.<br>This register does not show the correct value when the non-blocking cache mode is enabled. |

### 56.2.81 Number of Source Cache Miss Register (PEFCSCMR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEFCSCM | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PEFCSCM | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PEFCSCM | H'0000 0000 | R | Performance counter to count the number of times the source cache is missed.<br>This register does not show the correct value when the non-blocking cache mode is enabled. |

## 56.3 Operation

### 56.3.1 Triangle Drawing

According to the destination (DST) vertex coordinates and source (SRC) vertex coordinates given by the DL, a triangle is drawn in DST using the SRC texture.

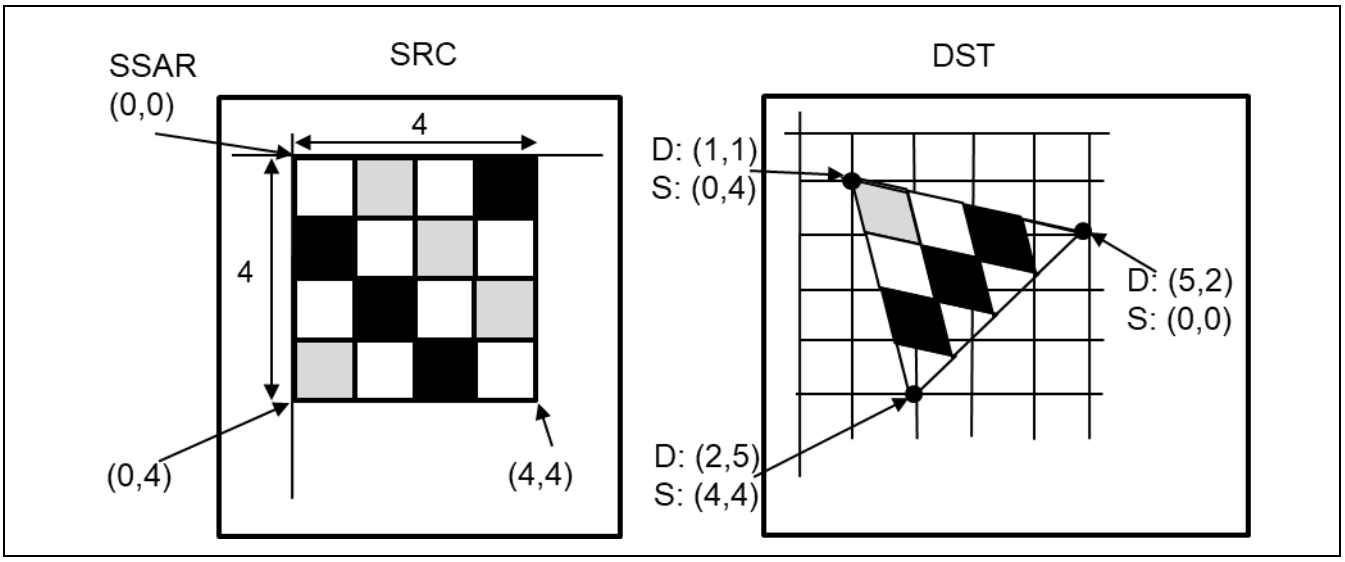



**Figure 56.2 Drawing with Texture Mapping**

For drawing, the DL is used to specify the (X, Y) coordinates in the DST area and the corresponding (u, v) coordinates in the SRC area of the desired triangle. Figure 56.2 shows an example in which a triangle is drawn in the DST area such that X, Y coordinates of its vertexes are (1, 1), (5, 2), and (2, 5) with texture mapping defined by uv coordinates (0, 4), (0, 0), and (4, 4). Specify the destination coordinates X and Y in an

integer or a fixed-point number (lower 2 bits) according to the mode set by the DDP bit in UVDPOR. Specify source coordinates (u, v) with the precision by the UVDPO bits in UVDPOR.

When the SRC coordinates exceed the values given by the source width SUSR and source height SVSR, SUSR and SVSR are replacing these values.

The size of the circumscribed rectangle of a triangle must be less than 128 pixels for both X-direction and Y-direction.

For setting coordinates to the DL, refer to section 56.5, Display List.

### (1) N-Vertex Strip

Vertex data can be shared by multiple triangles to draw strip-type triangles, thus saving DL capacity and improving performance.

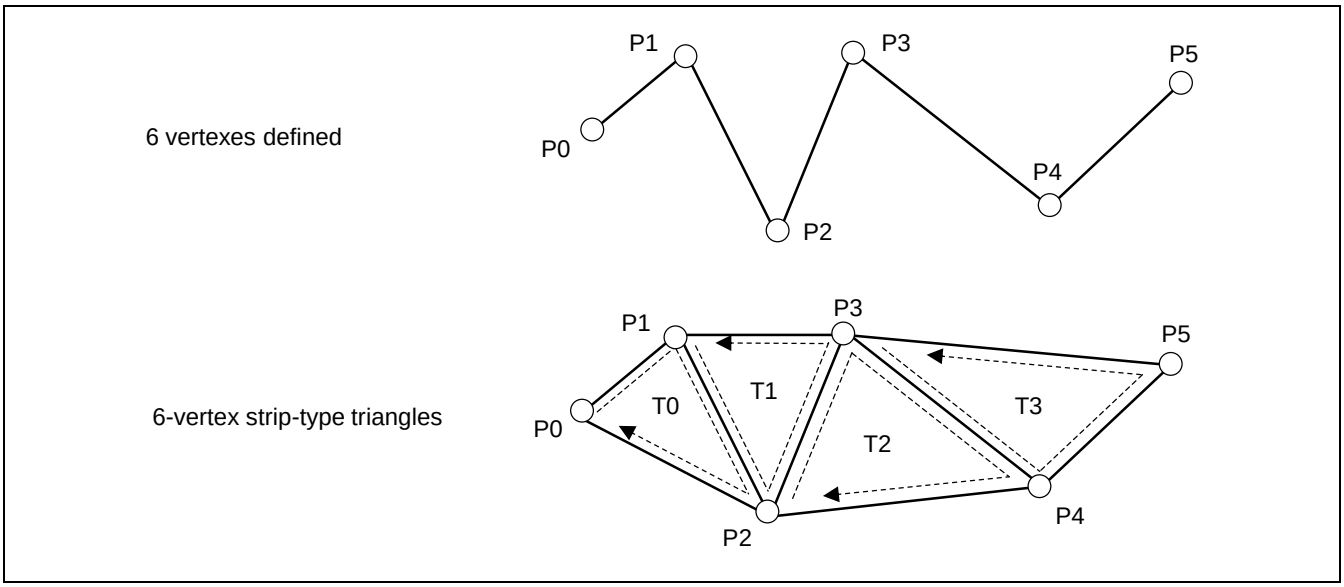



**Figure 56.3 N-Vertex Strip**

Figure 56.3 shows an example of 6-vertex triangles. First, triangle T0 is drawn using the first three vertexes P0, P1, and P2, and then triangle T1 is drawn using the last two vertexes P1 and P2 of triangle T0 and the next vertex P3. Similarly, the next triangle is drawn using the last two vertexes of the preceding triangle and the next vertex. This process is repeated until the strip is completed.

### (2) Clockwise

Drawing is performed only if the order of the first three vertexes of a strip-type triangle matches the order that is specified by the TCM bit in the triangle mode register. If there is no match, the strip-type triangle will not be drawn. In the example given in Figure 56.3, since the order of vertexes of the triangle composed of P0, P1, and P2 is clockwise, the strip-type triangle composed of P0 to P5 will be drawn only when the TCM bit in the triangle mode register is 1.

### (3) Rule to Prevent Duplicate Drawing

This module applies the rule described below to prevent duplicate drawing of the pixels for a side shared by the adjacent triangles when drawing triangles. For this reason, some pixels which exist in response to the specified vertex coordinates of triangles may not be drawn if they are for sides.

If either of the two conditions below is met and the vector passes through the corresponding pixel grid (the fractional part in the X and Y coordinates is 0), the pixels will not be drawn.
In case that a triangle is handled as three clockwise vectors, there are the following two conditions:
(a) Component y of the vector is positive.
(b) Component y of the vector is 0 and component x is negative.
In case that a triangle is handled as three counter-clockwise vectors, there are the following two conditions:
(a) Component y of the vector is negative.
(b) Component y of the vector is 0 and component x is positive.
An example is given in the figure below. For the sides indicated by the solid lines, the pixels will be drawn even if the corresponding vector passes through the pixel grid. For the sides indicated by the broken lines, the pixels will not be drawn if the corresponding vector passes through the pixel grid. As result, the pixels with a cross on the pixel grid will be drawn.

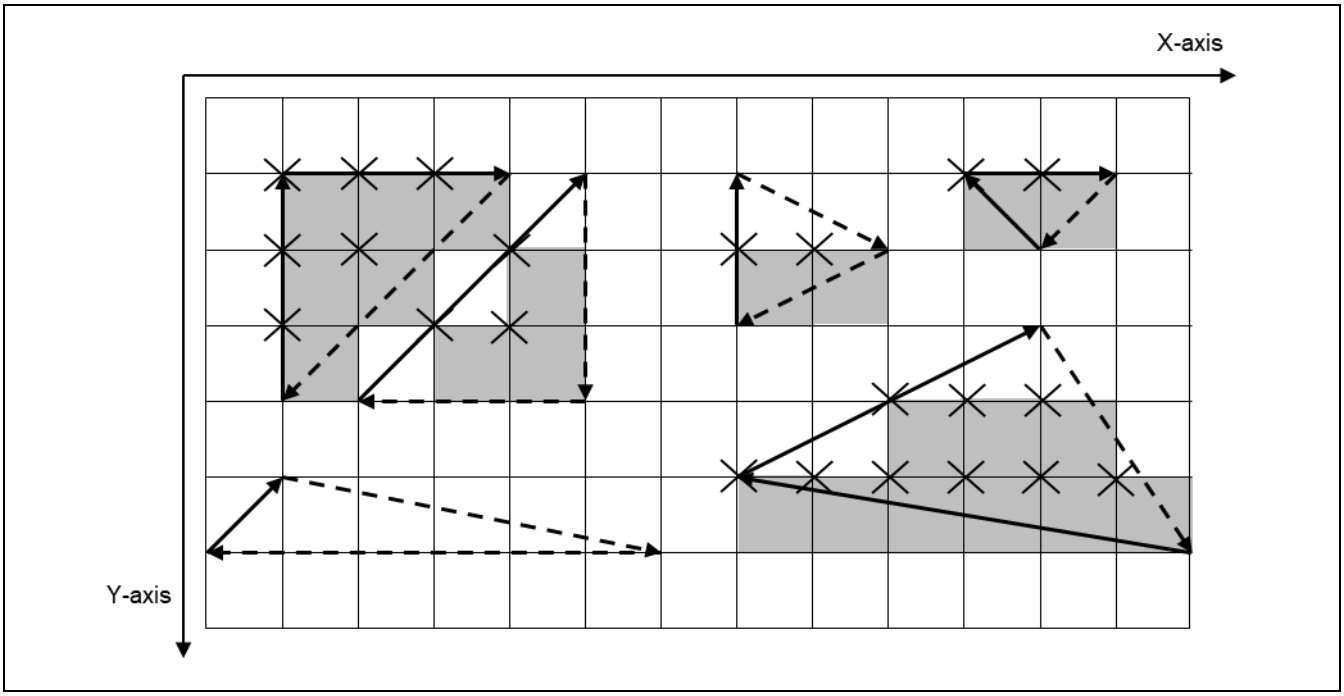



**Figure 56.4 Example of Rule to Prevent Duplicate Drawing**

The figure below shows an example of using the automatic vertex generation function to draw contiguous triangles with the origins located at coordinates (1, 1) and (1, 4) and the two 3 x 3 meshes used. Each of the triangles has pixels that are not drawn because of this rule. If two triangles are adjacent to each other, the shared side will be drawn on the either of the two sides, so no required pixels are missed. However, since there are no adjacent triangles at the right and bottom edges in this example, the pixels (points indicated by the broken circles) will not be drawn because of this rule.

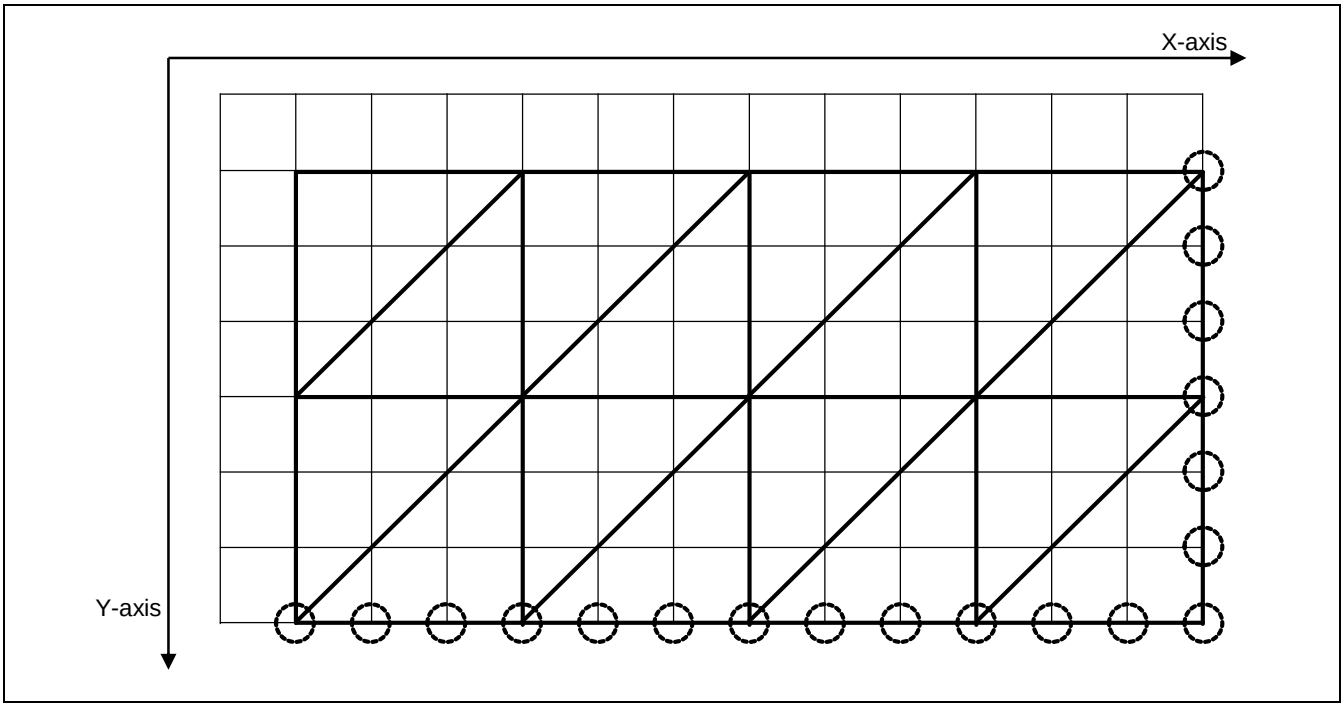



**Figure 56.5 Example of Drawing Contiguous Triangles**

### 56.3.2 Coordinate Space

The size of the DST drawing coordinate space is 2,048 x 2,048, in which $0 \leq X \leq 2{,}048$ and $0 \leq Y \leq 2{,}048$. An object (triangle) specified with the DL is actually drawn only when it is in the clipping area defined by the MIN and MAX clipping X and Y registers in the drawing coordinate space. In the DL, vertex coordinates can be specified with signed 16 bits if the destination coordinates are in integer, or with a signed fixed-point number (where the lower 2 bits are below the decimal point) if the destination coordinates are in sub-pixels.

The size of the SRC drawing coordinate space is $0 \leq u < 2{,}048$ and $0 \leq v < 2{,}048$. The (u, v) values should always be specified in the SRC coordinate space. Note that negative values cannot be used for u or v. If used, operation will be undefined.

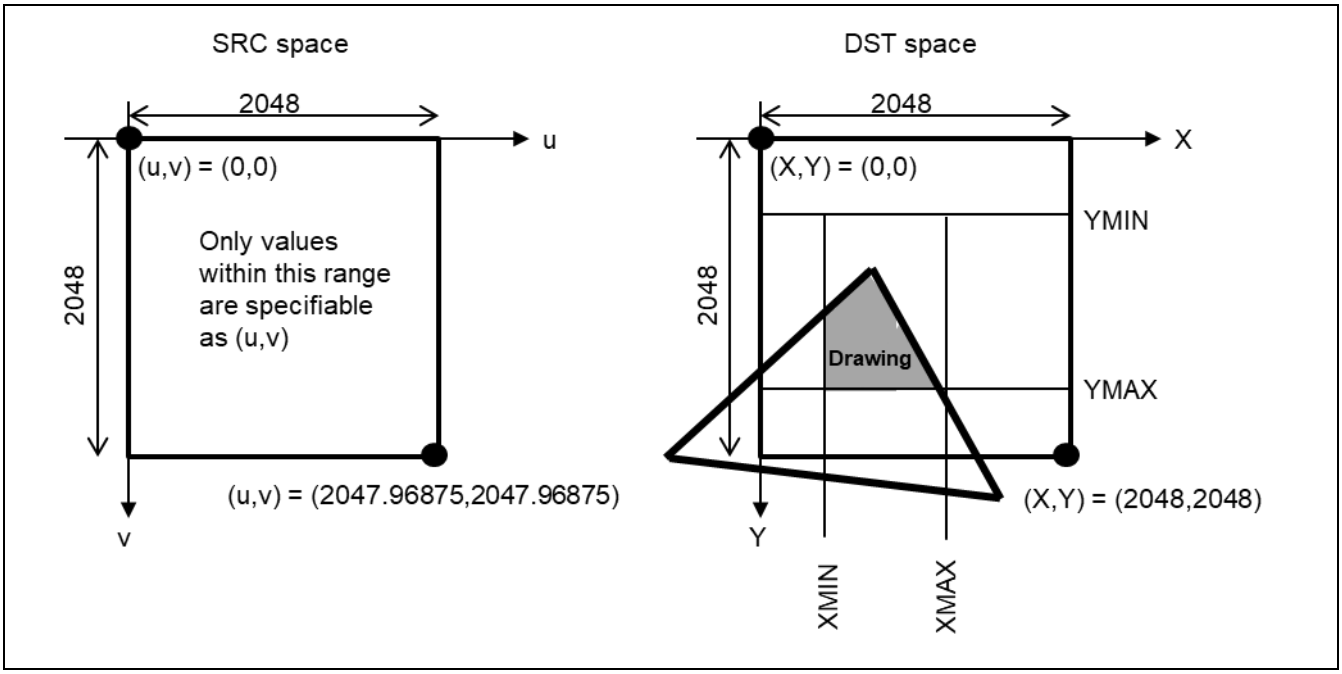



**Figure 56.6 Coordinate Space**

### 56.3.3 Automatic Coordinate Generation

When either SRC or DST is a rectangular mesh and the mesh widths (X and Y directions) are set, either SRC or DST coordinates can automatically be generated using the internal counter. According to the mesh width settings, N-vertex triangle strip coordinates are internally generated.Mesh position and size can be specified with registers AMXSR, AMYSR, AMXOR, and AMYOR. Automatic coordinate generation will be available when the AUTOSG and AUTODG bits in the triangle mode register are set to 1 as required.

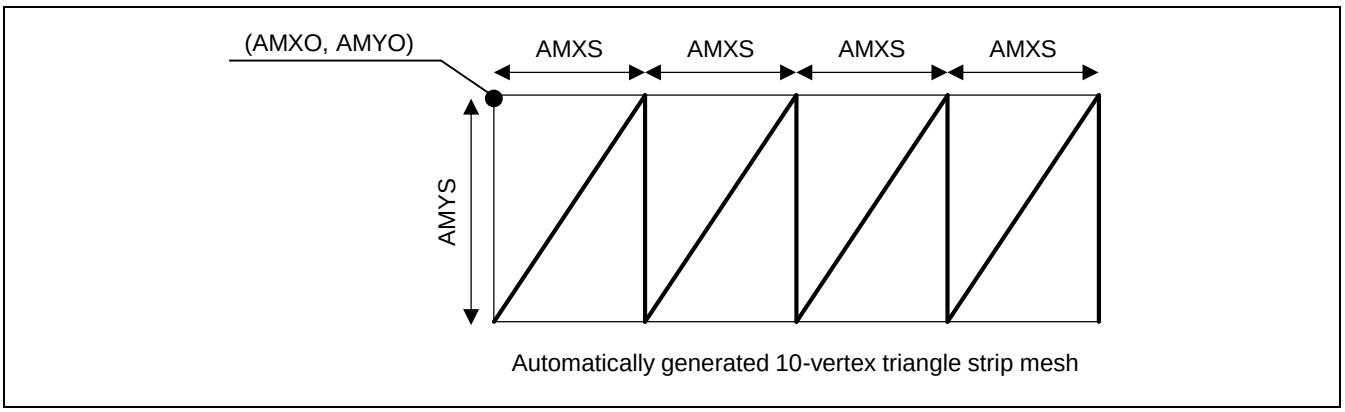



**Figure 56.7 Automatic Coordinate Generation**

The maximum value of coordinates that can be generated is 2,047 if the destination coordinates are in destination sub-pixel mode (i.e. the DDP bit in UVDPOR is 1) and it is 8,191 if not in destination sub-pixel mode. The source coordinate is 2,047/(2^(UVDPO value in UVDPOR)).

### 56.3.4 Adding offset to coordinate

It is possible to add offset to the X/Y and u/v coordinates automatically generated or specified by DL. Therefore, it is feasible to parallel shift the triangular mesh in X/Y or u/v space.

Specify the offset to X/Y coordinates using XYOFSR register. Specify the offset to u/v coordinates using UVOFSR register. Note that X/Y and u/v coordinate specified in Figure 56.6 are the coordinates after the addition of the offset set in XYOFSR register and UVOFSR register. Therefore, user need to set the offset XYOFSR register and UVOFSR register so that XY coordinates after addition are not exceed the maximum of X/Y ( signed 16bit values ) and u/v coordinates before and after the addition are not exceed u/v ($0 \leq u < 2{,}048$ and $0 \leq v < 2{,}048$) specified in Figure 56.6.

$X = X' + XOFS$

$Y = Y' + YOFS$

$u = u' + UOFS$

$v = v' + VOFS$

Where $X'$, $Y'$, $u'$, and $v'$ are the coordinates that are specified by DL or generated by Automatic Coordinate Generation.

The value space of XOFS and YOFS are determined by DDP bit in the UVDPOR register.

DDP bit = 0: $-2048 \leq XOFS/YOFS \leq 2047$

DDP bit = 1: $-512.00 \leq XOFS/YOFS \leq 511.75$

The value space of UOFS and VOFS are from $-1024 \leq UOFS/VOFS < 1024$.

If the lower bits of X/Y coordinates or u/v coordinates in UVDPOR register are set as fractional part, the same number of lower bits of the offset will also be set as fractional part.

Note: In case of using Rotation and scaling-down, X, Y, u and v coordinate after addition of XOFS/YOFS and UOFS/VOFS are used in Sub-divided area.

### 56.3.5 Bilinear Filtering

In texture mapping by the TRI instruction, when the calculated Y (luminance) coordinates are not integers, the color data of Y is defined by linearly interpolating (bilinear filtering) the four pixels nearby. Bilinear filtering will be unavailable when the BFE bit in the triangle mode register (TRIMR) is set to 0; in this case, the decimal parts of the uv coordinates are omitted and the integer parts alone are used.

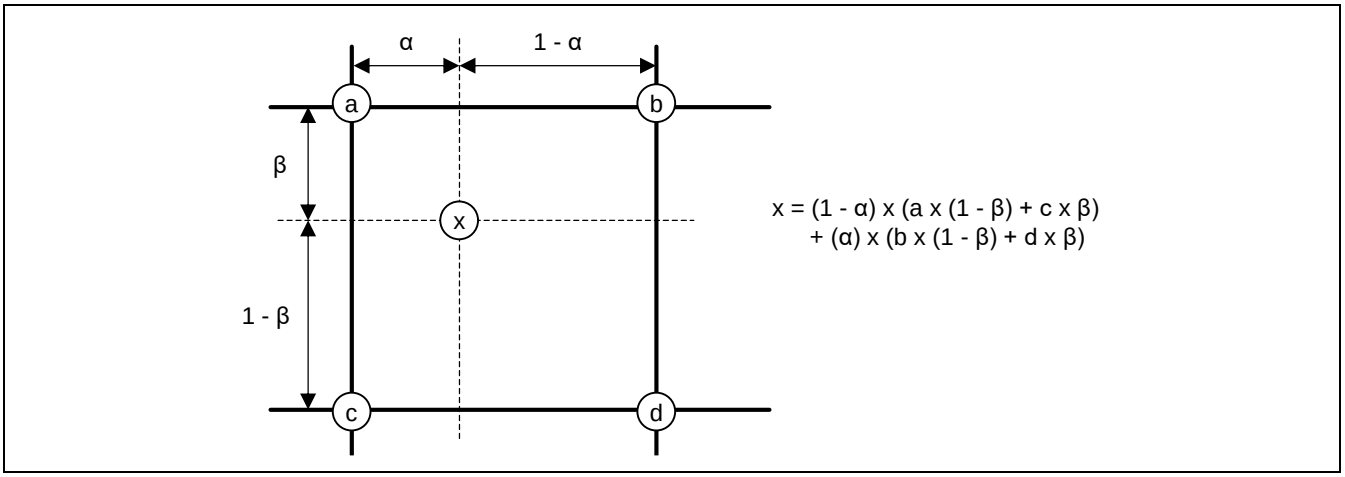



**Figure 56.8 Bilinear Filtering for Y (Luminance)**

When the UV (color difference) coordinates are not integers, a special bilinear filtering for UV is used. Specifically, for drawing the even-number data (U) of the color difference in the destination, U is defined by linearly interpolating the four even-number data nearby in the source, whereas for drawing the odd-number data (V) of the color difference, V is defined by linearly interpolating the four odd-number data nearby in the source.

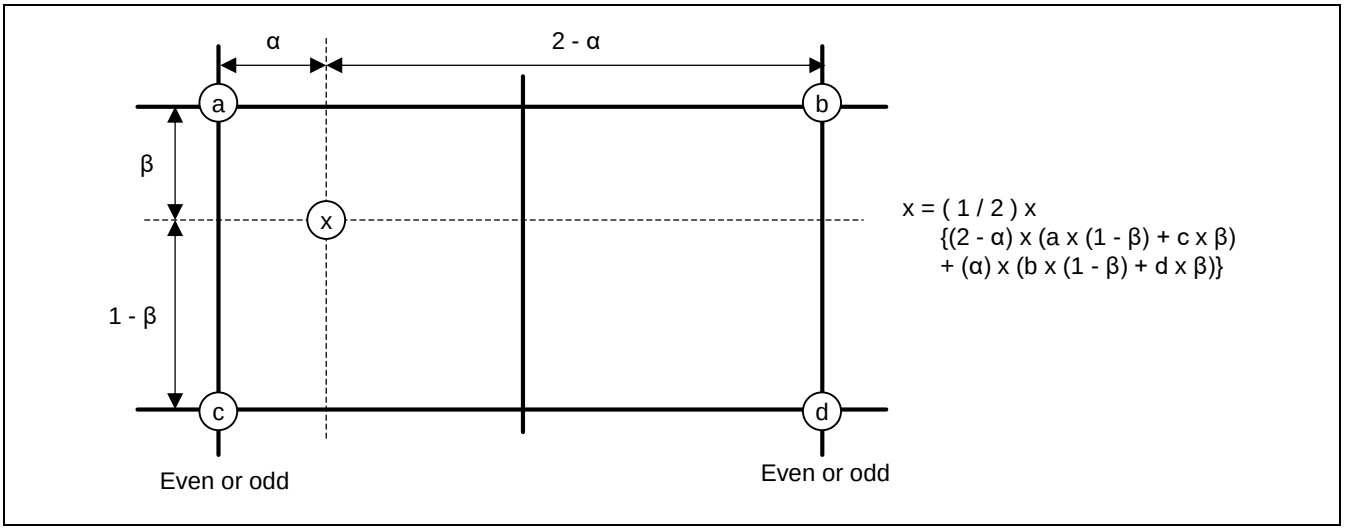



**Figure 56.9 Bilinear Filtering for UV (Color Difference)**

### 56.3.6 Data Location in Memory

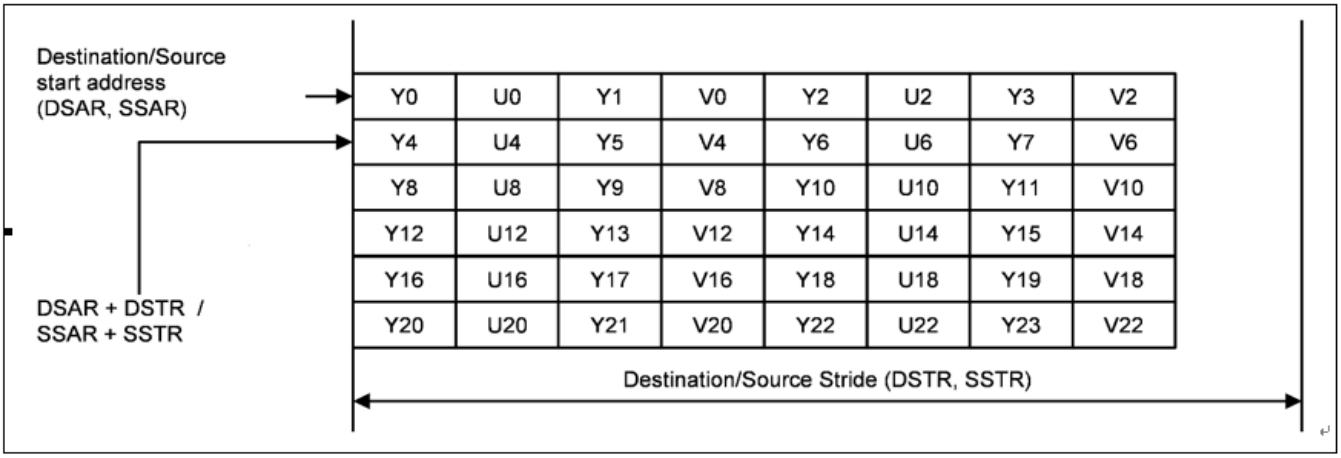



**Figure 56.10 Allocation of YUV Data to Memory (YUV422 Interleaved)**

In interleaved mode, the YUV data after the drawing are laid out as U data, V data, and Y data, in that order. The vertical orientation on the screen is governed by the memory stride (destination stride) which is used to handle the data in the memory two-dimensionally. On the screen, the pixels below a line are assigned to the addresses that are the original addresses plus the stride. The YUV data are handled in units of 16 bits or 32 bits, each being a combination of Y and U/V values. When using interleaved mode, use the YUV422E bit in CMRCR2. When interchanging Y and U/V, use the YCFORM bit in TRICR for the destination data and YUV422FORM bit in CMRCR2 for the source data. After the replacement, the data will be laid out as shown in the figure below:

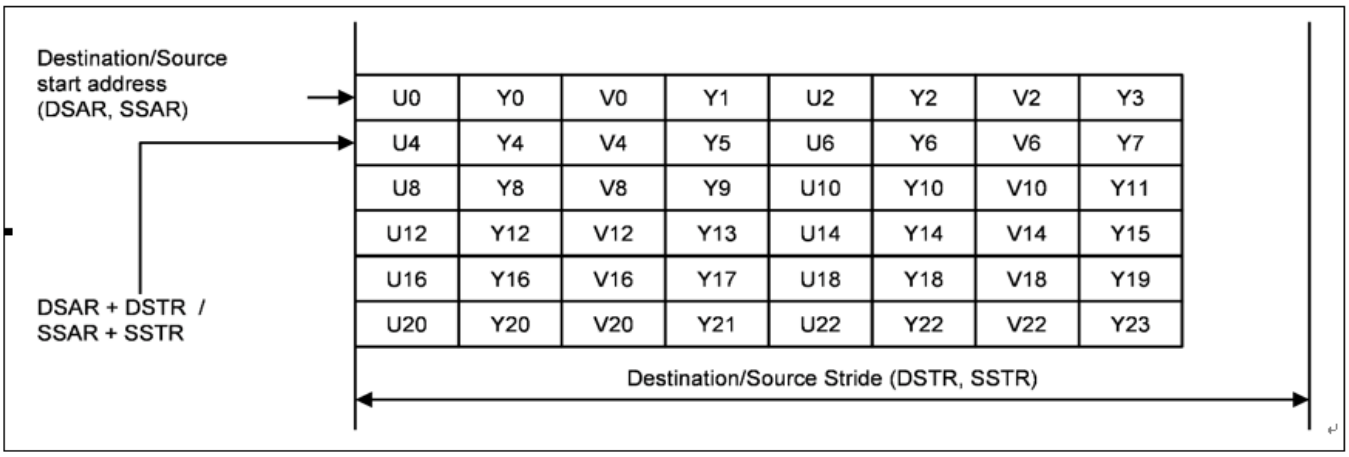



**Figure 56.11 Allocation of YUV Data to Memory (Interleaved Output, YCFORM = 1 or Interleaved Input, YUV422FORM = 1)**

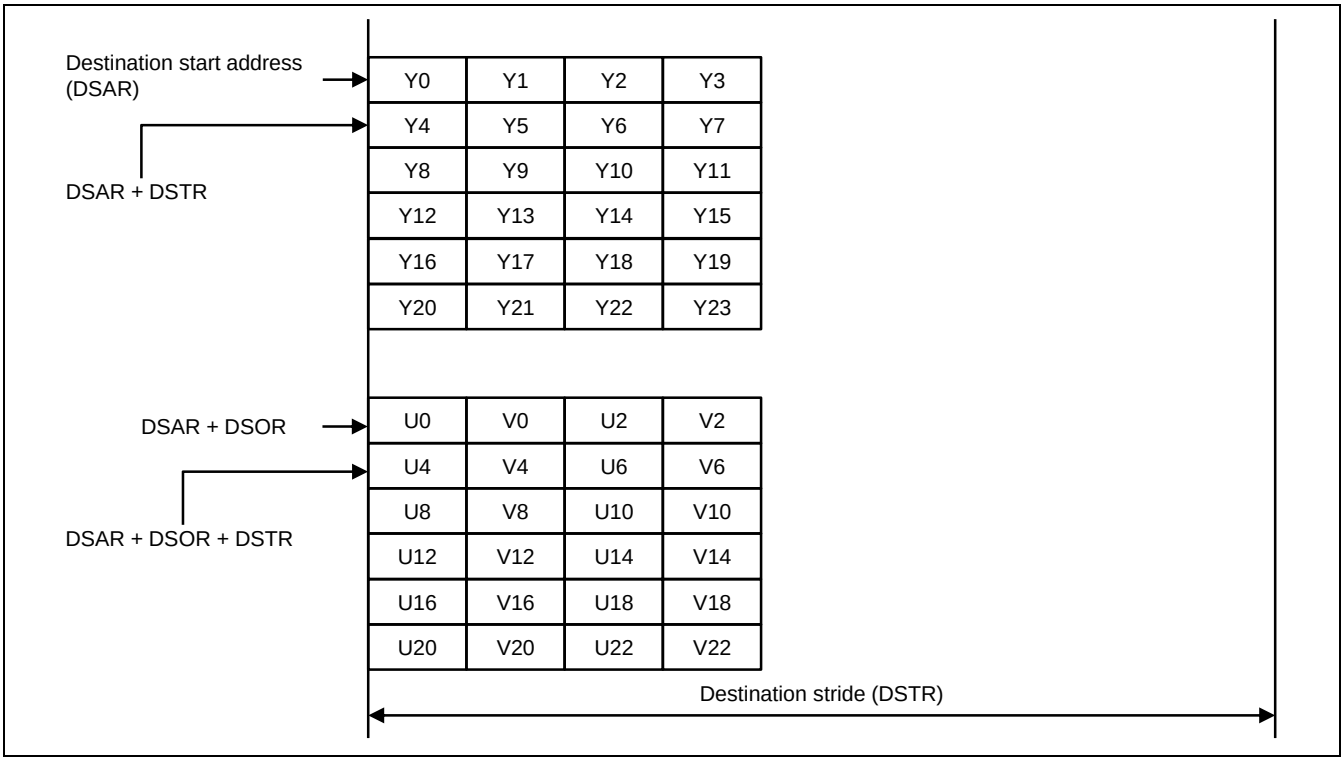



**Figure 56.12 Allocation of YUV Data in Memory (Separate Output)**

In the case of separate output of YUV422 data, the Y and UV data are assigned to separate areas. The address where the Y data start, as in the case of interleaved output, is specified in the DSAR register.

The size of the Y values is 8 bits if the Y data are 8 bpp; 16 bits (flush to the LSB; the 6 upper bits are 0) if the Y data are 10 bpp; 16 bits (flush to the LSB; the 4 upper bits are 0) if the Y data are 12 bpp; 16 bits (flush to the LSB; the 2 upper bits are 0) if the Y data are 14 bpp; and 16 bits if the Y data are 16 bpp.

The size of both the U and V values is 8 bits if the UV data are 8 bpp; 16 bits (flush to the LSB; the upper 6 bits are 0) if the UV data are 10 bpp; 16 bits (flush to the LSB; the upper 4 bits are 0) if the UV data are 12 bpp; 16 bits (flush to the LSB; the upper 2 bits are 0) if the UV data are 14 bpp; and 16 bits if the UV data are 16 bpp.

The address where the UV data start is the address specified in DSOR added to the start address of the Y data specified in DSAR. The stride is common to both cases, and is specified in DSTR. Set YISM bit in CMRCR to use the separate output mode.

In the case of Y exclusive mode, assignment is in the same manner as the Y data for separate output. In the case of U/V exclusive mode, assignment is in the same manner as the UV data for separate output with DSOR==0. To use the Y or U/V exclusive mode, clear the YUV422E bit in CMRCR2. Also, clear the YCM bit in CMRCR for Y exclusive mode, and set the bit for U/V exclusive mode.

**Tile Addressing Mode**

This module supports not only linear addressing where the two-dimensional image data are arranged like raster in memory but also a tile addressing mode where the data are arranged like tile.

In tile addressing mode, continuous memory addresses are interleaved with data for multiple Y coordinates, thus improving the cache-hit ratio for vertical access to the image data (e.g., for rotation of an image) and also improving processing performance.

In tile addressing mode, the data are stored with the Y coordinate incremented in 16-byte units within 64-byte sections. An example where 8bpp Y data are exclusively output is shown below. (x, y) in the figure below indicate the x and y coordinates of pixels.

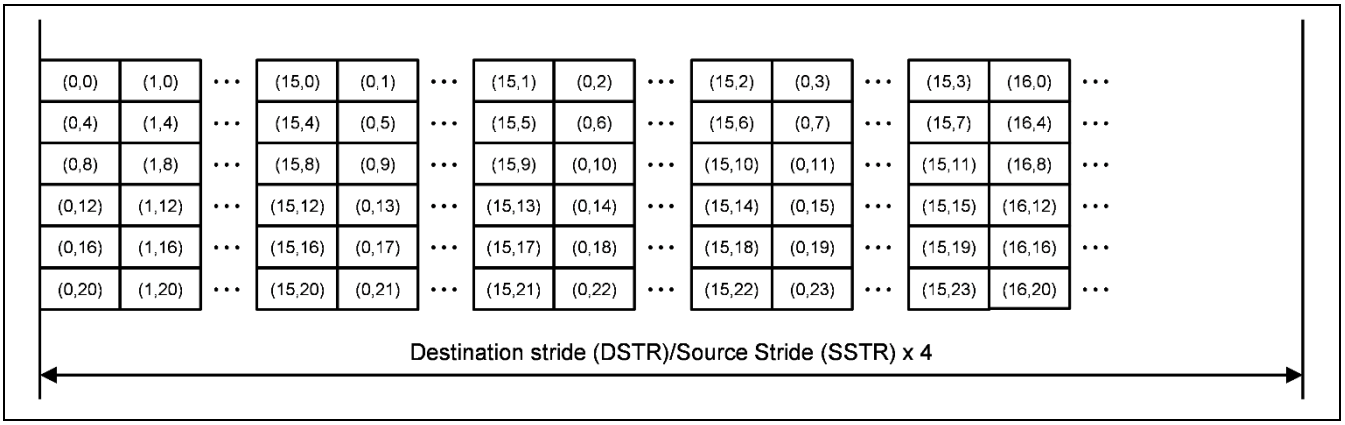



**Figure 56.13 Data in Memory in Tile addressing mode (Y Exclusive, 8 bpp)**

The tile addressing mode can be set up individually for source and destination images. To set up the tile addressing mode for the source image, set the TCTE bit in CMRCR2 to 1. To set up the tile addressing mode for the destination image, set the DCTE bit in CMRCR2 to 1.

To set the stride in tile addressing mode, set the corresponding bits in DSTR, SSTR, and SSTRn to a value equal to one quarter of the actual memory stride.

### 56.3.7 Supported data format and data location in Memory

This module supports YUV422 format and YUV420 format for the input/output image. Besides, each format supports interleaved format and semi-planar format. It is also possible to process only the Y plane or only the UV plane. The combinations of the supported formats and the relevant register settings are shown below. The combinations other than following table are prohibited.

**Table 56.13 Combinations of the supported input/output format and the relevant register settings**

| Input Format | Output Format | CMRCR2 | | | CMRCR | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | YUV422E | YUVSEMI | UVC | YISM | UVS | YCM | YOM | When EFPE=1, following bits of TRIMR register are use. |
| YUV422 Interleave*2 | YUV422 Interleave | 1 | 0 | 0 | 0 | 0 | 0 | 0 | SHFE/SHFVAL*1<br>Both Y and UV shift |
| | YUV422 Semi-planar | 1 | 0 | 0 | 1 | 0 | 0 | 0 | SHFE/SHFVAL<br>UVSHFE/UVSHFVAL |
| | Y Plane Only | 1 | 0 | 0 | 1 | 0 | 0 | 1 | SHFE/SHFVAL*1<br>Y-only shift |
| YUV422 Semi-planar*2 | YUV422 Interleave | 0 | 1 | 0 | 0 | 0 | 0 | 0 | SHFE/SHFVAL<br>UVSHFE/UVSHFVAL |
| | YUV422 Semi-planar | 0 | 1 | 0 | 1 | 0 | 0 | 0 | SHFE/SHFVAL<br>UVSHFE/UVSHFVAL |
| | YUV420 Semi-planar*4 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | SHFE/SHFVAL<br>UVSHFE/UVSHFVAL |
| YUV420 Semi-planar*2 | YUV422 Interleave*5 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | SHFE/SHFVAL<br>UVSHFE/UVSHFVAL |
| | YUV422 Semi-planar*5 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | SHFE/SHFVAL<br>UVSHFE/UVSHFVAL |
| | YUV420 Semi-planar *6 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | SHFE/SHFVAL<br>UVSHFE/UVSHFVAL |
| Y Plane Only<br>(Semi-planar format) | Y Plane Only*3<br>(Semi-planar format) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SHFE/SHFVAL*1<br>Y-only shift |
| UV Plane Only (YUV422 Semi-planar format) | UV Plane Only*3 (YUV422 Semi-planar format) | 0 | 0 | 0 | 0 | 0 | 1 | 0 | SHFE/SHFVAL*1<br>UV-only shift |
| | UV Plane Only*3 *4 (YUV420 Semi-planar format) | 0 | 0 | 1 | 0 | 0 | 1 | 0 | SHFE/SHFVAL*1<br>UV-only shift |
| UV Plane Only (YUV420 Semi-planar format) | UV Plane Only*3 *5 (YUV422 Semi-planar format) | 0 | 0 | 1 | 0 | 1 | 1 | 0 | SHFE/SHFVAL*1<br>UV-only shift |
| | UV Plane Only*3 *7 (YUV420 Semi-planar format) | 0 | 0 | 0 | 0 | 1 | 1 | 0 | SHFE/SHFVAL*1<br>UV-only shift |

Notes: 1. Set UVSHFE=0 and UVSHFVAL=0 when not using these bits.

2. Set RSCR.RSE=0 when using these formats. These formats read Y data and UV data simultaneously.

3. These output from the IMR-LX6 basic function block are input to Rotator & Scaler extension when RSCR.RSE = 1.

4. For these IMR basic block's outputs, IMR-LX6 hardware automatically change the output Y-coordinate to half (1/2) of the input Y-coordinate for UV plane.

5. For these IMR basic block's outputs, IMR-LX6 hardware automatically change the output v-coordinate to half (1/2) of the input v-coordinate for UV plane.

6. For these IMR basic block's outputs, IMR-LX6 hardware automatically change the output Y- and v-coordinates to half (1/2) of the input Y-and v-coordinates for UV plane.

7. For these IMR basic block's outputs, IMR-LX6 hardware automatically change the output Y- and v-coordinates to half (1/2) of the input Y-and v-coordinates for UV plane. For the Rotator & Scaler extension, when processing uv420 plane only with UVS=1, from the sub-divided area of 4 rows, only 2 rows from the top can be output as described in Figure 56.40. Therefore, the output data in one line corresponds to the image of two pixels in vertical direction. User needs to double the Y and v coordinate for the original UV420 data and destination to generate the coordinates for the TRI command. IMR-LX6 hardware automatically halves the Y and v coordinates when outputting to Rotator & Scaler function, so drawing 128x128 or 64x128 pixels writes to one sub-divided area 128 x 64 or 64 x 64 units.

The supported bit depth and the relevant settings are as below. Any operations performed using settings other than those listed in this table cannot be guaranteed. If the bit depth is 8bpp, the data will be stored in 1-byte unit, otherwise, it will be stored in 2-byte unit with bottom alignment. For example, if the bit depth is 12bpp, the upper 4-bits of 2-byte data are always 0.

There may be restrictions regarding the combinations of bit depth of Y plane and UV plane, so please refer to the detailed description of each format.

**Table 56.14 Supported bit depth and the relevant register settings (Y plane, input)**

| Bit depth (Y plane) | CMRCR | | |
|---|---|---|---|
| | SY1416 | SY12 | SY10 |
| 8bpp | 00 | 0 | 0 |
| 10bpp | 00 | 0 | 1 |
| 12bpp | 00 | 1 | 0 |
| 14bpp*1 | 10 | 0 | 0 |
| 16bpp*1 | 11 | 0 | 0 |

Note: *1 This bit depth setting is supported only when using the extended filtering pipeline.

**Table 56.15 Supported bit depth and the relevant register settings (UV plane, input)**

| Bit depth (UV plane) | CMRCR | |
|---|---|---|
| | SUV1416 | SUV |
| 8bpp | 00 | 00 |
| 10bpp | 00 | 01 |
| 12bpp | 00 | 10 |
| 14bpp*1 | 10 | 00 |
| 16bpp*1 | 11 | 00 |

Note: *1 This bit depth setting is supported only when using the extended filtering pipeline.

**Table 56.16 Supported bit depth and the relevant register settings (Y plane, output)**

| Bit depth (Y plane) | CMRCR | | |
|---|---|---|---|
| | DY1416 | Y12 | Y10 |
| 8bpp | 00 | 0 | 0 |
| 10bpp | 00 | 0 | 1 |
| 12bpp | 00 | 1 | 0 |
| 14bpp*1 | 10 | 0 | 0 |
| 16bpp*1 | 11 | 0 | 0 |

Note: *1 This bit depth setting is supported only when using the extended filtering pipeline.

**Table 56.17 Supported bit depth and the relevant register settings (UV plane, output)**

| Bit depth (UV plane) | CMRCR | |
|---|---|---|
| | DUV1416 | DUV |
| 8bpp | 00 | 00 |
| 10bpp | 00 | 01 |
| 12bpp | 00 | 10 |
| 14bpp*1 | 10 | 00 |
| 16bpp*1 | 11 | 00 |

Note: *1 This bit depth setting is supported only when using the extended filtering pipeline.

The details of each format and the memory allocation in Linear Addressing Mode are described below. Tile Addressing Mode will also be described at the last part of the section.

#### 56.3.7.1 YUV422 Interleaved format

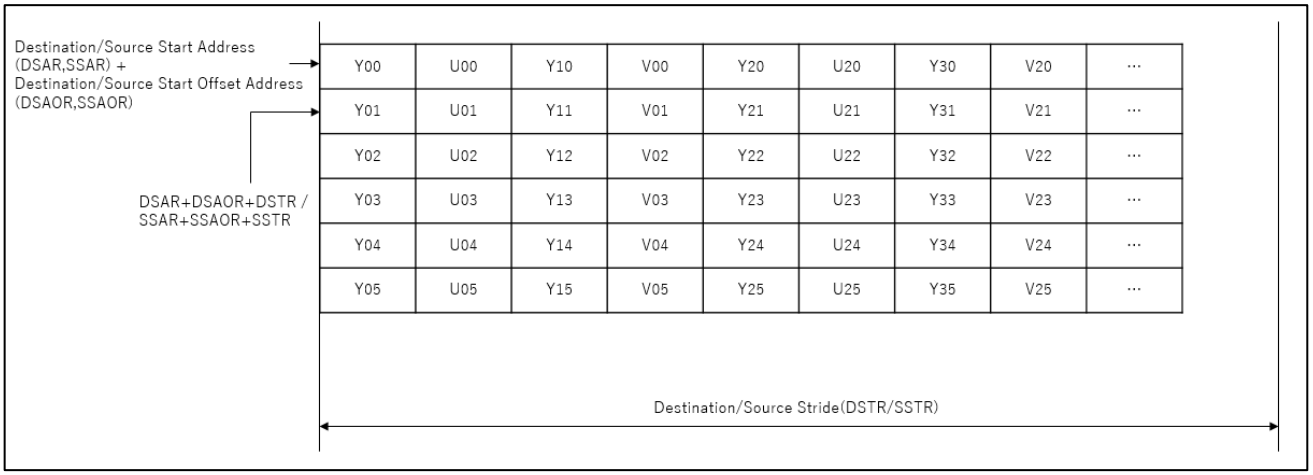



**Figure 56.14 Allocation of YUV Data to Memory (YUV422 Interleaved Output, YCFORM = 0 or YUV422 Interleaved Input, YUV422FORM = 0)**

In YUV422 Interleaved format, YUV data is arranged in the memory in the order of Y data, U data, Y data, and V data. The vertical direction on the screen follows the memory stride used to handle memory data in two-dimension state. The pixels one line below on the screen are placed at the stride-added address. The YUV data is handled in 16-bit and 32-bit unit for Y and U/V respectively.

The start address from which input data is read is obtained by adding SSAR and SSAOR. Also, the start address to which output data is written is obtained by adding DSAR and DSAOR. Memory stride is set using SSTR for input data and DSTR for output data.

If the order of Y and U/V is changed, use the YCFORM bit of TRICR register for output and the YUV422FORM bit of CMRCR2 for input. Memory allocation in this case is shown in the figure below.

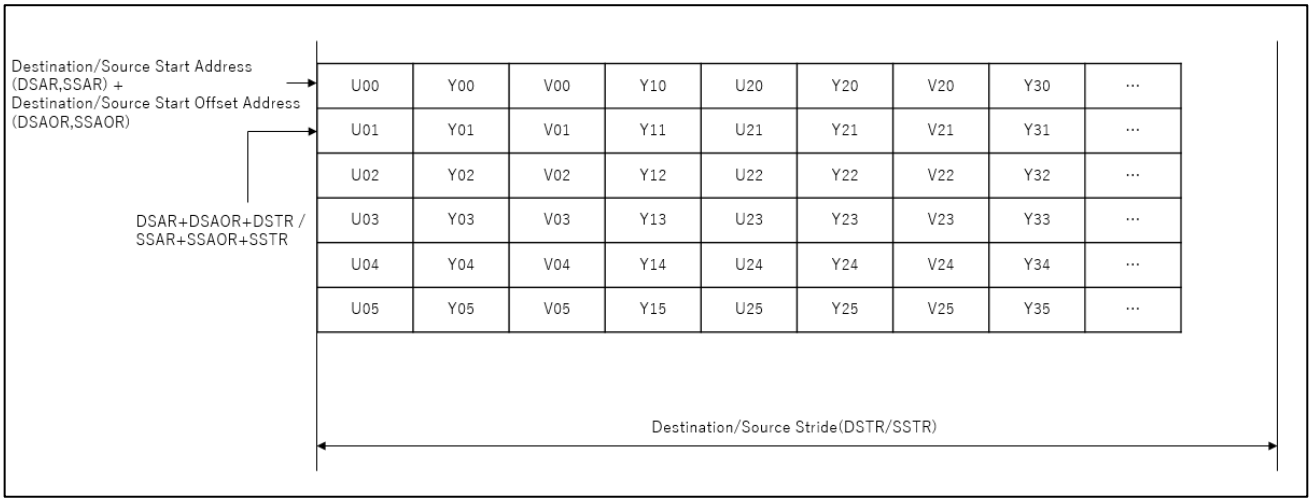



**Figure 56.15 Allocation of YUV Data to Memory (YUV422 Interleaved Output, YCFORM = 1 or YUV422Interleaved Input, YUV422FORM = 1)**

To change the order of the input U and V, set UVFORM bit of CMRCR2 to 1. The order of the output U and V cannot be changed. The order of the output follows the order changed according to the setting of UVFORM of CMRCR2.

For YUV422 Interleaved format, the bit depth (bpp) of Y plane and UV plane should be the same for input or output. Otherwise, the operation cannot be guaranteed.

#### 56.3.7.2 YUV422 Semi-planar format

For YUV422 Semi-planar format, the YUV data has the Y plane and the UV plane placed separately in the memory. The UV plane is placed in the memory in the order of U data, V data, U data, and V data. The vertical direction on the screen follows the memory stride used to handle memory data in two-dimension state. The pixels one line below on the screen are placed at the stride-added address. The Y/UV data is handled in 8-bit or 16-bit unit.

For the YUV422 Semi-planar format for input or output, set the Y plane bit depth (bpp) greater than or equal to the UV plane bit depth. Otherwise, the operation cannot be guaranteed.

Create the vertex data on DL based on Y plane for both the input and output data.

**(1) Input data**

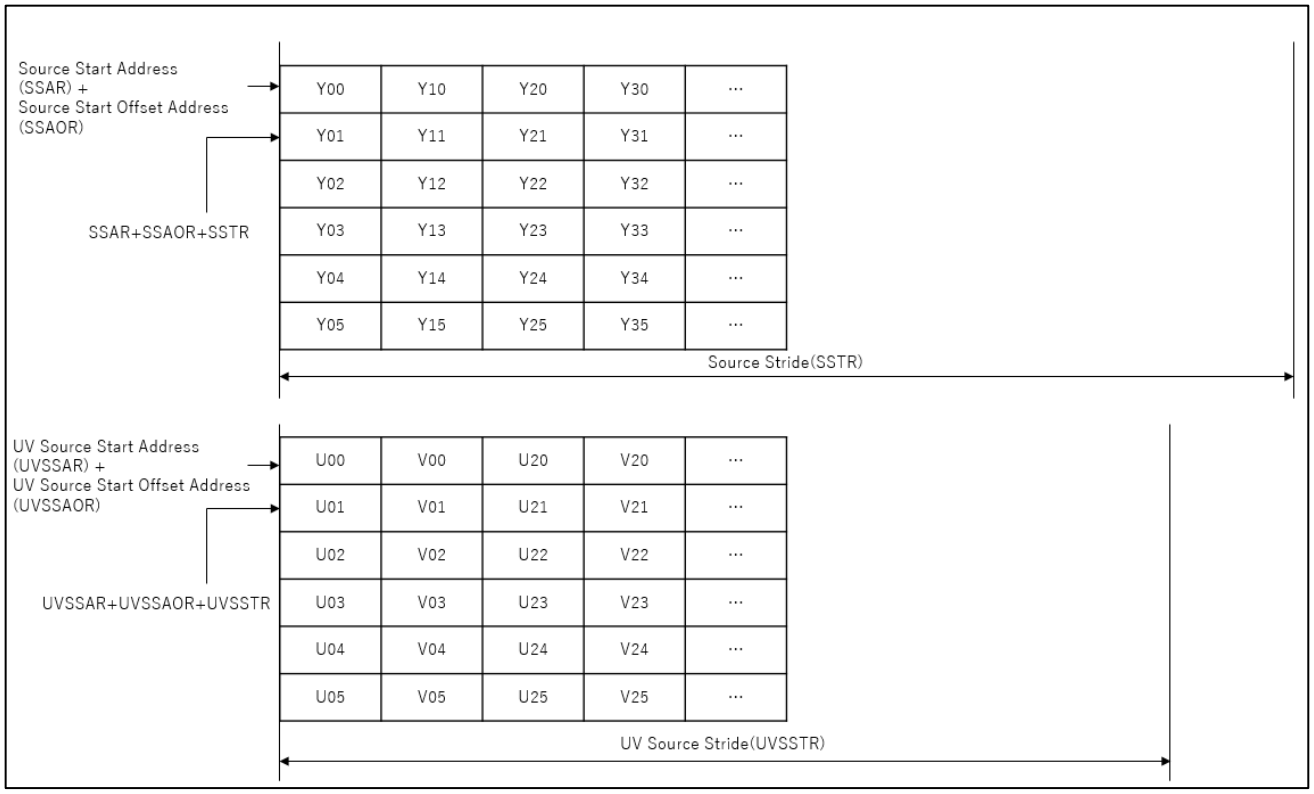



**Figure 56.16 Allocation of YUV Data to Memory (YUV422 Semi-planar, Input)**

The figure above shows the memory allocation of the input in YUV422 Semi-planar format. The start address from which the input data is read is obtained by adding SSAR and SSAOR for Y plane, and is obtained by adding UVSSAR and UVSSAOR for UV plane. The memory stride is set using SSTR for Y plane and UVSSTR for UV plane.

To change the order of the input U and V, set UVFORM of CMRCR2 to 1.

**(2) Output data**

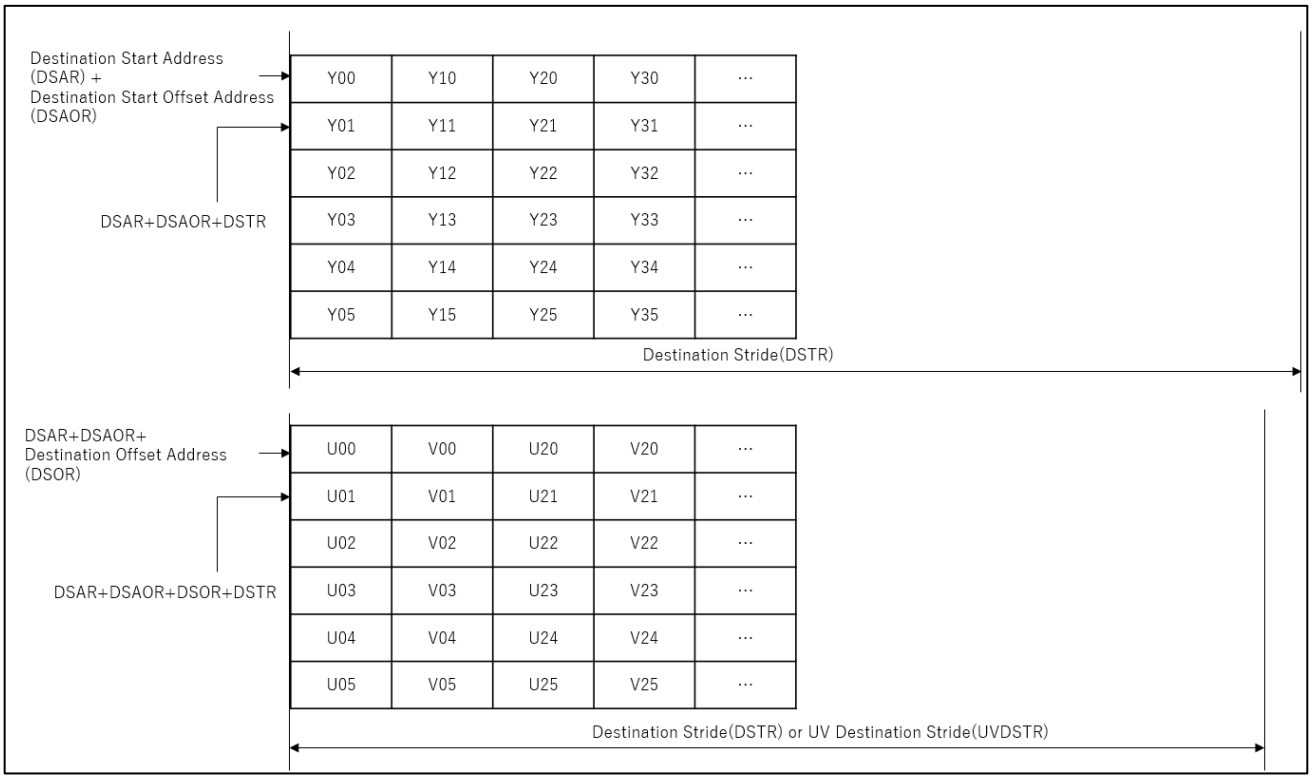



**Figure 56.17 Allocation of YUV Data to Memory (YUV422 Semi-planar, Output)**

The figure above shows the memory allocation of the output in YUV422 Semi-planar format. The start address to which output data is written is obtained by adding DSAR and DSAOR for Y plane, and is obtained by adding DSAR, DSAOR and DSOR for UV plane. The memory stride is set using DSTR for Y plane. The UV plane is set using DSTR when UVDSTSEL of DCCR is 0, and using UVDSTR when UVDSTSEL is 1.

The order of the output U and V cannot be changed. The order of the output follows the order changed according to the setting of UVFORM of CMRCR2.

### 56.3.7.3 YUV420 Semi-planar format

In the YUV420 Semi-planar format, the YUV data has the Y plane and the UV plane placed separately in the memory. The UV plane is placed in the memory in the order of U data, V data, U data, and V data. The vertical direction on the screen follows the memory stride used to handle memory data in two-dimension state. The pixels one line below on the screen are placed at the stride-added address. The Y/UV plane is handled in 8-bit or 16-bit unit. Besides, the UV plane is halved in the vertical direction compared to the YUV422 Semi-planar format.

For the YUV420 Semi-planar format for input or output, set the Y plane bit depth (bpp) greater than or equal to the UV plane bit depth. Otherwise, the operation cannot be guaranteed.

Create the vertex data on DL based on Y plane for both the input and output data.

### (1) Input data

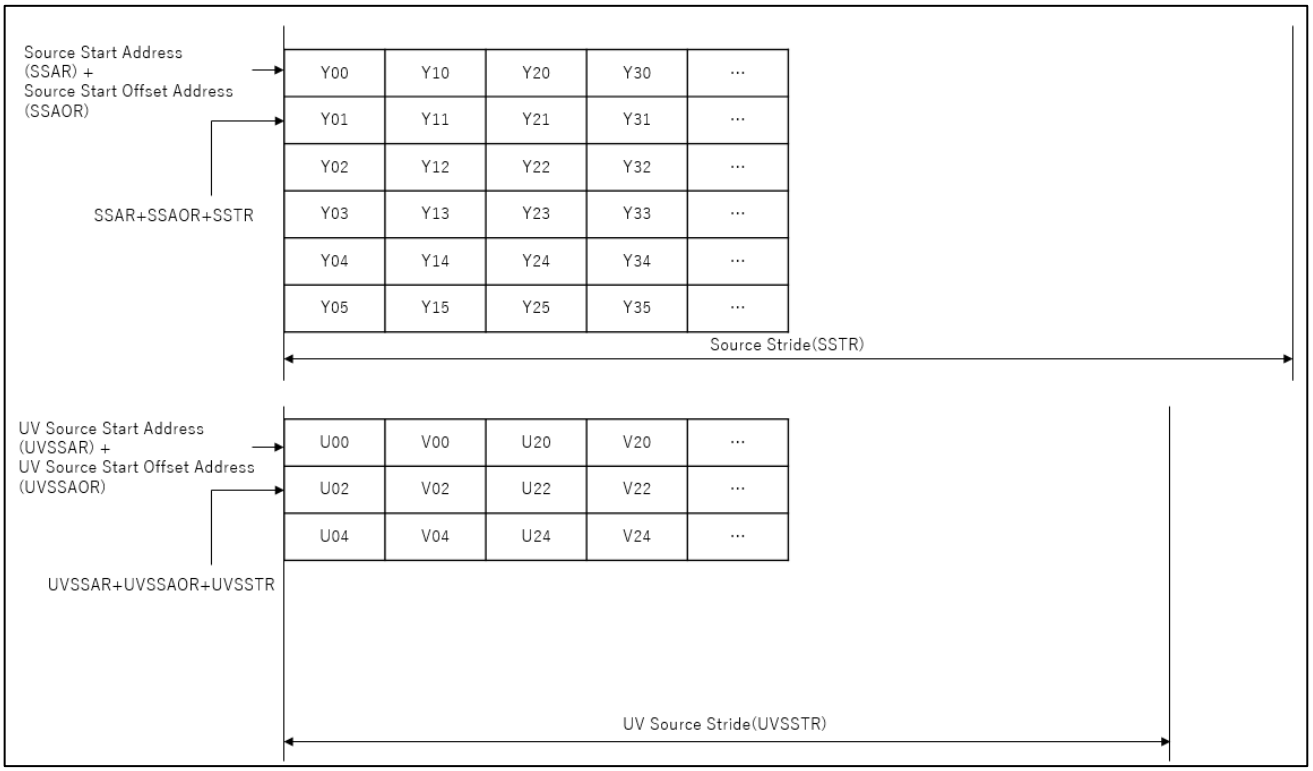



**Figure 56.18 Allocation of YUV Data to Memory (YUV420 Semi-planar, Input)**

The figure above shows the memory allocation of the input in YUV420 Semi-planar format. The start address from which the input data is read is obtained by adding SSAR and SSAOR for Y plane, and is obtained by adding UVSSAR and UVSSAOR for UV plane. The memory stride is set using SSTR for Y plane and UVSSTR for UV plane.

To change the order of the input U and V, set UVFORM of CMRCR2 to 1.

**(2) Output data**

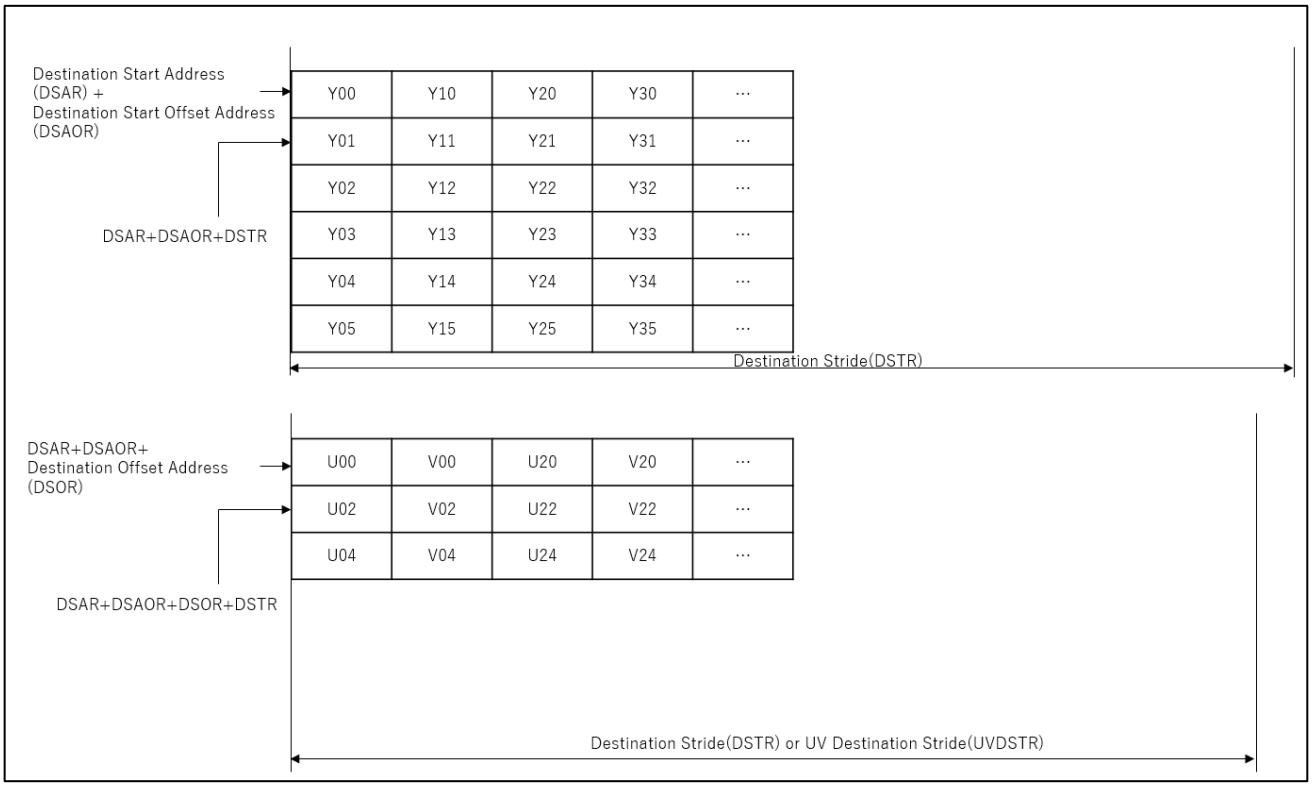



**Figure 56.19 Allocation of YUV Data to Memory (YUV420 Semi-planar, Output)**

The figure above shows the memory allocation of the output in YUV420 Semi-planar format. The start address to which output data is written is obtained by adding DSAR and DSAOR for Y plane, and is obtained by adding DSAR, DSAOR and DSOR for UV plane. The memory stride is set using DSTR for Y plane. The UV plane is set using DSTR when UVDSTSEL of DCCR is 0, and using UVDSTR when UVDSTSEL is 1.

The order of the output U and V cannot be changed. The order of the output follows the order changed according to the setting of UVFORM of CMRCR2.

### 56.3.7.4 Y Plane Only format

For the Y Plane Only format, only Y data is processed while UV data is ignored. The vertical direction on the screen follows the memory stride used to handle memory data in two-dimension state. The pixels one line below on the screen are placed at the stride-added address. The data is handled in 8-bit or 16-bit unit.

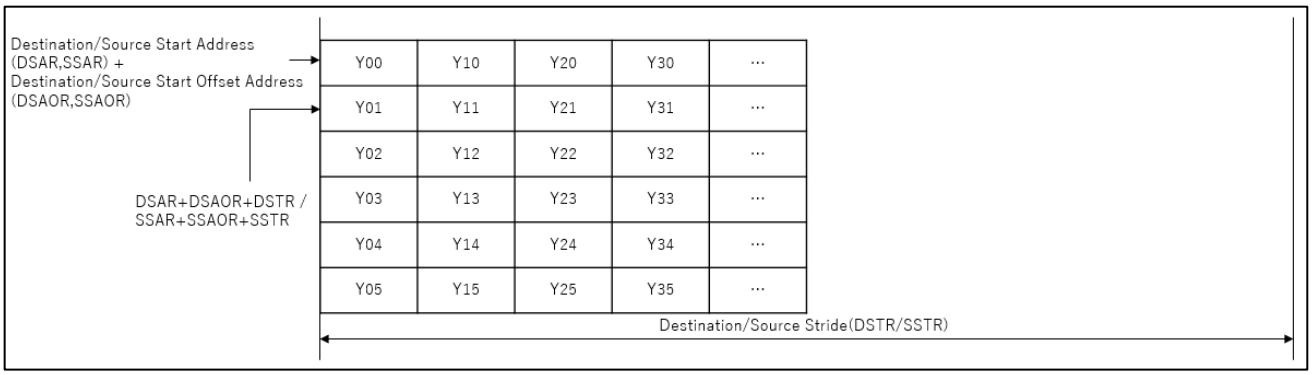



**Figure 56.20 Allocation of YUV Data to Memory (Y Plane Only)**

The figure above shows the memory allocation of the output in Y Plane Only format. The start address from which input data is read is obtained by adding SSAR and SSAOR. Also, the start address to which output data is written is obtained by adding DSAR and DSAOR. Memory stride is set using SSTR for input data and DSTR for output data.

### 56.3.7.5 UV Plane Only format

For the UV Plane Only format, only UV data is processed while Y data is ignored. The vertical direction on the screen follows the memory stride used to handle memory data in two-dimension state. The pixels one line below on the screen are placed at the stride-added address. The data is handled in 8-bit or 16-bit unit.

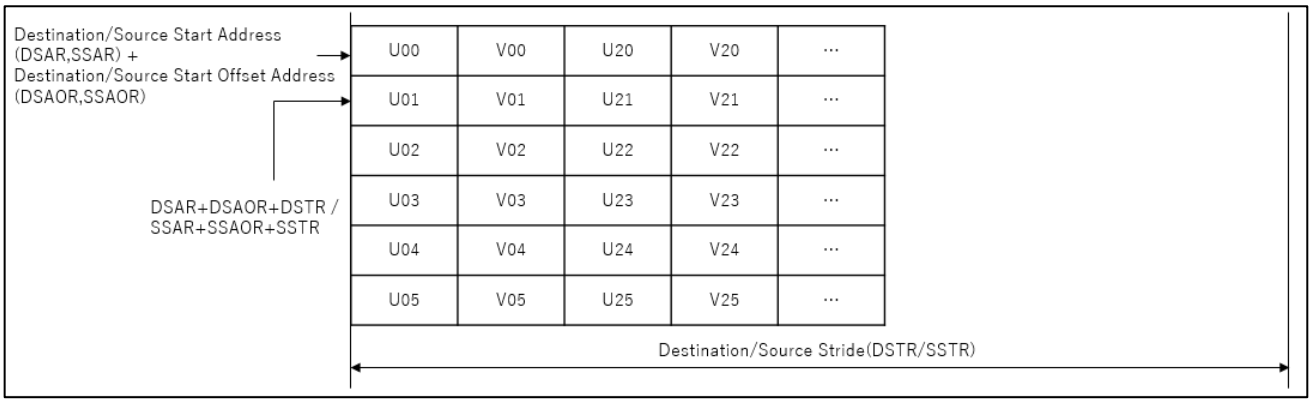



**Figure 56.21 Allocation of YUV Data to Memory (UV Plane Only)**

The figure above shows the memory allocation of the output in UV Plane Only format. The start address from which input data is read is obtained by adding SSAR and SSAOR. Also, the start address to which output data is written is obtained by adding DSAR and DSAOR. Memory stride is set using SSTR for input data and DSTR for output data.

### 56.3.7.6 Tile Addressing Mode

This module supports not only linear addressing where the two-dimensional image data are arranged like raster in memory but also a tile addressing mode where the data are arranged like tile.

In tile addressing mode, continuous memory addresses are interleaved with data for multiple Y coordinates, thus improving the cache-hit ratio for vertical access to the image data (e.g., for rotation of an image) and also improving processing performance.

In tile addressing mode, the data are stored with the Y coordinate incremented in 16-byte units within 64-byte sections. An example where 8-bpp Y data are exclusively output is shown below. (x, y) in the figure below indicate the x and y coordinates of pixels.

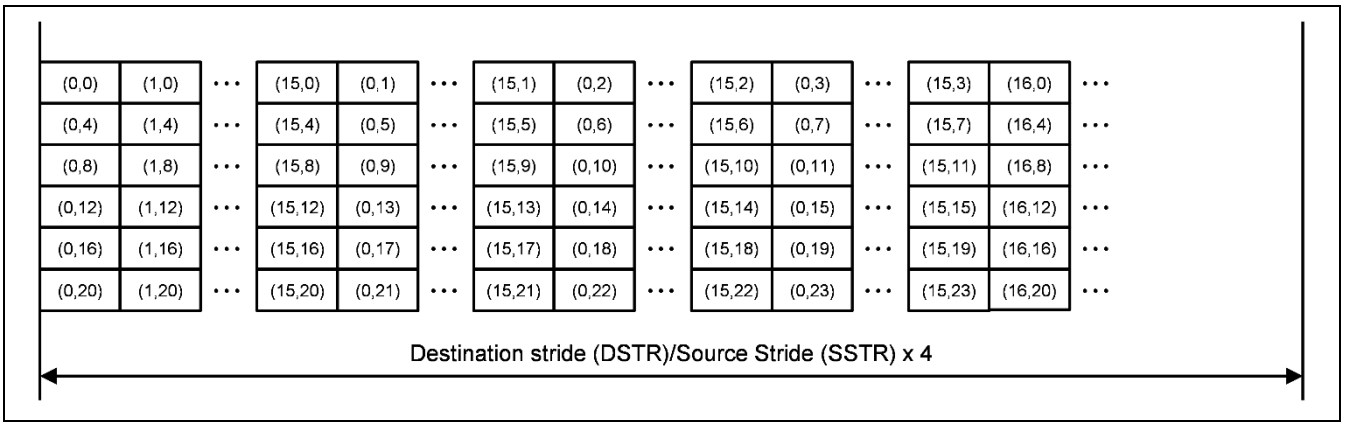



**Figure 56.22 Data in Memory in Tile addressing mode (Y Exclusive, 8 bpp)**

The tile addressing mode can be set up individually for source and destination images. To set up the tile addressing mode for the source image, set the TCTE bit in CMRCR2 to 1. To set up the tile addressing mode for the destination image, set the DCTE bit in CMRCR2 to 1.

To set the stride in tile addressing mode, set the corresponding bits in DSTR and SSTR, SSTR, SSTRn, UVSSTR, and UVSSTRn to a value equal to one quarter of the actual memory stride.

#### 56.3.7.7 How to use the address offset register

This module includes offset address register for each start address register. As explained in section 56.3.7, the start address is obtained by summing the setting of each start address register and the offset address register.

These offset address registers can be used, for example, when dividing a screen and when using one DL in different memory spaces.

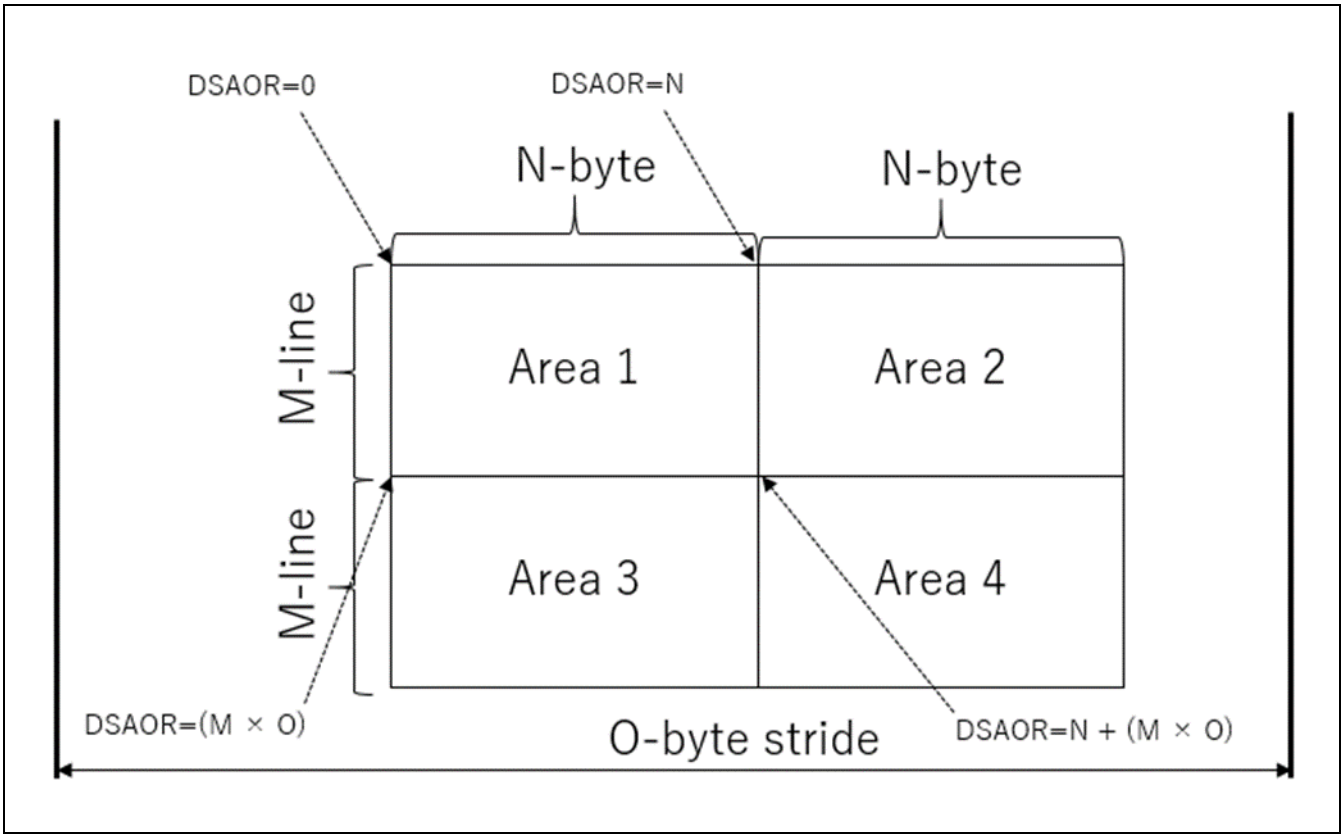



**Figure 56.23 Example of offset address register utilization**

Figure 56.23 shows an example of dividing the screen into four parts and rendering it. The width of each divided area is N bytes, the height is M lines, and the stride of the whole screen is O bytes.

When rendering these four areas using one DL, it is necessary to rewrite the destination start address register three or four times inside the DL. If the memory allocation area for this screen rendering is different from when the DL is initially created, it is necessary to rewrite and update all the destination start address registers on the DL.When using the offset address register, describe it in DL so that the offset address register of Area 1 in this figure is 0, Area 2 is N, Area 3 is MxO, Area 4 is N+(MxO). After the memory allocation area is determined, set this address to the destination start address register via the register access from the CPU. By doing so, it is possible to update the address of all the divided areas using one register setting.

For the Rotator & Scaler Extension function which will be described later, since the screen dividing process is required, it is possible to make better use of the offset address register.

### 56.3.8 Luminance/Hue Correction

After texture mapping using the TRI instruction, luminance and hue correction can be performed on the image data by using scale and offset values. This makes it possible to correct for factors such as differences in luminance at the center and periphery of the camera lens or differences in the hue characteristics of multiple cameras.

Luminance correction is enabled when the LUCE bit in the CMRCR register is set to 1. The scale value is set either in a register (the YLCPR register), when the LUSM bit in the CMRCR register is set to 1, or in the DL for each vertex, when the LUSM bit is cleared to 0. The offset is set either in a register (the YLCPR register), when the LUOM bit in the CMRCR register is set to 1, or in the DL for each vertex, when the LUOM bit is cleared to 0. (For information on specifying values in the DL, see section 56.5, Display List.) Scale values should be specified as unsigned, and offset values as signed, 8-bit data.

The luminance correction is calculated according to the following equation, with Y designating the luminance value after texture mapping and Y' the luminance value after luminance correction.

$$Y' = ((Y \times lscal) >> YLDPO) + lofst$$

Here YLDPO is the value specified by the YLDPO field in the CPDPOR register. For each pixel, lscal and lofst are calculated as scale and offset values for each vertex of a triangle and mapped to the triangle in the same manner as texture coordinates.

If the value calculated for Y' is less than the value specified in the YLMINR register, Y' is clamped to the YLMINR value. Conversely, if the value calculated for Y' is greater than the value specified in the YLMAXR register, Y' is clamped to the YLMAXR value.

When the scale or offset values are specified in registers, the register values are applied uniformly to all vertexes. Scale and offset values may not be omitted from the DL if they are specified in registers. In that case, make sure to insert dummy values in the DL. If scale and offset values are specified in registers and setting in TRIMR.DIDL bit is 1, do not need to insert dummy parameters in DL.

Hue correction is enabled when the CLCE bit in the CMRCR register is set to 1. The scale values are set either in registers (the UBCPR and VRCPR registers), when the CLSM bit in the CMRCR register is set to 1, or in the DL for each vertex, when the CLSM bit is cleared to 0. The offset values for each vertex are set either in registers (the UBCPR and VRCPR registers), when the CLOM bit in the CMRCR register is set to 1, or in the DL for each vertex, when the CLOM bit is cleared to 0. (For information on specifying values in the DL, see section 56.5, Display List.) Scale vales should be specified as unsigned, and offset values as signed, 8-bit data.

The hue correction is calculated according to the following equation, with U and V designating the hue values after texture mapping and U' and V' the hue values after hue correction.

$$U' = ((U + ubofs) \times ubscl) >> UBDPO$$

$$V' = ((V + vrofs) \times vrscl) >> VRDPO$$

Here UBDPO and VRDPO are the values specified by the UBDPO and VRDPO fields in the CPDPOR register. For each pixel, ubscl, ubofs, vrscl, and vrofs are calculated as scale and offset values for each vertex of a triangle and mapped to the triangle in the same manner as texture coordinates.

If the value calculated for U' is less than the value specified in the UBMINR register, U' is clamped to the UBMINR value. Conversely, if the value calculated for U' is greater than the value specified in the UBMAXR register, U' is clamped to the UBMAXR value. In like manner, V' is clamped to VRMINR if it is less than VRMINR and to VRMAXR if it is greater than VRMAXR.

When the scale or offset values are specified in registers, the register values are applied uniformly to all vertexes. Scale and offset values may not be omitted from the DL if they are specified in registers. In that case, make sure to insert dummy values in the DL. If scale and offset values are specified in registers and setting in TRIMR.DIDL bit is 1, do not need to insert dummy value in DL.

### 56.3.9 Luminance Correction Using Lookup Table (LUT)

By using the LUT, luminance can be converted from 10 bpp or 12 bpp to 8 bpp.

In 12 to 8 bpp conversion, the upper 10 bits are converted to 8 bits using the LUT.

Luminance conversion using the LUT is carried out after bilinear processing, and is followed by luminance correction. This function is enabled when the LUTE bit in the CMRCR2 register is 1.

Before using this function, set 8-bit data corresponding to luminance to the LUTDR registers (0 to 1,023) at addresses H'1000 to H'1FFC.

The LUTDRn registers can be set from the CPU and DL. When using the DL to set data, use the WTL2 instruction.

When using the LUT to set data, output data should be 8 bits; set the Y12 and Y10 bits in the CMRCR register to 0.

Note that Conversion by the LUT is not support for Extended Filtering Pipeline and DYP mode.

### 56.3.10 Extended Filtering Pipeline

The IMR-LX6 incorporated in this product has a functionally extended filtering pipeline in addition to the filtering pipeline in previous products.

#### (1) Overview

The table below shows the differences between the previous and extended pipelines.

**Table 56.18 Comparison of Functions between the Previous and Extended Filtering Pipelines**

| | Previous Pipeline | Extended Pipeline |
|---|---|---|
| Bilinear filtering | √ | √ |
| Filtering with the use of pyramidal images | √ | √ |
| Luminance and hue correction | √ | √ |
| Filtering in the 8-bpp, 10-bpp, or 12-bpp format | √ | √ |
| Filtering in the 14-bpp or 16-bpp format | — | √ |
| Selecting whether to retain the fractional part following filtering operations | — | √ |
| Rounding off the results of filtering operations | — | √ |
| Logical right-shifting of the value following processing for filtering | — | √ |
| Conversion by the LUT | √ | — |

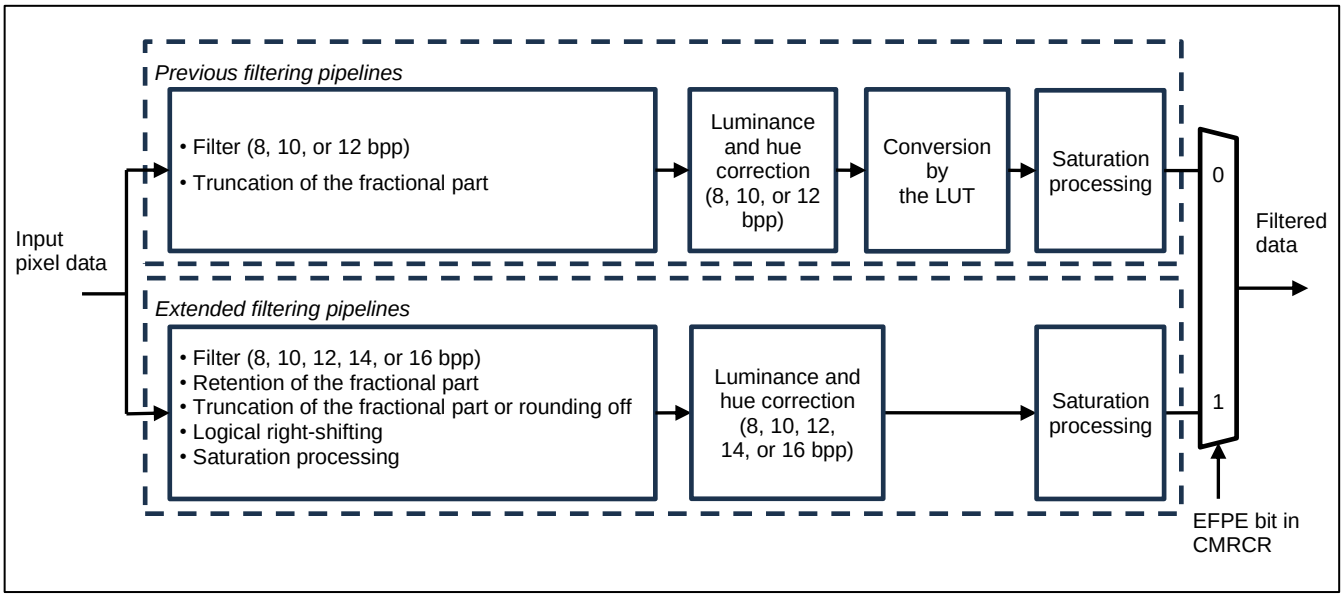



**Figure 56.24 Configurations of the Previous and Extended Filtering Pipelines**

Figure 56.24 is a block diagram of the previous and extended filtering pipelines. Switching between the two pipelines is possible by using the EFPE bit in CMRCR. The two pipelines have the same register functions. However, they may output different data for the same input data even though settings such as those for pixel format are the same because of the differences in functionality between the pipelines.

The following explains the functions of the extended filtering pipeline.

### (2) Filtering of data in the 14-bpp or 16-bpp format

The extended filtering pipeline handles data in the 14-bpp or 16-bpp format.

When handling data in the 14-bpp or 16-bpp format, set bits SY1416, DY1416, SUV1416, and DUV1416 in CMRCR. In such cases, set bits SY12, SY10, Y12, Y10, SUV, and DUV in CMRCR to 0.

When handling data in the YUV422 interleaved mode, specify the same luminance and hue for the pixel format.

### (3) Retaining and rounding the fractional parts following extended filtering

The filtering pipeline from the previous products truncates the fractional part and only retains the integer part following filtering operations. The extended filtering pipeline can output data with the specified number of fractional digits being retained after filtering operations.

Refer to the Table 56.13 for possible combinations of SHFE, SHFVAL, UVSHFE and UVSHFVAL in TRIMR.

The below part will describe about the Y plane. For the UV plane, if the image format of external memory is YUV422 Semi-planar or YUV420 Semi-planar, read SHFE as UVSHFE and SHFVAL as UVSHFVAL.

Additionally, while the filtering pipeline from the previous products simply truncates the fractional part following filtering, how to round the fractional part in the results of filtering is specifiable for the extended filtering pipeline as truncation or as rounding up and down. Selecting rounding might reduce the errors in the mathematical operations of filtering.

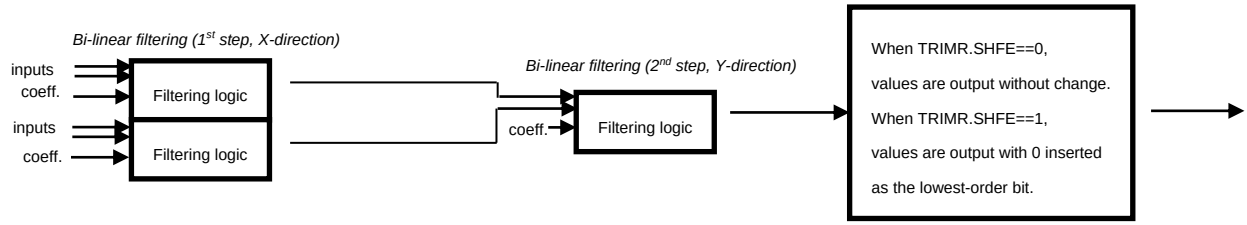



**Figure 56.25 Configuration of the Filtering Circuit (when Bilinear Filtering is Enabled)**

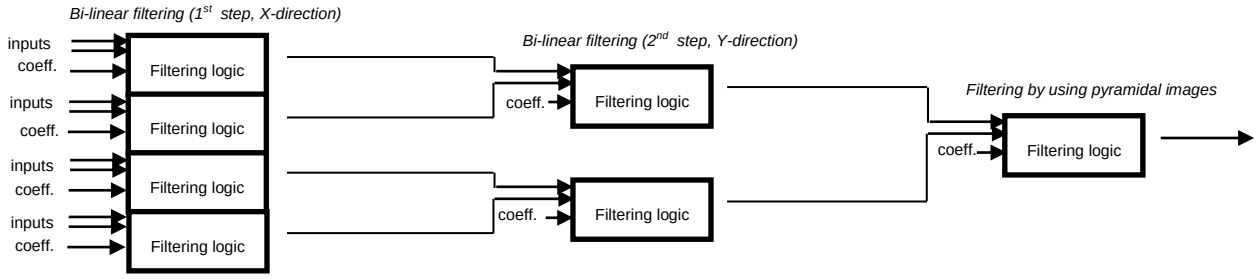



**Figure 56.26 Configuration of the Filtering Circuit (when Filtering by using pyramidal images is Enabled)**

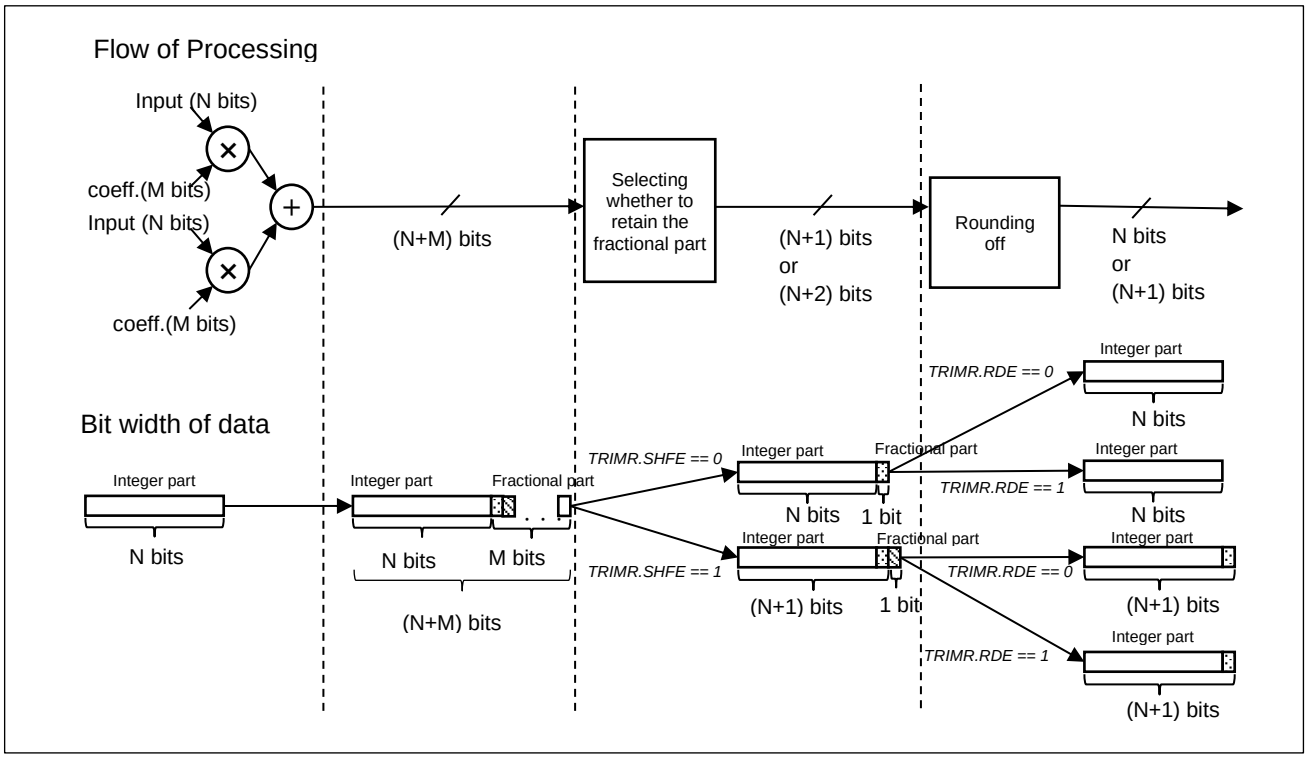



**Figure 56.27 Details of the Filtering Circuit**

Figure 56.25 and Figure 56.26 show the configuration of the filtering circuit in the extended filtering pipeline. Figure 56.27 shows the details of the filtering circuit.

Each of the filtering circuits first multiplies and accumulates the input values and coefficients. Assume the input values are N bits (positive integer) and the coefficient is M bits (a value between 0.0 and 1.0), the output value from the multiplier-accumulator is (N + M) bits. The higher-order N bits are the integer part and the lower-order M bits are the fractional part.

The output from the multiplier-accumulator is then input to the block for selecting whether to retain the fractional part. When the setting is for truncating the fractional part (the SHFE bit in TRIMR is set to 0), the block outputs an (N + 1)-bit value, where the higher-order N bits are the integer and the highest-order bit of the lower-order M bits is the fractional part. When the setting is for retaining some of the digits of the fractional part (the SHFE bit in TRIMR is set to 1), the given block outputs an (N+2)-bit value, where the higher-order (N + 1) bits are the integer part, and the highest-order bit of the lower-order (M - 1) bits is the fractional part.

The output from the block for selecting whether to retain the fractional part is then input to the block for rounding off. When the setting is for truncation (the RDE bit in TRIMR is 0), the block for rounding off outputs a truncated value, where the bits handled as the fractional part*1 in the previous block are discarded. When the setting is for rounding up or down (the RDE bit in TRIMR is 1), the given block outputs the value after the bit handled as the fractional part in the previous block is added to the integer part.*2

Note: 1. The lowest-order bit of the input to the block for selecting whether to retain the fractional part

2. Figure 56.27 does not show a difference between the output values when TRIMR.RDE is 1 and when it is 0. However, since the output value when TRIMR.RDE is 1 may have been rounded up, the value may be different from that when TRIMR.RDE is 0.

When the setting is for some of the digits of the fractional part to be retained (the SHFE bit in TRIMR is 1), each of the filtering circuits outputs a value with one more bit than the input value. That is, if the input pixel

format before filtering is N bpp, data are output in the (N+3) bpp format after all three filtering operations. Accordingly, the input and output pixel formats, and the number of digits of logical right-shifting must be set in accord with the combination shown in Table 56.19 or Table 56.20.

**Table 56.19 Settings for Bilinear Filtering**

| Input Pixel Format*1 | Output Pixel Format*2 | Number of Digits of Logical Right-Shifting*3 |
|---|---|---|
| 8 bpp | 10 bpp | 1 |
| 10 bpp | 12 bpp | 1 |
| 12 bpp | 14 bpp | 1 |
| 14 bpp | 16 bpp | 1 |

**Table 56.20 Setting for Filtering with the Use of Pyramidal Images**

| Input Pixel Format*1 | Output Pixel Format*2 | Number of Digits of Logical Right-Shifting*3 |
|---|---|---|
| 8 bpp | 10 bpp | 1 |
| 8 bpp | 12 bpp (11 bits)*4 | 0 |
| 10 bpp | 12 bpp | 1 |
| 10 bpp | 14 bpp (13 bits)*4 | 0 |
| 12 bpp | 14 bpp | 1 |
| 12 bpp | 16 bpp (15 bits)*4 | 0 |
| 14 bpp | 16 bpp | 1 |

Notes: 1. The input pixel format can be set by using bits SUV1416, SY1416, SY12, SY10, and SUV in the CMRCR register.
2. The output pixel format can be set by using bits DUV1416, DY1416, Y12, Y10, and DUV in the CMRCR register.
3. The number of the digits of logical right-shifting can be set by using the SHFVAL bit in the TRIMR register.
4. The Output Pixel Formats need to be set as 12 bpp, 14 bpp or 16 bpp, however actual output image data are (N+3) bpp.

**(4) Logical right-shifting following filtering**

The extended filtering pipeline can apply logical right-shifting on the data following filtering.

The saturation processing applied according to the pixel format of the destination data is handled after the logical right shifting. Accordingly, the saturation processing can be used to limit the data following filtering to be within a specified range.

Use the SHFVAL field in the SHFVAL register to specify the number of bits of logical right-shifting. The parts of the value shifted out of the range by logical right-shifting are always truncated.

### 56.3.11 Synchronized system between modules

This module has functionality for synchronizing operations with other image processing modules.

**(1) Overview**

Instructions for inter-module synchronization (WUP and SLP) are available for use in the display list. A module executing one of the inter-module synchronization signals (SLP) causes it to wait for the other inter-module synchronization signal (WUP) from another module. These instructions can be executed in desired positions in the display lists.

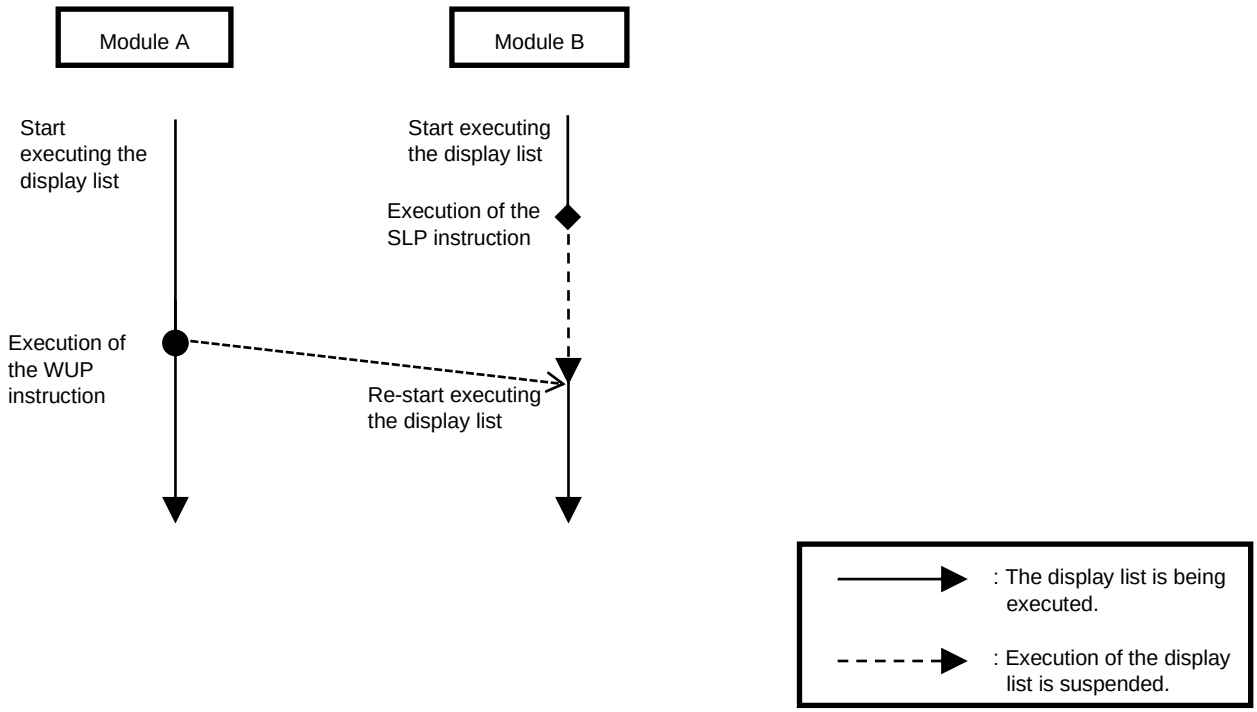



**Figure 56.28 Example of One Module Synchronizing its Operation with Another**

Figure 56.28 is an example of one module synchronizing its operation with another. In this figure, module B suspends execution of its display list in response to execution of the SLP instruction, and re-starts execution of the subsequent instructions in the display list once it receives the signal produced by module A executing the WUP instruction at the desired position.

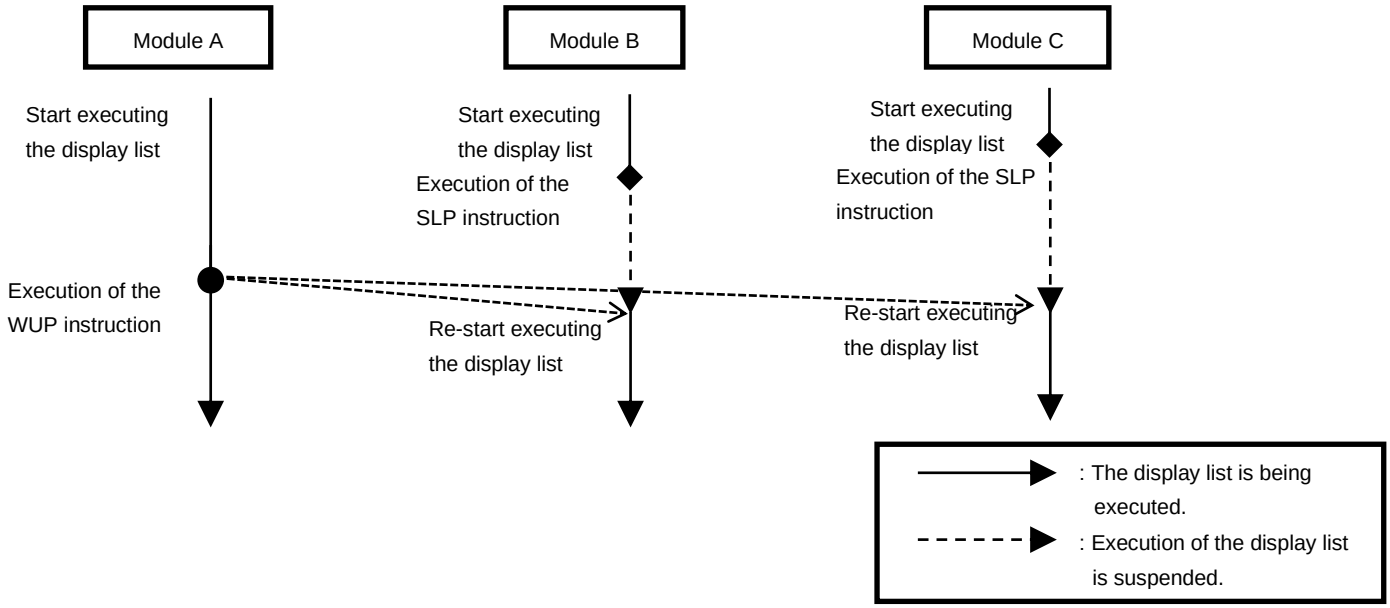



**Figure 56.29 Example of One Module Synchronizing its Operation with Two Others**

Synchronization between modules is also possible with more than two modules. In Figure 56.29, modules B and C suspend execution of their display lists to wait for execution of the WUP instruction in a desired position in the display list of module A.

In Figure 56.28 and Figure 56.29, the SLP instruction is executed before the WUP instruction, however, there is no particular order of instructions, because the synchronization signal generated by the WUP instruction is held pending in the modules receiving the signal. Also, the signal can be held pending by each of the modules receiving synchronization signals. Therefore, the order of instructions can be reversed for some modules when synchronizing the operations of more than two modules. *

When the module receives another synchronization signal while one is already held pending, a synchronization error is generated. Refer to 56.3.11 (4) Detection of Synchronization Errors for more details.

If a specified module to be synchronized has not been activated, wait till the module is activated and then transmit the WUP signal to it. Note, however, that if the IMR-LX6 receives the WUP signal from the ISP before the IMR-LX6 is activated, a synchronization signal reception error is detected. For details on this type of error, refer to 56.3.11 (4), Detection of Synchronization Errors.

Notes: When a synchronization signal is held pending in the module, executing the TRAP instruction without executing the SLP instruction is prohibited.

### (2) Possible Combinations of Modules for Synchronization of Operations

Table 56.21 shows the possible combinations of modules for synchronization with the IMR-LX6 modules. √ indicates that synchronization between the receiving and transmitting modules is possible.

Synchronizing operations with modules not in the table below is not possible.

**Table 56.21 Possible Combinations of Modules for Synchronized Operations**

| Receiving module / Transmitting module | IMR-LX6 ch0 | IMR-LX6 ch1 | IMR-LX6 ch2 | IMR-LX6 ch3 | IMP DMAC0 (thread0) | IMP DMAC0 (thread1) | IMP DMAC1 (thread0) | IMP DMAC1 (thread1) | Slim-IMP DMAC0 (thread0) | Slim-IMP DMAC0 (thread1) | Slim-IMP DMAC1 (thread0) | Slim-IMP DMAC1 (thread1) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMR-LX6 ch0 | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| IMR-LX6 ch1 | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| IMR-LX6 ch2 | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| IMR-LX6 ch3 | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ |
| IMP DMAC0 (thread0) | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ |
| IMP DMAC0 (thread1) | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ |
| IMP DMAC1 (thread0) | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ |
| IMP DMAC1 (thread1) | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ |
| Slim-IMP DMAC0 (thread0) | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ |
| Slim-IMP DMAC0 (thread1) | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ |
| Slim-IMP DMAC1 (thread0) | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ |
| Slim-IMP DMAC1 (thread1) | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — |
| ISP | √ | √ | √ | √ | — | — | — | — | — | — | — | — |
| VIN[0].ch[n] | VIN[0].ch[3] | VIN[0].ch[7] | VIN[0].ch[0] /ch[4] | VIN[0].ch[1] /ch[5] | — | — | — | — | — | — | — | — |
| VIN[1].ch[n] | VIN[1].ch[3] | VIN[1].ch[7] | VIN[1].ch[0] /ch[4] | VIN[1].ch[1] /ch[5] | — | — | — | — | — | — | — | — |

### (3) Setting the Synchronizing Operations

The modules with operations to be synchronized with the IMR-LX6 can be specified by setting the fields of the synchronization control registers (SYNCCRn), and using the WUP and SLP instructions.

Before execution of the WUP and SLP instructions, the numbers corresponding to the modules for synchronization of operations must be set in the SYNCCm (m = 0 to 15) fields in SYNCCRn. The correspondence between the modules and numbers for selecting the modules is shown in Table 56.22. The Number for Specifying the module from IMP DMAC and Slim-IMP DMAC to IMR-LX6 is different than IMR-LX6. Refer to the U30.IMP DMAC HWM for details.

**Table 56.22 Correspondence between the Modules and Numbers for Specifying the Modules**

| Module | Number for Specifying the Module |
|---|---|
| IMP DMAC0 (thread0) | H'10 |
| IMP DMAC0 (thread1) | H'11 |
| IMP DMAC1 (thread0) | H'12 |
| IMP DMAC1 (thread1) | H'13 |
| Slim-IMP DMAC0 (thread0) | H'14 |
| Slim-IMP DMAC0 (thread1) | H'15 |
| Slim-IMP DMAC1 (thread0) | H'16 |
| Slim-IMP DMAC1 (thread1) | H'17 |
| IMR-LX6 ch0 | H'40 |
| IMR-LX6 ch1 | H'41 |
| IMR-LX6 ch2 | H'42 |
| IMR-LX6 ch3 | H'43 |
| ISP | H'50 |
| VIN[0].ch[0] | H'60 |
| VIN[0].ch[1] | H'61 |
| VIN[0].ch[2] | H'62 |
| VIN[0].ch[3] | H'63 |
| VIN[0].ch[4] | H'64 |
| VIN[0].ch[5] | H'65 |
| VIN[0].ch[6] | H'66 |
| VIN[0].ch[7] | H'67 |
| VIN[1].ch[0] | H'68 |
| VIN[1].ch[1] | H'69 |
| VIN[1].ch[2] | H'6A |
| VIN[1].ch[3] | H'6B |
| VIN[1].ch[4] | H'6C |
| VIN[1].ch[5] | H'6D |
| VIN[1].ch[6] | H'6E |
| VIN[1].ch[7] | H'6F |

After the number corresponding to the module for synchronizing operations is set in the SYNCCm field of SYNCCRn, set the corresponding bit in the SYNCC enable field of the WUP or SLP instruction. For example, when the module for synchronizing operations is set in the SYNCC0 field, set bit 0 of the SYNCC enable field to 1. When the module for synchronizing operations is set in the SYNCC14 field, set bit 14 of the SYNCC enable field to 1.

Executing the WUP or SLP instruction enables the module specified in the SYNCCm field and SYNCC enable field to transmit the WUP signal or to suspend its operation until it receives the WUP signal, respectively.

Setting more than one bit in the SYNCC enable field to 1 at the same time is also possible. In such cases, the synchronization signals are transmitted to or received from the respective modules simultaneously. *

Notes: It is prohibited to execute WUP or SLP instruction with two or more same modules specified for synchronization. For example, the following settings are prohibited.
The SYNCC enable field of the WUP or SLP instruction is B'0000 0000 0000 1001.
The syncc0 field in SYNCCR0 is H'10. (IMP DMAC0/thread0 is selected)
The syncc3 field in SYNCCR0 is H'10. (IMP DMAC0/thread0 is selected)

**(4) Detection of Synchronization Errors**

Detection of an overflow of the synchronization signal and the reception of a synchronization signal while the given module is not executing a display list is possible. The detected error can be indicated in the respective bit in SR, and an indicator can be transmitted to the CPU. Operations cannot be guaranteed if a synchronization error is generated.

**(a) Overflow Errors of the WUP Signal**

The WUP signal generated by executing the WUP instruction is held pending in the receiving module to wait for the execution of the SLP instruction. When another WUP signal is received while one is already held pending, an overflow error is detected.

The number of signals that can be held pending in this product is one.

If an overflow is detected when the setting of the WOVFENB bit in ICR is 1, the WOVF bit in SR is set to 1. If the setting of the WOVFM bit in IMR is 0, an interrupt signal is transmitted to the CPU in response to setting of the WOVF bit in SR to 1.

This error is only used for synchronization between the ISP or VIN and this module. Avoid generating errors in the synchronization of operations between the IMR-X6 and modules other than the ISP or VIN by not executing the WUP instruction two or more consecutive times for this module. Do not set the same module in the different SYNCC enable fields.

**(b) Synchronization Signal Reception Errors while the IMR-LX6 is not Executing a Display List**

When the IMR-LX6 receives a synchronization signal from the ISP or VIN while it is not executing a display list and the setting of the WERRENB bit in ICR is 1, the WERR bit in SR is set to 1.

If the setting of the WERRM bit in IMR is 0, an interrupt signal is transmitted to the CPU in response to setting of the WERR bit in SR to 1.

This error is only used for synchronization between the ISP or VIN and this module, and is not detected in synchronization between the IMR-X6 and modules other than the ISP or VIN.

In the case that VIN WUP is used by another IP (not used by IMR-X6), WERRENB shall be set to 0 before VIN send the WUP to IMR. Otherwise, IMR may assert the pseudo-WERR. Because, if VIN sends WUP to VSPX or sends WUP interrupts to the CPU, IMR also receives these WUP although IMR does not use them. Therefore, users need to disable the pseudo-WERR.

### 56.3.12 Filtering for Improved Image Quality in Scaling Images Down by Using Pyramidal Images

When texture mapping is executed, if the destination image is smaller than the source image (i.e. the latter is being scaled-down), using bilinear filtering for texture mapping may lead to deterioration of the image quality. Meanwhile, the image quality in scaling images down can be improved by selecting a texture in accord with the scale-down rate from among the pyramidal images, which are a group of multiple reduced images that are usable as textures, and using these for texture mapping.

#### 56.3.12.1 Filtering by Using Pyramidal Images

An example of scaling down a source image composed of pixels No. 1 to 12 vertically and horizontally into half the original size is given below. To simplify the description, bilinear filtering is off in this example, but even if bilinear filtering is on, the image quality will deteriorate similarly.

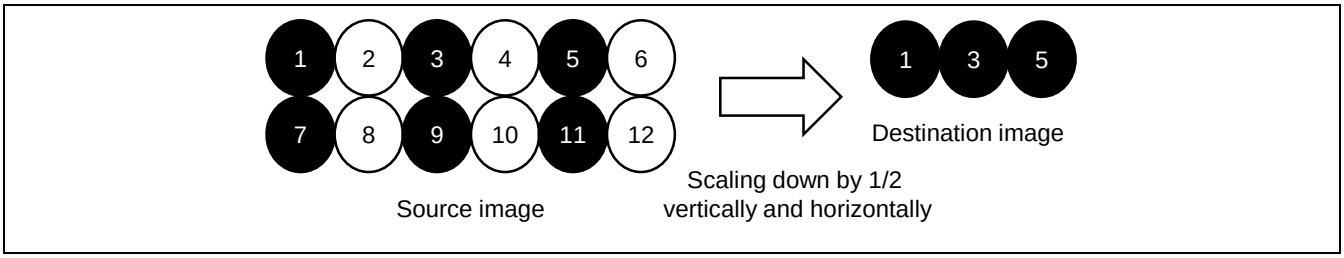



**Figure 56.30 Deterioration of Image Quality in Scaling Down**

In this example, the reduced image has lost all pixels except for pixels No. 1, 3, and 5. As a result, the original image having continuous black and white vertical lines becomes an image which is entirely black, and the image is thus changed to a different shade and shape.

This module incorporates a function for suppressing such deterioration of image quality by using pyramidal images. A pyramidal image is created by recursively scaling down the original image by 1/2 both vertically and horizontally. In scaling down, the four adjacent pixels in an original image are equally averaged into the pixel in the subsequent image. The Nth image to be created, here, is referred to as map n. This module supports up to map 4.

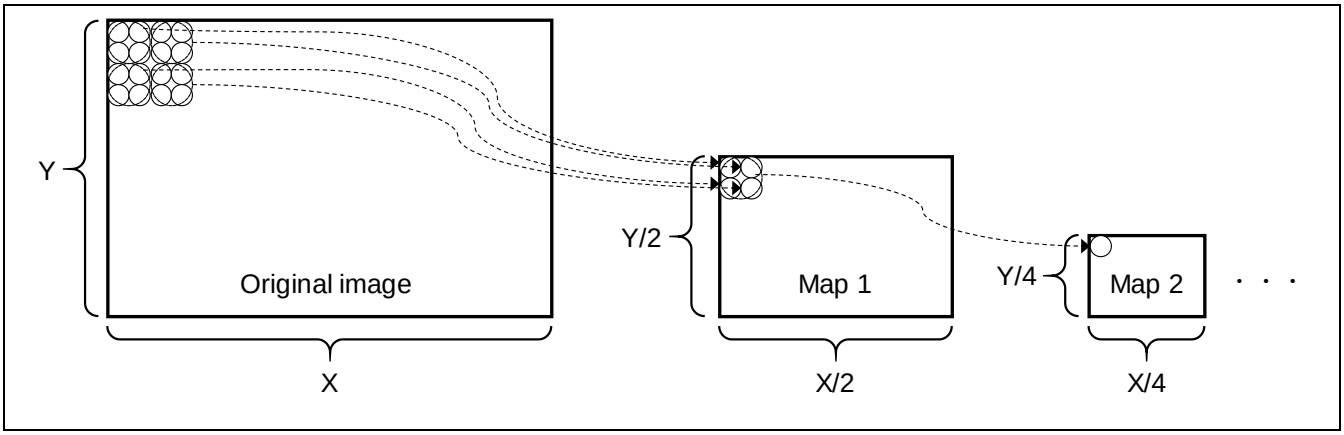



**Figure 56.31 Example of Pyramidal Images**

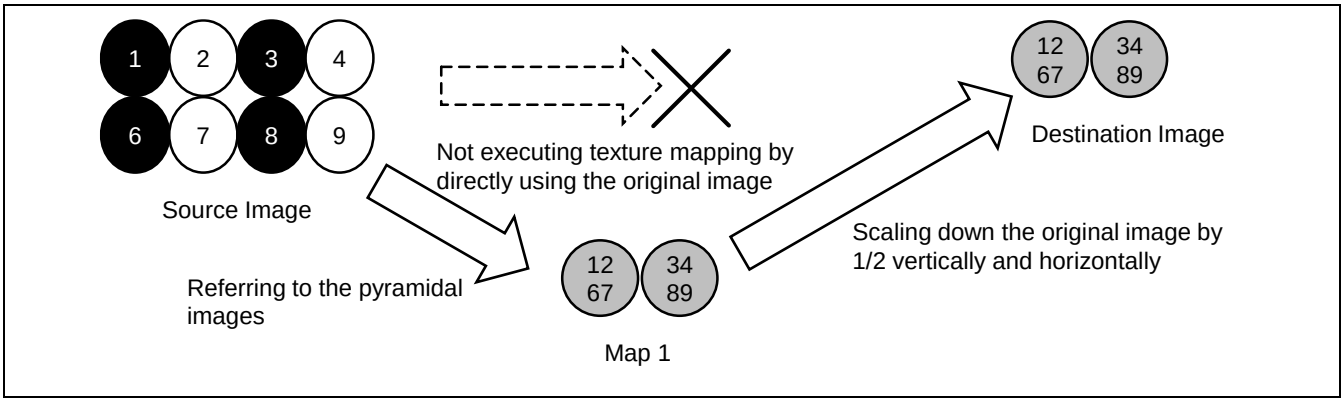



**Figure 56.32 Texture Mapping by Using Pyramidal Images**

In texture mapping, this module calculates the rate of scaling up or down of the destination image from a source image. For scaling up (scale-up rate ≥ 1), the module refers to the original image, and for scaling down (scale-up rate < 1), the module refers to the pyramidal images whose scale-down rate is the closest to the result of calculation.

The number “n” of the map image to be used is calculated as follows. The original image is taken as map 0.

n_u = min{log2(shrink_ratio_u), MAXLODU }

n_v = min{log2(shrink_ratio_v), MAXLODV }

n = max{n_u, n_v}

shrink_ratio_u: Scale-down rate in the u direction, shrink_ratio_v: Scale-down rate in the v direction

MAXLODU: Maximum setting of the map number for the u direction to be used

MAXLODV: Maximum setting of the map number for the v direction to be used

The value with the part below the floating-point discarded from “n” becomes the number of the map image. In the above example, as the scale-down rate is 1/2 both vertically and horizontally, “n” becomes 1 and texture mapping is executed by referring to map 1. As a result, the destination image does not become an image which is entirely black but a gray image into which the black and white lines in the original image have been blended, thus improving the quality of the resulting image.

If the maximum scale-down rates in the u and v directions are extremely different, the larger number is selected as the map number, which may lead to a fuzzy destination image. For example, if texture mapping where the scale-down rates in the u and v directions are 2 and 8 respectively is executed, map 3 is selected. However, this image is unnecessarily scaled down in the u direction, and thus looks fuzzy. In this case, setting the smaller number by specifying MAXLODU and MAXLODV to 1 and 3 respectively enables suppression of the blurring of the image.

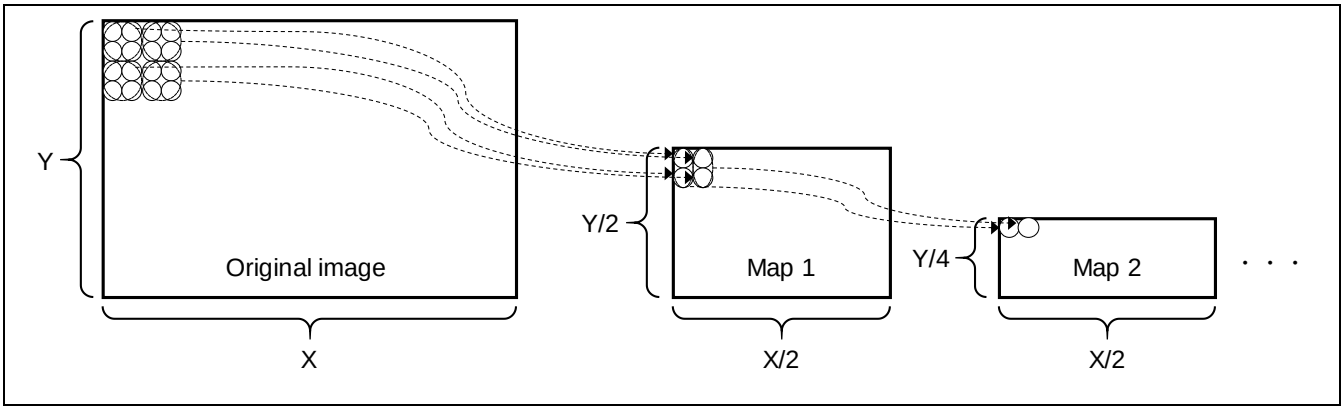



**Figure 56.33 Example of Pyramidal Images (MAXLODU = 1, MAXLODV = 4)**

The above figure is an example of the pyramidal images where MAXLODU and MAXLODV are set to 1 and 4, respectively. Though the original image is recursively scaled down by 1/2 in the Y direction, it's scaled down by 1/2 into map 1 and then not scaled down into map 2 and subsequent levels in the X direction. This enables retention of the details of the image in the X direction and thus allows texture mapping without images being out of focus even if a large number is selected as the map number.

**How to Use Filtering by Pyramidal Images**

To enable this function, set the MAXLODU and MAXLODV fields in the TRIMR2 register to any value from 1 to 4. Taking N as whichever is larger of the values of MAXLODU and MAXLODUV, drawing is handled with the use of the single image from among map images 1 to N and the original image.

Also prepare and arrange in memory in advance the pyramidal images in accord with the MAXLODU and MAXLODV settings, and set their addresses in memory and the memory stride. To make a setting for map n, use the SSARn, and SSTRn registers. Also use the SSAORn, UVSSARn, UVSSAORn, and UVSSTRn. To set for the original image, use the SSAR and SSTR registers regardless of whether or not you will be using this function. Also, use SSAOR, UVSSAR, UVSSAORn and UVSSTRn.

To create the pyramidal images by recursively scaling down by 1/2, the size of the original image must be a multiple of the MAXLODUth power of two in the X direction and a multiple of the MAXLODVth power of two in the Y direction.

When creating pyramidal images by using the IMR-LX6, use bilinear filtering with the source coordinates (u, v) corresponding to the destination coordinates (X, Y) shifted by +0.5 in both the u and v directions*. Shifting by +0.5 places the four vertices of the source coordinates in the middle of four neighboring pixels, which allows rendering of a pixel that is the average of the four neighboring pixels. When scaling down a source image vertically and horizontally into half the original size as shown in the Figure below, set the source coordinates (0, 0) corresponding to the destination coordinates (0, 0) to (0.5, 0.5).

If you create another map following the first one, do so based on the first map to have been created. Similarly, if you create more maps, do so based on the preceding one each time.

Note: Do not shift both source coordinates by +0.5 unless you intend to scale down the original source image in both the u and v directions. For example, if scaling down is to be in the u direction but not in the v direction, only shift the u source coordinate by +0.5, and leave the v source coordinate unchanged.

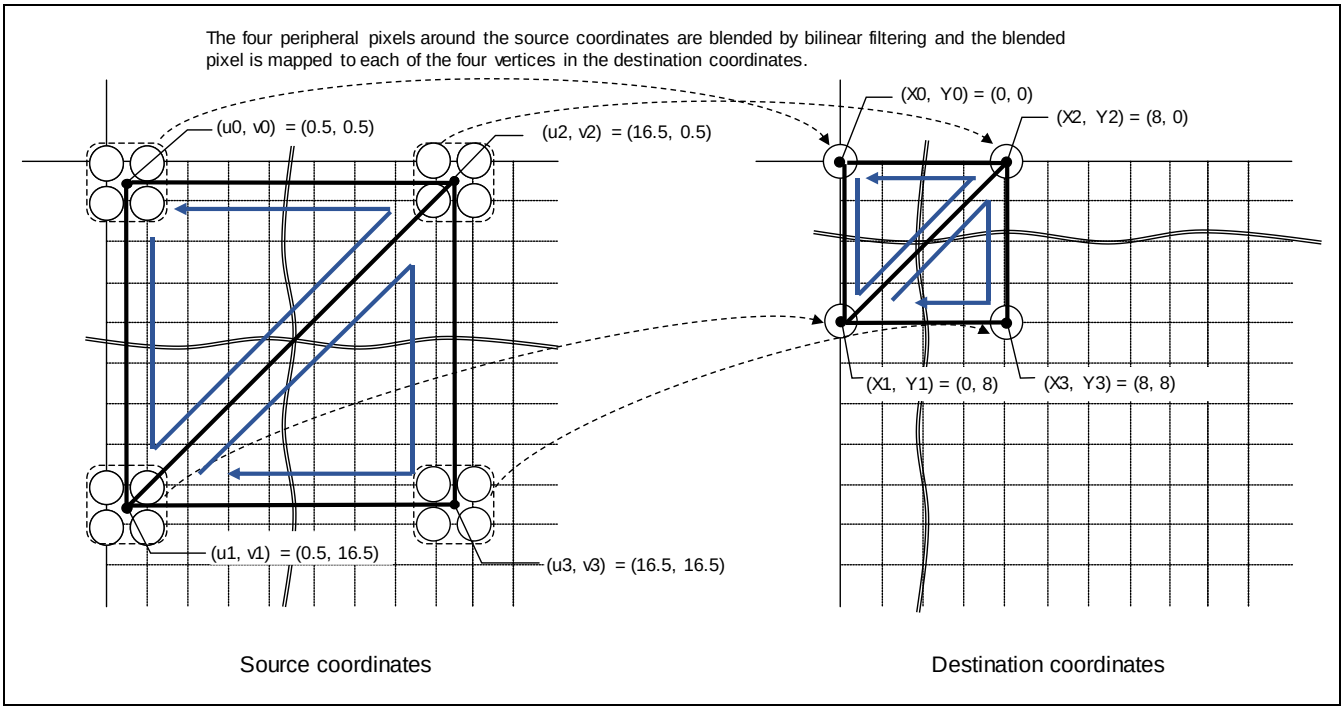



**Figure 56.34 Example of the Setting of Coordinates when Creating a Pyramidal Image**

### 56.3.12.2 Blending between Pyramidal Images

While pyramidal images are in use to improve image quality in scaling down, a smooth change in the scale-down rate may lead to the interruption of images in the interstices between the maps in use. To prevent this, set the TFE bit in the TRIMR register to 1 as well as enabling the filtering by using pyramidal images. This enables blending of the map images corresponding to the rates around a scale-down rate which is not a power of two, thus improving the image quality in the case where the scale-down rate changes smoothly.

Specifically, in the following expressions used for selection of the map number which are described on the previous page,

n_u = log2(shrink_ratio_u) > MAXLODU?MAXLODU:log2(shrink_ratio_u),

n_v = log2(shrink_ratio_v) > MAXLODV?MAXLODV:log2(shrink_ratio_v), and

n = max{n_u, n_v},

the calculated n is used to execute a floor function (n_floor) and a ceiling function (n_ceil), and then the map images with the numbers equivalent to the results (n_floor) and (n_ceil) are used for texture mapping (these map numbers are referred to as pix_floor and pix_ceil). The final value of pixels can be calculated from the following expression by using pix_floor, pix_ceil and the fractional part of n.

pix = pix_floor * (1-n_fraction) + pix_ceil * n_fraction

An example for n = 1.4 is given below.

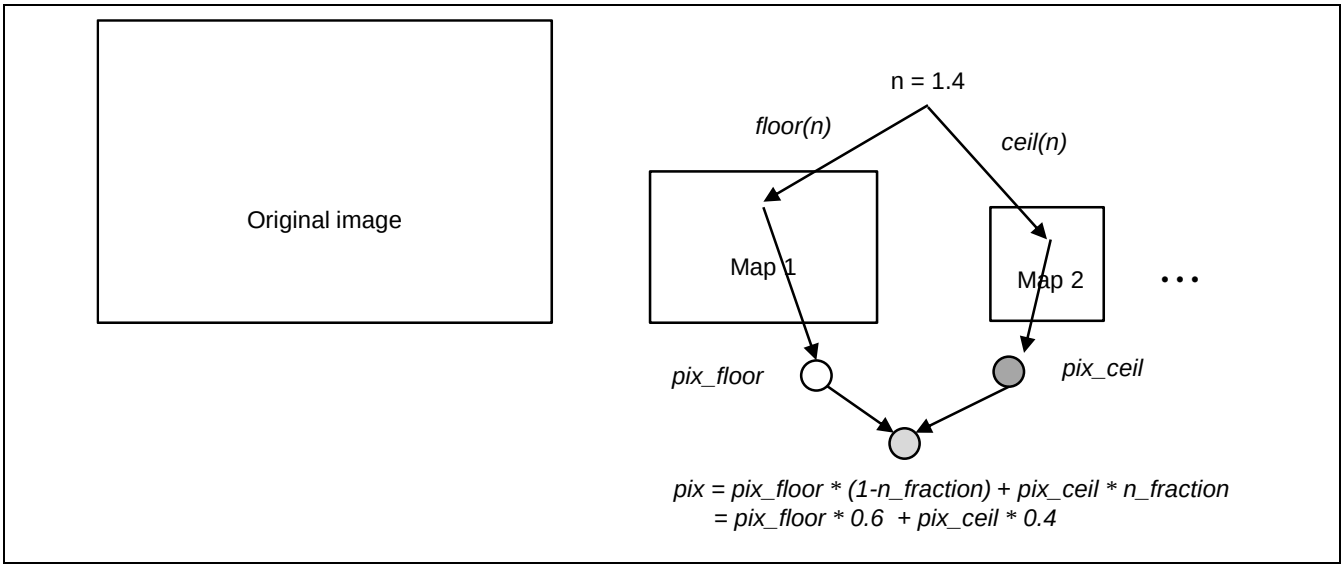



**Figure 56.35 Example of Blending between Maps (n = 1.4)**

Bilinear filtering must be enabled for the filtering of images by using pyramidal images. Set the TFE and BFE bits in TRIMR to 1 to enable bilinear filtering. Bilinear filtering is applied to the texture mapping of each of pix_floor and pix_ceil described above.

### 56.3.13 Non-blocking Cache Mode

This module issues source data read request to external memory where the source data is stored. The maximum number of the on-the-fly requests can be controlled by NCME bit of NCMR register.

If the bit is set to 1, this module issues multiple requests even if the leading on-the-fly requests haven't received the responses from the corresponding external memory. Thus, the rendering performance of this module can be improved compared to the case NCME bit is clear to 0.

If the bit is set to 0, this module does not issue any request if the leading on-the-fly request haven't received the responses. Thus, the memory band width consumed by this module can be smaller in a time unit.

The maximum number of destination data write requests is not constrained by NCME bit.

### 56.3.14 Double luminance plane processing mode

This additional mode will support input precision is 8bpp,10bpp, 12bpp,14bpp,16bpp and setting separated for double Y plane. In case of using DYP mode, set the enable bit CMRCR2.DYP=1.

1. 1st Y plane use CMRCR (SY12, SY10, Y12, Y10, DY1416, SY1416).
2. 2nd Y plane use CMRCR (SUV, DUV, SUV1416, DUV1416).
3. Source address and stride: Use setting of UVSSAR and UVSSTR for second Y plane.
4. Destination address: Use same setting registers with YUV semi-planar mode: DSAR + DSOR for second Y plane.
5. Destination Stride: Using UVDSTR if bit DCCR.UVDSTSEL = 1. Otherwise, use the same setting of DSTR for second Y plane.
6. Support only bilinear filter (BFE), don't support Pyramidal Image Filtering (TFE).
7. Do not support LUT for DYP mode (CMRCR2.LUTE=0).
8. Support output separate mode only.
9. Use same data format for input and output when using double luminance plane. And use same bit depth for double (both) Y plane.
10. Set CMRCR. EFPE = 1 and CMRCR.YISM = 1 when using DYP mode.
11. 2nd Y source data in single color mode depend on value of TRICR3.
12. DYP mode do not support Rotator and Scalar function. Set RSE bit in the RSCR to 0 when DYP mode=1.

Luminance correction function support as below:

1. Set YLMINR/YLMAXR for both Y in DYP mode.
2. Set CMRCR.LUSM, CMRCR.LUOM for both Y in DYP mode.
3. Set DYP_LSCAL and DYP_LOFST of YLCPR for second Y plane in DYP mode.
4. When DYP mode=1, use type (m) and using LUCE values from DL.

## 56.4 Display List

It is prohibited to use instructions not listed on this section.

### 56.4.1 TRI (Triangle)

#### (1) Function

TRI draws N-vertex strip-type triangles according to the number of vertexes, texture coordinates, and destination coordinates specified.

#### (2) Instruction Format

When destination coordinates (X, Y) or source coordinates (u, v) are automatically generated, omit them from the DL.When destination coordinates or source coordinates are not automatically generated and thus not omitted, arrange u, v, X, and Y in the memory in this order.
For absolute specification, arrange u, v, X, and Y in 16-bit units. Do not allow the automatically generated coordinates to exceed the drawing space in auto mode. If the destination coordinates are equal or larger 2,048, which is the maximum drawing space size, the corresponding pixels are not drawn.
The source drawing space is $0 \leq u < 2,048$ and $0 \leq v < 2,048$. If u or v exceeds the specified range, operation will be undefined.

The size of the circumscribed rectangle of a triangle must be less than 128 pixels for both X-direction and Y-direction.

For inputting the source coordinates to the DL in 16-bit units, the coordinates can be specified using the fixed point. In this case, specify the number of decimal places with UVDPOR (H'70). No register is provided for turning on/off fixed-point mode. Setting 0 to UVDPO in UVDPOR causes integer mode. In the DL figures below, N indicates the number of vertexes. For example, substitute 5 for N to draw a 5-vertex image. N < 3 is prohibited.

Drawing is performed only if the order of the first three vertexes of a specified list of vertexes matches the order that is specified by the TCM bit in the triangle mode register (TRIMR); if there is no match, drawing by the TRI instruction will not be performed.

For luminance/hue correction, specify the parameters in the DL. Even in the mode in which the parameters should be specified in the register, specify dummy parameters in the DL. In case of setting as Table 56.23, do not need to insert dummy parameters in DL. In this case, supported DL types are only the (a), (b) and (c). For details on luminance/hue correction, refer to section 56.3.8, Luminance/Hue Correction.OP CODE is H'8A for all the DL types. The DL type is determined by selecting either automatic coordinate generation mode with the triangle mode register (TRIMR) and enabling or disabling luminance/hue correction with the rendering mode register (CMRCR). When setting in CP32E/CP16E bits of CMRCR is 1, select DL type (a), (b) or (c), selecting others are prohibited.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N | | | | | | | | | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### (a) uvXY Type (LUCE = 0, CLCE = 0)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| u0 | | | | | | | | | | | | | | | | v0 | | | | | | | | | | | | | | | |
| X0 | | | | | | | | | | | | | | | | Y0 | | | | | | | | | | | | | | | |
| u1 | | | | | | | | | | | | | | | | v1 | | | | | | | | | | | | | | | |
| X1 | | | | | | | | | | | | | | | | Y1 | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u(N - 1) | | | | | | | | | | | | | | | | v(N - 1) | | | | | | | | | | | | | | | |
| X(N - 1) | | | | | | | | | | | | | | | | Y(N - 1) | | | | | | | | | | | | | | | |

### (b) uv Automatically Generated; XY Type (LUCE = 0, CLCE = 0)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| X0 | | | | | | | | | | | | | | | | Y0 | | | | | | | | | | | | | | | |
| X1 | | | | | | | | | | | | | | | | Y1 | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| X(N - 1) | | | | | | | | | | | | | | | | Y(N - 1) | | | | | | | | | | | | | | | |

### (c) XY Automatically Generated; uv Type (LUCE = 0, CLCE = 0)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| u0 | | | | | | | | | | | | | | | | v0 | | | | | | | | | | | | | | | |
| u1 | | | | | | | | | | | | | | | | v1 | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u(N - 1) | | | | | | | | | | | | | | | | v(N - 1) | | | | | | | | | | | | | | | |

### (d) uvXY Type (LUCE = 0, CLCE = 1) when DYP mode=0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| u0 | | | | | | | | | | | | | | | | v0 | | | | | | | | | | | | | | | |
| X0 | | | | | | | | | | | | | | | | Y0 | | | | | | | | | | | | | | | |
| ubscl0 | | | | | | | | ubofs0 | | | | | | | | vrscl0 | | | | | | | | vrofs0 | | | | | | | |
| u1 | | | | | | | | | | | | | | | | v1 | | | | | | | | | | | | | | | |
| X1 | | | | | | | | | | | | | | | | Y1 | | | | | | | | | | | | | | | |
| ubscl1 | | | | | | | | ubofs1 | | | | | | | | vrscl1 | | | | | | | | vrofs1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u(N - 1) | | | | | | | | | | | | | | | | v(N - 1) | | | | | | | | | | | | | | | |
| X(N - 1) | | | | | | | | | | | | | | | | Y(N - 1) | | | | | | | | | | | | | | | |
| ubscl(N-1) | | | | | | | | ubofs(N-1) | | | | | | | | vrscl(N-1) | | | | | | | | vrofs(N-1) | | | | | | | |

### (e) uvXY Type (LUCE = 1, CLCE = 0)

Note: when DYP mode=0, lscal1_n and lofs1_n is not used (Reserved (all 0)).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| u0 | | | | | | | | | | | | | | | | v0 | | | | | | | | | | | | | | | |
| X0 | | | | | | | | | | | | | | | | Y0 | | | | | | | | | | | | | | | |
| lscal1_0 | | | | | | | | lofst1_0 | | | | | | | | lscal0_0 | | | | | | | | lofst0_0 | | | | | | | |
| u1 | | | | | | | | | | | | | | | | v1 | | | | | | | | | | | | | | | |
| X1 | | | | | | | | | | | | | | | | Y1 | | | | | | | | | | | | | | | |
| lscal1_1 | | | | | | | | lofst1_1 | | | | | | | | lscal0_1 | | | | | | | | lofst0_1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u(N - 1) | | | | | | | | | | | | | | | | v(N - 1) | | | | | | | | | | | | | | | |
| X(N - 1) | | | | | | | | | | | | | | | | Y(N - 1) | | | | | | | | | | | | | | | |
| lscal1_(N-1) | | | | | | | | lofst1_(N-1) | | | | | | | | lscal0_(N-1) | | | | | | | | lofst0_(N-1) | | | | | | | |

### (f) uvXY Type (LUCE = 1, CLCE = 1) when DYP mode=0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| u0 | | | | | | | | | | | | | | | | v0 | | | | | | | | | | | | | | | |
| X0 | | | | | | | | | | | | | | | | Y0 | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal0 | | | | | | | | lofst0 | | | | | | | |
| ubscl0 | | | | | | | | ubofs0 | | | | | | | | vrscl0 | | | | | | | | vrofs0 | | | | | | | |
| u1 | | | | | | | | | | | | | | | | v1 | | | | | | | | | | | | | | | |
| X1 | | | | | | | | | | | | | | | | Y1 | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal1 | | | | | | | | lofst1 | | | | | | | |
| ubscl1 | | | | | | | | ubofs1 | | | | | | | | vrscl1 | | | | | | | | vrofs1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u(N - 1) | | | | | | | | | | | | | | | | v(N - 1) | | | | | | | | | | | | | | | |
| X(N - 1) | | | | | | | | | | | | | | | | Y(N - 1) | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal(N-1) | | | | | | | | lofst(N-1) | | | | | | | |
| ubscl(N-1) | | | | | | | | ubofs(N-1) | | | | | | | | vrscl(N-1) | | | | | | | | vrofs(N-1) | | | | | | | |

### (g) uv Automatically Generated; XY Type (LUCE = 0, CLCE = 1) when DYP mode=0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| X0 | | | | | | | | | | | | | | | | Y0 | | | | | | | | | | | | | | | |
| ubscl0 | | | | | | | | ubofs0 | | | | | | | | vrscl0 | | | | | | | | vrofs0 | | | | | | | |
| X1 | | | | | | | | | | | | | | | | Y1 | | | | | | | | | | | | | | | |
| ubscl1 | | | | | | | | ubofs1 | | | | | | | | vrscl1 | | | | | | | | vrofs1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| X(N - 1) | | | | | | | | | | | | | | | | Y(N - 1) | | | | | | | | | | | | | | | |
| ubscl(N-1) | | | | | | | | ubofs(N-1) | | | | | | | | vrscl(N-1) | | | | | | | | vrofs(N-1) | | | | | | | |

### (h) uv Automatically Generated; XY Type (LUCE = 1, CLCE = 0)

Note: when DYP mode=0, lscal1_n and lofs1_n is not used (Reserved (all 0)).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| X0 | | | | | | | | | | | | | | | | Y0 | | | | | | | | | | | | | | | |
| lscal1_0 | | | | | | | | lofst1_0 | | | | | | | | lscal0_0 | | | | | | | | lofst0_0 | | | | | | | |
| X1 | | | | | | | | | | | | | | | | Y1 | | | | | | | | | | | | | | | |
| lscal1_1 | | | | | | | | lofst1_1 | | | | | | | | lscal0_1 | | | | | | | | lofst0_1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| X(N - 1) | | | | | | | | | | | | | | | | Y(N - 1) | | | | | | | | | | | | | | | |
| lscal1_(N-1) | | | | | | | | lofst1_(N-1) | | | | | | | | lscal0_(N-1) | | | | | | | | lofst0_(N-1) | | | | | | | |

### (i) uv Automatically Generated; XY Type (LUCE = 1, CLCE = 1) when DYP mode=0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| X0 | | | | | | | | | | | | | | | | Y0 | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal0 | | | | | | | | lofst0 | | | | | | | |
| ubscl0 | | | | | | | | ubofs0 | | | | | | | | vrscl0 | | | | | | | | vrofs0 | | | | | | | |
| X1 | | | | | | | | | | | | | | | | Y1 | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal1 | | | | | | | | lofst1 | | | | | | | |
| ubscl1 | | | | | | | | ubofs1 | | | | | | | | vrscl1 | | | | | | | | vrofs1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| X(N - 1) | | | | | | | | | | | | | | | | Y(N - 1) | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal(N-1) | | | | | | | | lofst(N-1) | | | | | | | |
| ubscl(N-1) | | | | | | | | ubofs(N-1) | | | | | | | | vrscl(N-1) | | | | | | | | vrofs(N-1) | | | | | | | |

### (j) XY Automatically Generated; uv Type (LUCE = 0, CLCE = 1) when DYP mode=0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| u0 | | | | | | | | | | | | | | | | v0 | | | | | | | | | | | | | | | |
| ubscl0 | | | | | | | | ubofs0 | | | | | | | | vrscl0 | | | | | | | | vrofs0 | | | | | | | |
| u1 | | | | | | | | | | | | | | | | v1 | | | | | | | | | | | | | | | |
| ubscl1 | | | | | | | | ubofs1 | | | | | | | | vrscl1 | | | | | | | | vrofs1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u(N - 1) | | | | | | | | | | | | | | | | v(N - 1) | | | | | | | | | | | | | | | |
| ubscl(N-1) | | | | | | | | ubofs(N-1) | | | | | | | | vrscl(N-1) | | | | | | | | vrofs(N-1) | | | | | | | |

### (k) XY Automatically Generated; uv Type (LUCE = 1, CLCE = 0)

Note: when DYP mode=0, lscal1_n and lofs1_n is not used (Reserved (all 0)).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| u0 | | | | | | | | | | | | | | | | v0 | | | | | | | | | | | | | | | |
| lscal1_0 | | | | | | | | lofst1_0 | | | | | | | | lscal0_0 | | | | | | | | lofst0_0 | | | | | | | |
| u1 | | | | | | | | | | | | | | | | v1 | | | | | | | | | | | | | | | |
| lscal1_1 | | | | | | | | lofst1_1 | | | | | | | | lscal0_1 | | | | | | | | lofst0_1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u(N - 1) | | | | | | | | | | | | | | | | v(N - 1) | | | | | | | | | | | | | | | |
| lscal1_(N-1) | | | | | | | | lofst1_(N-1) | | | | | | | | lscal0_(N-1) | | | | | | | | lofst0_(N-1) | | | | | | | |

### (l) XY Automatically Generated; uv Type (LUCE = 1, CLCE = 1)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1010 (H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | N (3 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |
| u0 | | | | | | | | | | | | | | | | v0 | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal0 | | | | | | | | lofst0 | | | | | | | |
| ubscl0 | | | | | | | | ubofs0 | | | | | | | | vrscl0 | | | | | | | | vrofs0 | | | | | | | |
| u1 | | | | | | | | | | | | | | | | v1 | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal1 | | | | | | | | lofst1 | | | | | | | |
| ubscl1 | | | | | | | | ubofs1 | | | | | | | | vrscl1 | | | | | | | | vrofs1 | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| u(N - 1) | | | | | | | | | | | | | | | | v(N - 1) | | | | | | | | | | | | | | | |
| Reserved (all 0) | | | | | | | | | | | | | | | | lscal(N-1) | | | | | | | | lofst(N-1) | | | | | | | |
| ubscl(N-1) | | | | | | | | ubofs(N-1) | | | | | | | | vrscl(N-1) | | | | | | | | vrofs(N-1) | | | | | | | |

**Table 56.23 Support disable input dummy data in DL if use value of register.**

| TRIMR | CMRCR | | | | | |
|---|---|---|---|---|---|---|
| DIDL | CLSM | CLOM | LUSM | LUOM | CLCE | LUCE |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### 56.4.2 NOP (No Operation)

#### (1) Function

NOP inserts N cycles of NOP into a drawing pipeline.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_0000 (H'80) | | | | | | | | Reserved (all 0) | | | | | | | | N (1 ≤ N ≤ 65,535) | | | | | | | | | | | | | | | |

It is not guaranteed that inserting N cycles of NOP into the drawing pipeline always delays the drawing completion exactly for N cycles. This is because the NOP cycles may overlap with the periods while drawing is stalled due to bus arbitration or cache hit/miss in the pixel-processing unit.

Substitute the number of NOP cycles to be inserted for N. Therefore, substitute X for N to insert X-number of NOP. Substituting 0 is prohibited. Note that N does not indicate the number of cycles in which stall actually occurs; it indicates the number of NOP instructions to be issued to the following stage.

### 56.4.3 TRAP

#### (1) Function

TRAP leads to completion of display list execution followed by the generation of an interrupt. If the TRAP instruction is to be used, insert a SYNCM instruction immediately before the TRAP instruction. If there is no SYNCM instruction, the operation will be undefined.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1111 (H'8F) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |

### 56.4.4 WTL (Write Register Long)

#### (1) Function

WTL writes data in 32-bit units to the N-number of consecutive registers starting at ADD. Set the address to ADD[23:16] in 32-bit (4-byte) units. For example, specify H'19 for TRIMSR (H'064). Note that WTL cannot write data to the registers in the area whose address exceeds 10 bits.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_0001 (H'81) | | | | | | | | ADD | | | | | | | | N | | | | | | | | | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA N-1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

N indicates the number of consecutive addresses to be written to. Substitute X for N to write to X-number of registers. Substituting 0 is prohibited. Immediately before writing to the registers related to memory access, a SYNCM instruction must be inserted. For details, refer to the descriptions on the registers.

### 56.4.5 WTL2 (Write Register Long 2)

#### (1) Function

WTL2 writes data in 32-bit units to the N-number of consecutive registers starting at ADD. Set the address to ADD[23:10] in 32-bit (4-byte) units. Note that WTL2 cannot write data to the registers in the area whose address exceeds 16 bits.

Although WTL and WTS can only set the registers allocated to the 10-bit address space, WTL2 can set the registers allocated to the 16-bit address space.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_0011 (H'83) | | | | | | | | ADD | | | | | | | | | | | | | | N | | | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA N-1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

N indicates the number of consecutive addresses to be written to. Substitute X for N to write to X-number of registers. Substituting 0 is prohibited. A maximum of 512 can be set. Immediately before writing to the registers related to memory access, a SYNCM instruction must be inserted. For details, refer to the descriptions on the registers.

### 56.4.6 WTS (Write Register Short)

#### (1) Function

WTS writes 16-bit data to the [15:0] bits in the register indicated by ADD.

Set the address to ADD[23:16] in 32-bit (4-byte) units. For example, specify H'19 for TRIMSR (H'064). Note that WTS cannot write data to the registers in the area whose address exceeds 10 bits.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_0010 (H'82) | | | | | | | | ADD | | | | | | | | DATA | | | | | | | | | | | | | | | |

WTS is a simplified version of WTL for reducing the DL size when writing to non-consecutive registers. Immediately before writing to the registers related to memory access, a SYNCM instruction must be inserted. For details, refer to the descriptions on the registers.

### 56.4.7 INT (Interrupt)

#### (1) Function

INT generates an interrupt if interrupts are enabled. To maintain coherence between the main memory and the internal cache when an interrupt is generated, insert a SYNCM instruction immediately before the INT instruction.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1000 (H'88) | | | | | | | | Reserved (all 0) | | | | | | | | N = 0 | | | | | | | | | | | | | | | |

### 56.4.8 SYNCM (Synchronize Memory)

#### (1) Function

After completion of the preceding instruction, SYNCM stalls execution of the next instruction until all dirty lines are written from the pixel cache to the memory. For example, SYNCM can be used to control coherence with data in memory. This instruction is essential to handle pre-processing for TRAP instructions. Always place a SYNCM instruction before any TRAP instruction.

For a display list read out to an internal buffer, a SYNCM instruction does not lead to any updating. Even if a display list in main memory is updated after the execution of a SYNCM instruction, the SYNCM instruction may not have any effect on subsequent execution of the display list.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_0110 (H'86) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |

### 56.4.9 GOSUB (Go Subroutine)

#### (1) Function

GOSUB allows use of a subroutine call. DL execution jumps to the address specified with GSADD, and is returned to the caller by the RET instruction, which is described in the next section. Only one level of nesting is permitted. With the GOSUB instruction, it is not necessary to prepare two types of DL when issuing TRI instructions to the same coordinates for the same number of times, with different rendering modes and base addresses.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1100 (H'8C) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |
| GSADD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Reserved (all 0) | | |

### 56.4.10 RET (Return)

#### (1) Function

RET allows jumping to the DL address next to the address at which the GOSUB instruction is executed.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1101 (H'8D) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |

### 56.4.11 WUP (Wakeup)

#### (1) Function

WUP is used to transmit the corresponding synchronization signal to other modules. Once a module has transmitted this synchronization signal, it continues with execution of the subsequent instructions in the display list.

To transmit the synchronization signal to the module specified in the SYNCCm (m = 0 to 15) field of SYNCCRn (n = 0 to 3), set the corresponding bit of the SYNCC enable field to 1. For details, refer to section 56.3.11, Synchronized system between modules.

Be sure to insert at least eight instructions other than WUP between any two WUP instructions in the display list. Do not execute the WUP instruction to send the signal to a channel for which use of this instruction is prohibited. Refer to Table 56.21 Possible Combinations of Modules for Synchronized Operations .

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1010_0000 (H'A0) | | | | | | | | Reserved (all 0) | | | | | | | | SYNCC enable | | | | | | | | | | | | | | | |

### 56.4.12 SLP (Sleep)

#### (1) Function

SLP is used to initiate a wait for the reception of the synchronization signal from other modules. The SLP instruction suspends execution of the subsequent instructions in the display list until the module has received a synchronization signal from the other module with which it is to be synchronized. Once a module has received the synchronization signal, it continues with the execution of the subsequent instructions in the display list.

To cause a wait for the reception of the synchronization signal from the module specified in the SYNCCm (m = 0 to 15) field of SYNCCRn (n = 0 to 3), set the corresponding bit of the SYNCC enable field to 1. For details, refer to section 56.3.11, Synchronized system between modules.

Do not execute the SLP instruction in the channel where use of this instruction is prohibited. and Table 56.21 Possible Combinations of Modules for Synchronized Operations for the applicable channels.

#### (2) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | |
|---|---|---|
| OP CODE = 1010_0001 (H'A1) | Reserved (all 0) | SYNCC enable |

## 56.5 Rotator & Scaler Extension

### 56.5.1 Overview

This LSI chip incorporates extended functionality to handle the rotation and scaling-down of data processed by the IMR-LX6 and output of the results to external memory. This functionality is only present on certain channels. Rotation and scaling-down are only available for some channels. See Table 56.2 for availability of this function.

Figure 56.36 shows the relationship between the extended functionality for rotation and scaling-down and the basic function blocks. The figure also includes brief names that indicate the operations handled by the individual blocks of the extension.

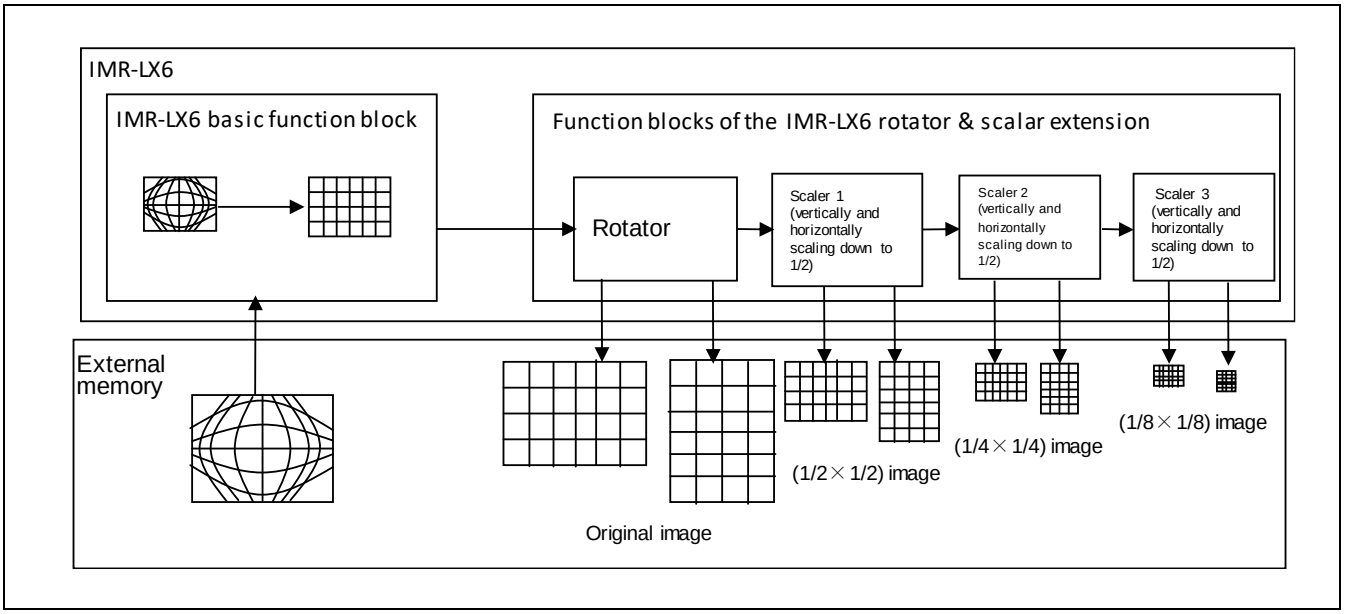



**Figure 56.36 Block Diagram of the Rotator & Scaler Extension**

The basic function block handles the correction of distortion in image data based on the instructions in the display list in the same way as the standard IMR-LX6. When the extended functions are in use, the basic function block outputs the distortion-corrected data to the extended function blocks instead of to the external memory.

The extended function blocks can handle the following processing of the input data.

- Rotation
  - Generation of flipped images
  - Simultaneous generation of rotated and non-rotated images in response to a single input from the basic function block
- Scaling-down
  - Recursive or simple vertical and horizontal scaling down by 1/2
    - Vertical and horizontal scaling down to 1/8
    - Filtering the results of scaling down (averaging the values of sets of neighboring four pixels)
    - 3x3 Gaussian Filter on Y Plane
  - Rotated or non-rotated images can be scaled down by 1/2, 1/4, or 1/8.
- Supported formats:
  - YUV422/YUV420 semi-planar as described in Table 56.13 (Y processing / UV processing is switched by CMRCR.YCM bit)
  - Bit depths: 8 bpp, 10 bpp, 12 bpp, 14 bpp, or 16 bpp

#### 56.5.1.1 Register Configuration (1)

The registers of the extended functions are mapped to the same space as those of the basic function block. For the base addresses of each channel, see Table 56.2.

Do not write to any address other than those listed in the tables below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined.

**Table 56.24 Register Configuration (1) - Control Registers**

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| Rotator and Scaler Control Register | RSCR | R | H'0200 | 32 |
| Rotator and Scaler Control Set Register | RSCSR | —/W | H'0204 | 32 |
| Rotator and Scaler Control Clear Register | RSCCR | —/W | H'0208 | 32 |
| Rotator and Scaler Output Format Register | RSOFR | R/W | H'020C | 32 |
| Destination Start Address for Non-Rotated Image Register 0 | DSANRR0 | R/W | H'0210 | 32 |
| Destination Stride for Non-Rotated Image Register 0 | DSTNRR0 | R/W | H'0214 | 32 |
| Destination Start Address for Rotated Image Register 0 | DSARR0 | R/W | H'0218 | 32 |
| Destination Stride for Rotated Image Register 0 | DSTRR0 | R/W | H'021C | 32 |
| Destination Start Address for Non-Rotated Image Register 1 | DSANRR1 | R/W | H'0220 | 32 |
| Destination Stride for Non-Rotated Image Register 1 | DSTNRR1 | R/W | H'0224 | 32 |

| Name | Symbol | R/W | Offset | Access Size |
|---|---|---|---|---|
| Destination Start Address for Rotated Image Register 1 | DSARR1 | R/W | H'0228 | 32 |
| Destination Stride for Rotated Image Register 1 | DSTRR1 | R/W | H'022C | 32 |
| Destination Start Address for Non-Rotated Image Register 2 | DSANRR2 | R/W | H'0230 | 32 |
| Destination Stride for Non-Rotated Image Register 2 | DSTNRR2 | R/W | H'0234 | 32 |
| Destination Start Address for Rotated Image Register 2 | DSARR2 | R/W | H'0238 | 32 |
| Destination Stride for Rotated Image Register 2 | DSTRR2 | R/W | H'023C | 32 |
| Destination Start Address for Non-Rotated Image Register 3 | DSANRR3 | R/W | H'0240 | 32 |
| Destination Stride for Non-Rotated Image Register 3 | DSTNRR3 | R/W | H'0244 | 32 |
| Destination Start Address for Rotated Image Register 3 | DSARR3 | R/W | H'0248 | 32 |
| Destination Stride for Rotated Image Register 3 | DSTRR3 | R/W | H'024C | 32 |
| Destination Start Address Offset for Non-Rotated Image Register 0 | DSONRR0 | R/W | H’0250 | 32 |
| Destination Start Address Offset for Non-Rotated Image Register 1 | DSONRR1 | R/W | H’0254 | 32 |
| Destination Start Address Offset for Non-Rotated Image Register 2 | DSONRR2 | R/W | H’0258 | 32 |
| Destination Start Address Offset for Non-Rotated Image Register 3 | DSONRR3 | R/W | H’025C | 32 |
| Destination Start Address Offset for Rotated Image Register 0 | DSORR0 | R/W | H’0260 | 32 |
| Destination Start Address Offset for Rotated Image Register 1 | DSORR1 | R/W | H’0264 | 32 |
| Destination Start Address Offset for Rotated Image Register 2 | DSORR2 | R/W | H’0268 | 32 |
| Destination Start Address Offset for Rotated Image Register 3 | DSORR3 | R/W | H’026C | 32 |
| Gaussian Filter Control Register | GFCR | R/W | H’0270 | 32 |
| Rotator and Scaler Control Register 2 | RSCR2 | R | H’0274 | 32 |
| Rotator and Scaler Control Set Register 2 | RSCSR2 | —/W | H’0278 | 32 |
| Rotator and Scaler Control Clear Register 2 | RSCCR2 | —/W | H’027C | 32 |
| Gaussian Filter Output Format Register | GFOFR | R/W | H’02C8 | 32 |

#### 56.5.1.2 Register Configuration (2)

Some of the registers in the IMR-LX6 can be rewritten using the WTS, WTL, or WTL2 instruction in the DL or accessed from a CPU while the IMR-LX6 is drawing data. In the following tables, √ indicates that the register can be rewritten using the WTS, WTL, or WTL2 instruction or that the register can be rewritten from a CPU during drawing operation.

**Table 56.25 Register Configuration (2) - Control Registers**

| Name | Symbol | WTL/WTL2/WTS | CPU Write Access during Drawing |
|---|---|---|---|
| Rotator and Scaler Control Register | RSCR | — | — |
| Rotator and Scaler Control Set Register | RSCSR | √ | — |
| Rotator and Scaler Control Clear Register | RSCCR | √ | — |
| Rotator and Scaler Output Format Register | RSOFR | √ | — |
| Destination Start Address for Non-Rotated Image Register 0 | DSANRR0 | √ | — |
| Destination Stride for Non-Rotated Image Register 0 | DSTNRR0 | √ | — |
| Destination Start Address for Rotated Image Register 0 | DSARR0 | √ | — |
| Destination Stride for Rotated Image Register 0 | DSTRR0 | √ | — |
| Destination Start Address for Non-Rotated Image Register 1 | DSANRR1 | √ | — |
| Destination Stride for Non-Rotated Image Register 1 | DSTNRR1 | √ | — |
| Destination Start Address for Rotated Image Register 1 | DSARR1 | √ | — |
| Destination Stride for Rotated Image Register 1 | DSTRR1 | √ | — |
| Destination Start Address for Non-Rotated Image Register 2 | DSANRR2 | √ | — |
| Destination Stride for Non-Rotated Image Register 2 | DSTNRR2 | √ | — |
| Destination Start Address for Rotated Image Register 2 | DSARR2 | √ | — |
| Destination Stride for Rotated Image Register 2 | DSTRR2 | √ | — |
| Destination Start Address for Non-Rotated Image Register 3 | DSANRR3 | √ | — |
| Destination Stride for Non-Rotated Image Register 3 | DSTNRR3 | √ | — |
| Destination Start Address for Rotated Image Register 3 | DSARR3 | √ | — |

| Name | Symbol | WTL/WTL2/WTS | CPU Write Access during Drawing |
|---|---|---|---|
| Destination Stride for Rotated Image Register 3 | DSTRR3 | √ | — |
| Destination Start Address Offset for Non-Rotated Image Register 0 | DSONRR0 | √ | — |
| Destination Start Address Offset for Non-Rotated Image Register 1 | DSONRR1 | √ | — |
| Destination Start Address Offset for Non-Rotated Image Register 2 | DSONRR2 | √ | — |
| Destination Start Address Offset for Non-Rotated Image Register 3 | DSONRR3 | √ | — |
| Destination Start Address Offset for Rotated Image Register 0 | DSORR0 | √ | — |
| Destination Start Address Offset for Rotated Image Register 1 | DSORR1 | √ | — |
| Destination Start Address Offset for Rotated Image Register 2 | DSORR2 | √ | — |
| Destination Start Address Offset for Rotated Image Register 3 | DSORR3 | √ | — |
| Gaussian Filter Control Register | GFCR | √ | — |
| Rotator and Scaler Contol Register 2 | RSCR2 | — | — |
| Rotator and Scaler Contol Set Register 2 | RSCSR2 | √ | — |
| Rotator and Scaler Contol Clear Register 2 | RSCCR2 | √ | — |
| Gaussian Filter Output Format Register | GFOFR | √ | — |

### 56.5.2 Register Description

**[Legend]**

Initial value: Register value after a power-on reset
—: Undefined value
R/W: Readable/writable. The written value can be read.
R: Read-only. The write value should be the same as the read value. Only modify the target bit (Read Modify Write) because the initial value should not be changed.
–/W: Write-only. The read value is undefined.

#### 56.5.2.1 Rotator and Scaler Control Register (RSCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSE | RSUVFLT | — | — | — | — | SC8SHFVAL | | — | — | SC4SHFVAL | | — | — | SC2SHFVAL | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSUV420 | — | BP8E | RD8E | BP4E | RD4E | BP2E | RD2E | RT8E | NR8E | RT4E | NR4E | RT2E | NR2E | RT1E | NR1E |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RSE | 0 | R | This bit enables the extended functionality to handle rotation and scaling down.<br>0: The extended functionality is disabled.<br>1: The extended functionality is enabled.<br>When the extended functionality is enabled, output from the basic function block is disabled.<br>Set this bit to 0 when the setting of the YUV422E or DYP bit in the CMRCR2 is 1. Operation cannot be guaranteed if this bit is set to 1 while the setting of the YUV422E or DYP bit in the CMRCR2 is 1. |
| 30 | RSUVFLT | 0 | R | Specifies the algorithm to rotate the UV plane in YUV422 format.<br>0: Rotated in 2x2 block unit<br>1: Rotated during interpolation<br>When this bit is 1, set this RSUV420 bit to 0. |
| 29 to 26 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 25, 24 | SC8SHFVAL | 00 | R | Number of bit positions for logical right shifting of the value following processing for vertical and horizontal reduction to 1/8.<br>Set this field in accord with the combinations shown in Table 56.26 or Table 56.27. |
| 23, 22 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 21, 20 | SC4SHFVAL | 00 | R | Number of bit positions for logical right shifting of the value following processing for vertical and horizontal reduction to 1/4.<br>Set this field in accord with the combinations shown in Table 56.26 or Table 56.27. |
| 19, 18 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 17 to 16 | SC2SHFVAL | 00 | R | Number of bit positions for logical right shifting of the value following processing for vertical and horizontal reduction to 1/2.<br>Set this field in accord with the combinations shown in Table 56.26 or Table 56.27. |
| 15 | RSUV420 | 0 | R | Specifies the YUV format of the input image. The algorithm to rotate the UV plane varies depending on the setting of this bit. The operation when the Y plane is input is not affected.<br>0: YUV422 format<br>1: YUV420 format<br>When this bit is 1, set this RSUVFLT bit to 0.<br>When processing the YUV420 format, setting 0 in this bit is prohibited. |
| 14 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 13 | BP8E | 0 | R | This bit selects whether to average or add the values of sets of neighboring four pixels as filtering in scaling down.<br>0: Four times of the average of the neighboring four pixels.<br>1: The values of the neighboring four pixels are added.<br>This bit controls the output of data vertically and horizontally in scaling down from 1/4 to 1/8 of the size of the input.<br>Regardless of the setting of this bit, the pixel format of the scaled down data is increased by two bits from that before scaling down proceeds. For example, if the pixel format before scaling down was 10 bpp, the pixel format after scaling down will become 12 bpp. Set the SC8SHFVAL field in this register and OFM8N and OFM8R fields in RSOFR so that the pixel value after scaling down is suitable. |
| 12 | RD8E | 0 | R | This bit specifies the rounding mode in the division when averaging of the neighboring four pixels in scaling down is selected.<br>0: Rounding down<br>1: Rounding to the nearest number<br>This bit controls the output of data vertically and horizontally in scaling down from 1/4 to 1/8 of the size of the input.<br>This bit is only valid when the setting of the BP8E bit is 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 | BP4E | 0 | R | This bit selects whether to average or add the values of sets of neighboring four pixels as filtering in scaling down.<br>0: Four times of the average of the neighboring four pixels.<br>1: The values of the neighboring four pixels are added.<br>This bit controls the output of data vertically and horizontally in scaling down from 1/2 to 1/4 of the size of the input.<br>Regardless of the setting of this bit, the pixel format of the scaled down data is increased by two bits from that before scaling down proceeds. For example, if the pixel format before scaling down was 10 bpp, the pixel format after scaling down will become 12 bpp. Set the SC4SHFVAL field in this register and OFM4N and OFM4R fields in RSOFR so that the pixel value after scaling down is suitable. |
| 10 | RD4E | 0 | R | This bit specifies the rounding mode in the division when averaging of the neighboring four pixels in scaling down is selected.<br>0: Rounding down<br>1: Rounding to the nearest number<br>This bit controls the output of data vertically and horizontally in scaling down from 1/2 to 1/4 of the size of the input.<br>This bit is only valid when the setting of the BP4E bit is 0. |
| 9 | BP2E | 0 | R | This bit selects whether to average or add the values of sets of neighboring four pixels as filtering in scaling down.<br>0: Four times of the average of the neighboring four pixels.<br>1: The values of the neighboring four pixels are added.<br>This bit controls the output of data vertically and horizontally in scaling down to 1/2 of the size of the input.<br>Regardless of the setting of this bit, the pixel format of the scaled down data is increased by two bits from that before scaling down proceeds. For example, if the pixel format before scaling down was 10 bpp, the pixel format after scaling down will become 12 bpp. Set the SC2SHFVAL field in this register and OFM2N and OFM2R fields in RSOFR so that the pixel value after scaling down is suitable. |
| 8 | RD2E | 0 | R | This bit specifies the rounding mode in the division when averaging of the neighboring four pixels in scaling down is selected.<br>0: Rounding down<br>1: Rounding to the nearest number<br>This bit controls the output of data vertically and horizontally in scaling down to 1/2 of the size of the input.<br>This bit is only valid when the setting of the BP2E bit is 0. |
| 7 | RT8E | 0 | R | This bit controls output of the data vertically and horizontally scaled down to 1/8 in response to execution of the OUTROT instruction.<br>0: The data vertically and horizontally scaled down to 1/8 are not output in response to execution of the OUTROT instruction.<br>1: The data vertically and horizontally scaled down to 1/8 are output in response to execution of the OUTROT instruction. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | NR8E | 0 | R | This bit controls output of the data vertically and horizontally scaled down to 1/8 in response to execution of the OUTNOR instruction.<br>0: The data vertically and horizontally scaled down to 1/8 are not output in response to execution of the OUTNOR instruction.<br>1: The data vertically and horizontally scaled down to 1/8 are output in response to execution of the OUTNOR instruction. |
| 5 | RT4E | 0 | R | This bit controls output of the data vertically and horizontally scaled down to 1/4 in response to execution of the OUTROT instruction.<br>0: The data vertically and horizontally scaled down to 1/4 are not output in response to execution of the OUTROT instruction.<br>1: The data vertically and horizontally scaled down to 1/4 are output in response to execution of the OUTROT instruction. |
| 4 | NR4E | 0 | R | This bit controls output of the data vertically and horizontally scaled down to 1/4 in response to execution of the OUTNOR instruction.<br>0: The data vertically and horizontally scaled down to 1/4 are not output in response to execution of the OUTNOR instruction.<br>1: The data vertically and horizontally scaled down to 1/4 are output in response to execution of the OUTNOR instruction. |
| 3 | RT2E | 0 | R | This bit controls output of the data vertically and horizontally scaled down to 1/2 in response to execution of the OUTROT instruction.<br>0: The data vertically and horizontally scaled down to 1/2 are not output in response to execution of the OUTROT instruction.<br>1: The data vertically and horizontally scaled down to 1/2 are output in response to execution of the OUTROT instruction. |
| 2 | NR2E | 0 | R | This bit controls output of the data vertically and horizontally scaled down to 1/2 in response to execution of the OUTNOR instruction.<br>0: The data vertically and horizontally scaled down to 1/2 are not output in response to execution of the OUTNOR instruction.<br>1: The data vertically and horizontally scaled down to 1/2 are output in response to execution of the OUTNOR instruction. |
| 1 | RT1E | 0 | R | This bit controls output of the non-scaled data in response to execution of the OUTROT instruction.<br>0: The non-scaled data are not output in response to execution of the OUTROT instruction.<br>1: The non-scaled data are output in response to execution of the OUTROT instruction. |
| 0 | NR1E | 0 | R | This bit controls output of the non-scaled data in response to execution of the OUTNOR instruction.<br>0: The non-scaled data are not output in response to execution of the OUTNOR instruction.<br>1: The non-scaled data are output in response to execution of the OUTNOR instruction. |

### 56.5.2.2 Rotator and Scaler Control Set Register (RSCSR)

This register is used to allow or disallow setting of the corresponding bits in the rotator and scaler control register (RSCR) to 1. Before setting a bit in RSCR to 1, write 1 to the corresponding bit in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSES | RSUVFLTS | — | — | — | — | SC8SHFVALS | | — | — | SC4SHFVALS | | — | — | SC2SHFVALS | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | –/W | –/W | R | R | R | R | –/W | –/W | R | R | –/W | –/W | R | R | –/W | –/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSUV420S | — | BP8ES | RD8ES | BP4ES | RD4ES | BP2ES | RD2ES | RT8ES | NR8ES | RT4ES | NR4ES | RT2ES | NR2ES | RT1ES | NR1ES |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | –/W | R | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RSES | 0 | –/W | 0: The RSE bit in RSCR is not set to 1.<br>1: The RSE bit in RSCR is set to 1. |
| 30 | RSUVFLTS | 0 | –/W | 0: The RSUVFLT bit in RSCR is not set to 1<br>1: The RSUVFLT bit in RSCR is set to 1. |
| 29 to 26 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 25, 24 | SC8SHFVALS | 00 | –/W | The bits of this field set the respective bits of the SC8SHFVAL field to 1.<br>That is, writing 1 to a bit of this field sets the corresponding bit of the SC8SHFVAL field to 1.<br>The corresponding bit of the SC8SHFVAL field retains its value in response to 0 being written to a bit of this field. |
| 23, 22 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 21, 20 | SC4SHFVALS | 00 | –/W | The bits of this field set the respective bits of the SC4SHFVAL field to 1.<br>That is, writing 1 to a bit of this field sets the corresponding bit of the SC4SHFVAL field to 1.<br>The corresponding bit of the SC4SHFVAL field retains its value in response to 0 being written to a bit of this field. |
| 19, 18 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17, 16 | SC2SHFVALS | 00 | –/W | The bits of this field set the respective bits of the SC2SHFVAL field to 1.<br>That is, writing 1 to a bit of this field sets the corresponding bit of the SC2SHFVAL field to 1.<br>The corresponding bit of the SC2SHFVAL field retains its value in response to 0 being written to a bit of this field. |
| 15 | RSUV420S | 0 | –/W | 0: The RSUV420 bit in RSCR is not set to 1.<br>1: The RSUV420 bit in RSCR is set to 1 |
| 14 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 13 | BP8ES | 0 | –/W | 0: The BP8E bit in RSCR is not set to 1.<br>1: The BP8E bit in RSCR is set to 1. |
| 12 | RD8ES | 0 | –/W | 0: The RD8E bit in RSCR is not set to 1.<br>1: The RD8E bit in RSCR is set to 1. |
| 11 | BP4ES | 0 | –/W | 0: The BP4E bit in RSCR is not set to 1.<br>1: The BP4E bit in RSCR is set to 1. |
| 10 | RD4ES | 0 | –/W | 0: The RD4E bit in RSCR is not set to 1.<br>1: The RD4E bit in RSCR is set to 1. |
| 9 | BP2ES | 0 | –/W | 0: The BP2E bit in RSCR is not set to 1.<br>1: The BP2E bit in RSCR is set to 1. |
| 8 | RD2ES | 0 | –/W | 0: The RD2E bit in RSCR is not set to 1.<br>1: The RD2E bit in RSCR is set to 1. |
| 7 | RT8ES | 0 | –/W | 0: The RT8E bit in RSCR is not set to1.<br>1: The RT8E bit in RSCR is set to 1. |
| 6 | NR8ES | 0 | –/W | 0: The NR8E bit in RSCR is not set to 1.<br>1: The NR8E bit in RSCR is set to 1. |
| 5 | RT4ES | 0 | –/W | 0: The RT4E bit in RSCR is not set to 1.<br>1: The RT4E bit in RSCR is set to 1. |
| 4 | NR4ES | 0 | –/W | 0: The NR4E bit in RSCR is not set to 1.<br>1: The NR4E bit in RSCR is set to 1. |
| 3 | RT2ES | 0 | –/W | 0: The RT2E bit in RSCR is not set to 1.<br>1: The RT2E bit in RSCR is set to 1. |
| 2 | NR2ES | 0 | –/W | 0: The NR2E bit in RSCR is not set to 1.<br>1: The NR2E bit in RSCR is set to 1. |
| 1 | RT1ES | 0 | –/W | 0: The RT1E bit in RSCR is not set to 1.<br>1: The RT1E bit in RSCR is set to 1. |
| 0 | NR1ES | 0 | –/W | 0: The NR1E bit in RSCR is not set to 1.<br>1: The NR1E bit in RSCR is set to 1. |

### 56.5.2.3 Rotator and Scaler Control Clear Register (RSCCR)

This register is used to allow or disallow clearing of the corresponding bits in the rotator and scaler control register (RSCR). Before clearing a bit in RSCR, write 1 to the corresponding bit in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSEC | RSUVFLTC | — | — | — | — | SC8SHFVALC | | — | — | SC4SHFVALC | | — | — | SC2SHFVALC | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | –/W | –/W | R | R | R | R | –/W | –/W | R | R | –/W | –/W | R | R | –/W | –/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RSUV420C | — | BP8EC | RD8EC | BP4EC | RD4EC | BP2EC | RD2EC | RT8EC | NR8EC | RT4EC | NR4EC | RT2EC | NR2EC | RT1EC | NR1EC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | –/W | R | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W | –/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RSEC | 0 | –/W | 0: Does not clear the RSE bit in RSCR to 0.<br>1: Clears the RSE bit in RSCR to 0. |
| 30 | RSUVFLTC | 0 | –/W | 0: Does not clear the RSUVFLT bit in RSCR to 0.<br>1: Clears the RSUVFLT bit in RSCR to 0. |
| 29 to 26 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 25, 24 | SC8SHFVALC | 00 | –/W | The bits of this field clear the respective bits of the SC8SHFVAL field to 0.<br>That is, writing 1 to a bit of this field clears the corresponding bit of the SC8SHFVAL field to 0.<br>The corresponding bit of the SC8SHFVAL field retains its value in response to 0 being written to a bit of this field. |
| 23, 22 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 21, 20 | SC4SHFVALC | 00 | –/W | The bits of this field clear the respective bits of the SC4SHFVAL field to 0.<br>That is, writing 1 to a bit of this field clears the corresponding bit of the SC4SHFVAL field to 0.<br>The corresponding bit of the SC4SHFVAL field retains its value in response to 0 being written to a bit of this field. |
| 19, 18 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17, 16 | SC2SHFVALC | 00 | –/W | The bits of this field clear the respective bits of the SC2SHFVAL field to 0.<br>That is, writing 1 to a bit of this field clears the corresponding bit of the SC2SHFVAL field to 0.<br>The corresponding bit of the SC2SHFVAL field retains its value in response to 0 being written to a bit of this field. |
| 15 | RSUV420C | 0 | –/W | 0: Does not clear the RSUV420 bit in RSCR to 0.<br>1: Clears the RSUV420 bit in RSCR to 0. |
| 14 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 13 | BP8EC | 0 | –/W | 0: Does not clear the BP8E bit in RSCR to 0.<br>1: Clears the BP8E bit in RSCR to 0. |
| 12 | RD8EC | 0 | –/W | 0: Does not clear the RD8E bit in RSCR to 0.<br>1: Clears the RD8E bit in RSCR to 0. |
| 11 | BP4EC | 0 | –/W | 0: Does not clear the BP4E bit in RSCR to 0.<br>1: Clears the BP4E bit in RSCR to 0. |
| 10 | RD4EC | 0 | –/W | 0: Does not clear the RD4E bit in RSCR to 0.<br>1: Clears the RD4E bit in RSCR to 0. |
| 9 | BP2EC | 0 | –/W | 0: Does not clear the BP2E bit in RSCR to 0.<br>1: Clears the BP2E bit in RSCR to 0. |
| 8 | RD2EC | 0 | –/W | 0: Does not clear the RD2E bit in RSCR to 0.<br>1: Clears the RD2E bit in RSCR to 0. |
| 7 | RT8EC | 0 | –/W | 0: Does not clear the RT8E bit in RSCR to 0.<br>1: Clears the RT8E bit in RSCR to 0. |
| 6 | NR8EC | 0 | –/W | 0: Does not clear the NR8E bit in RSCR to 0.<br>1: Clears the NR8E bit in RSCR to 0. |
| 5 | RT4EC | 0 | –/W | 0: Does not clear the RT4E bit in RSCR to 0<br>1: Clears the RT4E bit in RSCR to 0. |
| 4 | NR4EC | 0 | –/W | 0: Does not clear the NR4E bit in RSCR to 0.<br>1: Clears the NR4E bit in RSCR to 0. |
| 3 | RT2EC | 0 | –/W | 0: Does not clear the RT2E bit in RSCR to 0.<br>1: Clears the RT2E bit in RSCR to 0. |
| 2 | NR2EC | 0 | –/W | 0: Does not clear the NR2E bit in RSCR to 0.<br>1: Clears the NR2E bit in RSCR to 0. |
| 1 | RT1EC | 0 | –/W | 0: Does not clear the RT1E bit in RSCR to 0.<br>1: Clears the RT1E bit in RSCR to 0. |
| 0 | NR1EC | 0 | –/W | 0: Does not clear the NR1E bit in RSCT to 0.<br>1: Clears the NR1E bit in RSCR to 0. |

### 56.5.2.4 Rotator and Scaler Output Format Register (RSOFR)

This register is used to specify the pixel format in the three scalers within the extended function block. Set this register in accord with the combinations shown in Table 56.26 or Table 56.27. Use the CMRCR register to specify the pixel format for the rotator within the extended function block. When changing the setting of this register while the module is drawing data, execute the SYNCM instruction immediately before writing to this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | OFM8R | | | — | OFM4R | | | — | OFM2R | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | OFM8N | | | — | OFM4N | | | — | OFM2N | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

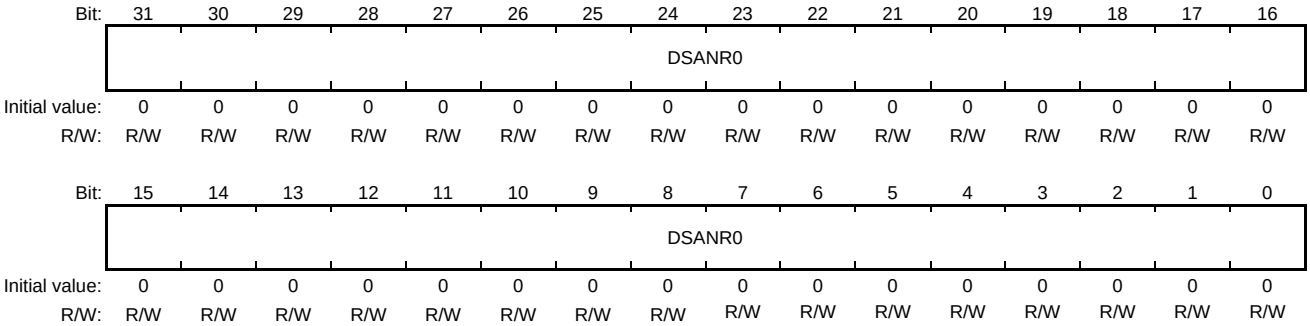
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 26 to 24 | OFM8R | 000 | R/W | These bits specify the format of the image rotated and vertically and horizontally scaled down to 1/8.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when RT8E bit in RSCR is 1. When RT8E and NR8E bits in RSCR are 1, set the same value in OFM8R and OFM8N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 23 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 22 to 20 | OFM4R | 000 | R/W | These bits specify the format of the image rotated and vertically and horizontally scaled down to 1/4.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when RT4E bit in RSCR is 1 and input to the scaler for the next stage of scaling. When RT4E and NR4E bits in RSCR are 1, set the same value in OFM4R and OFM4N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 19 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 18 to 16 | OFM2R | 000 | R/W | These bits specify the format of the image rotated and vertically and horizontally scaled down to 1/2.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when RT2E bit in RSCR is 1 and input to the scaler for the next stage of scaling. When RT2E and NR2E bits in RSCR are 1, set the same value in OFM2R and OFM2N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 15 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 to 8 | OFM8N | 000 | R/W | These bits specify the format of the non-rotated image vertically and horizontally scaled down to 1/8.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when NR8E bit in RSCR is 1. When RT8E and NR8E bits in RSCR are 1, set the same value in OFM8R and OFM8N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 7 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 6 to 4 | OFM4N | 000 | R/W | These bits specify the format of the non-rotated image vertically and horizontally scaled down to 1/4.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when NR4E bit in RSCR is 1 and input to the scaler for the next stage of scaling. When RT4E and NR4E bits in RSCR are 1, set the same value in OFM4R and OFM4N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 3 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

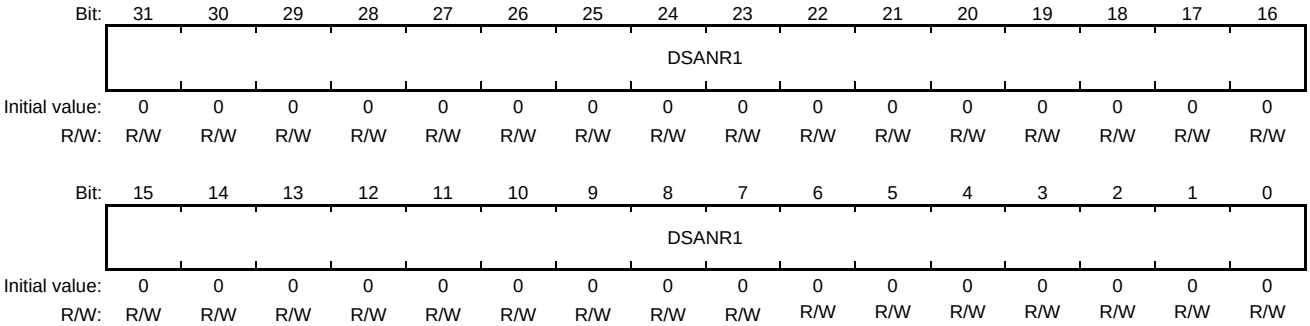
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 to 0 | OFM2N | 000 | R/W | These bits specify the format of the non-rotated image vertically and horizontally scaled down to 1/2.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when NR2E bit in RSCR is 1 and input to the scaler for the next stage of scaling. When RT2E and NR2E bits in RSCR are 1, set the same value in OFM2R and OFM2N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |

### 56.5.2.5 Destination Start Address for Non-Rotated Image Register 0 (DSANRR0)



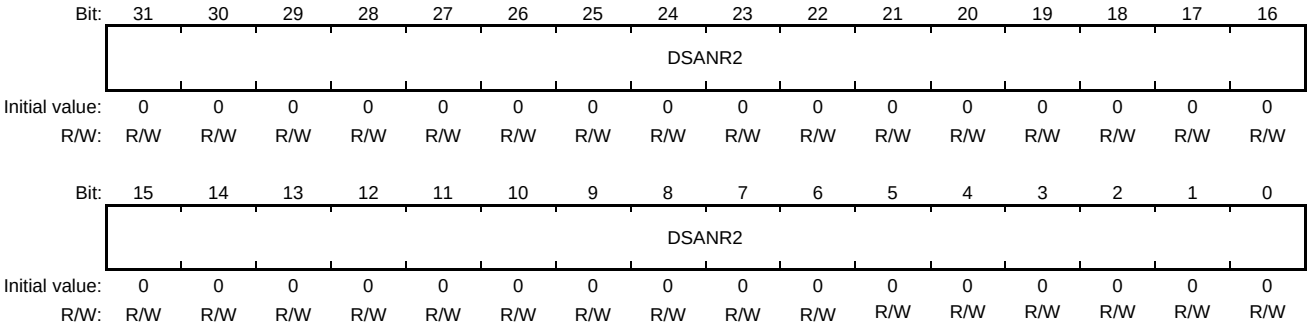

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSANR0 | All 0 | R/W | These bits specify the base address in bytes of the destination for the output of non-rotated images with unchanged size.<br>Setting should be aligned to 128-byte unit address for 16bpp and 64-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.6 Destination Start Address for Non-Rotated Image Register 1 (DSANRR1)



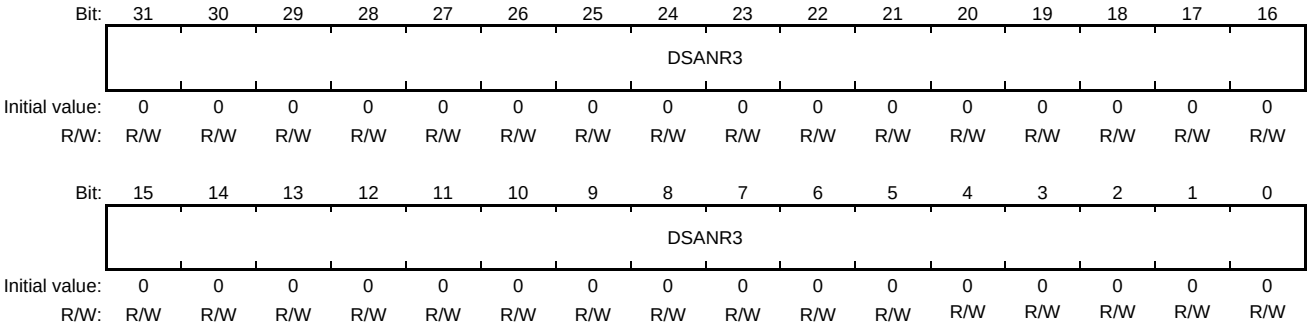

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSANR1 | All 0 | R/W | These bits specify the base address in bytes of the destination for non-rotated images horizontally and vertically scaled down to 1/2.<br>Setting should be aligned to 64-byte unit address for 16bpp and 32-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register |

### 56.5.2.7 Destination Start Address for Non-Rotated Image Register 2 (DSANRR2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSANR2 | All 0 | R/W | These bits specify the base address in bytes of the destination for non-rotated images horizontally and vertically scaled down to 1/4.<br>Setting should be aligned to 32-byte unit address for 16bpp and 16-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.8 Destination Start Address for Non-Rotated Image Register 3 (DSANRR3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSANR3 | All 0 | R/W | These bits specify the base address in bytes of the destination for non-rotated images horizontally and vertically scaled down to 1/8.<br>Setting should be aligned to 16-byte unit address for 16bpp and 8-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

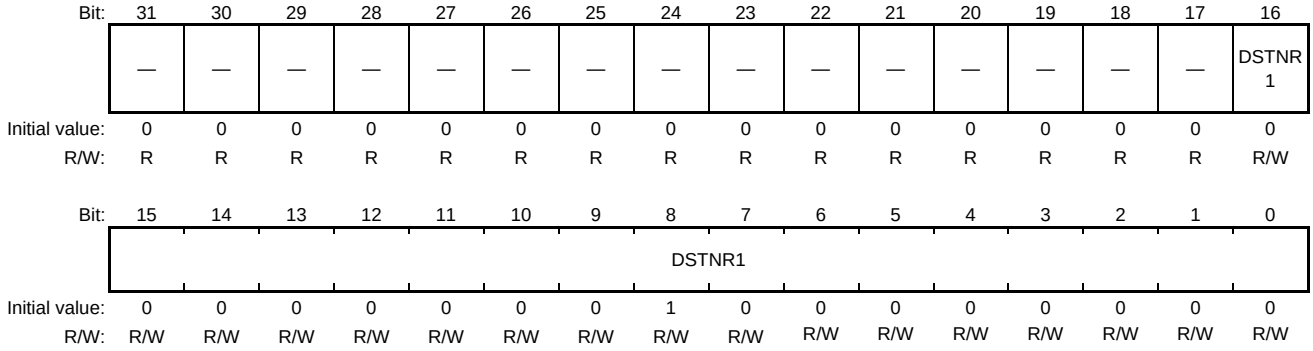
### 56.5.2.9 Destination Stride for Non-Rotated Image Register 0 (DSTNRR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTNR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTNR0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DSTNR0 | H'000200 | R/W | These bits specify the stride in bytes at the destination for the output of non-rotated images with unchanged size.<br>The minimum setting of this register is 64-byte and the maximum setting of this register is 65,536-byte. And the setting should be aligned to 128-byte unit for 16bpp and 64-byte unit for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.10 Destination Stride for Non-Rotated Image Register 1 (DSTNRR1)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DSTNR1 | H'000100 | R/W | These bits specify the stride in bytes at the destination for the output of non-rotated images horizontally and vertically scaled down to 1/2.<br>The minimum setting of this register is 32-byte and the maximum setting of this register is 65,536-byte. And the setting should be aligned to 64-byte unit for 16bpp and 32-byte unit for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

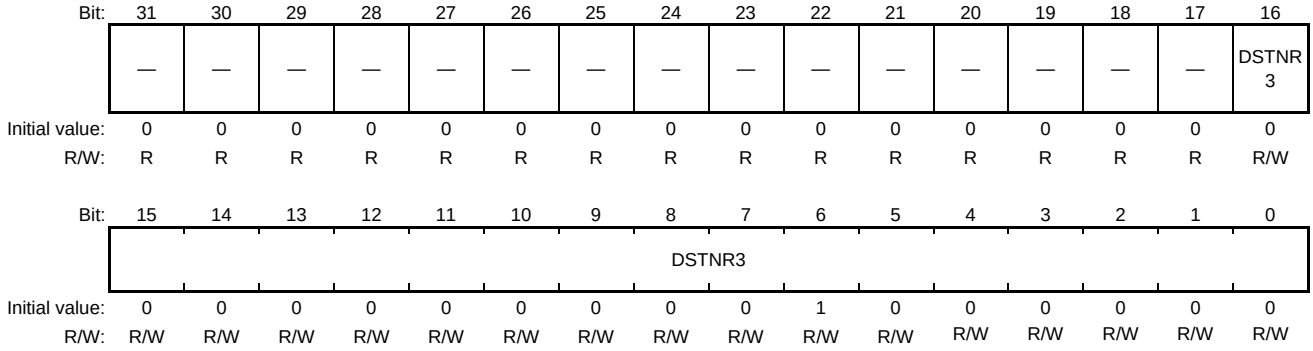
### 56.5.2.11 Destination Stride for Non-Rotated Image Register 2 (DSTNRR2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTNR2 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTNR2 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

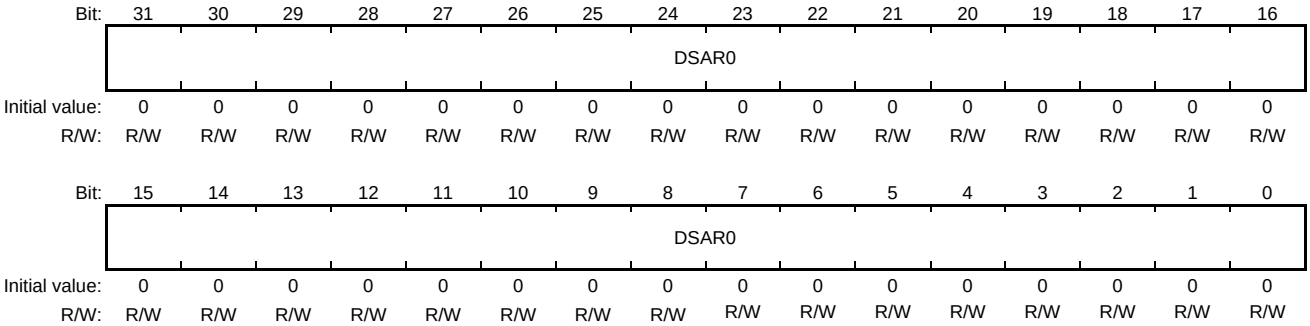
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DSTNR2 | H'00080 | R/W | These bits specify the stride in bytes at the destination for the output of non-rotated images horizontally and vertically scaled down to 1/4.<br>The minimum setting of this register is 16-byte and the maximum setting of this register is 65,536-byte. And the setting should be aligned to 32-byte unit for 16bpp and 16-byte unit for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.12 Destination Stride for Non-Rotated Image Register 3 (DSTNRR3)



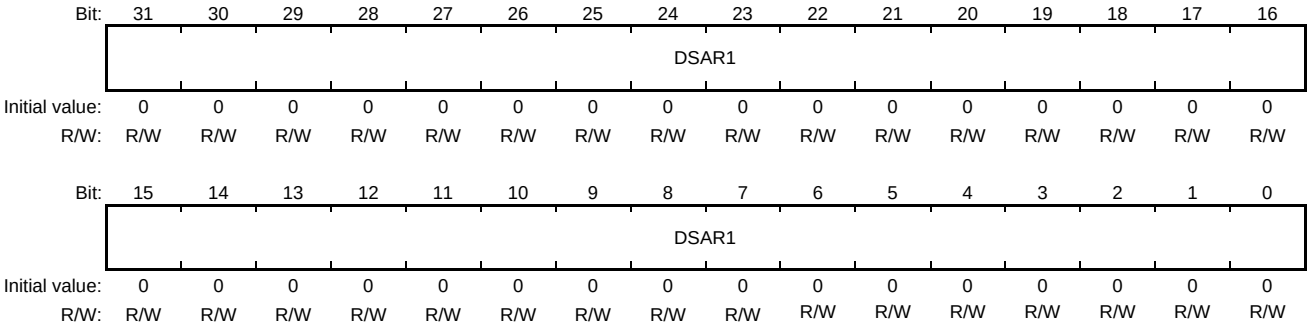

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DSTNR3 | H'00040 | R/W | These bits specify the stride in bytes at the destination for the output of non-rotated images horizontally and vertically scaled down to 1/8.<br>The minimum setting of this register is 8-byte and the maximum setting of this register is 65,536-byte. And the setting should be aligned to 16-byte unit for 16bpp and 8-byte unit for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.13 Destination Start Address for Rotated Image Register 0 (DSARR0)



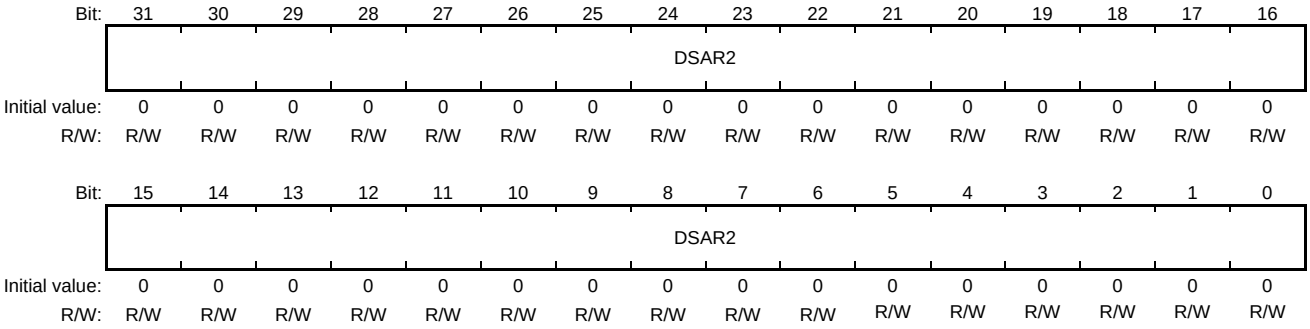

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSAR0 | All 0 | R/W | These bits specify the base address in bytes of the destination for the output of rotated images with unchanged size.<br>Setting should be aligned to 128-byte unit address for 16bpp and 64-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.14 Destination Start Address for Rotated Image Register 1 (DSARR1)



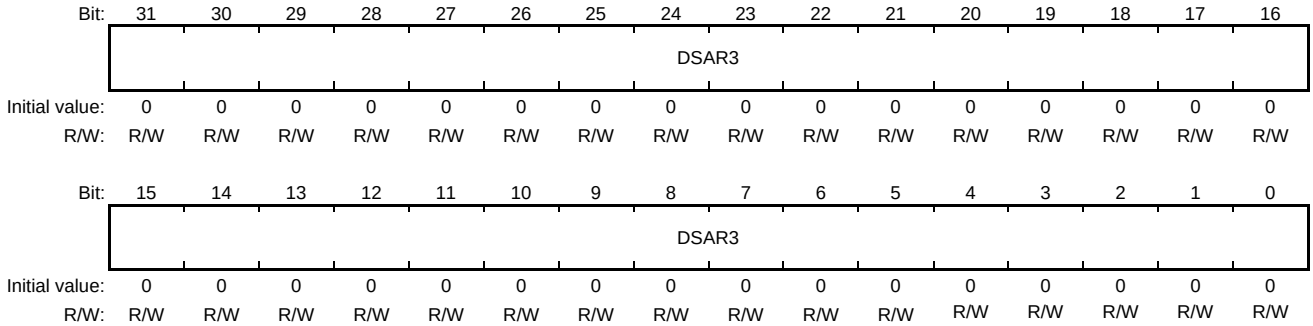

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSAR1 | All 0 | R/W | These bits specify the base address in bytes of the destination for the output of rotated images vertically and horizontally scaled down to 1/2.<br>Setting should be aligned to 64-byte unit address for 16bpp and 32-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.15 Destination Start Address for Rotated Image Register 2 (DSARR2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSAR2 | All 0 | R/W | These bits specify the base address in bytes of the destination for the output of rotated images vertically and horizontally scaled down to 1/4.<br>Setting should be aligned to 32-byte unit address for 16bpp and 16-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.16 Destination Start Address for Rotated Image Register 3 (DSARR3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSAR3 | All 0 | R/W | These bits specify the base address in bytes of the destination for the output of rotated images vertically and horizontally scaled down to 1/8.<br>Setting should be aligned to 16-byte unit address for 16bpp and 8-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.17 Destination Stride for Rotated Image Register 0 (DSTRR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTR0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DSTR0 | H'00100 | R/W | These bits specify the stride in bytes at the destination for the output of rotated images with unchanged size.<br>The minimum setting of this register is 64-byte and the maximum setting of this register is 65,536-byte. And the setting should be aligned to 128-byte unit for 16bpp and 64-byte unit for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.18 Destination Stride for Rotated Image Register 1 (DSTRR1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTR1 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTR1 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DSTR1 | H'00080 | R/W | These bits specify the stride in bytes at the destination for the output of rotated images horizontally and vertically scaled down to 1/2.<br>The minimum setting of this register is 32-byte and the maximum setting of this register is 65,536-byte.<br>And the setting should be aligned to 64-byte unit for 16bpp and 32-byte unit for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.19 Destination Stride for Rotated Image Register 2 (DSTRR2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTR2 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTR2 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DSTR2 | H'00040 | R/W | These bits specify the stride in bytes at the destination for the output of rotated images horizontally and vertically scaled down to 1/4.<br>The minimum setting of this register is 16-byte and the maximum setting of this register is 65,536-byte. And the setting should be aligned to 32-byte unit for 16bpp and 16-byte unit for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.20 Destination Stride for Rotated Image Register 3 (DSTRR3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DSTR3 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTR3 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 16 to 0 | DSTR3 | H'00020 | R/W | These bits specify the stride in bytes at the destination for the output of rotated images horizontally and vertically scaled down to 1/8.<br>The minimum setting of this register is 8-byte and the maximum setting of this register is 65,536-byte. And the setting should be aligned to 16-byte unit for 16bpp and 8-byte unit for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

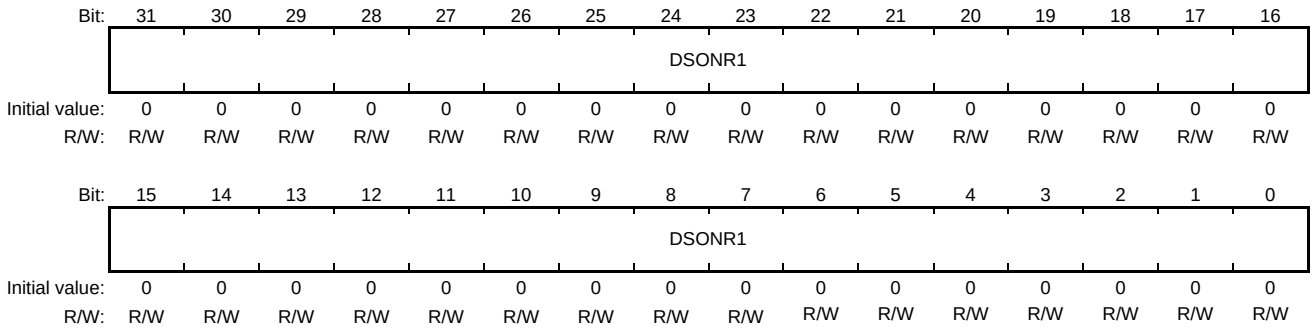
### 56.5.2.21 Destination Start Address Offset for Non-Rotated Image Register 0 (DSONRR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSONR0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSONR0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

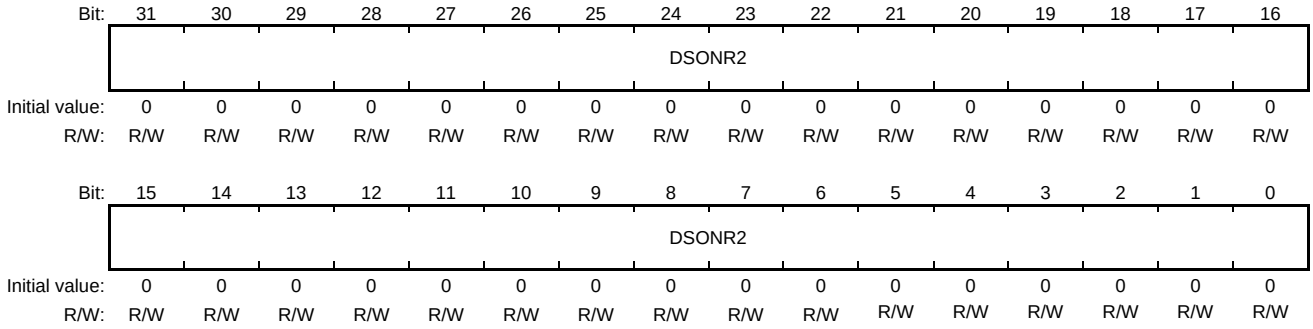
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSONR0 | All 0 | R/W | These bits specify the offset from the base address in bytes of the destination for the output of non-rotated images with unchanged size.<br>Setting should be aligned to 128-byte unit address for 16bpp and 64-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.22 Destination Start Address Offset for Non-Rotated Image Register 1 (DSONRR1)



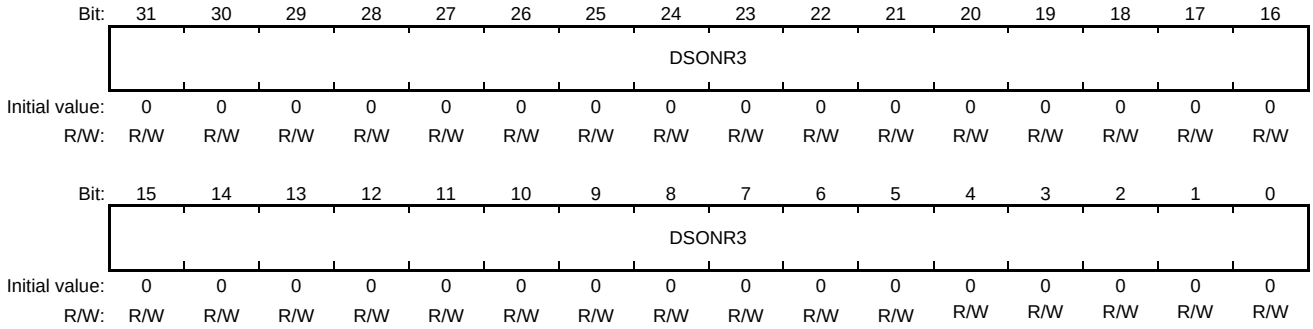

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSONR1 | All 0 | R/W | These bits specify the offset from the base address in bytes of the destination for the output of non-rotated images vertically and horizontally scaled down to 1/2.<br>Setting should be aligned to 64-byte unit address for 16bpp and 32-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.23 Destination Start Address Offset for Non-Rotated Image Register 2 (DSONRR2)



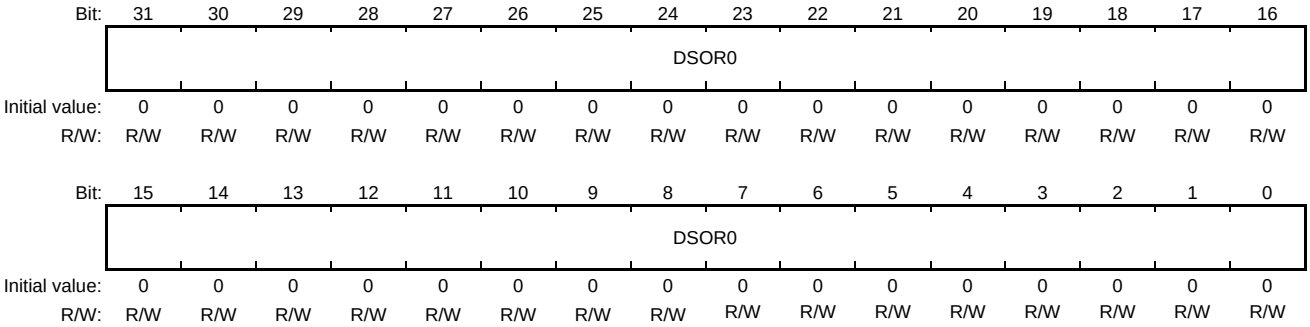

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSONR2 | All 0 | R/W | These bits specify the offset from the base address in bytes of the destination for the output of non-rotated images vertically and horizontally scaled down to 1/4.<br>Setting should be aligned to 32-byte unit address for 16bpp and 16-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.24 Destination Start Address Offset for Non-Rotated Image Register 3 (DSONRR3)



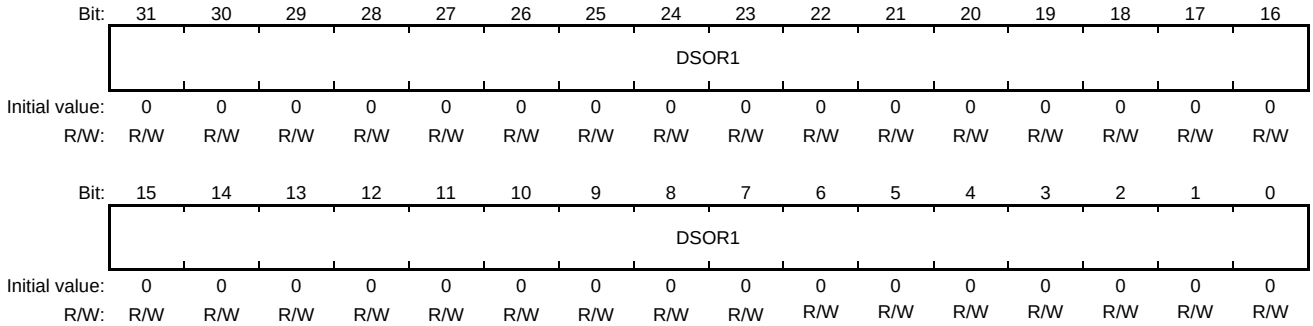

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSONR3 | All 0 | R/W | These bits specify the offset from the base address in bytes of the destination for the output of non-rotated images vertically and horizontally scaled down to 1/8.<br>Setting should be aligned to 16-byte unit address for 16bpp and 8-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.25 Destination Start Address Offset for Rotated Image Register 0 (DSORR0)



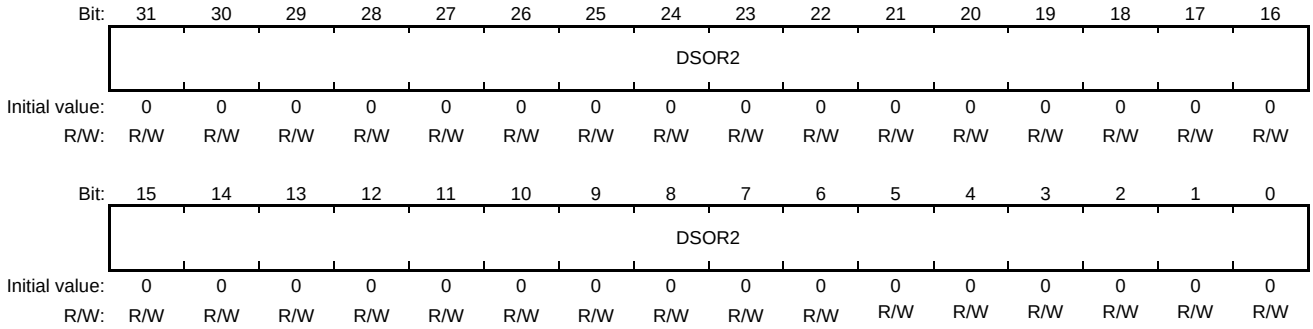

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSOR0 | All 0 | R/W | These bits specify the offset from the base address in bytes of the destination for the output of rotated images with unchanged size.<br>Setting should be aligned to 128-byte unit address for 16bpp and 64-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.26 Destination Start Address Offset for Rotated Image Register 1 (DSORR1)



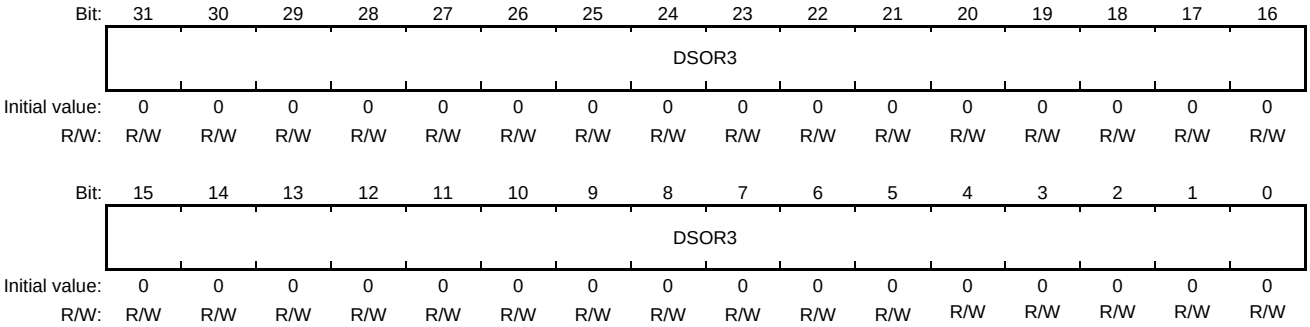

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSOR1 | All 0 | R/W | These bits specify the offset from the base address in bytes of the destination for the output of rotated images vertically and horizontally scaled down to 1/2.<br>Setting should be aligned to 64-byte unit address for 16bpp and 32-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.27 Destination Start Address Offset for Rotated Image Register 2 (DSORR2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSOR2 | All 0 | R/W | These bits specify the offset from the base address in bytes of the destination for the output of rotated images vertically and horizontally scaled down to 1/4.<br>Setting should be aligned to 32-byte unit address for 16bpp and 16-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.28 Destination Start Address Offset for Rotated Image Register 3 (DSORR3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSOR3 | All 0 | R/W | These bits specify the offset from the base address in bytes of the destination for the output of rotated images vertically and horizontally scaled down to 1/8.<br>Setting should be aligned to 16-byte unit address for 16bpp and 8-byte unit address for 8bpp.<br>When changing the setting of this bit field while the module is drawing data, execute the SYNCM instruction immediately before writing to this register. |

### 56.5.2.29 Gaussian Filter Control Register (GFCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SC8GSFHVAL | | | | SC4GSFHVAL | | | | SC2GSFHVAL | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | GRD 8E | GRD 4E | GRD 2E | UPE | DPE | LPE | RPE | — | — | — | GFE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 27 to 24 | SC8GSFHVAL | 4 | R/W | When calculating the Gaussian filter, this bit sets the number of digits of right-shifting after multiplying by the coefficient.<br>This setting is used for vertically and horizontally scaling down the input image to 1/8 of the original size. |
| 23 to 20 | SC4GSFHVAL | 4 | R/W | When calculating the Gaussian filter, this bit sets the number of digits of right-shifting after multiplying by the coefficient.<br>This setting is used for vertically and horizontally scaling down the input image to 1/4 of the original size. |
| 19 to 16 | SC2GSFHVAL | 4 | R/W | When calculating the Gaussian filter, this bit sets the number of digits of right-shifting after multiplying by the coefficient.<br>This setting is used for vertically and horizontally scaling down the input image to 1/2 of the original size. |
| 15 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 10 | GRD8E | 0 | R/W | Sets the handling of the value after the decimal point in the calculation result when calculating the Gaussian filter.<br>0: Truncates the value after the decimal point<br>1: Rounds off the value after the decimal point<br>This setting is used for vertically and horizontally scaling down the input image to 1/8 of the original size. |
| 9 | GRD4E | 0 | R/W | Sets the handling of the value after the decimal point in the calculation result when calculating the Gaussian filter.<br>0: Truncates the value after the decimal point<br>1: Rounds off the value after the decimal point<br>This setting is used for vertically and horizontally scaling down the input image to 1/4 of the original size. |
| 8 | GRD2E | 0 | R/W | Sets the handling of the value after the decimal point in the calculation result when calculating the Gaussian filter.<br>0: Truncates the value after the decimal point<br>1: Rounds off the value after the decimal point<br>This setting is used for vertically and horizontally scaling down the input image to 1/2 of the original size. |
| 7 | UPE | 0 | R/W | 0: Does not apply padding to the top of the image<br>1: Applies padding to the top of the image if the Gaussian filter is applied<br>This bit can be set to 1 only when the GFE bit of this register is 1. Otherwise, clear it to 0. |
| 6 | DPE | 0 | R/W | 0: Does not apply padding to the bottom of the image<br>1: Applies padding to the bottom of the image if the Gaussian filter is applied<br>This bit can be set to 1 only when the GFE bit of this register is 1. Otherwise, clear it to 0. |

| | | | | |
|---|---|---|---|---|
| 5 | LPE | 0 | R/W | 0: Does not apply padding to the left edge of the image<br>1: Applies padding to the left edge of the image if the Gaussian filter is applied<br>This bit can be set to 1 only when the GFE bit of this register is 1. Otherwise, clear it to 0. |
| 4 | RPE | 0 | R/W | 0: Does not apply padding to the right edge of the image<br>1: Applies padding to the right edge of the image if the Gaussian filter is applied<br>This bit can be set to 1 only when the GFE bit of this register is 1. Otherwise, clear it to 0. |
| 3 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 0 | GFE | 0 | R/W | Gaussian Filter Enable<br>0: 2x2 averaging filter for scaling-down<br>1: 3x3 Gaussian filter for scaling-down<br>This bit can be set to 1 only when processing Y plane. When processing UV plane, clear it to 0.<br>This bit should be set before starting to render using the TRI command, to which the desired filter is expected to be applied. It is prohibited to change the setting within one divided area. |

### 56.5.2.30 Rotator and Scaler Control Register 2 (RSCR2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | AFB8E | AFB4E | AFB2E | PRCM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 3 | AFB8E | 0 | R | This bit controls whether the 2x2 averaging filter is bypassed or not.<br>0: Bypass not allowed<br>1: Bypass allowed<br>This setting is used for vertically and horizontally scaling down the input image to 1/8 of the original size.<br>To enable the Gaussian filter, set this bit to 1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | AFB4E | 0 | R | This bit controls whether the 2x2 averaging filter is bypassed or not.<br>0: Bypass not allowed<br>1: Bypass allowed<br>This setting is used for vertically and horizontally scaling down the input image to 1/4 of the original size.<br>To enable the Gaussian filter, set this bit to 1. |
| 1 | AFB2E | 0 | R | This bit controls whether the 2x2 averaging filter is bypassed or not.<br>0: Bypass not allowed<br>1: Bypass allowed<br>This setting is used for vertically and horizontally scaling down the input image to 1/2 of the original size.<br>To enable the Gaussian filter, set this bit to 1. |
| 0 | PRCM | 0 | R | This bit sets the number of pixels to be processed per 1 cycle.<br>0: 1 pixel/cycle<br>1: 2 pixel/cycle<br>Setting this bit does not guaranteed that overall processing speed of IMR-LX6 is double in case of Rotator & Scalar.<br>SYNCM should be executed immediately before rewriting this bit. |

### 56.5.2.31 Rotator and Scaler Control Set Register 2 (RSCSR2)

This register is used to allow or disallow setting of the corresponding bits in the rotator and scaler control register 2 (RSCR2) to 1. Before setting a bit in RSCR2 to 1, write 1 to the corresponding bit in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 1w5 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | AFB8ES | AFB4ES | AFB2ES | PRCMS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | -/W | -/W | -/W | -/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 3 | AFB8ES | 0 | -/W | 0: The AFB8E bit in RSCR2 is not set to 1.<br>1: The AFB8E bit in RSCR2 is set to 1. |
| 2 | AFB4ES | 0 | -/W | 0: The AFB4E bit in RSCR2 is not set to 1.<br>1: The AFB4E bit in RSCR2 is set to 1. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | AFB2ES | 0 | -/W | 0: The AFB2E bit in RSCR2 is not set to 1.<br>1: The AFB2E bit in RSCR2 is set to 1. |
| 0 | PRCMS | 0 | -/W | 0: The PRCM bit in RSCR2 is not set to 1<br>1: The PRCM bit in RSCR2 is set to 1. |

### 56.5.2.32 Rotator and Scaler Control Clear Register 2 (RSCCR2)

This register is used to allow or disallow clearing of the corresponding bits in the rotator and scaler control register 2 (RSCR2). Before clearing a bit in RSCR2, write 1 to the corresponding bit in this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | AFB8EC | AFB4EC | AFB2EC | PRCMC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | -/W | -/W | -/W | -/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 3 | AFB8EC | 0 | -/W | 0: The AFB8E bit in RSCR2 is not clear to 0.<br>1: The AFB8E bit in RSCR2 is clear to 0. |
| 2 | AFB4EC | 0 | -/W | 0: The AFB4E bit in RSCR2 is not clear to 0.<br>1: The AFB4E bit in RSCR2 is clear to 0. |
| 1 | AFB2EC | 0 | -/W | 0: The AFB2E bit in RSCR2 is not clear to 0.<br>1: The AFB2E bit in RSCR2 is clear to 0. |
| 0 | PRCMC | 0 | -/W | .0: The PRCM bit in RSCR2 is not clear to 0.<br>1: The PRCM bit in RSCR2 is clear to 0. |

### 56.5.2.33 Gaussian Filter Output Format Register (GFOFR)

This register is used to specify the pixel format in the three scalers within the extended function block. Set this register in accord with the combinations shown in Table 56.26 or Table 56.27. Use the CMRCR register to specify the pixel format for the rotator within the extended function block. When changing the setting of this register while the module is drawing data, execute the SYNCM instruction immediately before writing to this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | GFM8R | | | — | GFM4R | | | — | GFM2R | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | GFM8N | | | — | GFM4N | | | — | GFM2N | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 26 to 24 | GFM8R | 000 | R/W | These bits specify the format of the image rotated and vertically and horizontally scaled down to 1/8 after the Gaussian Filter.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down of Gaussian Filter exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when RT8E bit in RSCR is 1. When RT8E bit in RSCR is 1, set the same value in OFM8R bits in RSOFR and GFM8R.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 23 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 22 to 20 | GFM4R | 000 | R/W | These bits specify the format of the image rotated and vertically and horizontally scaled down to 1/4 after the Gaussian Filter.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down of Gaussian Filterexceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when RT4E bit in RSCR is 1 and input to the scaler for the next stage of scaling. When RT4E bit in RSCR is 1, set the same value in OFM4R bits in RSOFR and GFM4R.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 19 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 18 to 16 | GFM2R | 000 | R/W | These bits specify the format of the image rotated and vertically and horizontally scaled down to 1/2 after the Gaussian Filter.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down of Gaussian Filter exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when RT2E bit in RSCR is 1 and input to the scaler for the next stage of scaling. When RT2E bit in RSCR is 1, set the same value in OFM2R bits in RSOFR and GFM2R.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 15 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 to 8 | GFM8N | 000 | R/W | These bits specify the format of the non-rotated image vertically and horizontally scaled down to 1/8 after the Gaussian Filter.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down of Gaussian Filter exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when NR8E bit in RSCR is 1. When NR8E bit in RSCR is 1, set the same value in OFM8N bits in RSOFR and GFM8N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 7 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |
| 6 to 4 | GFM4N | 000 | R/W | These bits specify the format of the non-rotated image vertically and horizontally scaled down to 1/4 after the Gaussian filter.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down of Gaussan filter exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when NR4E bit in RSCR is 1 and input to the scaler for the next stage of scaling. When NR4E bit in RSCR is 1, set the same value in OFM4N bits in RSOFR and GFM4N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |
| 3 | — | 0 | R | Reserved<br>This bit is always read as 0.<br>The write value should always be 0. Otherwise, the operation cannot be guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 to 0 | GFM2N | 000 | R/W | These bits specify the format of the non-rotated image vertically and horizontally scaled down to 1/2 after the Gaussianfilter.<br>0: 8 bpp<br>1: 10 bpp<br>2: 12 bpp<br>3: 14 bpp<br>4: 16 bpp<br>Others: Setting prohibited<br>When the pixel value calculated in scaling down of Gaussian filter exceeds the range of the format specified in this bit field, the calculated value is saturated to the maximum range of the format specified in this bit field. The saturated pixel value is output to the external memory when NR2E bit in RSCR is 1 and input to the scaler for the next stage of scaling. When NR2E bit in RSCR is 1, set the same value in OFM2N bits in RSOFR and GFM2N.<br>The output image is written to external memory in 1-byte/pixel units if the format setting is 8 bpp, and in 2-byte/pixel units for the other format settings. The output values are flushed to the LSB with all format settings. |

### 56.5.3 Operation

#### 56.5.3.1 Output of the non-rotated or rotated image data

This section explains how to use the extended function blocks on the condition that an executable display list is present for the basic function block. For usage of the basic function block, see sections 56.1 through 56.4.

The settings of some registers are common to the basic and extended function blocks. For example, switching between the Y processing mode and UV processing mode is handled by the YCM bit in CMRCR. The formats of input and output pixels in the rotator and of input pixels in scaler 1 in Figure 56.36 are set by the DUV1416, DY1416, Y12, Y10, and DUV fields in CMRCR.

To use the extended function blocks, set the RSE bit in RSCR to 1.

Note: The number of pixels of the area and the sub-area described in the following section indicates the number of data of the plane to be processed. For example, when processing the UV plane in YUV420 format, the data in one line of the UV plane corresponds to one pixel in the vertical direction of the plane as described in the following section. The line does not correspond to one pixel in the vertical direction of the Y plane.

**(1) Dividing up the Image to be Processed**

The image dividing method differs depending on the filtering method used for the scaling-down function which will described later in the document. This section first describes the basic 2x2 averaging filter and then its derivation, the 3x3 Gaussian filter. The Gaussian filter can be used only when processing Y plane. It cannot be used when processing UV plane.

**(a) Dividing method when applying the 2x2 averaging filter**

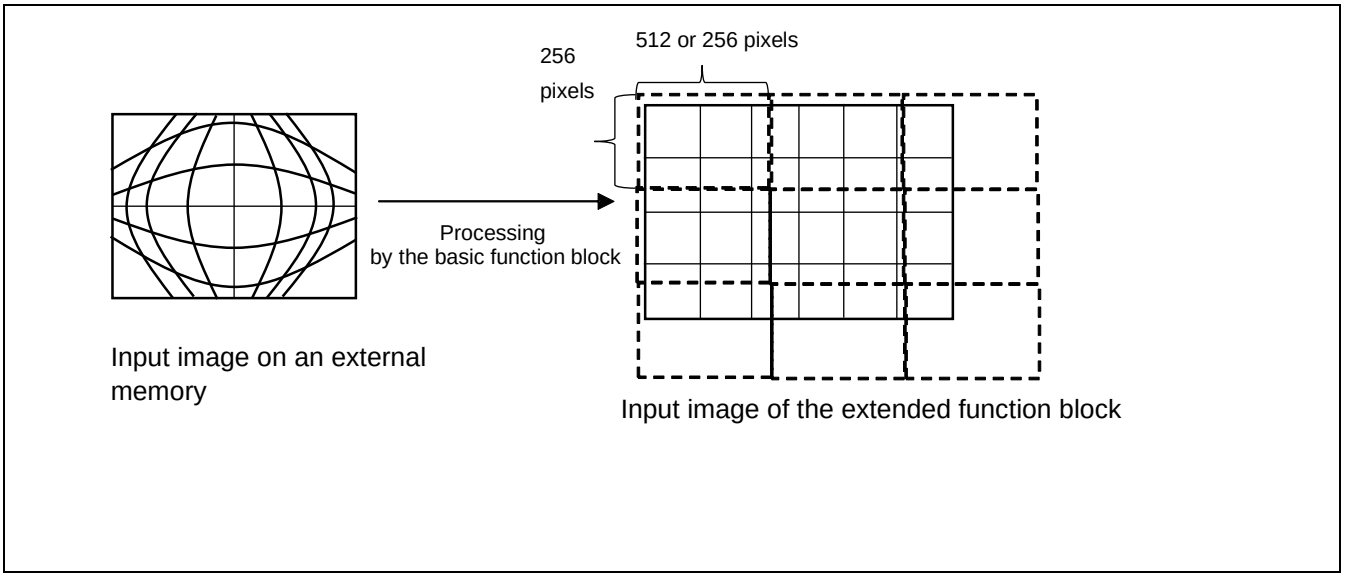



**Figure 56.37 Dividing up the Image to be Processed when the Extended Functions are in Use**

The function blocks of the extension can handle images with up to 512 pixels horizontally x 256 vertically*1 or with up to 256 pixels horizontally x 256 vertically*2. Accordingly, if the image to be processed in the basic function block exceeds these maximum numbers of pixels, the image has to be divided up.

Figure 56.37 shows the concept of dividing up the image. The dotted rectangles on the input image of the extended function block indicate the maximum size the extended function block can handle. In this example, the input image of the extended function block is divided into nine areas. Here, the display list for processing

by the basic function block must be created such that drawing proceeds in order through the units of the rectangular regions indicated by the broken lines. Drawing operations that straddle boundaries between the rectangular regions are prohibited.

Before starting to draw in each area, set the registers DSANRRn, DSTNRRn, DSARRn, DSTRRn, DSONRRn, and DSORRn so that each area is drawn in the desired location. Also set the NRnE and RTnE bits in RSCR as required. For details, see the descriptions of the respective registers.

Notes: 1. When the input from the basic function block is 8 bpp and all of the outputs from the extended function block are 8 bpp.
2. When the input from the basic function block is more than 8 bpp or any of the outputs from the extended function block is more than 8 bpp.

**(b) Dividing method when applying the 3x3 Gaussian filter**

The size of the output image after being applied the 3x3 Gaussian filter is one pixel smaller at each border per down-scale level. Therefore, for the image dividing process that requires this extended function, an overlapping area is required between the divided images.

The dividing process is as below:

1. Divide the entire image into 512 x 256 (8bpp) or 256 x 256 (others) similar to when the 2x2 averaging filter is applied[*1]. If there is any area smaller than 512 x 256 or 256 x 256 left, put it at the right bottom edge.
2. Expand the up, down, left, right edges of that area by 7 pixels and take it as the divided area. However, the areas that touch the top, bottom, left and right edges of the divided image are not expanded.

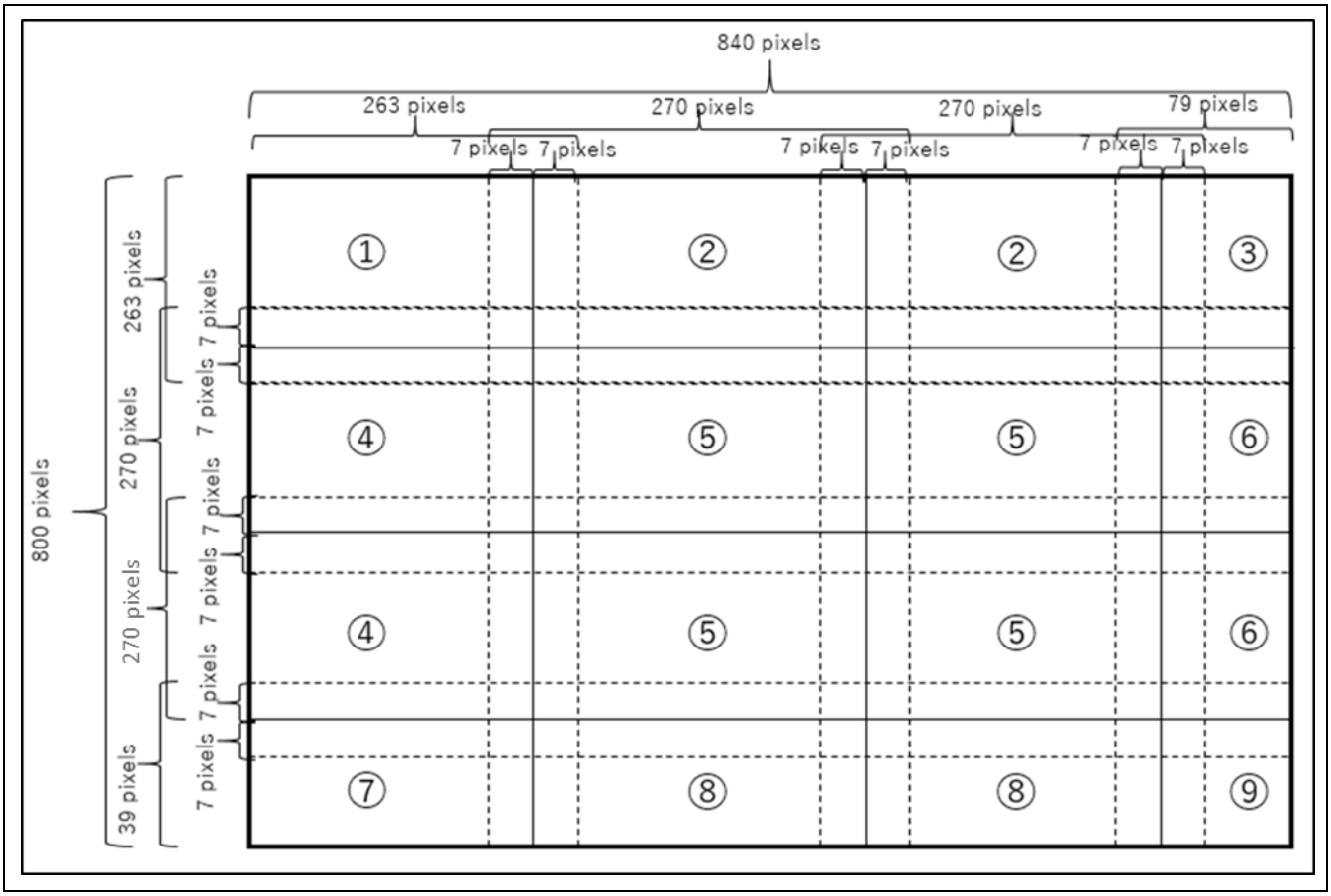



**Figure 56.38 Example of dividing an 840x800 image (2-Byte/pixel) using 3x3 Gaussian filter**

The figure above shows an example of dividing an input image with 2-Byte/pixel. 9 areas are created depending on the size after dividing. For example, after ⑤ is divided into 256x256 areas, expand all the up, down, left, right edges by 7 pixels, each area will increase to the size of 270x270. Additionally, ⑤ has areas that overlap with all areas that touch the up, down, left, right edges. On the other hand, in ⑨, dividing 840x800 area by 256 both vertically and horizontally results in 72x32. The right and bottom edges touch the outer circumference of the divided image and will not be expanded. The left and top edges will be expanded by 7 pixels and the size of the area increases to 79x39.

### (2) Further dividing up the first area following initial division

The method to divide into sub-areas described in this section is also different depending on whether the 2x2 averaging filter or 3x3 Gaussian filter is used. First, the section describes the dividing method when the 2x2 averaging filter is used and then the method when the 3x3 Gaussian filter is used.

**(a) Dividing method when using 2x2 averaging filter**

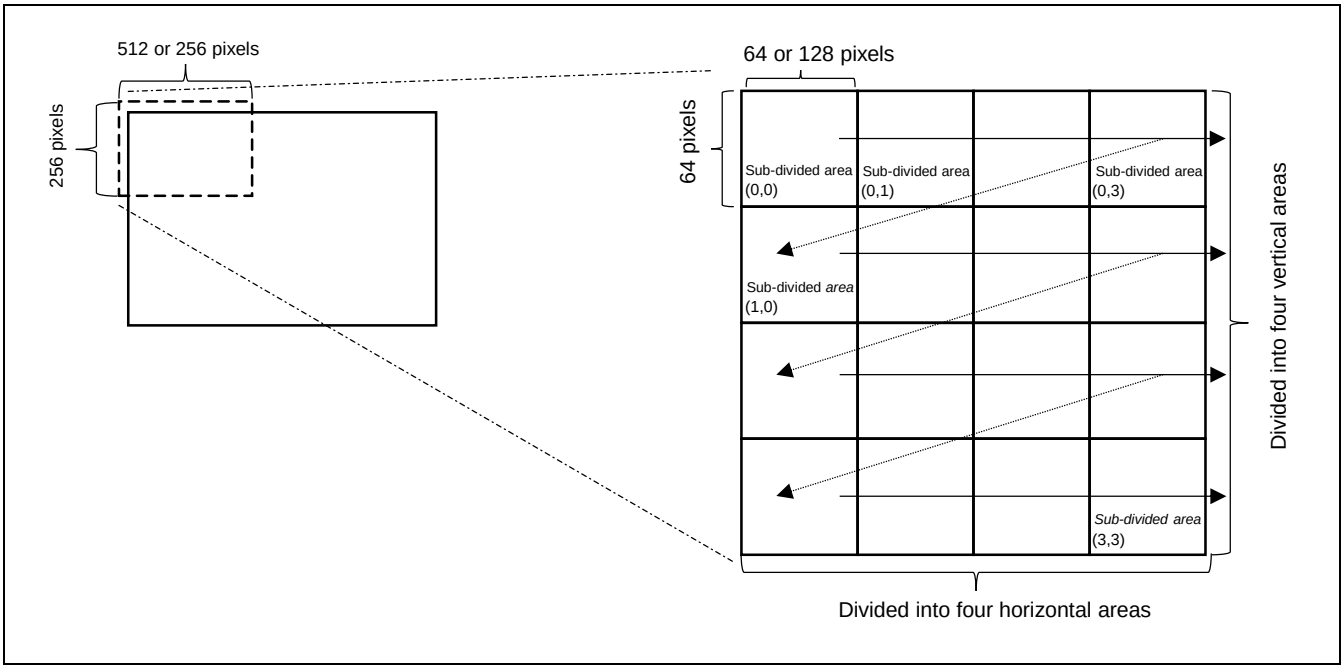



**Figure 56.39 Further Dividing the First Area Following Initial Division**

For the processing in the extended function block, the divided areas need to be further divided into four sub-divided areas vertically and horizontally.

Figure 56.39 shows the concept of further dividing the first divided area. Each divided area is further divided into sub-divided areas with 128 horizontal x 64 vertical pixels (when the input from the basic function block is 8 bpp) or 64 horizontal x 64 vertical pixels (when the input from the basic function block is not 8 bpp). If input from the basic function block is 8 bpp and any output of the extended function block is other than 8 bpp, usage with the 256-pixel horizontally x 256-pixel vertically area horizontally divided in two and vertically divided into four so that there are eight small areas in all is possible. The size of each small area thus becomes 128 x 64. In other cases, usage with the 512 x 256 or 256 x 256 area horizontally and vertically divided into four so that there are 16 small areas is possible. The explanation below is on the assumption of horizontal and vertical division into four.

You should organize the display list for the basic function block so that it specifies the image to be drawn in the 128 x 64 or 64 x 64 units of the sub-divided areas. It also should specify the X and Y coordinates of the top left of the sub-divided areas to be (0,0). The u and v values do not require a change.

In general, drawing should start from the top left and proceed along the X-axis in the positive direction, and then along the Y-axis in the positive direction, and so on (as indicated by the arrows in Figure 56.39). But, if necessary, user can change the drawing order, it is user responsibility to confirm the rendering is finished for the target ROW or COLUMN before OUTNOR or OUTROT instructions are executed.

Note: Figure 56.39 shows further dividing the first area following initial division except for processing the UV plane in YUV420 format, For UV plane in YUV420 format refer to Figure 56.40.

When processing the UV plane in YUV420 format, as described in Figure 56.40 for the sub-area of 4 rows, only 2 rows from the top can be used. Therefore, the output data in one line corresponds to the image of two pixels in vertical direction, the number of lines in each divided area is 128 lines. Please refer to the Table 56.13, Note 6 before creating the TRI command in DL for processing of the UV plane in YUV420 format in Rotator & Scaler extension.

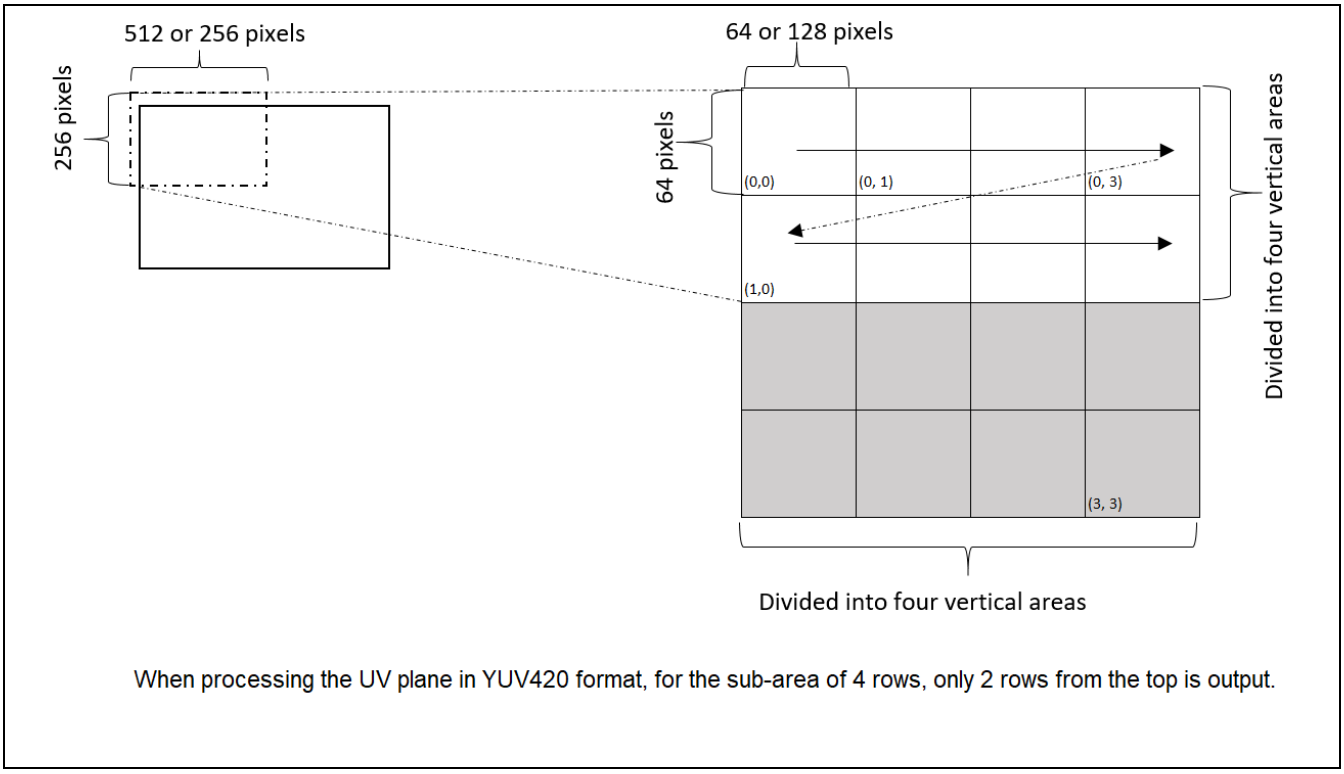



**Figure 56.40 Further Dividing the First Area Following Initial Division When processing the UV plane in YUV420 format**

### (b) Dividing method when using 3x3 Gaussian filter

When applying 3x3 Gaussian filter, the method to divide into 4x4 sub-areas is similar to when applying 2x2 averaging filter. However, the size of each sub-area is different. The size also varies depending on the position of the area to be divided into sub-areas in the entire output image. The rendering order is the same as when using the 2x2 averaging filter.

The content below explains the basic case in which the area does not touch the outer circumference of the entire output image, and then the case in which it touches each of four edges of the outer circumference.

In either case, regardless of whether there are any empty area, the origin of the x/y coordinate to be rendered is the upper left end of the sub-area specified by STRECT.

#### If the area does not touch the outer circumference

This section describes the division when the area to be divided does not touch any of the four edges of the entire output image. It corresponds to area ⑤ in Figure 56.38.

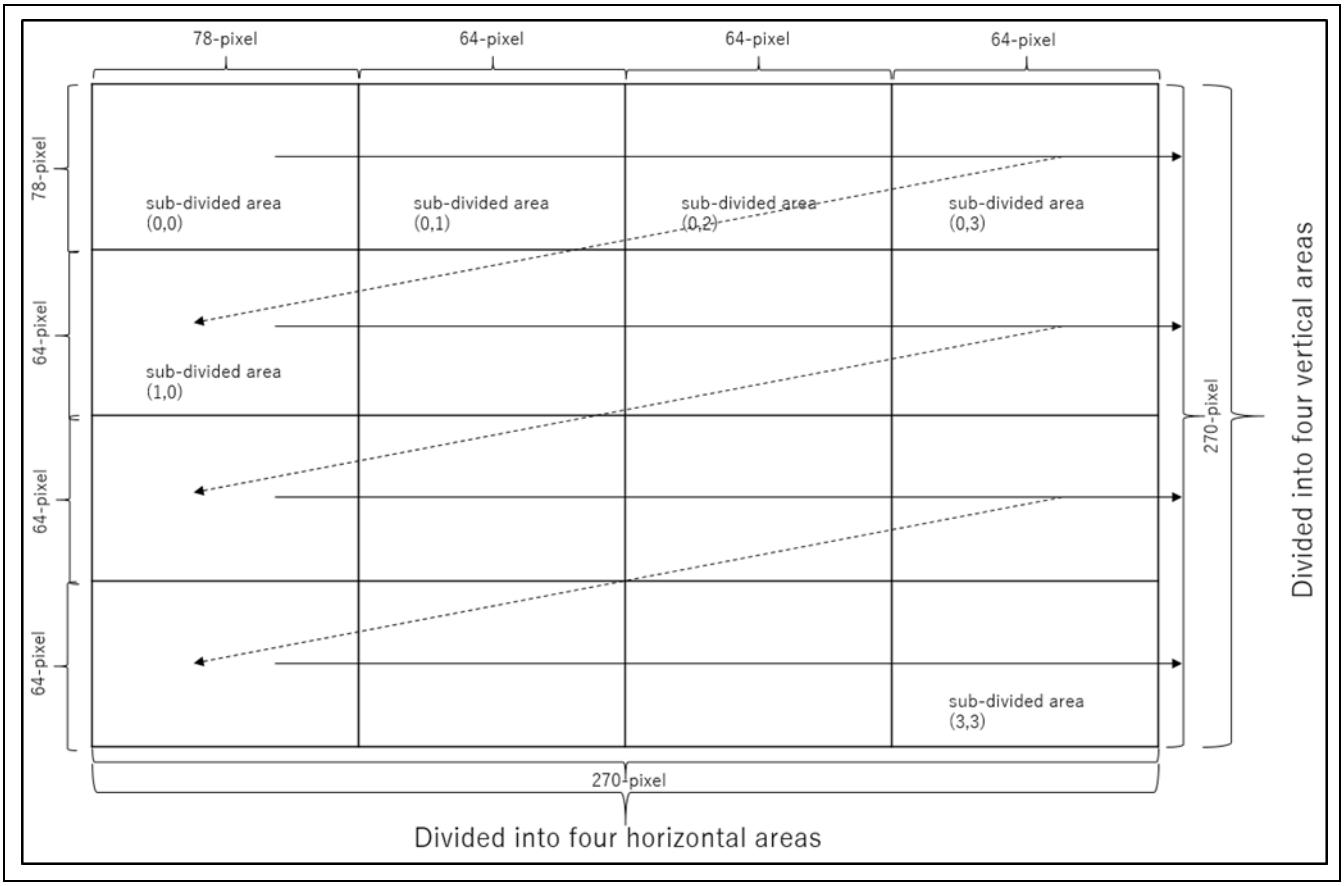



**Figure 56.41 Dividing into sub-areas – If the area does not touch any of the four edges of the output image (2-Byte/pixel)**

Figure 56.41 shows the division into sub-areas when the pixel input from the basic function block is 2-Byte/pixel. The 270x270 area is divided into 4x4 sub-areas. The height of the sub-area in the top-edge rows is 78 pixels, the width of the sub-area in the left-edge column is 78 pixels, the height and width of other small areas are both 64 pixels.

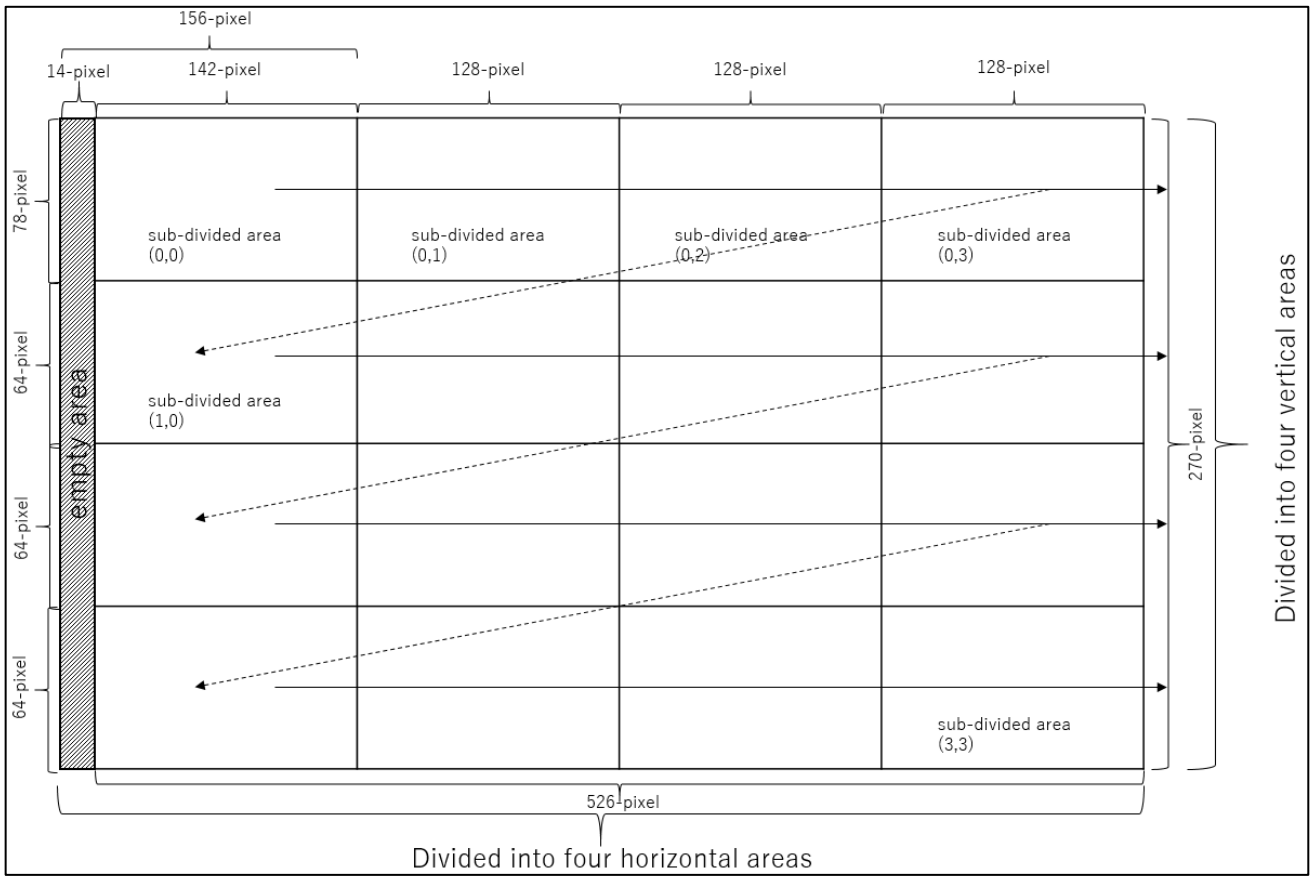



**Figure 56.42 Dividing into sub-areas – If the area does not touch any of the four edges of the output image (1-Byte/pixel)**

Figure 56.42 shows the division when the pixel input from the basic function block is 1-Byte/pixel. The 526x270 area is divided into 4x4 small areas. The width of the sub-area in the top-edge row is 142 pixels, the height of the sub-areas in the left-edge column is 78 pixels. The width and height of other sub-areas is 128 pixels and 64 pixels, respectively.

Besides, an empty area with a width of 14 pixels is generated at the left edge of the entire area (the shaded area). When specifying to render one of these sub-areas (0, 0), (1, 0), (2, 0), (3, 0) using the STRECT command, which is described later, the coordinate with the x coordinate corresponding from 0 to 13 (the place indicated by "empty area" in Figure 56.42) should not be written to and also should not be rendered.

#### When the area touches the upper edge

This section describes the division when the target area to be divided into sub-areas touches the upper edge of the entire output image. It corresponds to sub-areas ①, ②, ③ in Figure 56.38.

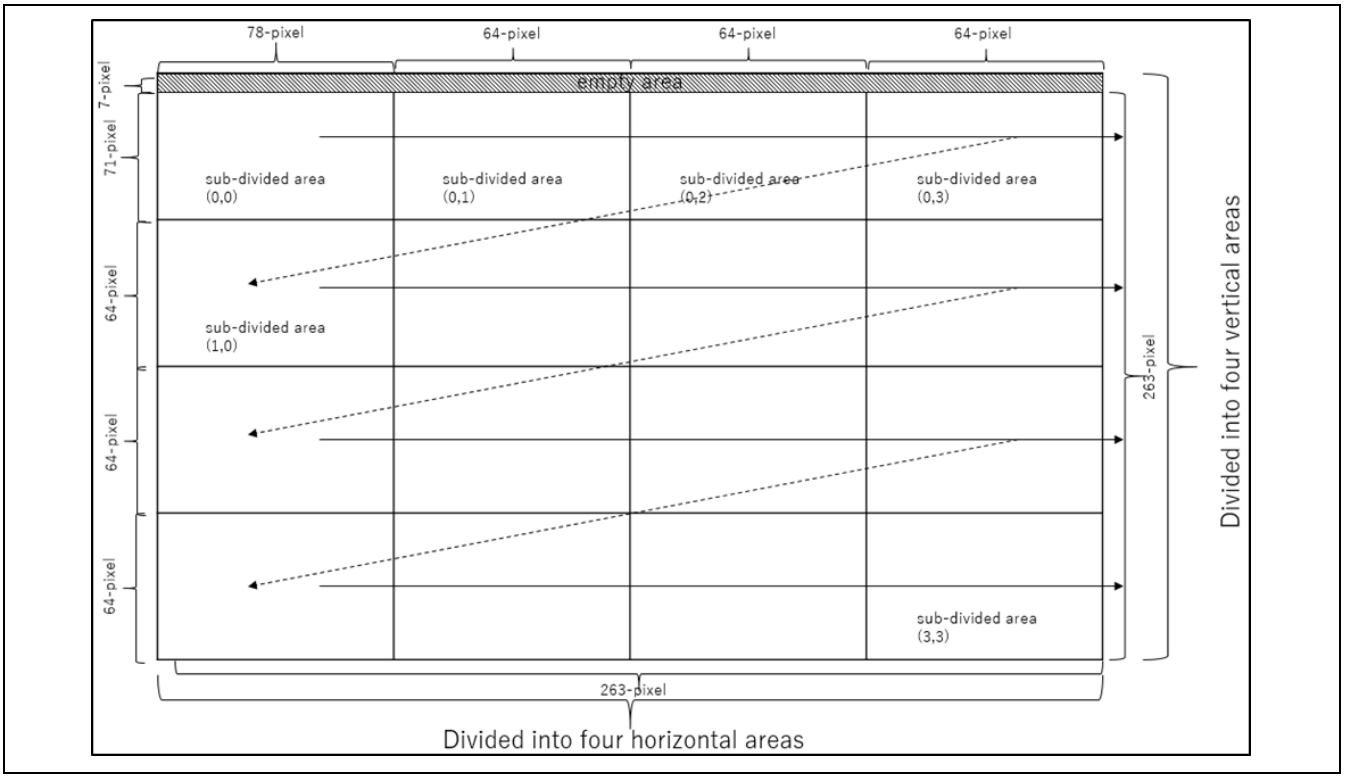



**Figure 56.43 Dividing into sub-areas – When the area touches the upper edge of the output image (2-Byte/pixel)**

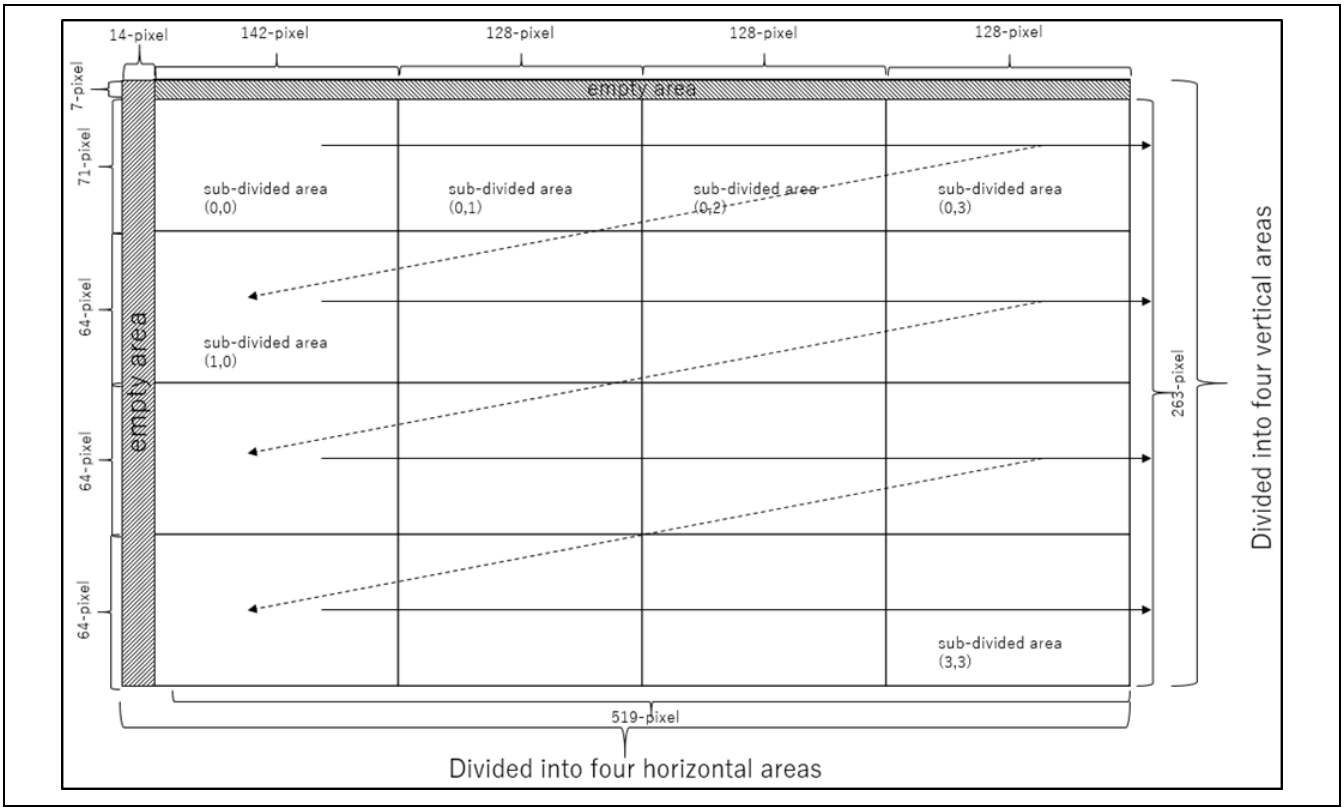



**Figure 56.44 Dividing into sub-areas – When the area touches the upper edge of the output image (1-Byte/pixel)**

Figure 56.43 shows the division when it touches the upper edge and the pixel input from the basic function block is 2-Byte/pixel. And Figure 56.44 shows the division when the area touches the upper edge and the pixel input from the basic function block is 1-Byte/pixel.

When the area touches the upper edge, an empty area with a height of 7 pixels will appear at the top edge of the entire area (the shaded area). Therefore, when specifying to render one of these sub-areas (0, 0), (0, 1), (0, 2), (0, 3) with STRECT, which is described later, the coordinate with the y coordinate corresponding from 0 to 6 (the place indicated by "empty area" at the top of Figure 56.43 and Figure 56.44) should not be written to and also should not be rendered. Also, similar to the case when the area does not touch the outer circumference, and the pixel input from the basic function block is 1-Byte/pixel, if specifying to render one of these sub-areas (0, 0), (1, 0), (2, 0), (3, 0) using the STRECT command, which is described later, the coordinate with the x coordinate corresponding from 0 to 13 (the place indicated by "empty area" at the left of Figure 56.44) should not be written to and also should not be rendered.

#### When the area touches the lower edge

This section describes the division when the target area to be divided into sub-areas touches the lower edge of the entire output image. It corresponds to small areas ⑦, ⑧, ⑨ in Figure 56.38.

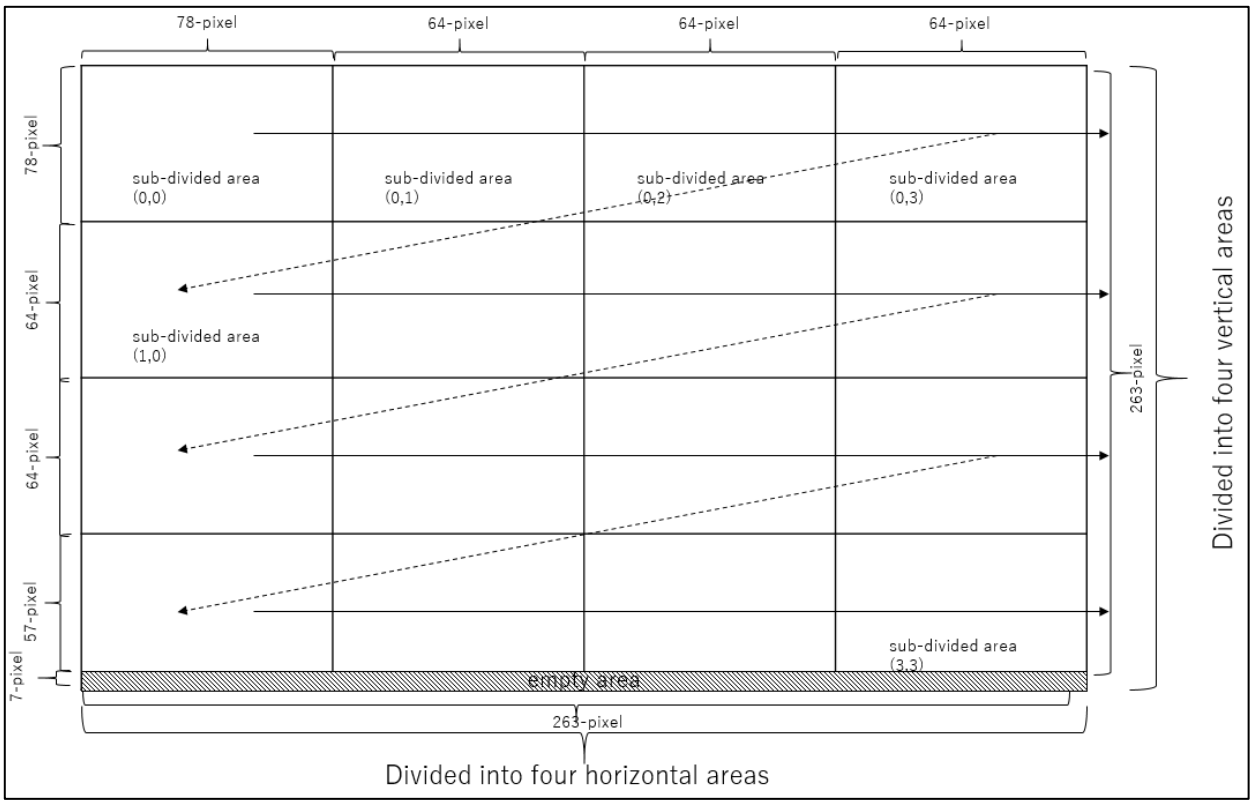



**Figure 56.45 Dividing into sub-areas – When the touches the lower edge of the output image (2-Byte/pixel)**

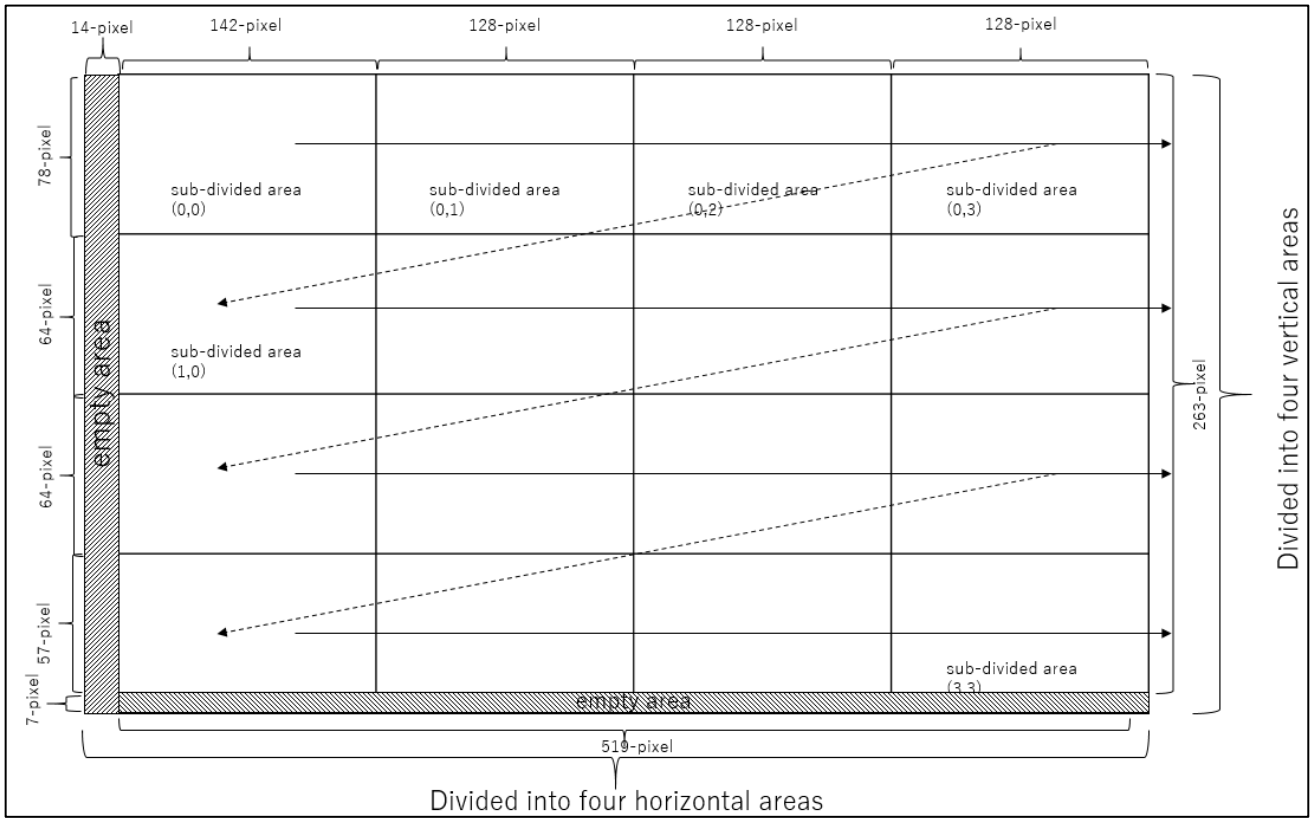



**Figure 56.46 Dividing into sub-areas – When the touches the lower edge of the output image (1-Byte/pixel)**

Figure 56.45 shows the division when the area touches the lower edge and the pixel input from the basic function block is 2-Byte/pixel. And Figure 56.46 shows the division when the area touches the lower edge and the pixel input from the basic function block is 1-Byte/pixel.

When the area touches the lower edge, an empty area with a height of 7 pixels will appear at the bottom edge of the entire area (the shaded area). Therefore, when specifying to render one of these sub-areas (3, 0), (3, 1), (3, 2), (3, 3) with STRECT, which is described later, the coordinate with the y coordinate corresponding from 57 to 63 (the place indicated by "empty area" at the bottom of Figure 56.45 and Figure 56.46 should not be written to and also should not be rendered. Also, similar to the case when the area does not touch the outer circumference, and the pixel input from the basic function block is 1-Byte/pixel, if specifying to render one of these sub-areas (0, 0), (1, 0), (2, 0), (3, 0) with the STRECT command, which is described later, the coordinate with the x coordinate corresponding from 0 to 13 (the place indicated by "empty area" at the left of Figure 56.46) should not be written to and also should not be rendered.

If the number of pixels in the vertical direction to be rendered is less than the number of pixels shown above, specify the number of pixels in the YWIDTH field of the OUTNOR and OUTROT commands, which are described later.

**When the area touches the left edge**

This section describes the division when the target area to be divided into sub-areas touches the left edge of the entire output image. It corresponds to small areas ①, ④, ⑦ in Figure 56.38.

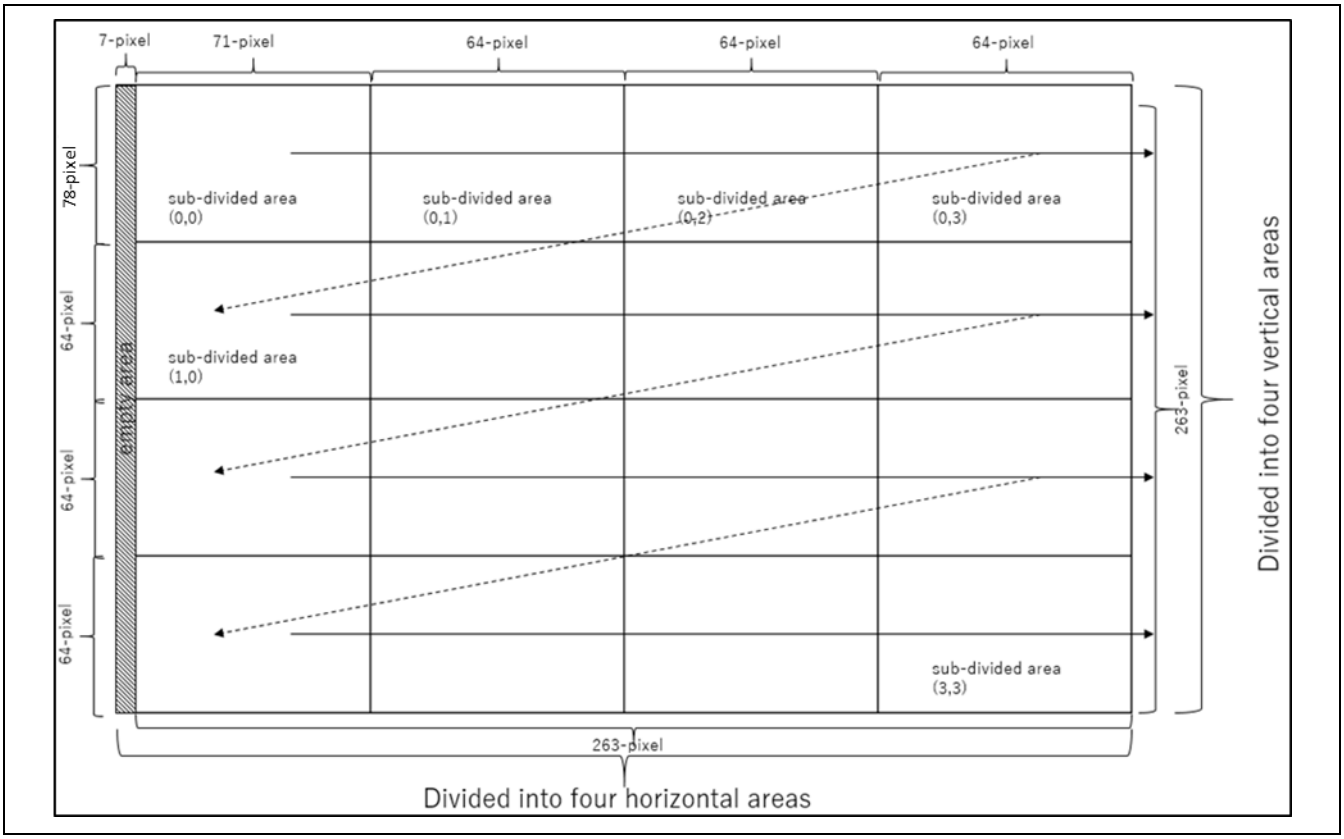



**Figure 56.47 Dividing into sub-areas – When the area touches the left edge of the output image (2-Byte/pixel)**

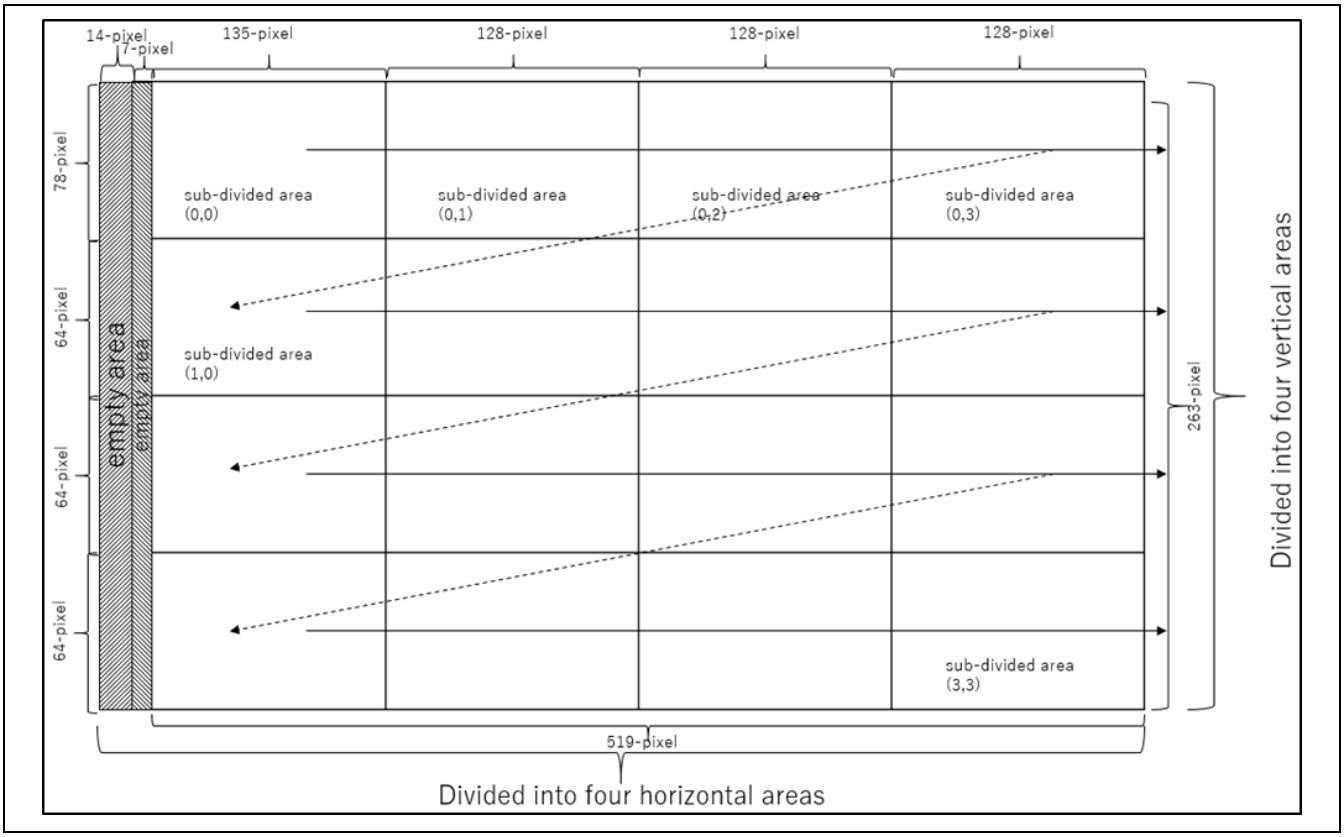



**Figure 56.48 Dividing into sub-areas – When the area touches the left edge of the output image (1-Byte/pixel)**

Figure 56.47 shows the division when the area touches the left edge and the pixel input from the basic function block unit is 2-Byte/pixel. And Figure 56.48 shows the division when the area touches the left edge and the pixel input from the basic function block is 1-Byte/pixel.

When the area touches the left side, an empty area with a width of 7 pixels will appear at the left edge of the entire area (the shaded area). Therefore, when specifying to render one of these sub-areas (0, 0), (1, 0), (2, 0), (3, 0) with the STRECT, which is described later, if the input from the basic function block is 1-Byte/pixel, the coordinate with the x coordinate corresponding from 0 to 20 (the 2 places indicated by "empty area" in Figure 56.48) should not be written to and should not be rendered; if the input from the basic function block is 2-Byte/pixel, the coordinate with the x coordinate corresponding from 0 to 6 (the place indicated by "empty area" in Figure 56.47) should not be written to and should not be rendered.

#### When the area touches the right edge

This section describes the division when the target area to be divided into sub-areas touches the right edge of the entire output image. It corresponds to small areas ③, ⑥, ⑨ in Figure 56.38.

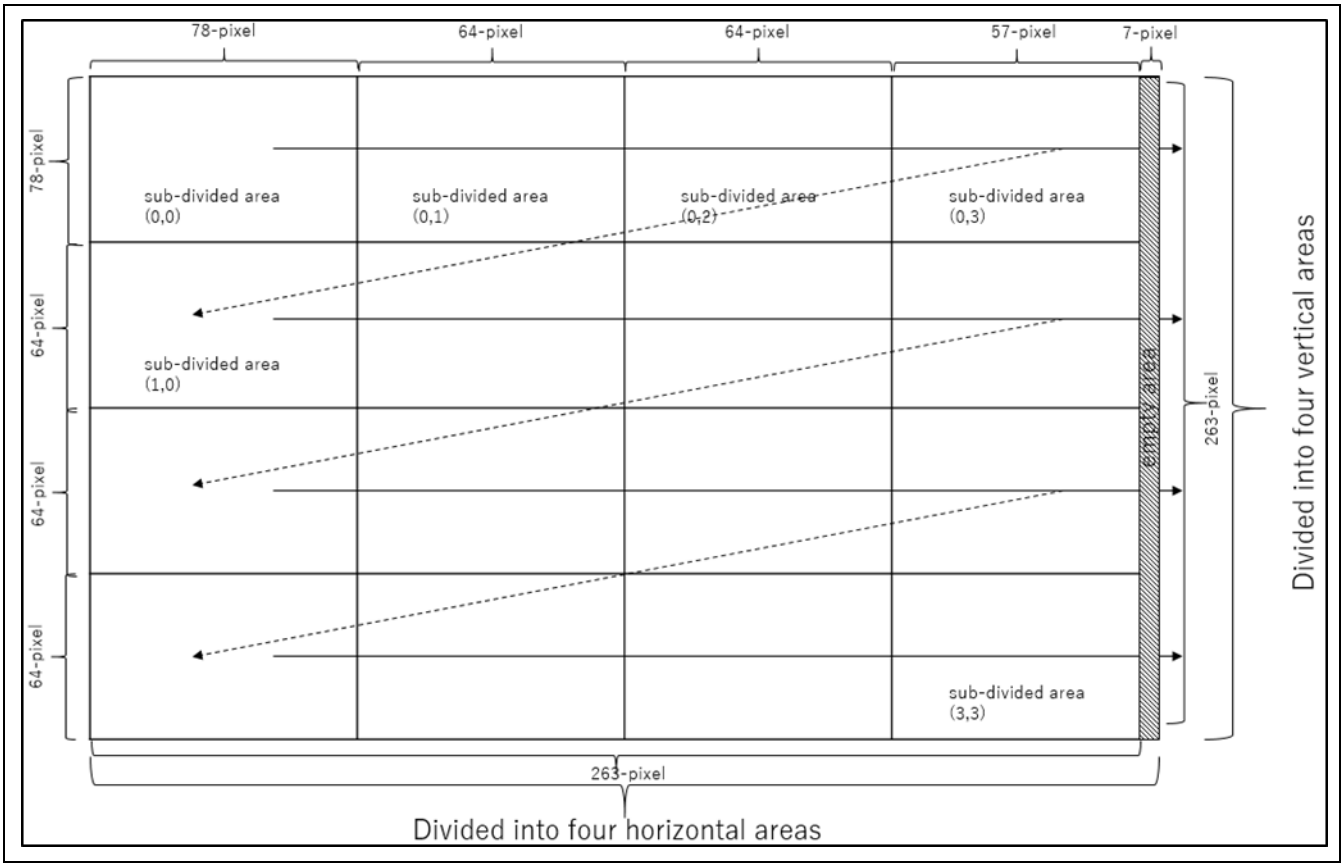



**Figure 56.49 Dividing into sub-areas – When the area touches the right edge of the output image (2-Byte/pixel)**

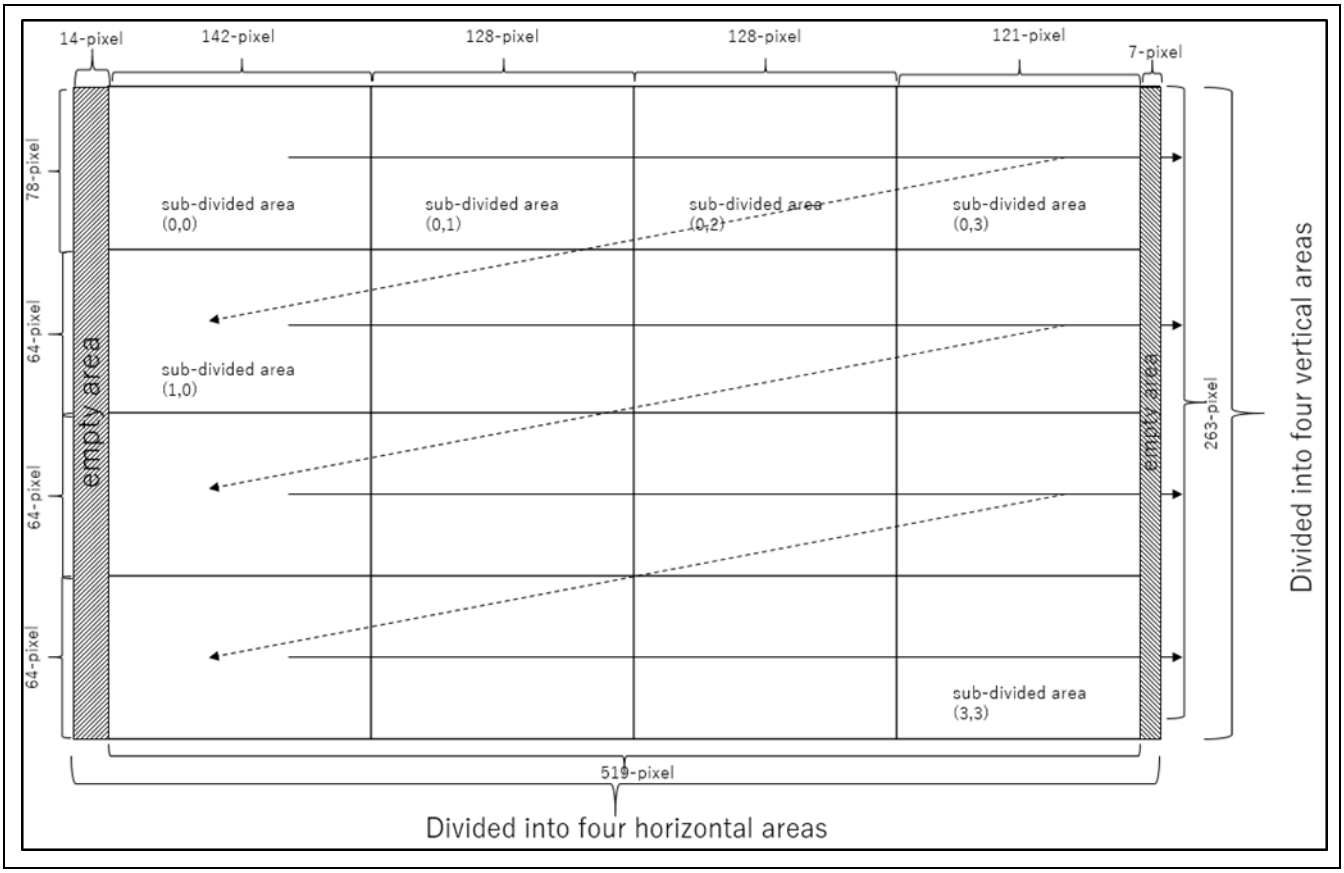



**Figure 56.50 Dividing into sub-areas – When the area touches the right edge of the output image (1-Byte/pixel)**

Figure 56.49 shows the division when the area touches the right edge and the pixel input from the basic function block is 2-Byte/pixel. And Figure 56.50 shows the division when the area touches the right edge and the pixel input from the basic function block is 1-Byte/pixel.

When the area touches the right edge, an empty area with a width of 7 pixels will appear at the right edge of the entire area (the shaded area). Therefore, when specifying to render one of these sub-areas (0, 3), (1, 3), (2, 3), (3, 3) with the STRECT, which is described later, if the input from the basic function block is 1-Byte/pixel, the coordinate with the x coordinate corresponding from 121 to 127 (the place indicated by "empty area" at the right of Figure 56.50) should not be written to and should not be rendered; if the input from the basic function basic function block is 2-Byte/pixel, the coordinate with the x coordinate corresponding from 57 to 63 (the place indicated by "empty area" at the right of Figure 56.49) should not be written to and should not be rendered. Also, similar to the case when the area does not touch the outer circumference, and the pixel input from the basic function block is 1-Byte/pixel, if specifying to render one of these sub-areas (0, 0), (1, 0), (2, 0), (3, 0) with the STRECT command, which is described later, the coordinate with the x coordinate corresponding from 0 to 13 (the place indicated by "empty area" at the left of Figure 56.50) should not be written to and also should not be rendered.

If the number of pixels in the horizontal direction to be rendered is less than the number of pixels shown above, specify the number of pixels in the XWIDTH field of the OUTNOR and OUTROT commands, which are described later.

**When the area touches multiple edges**

This section describes the division when the target area to be divided into sub-areas touches multiple edges of the entire output image, this combines 4 cases that were described before. It corresponds to small areas ①, ③, ⑦, ⑨ in Figure 56.38.

Figure 56.51 shows the division when the area touches the left edge and the upper edge and the pixel input from the basic function block is 2-Byte/pixel. And Figure 56.52 shows the division when the area touches the lower and right edge and the pixel input from the basic function block is 1-Byte/pixel.

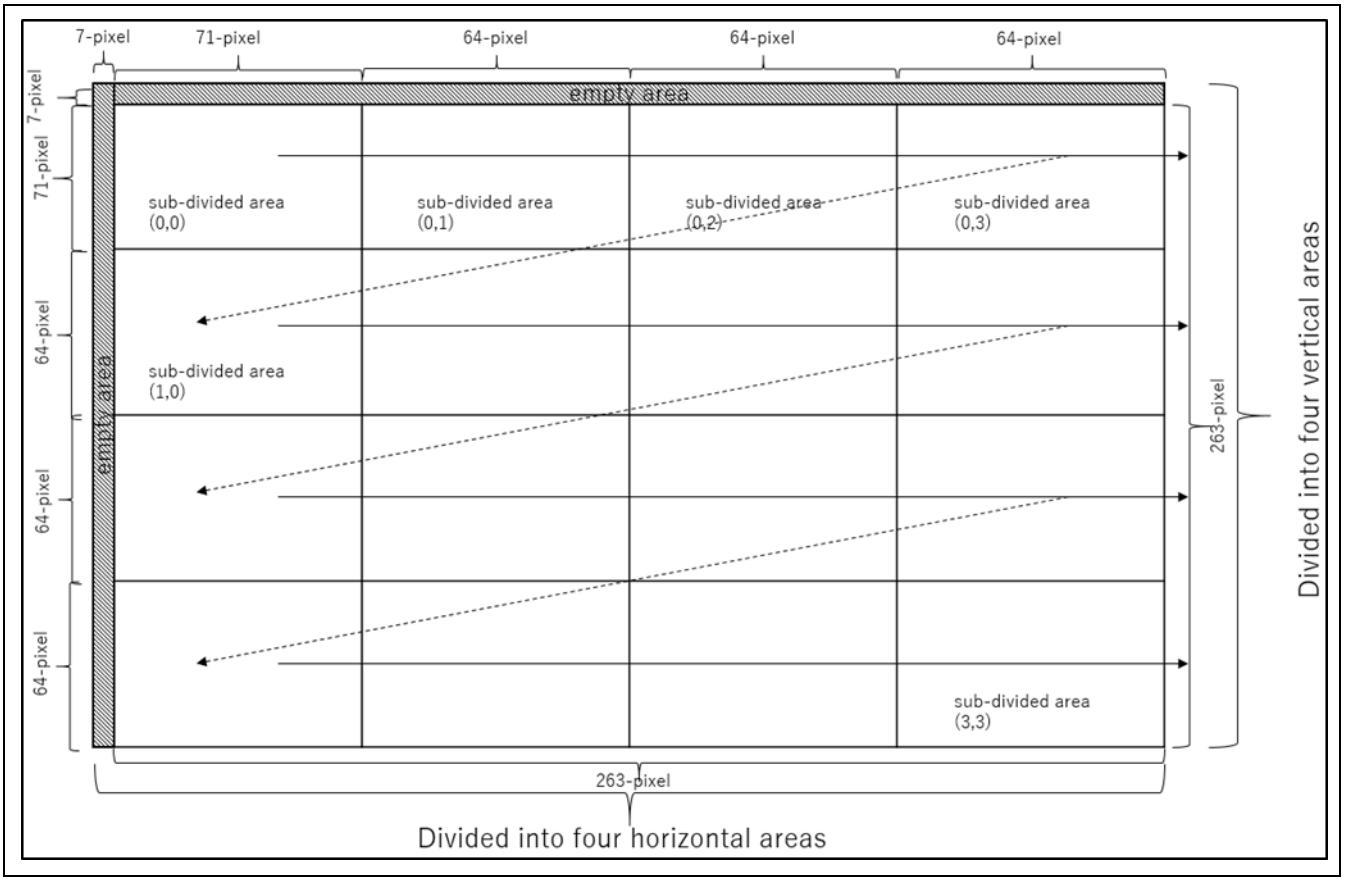



**Figure 56.51 Dividing into sub-areas – When the area touches the left edge and the upper edge of the output image (2-Byte/pixel)**

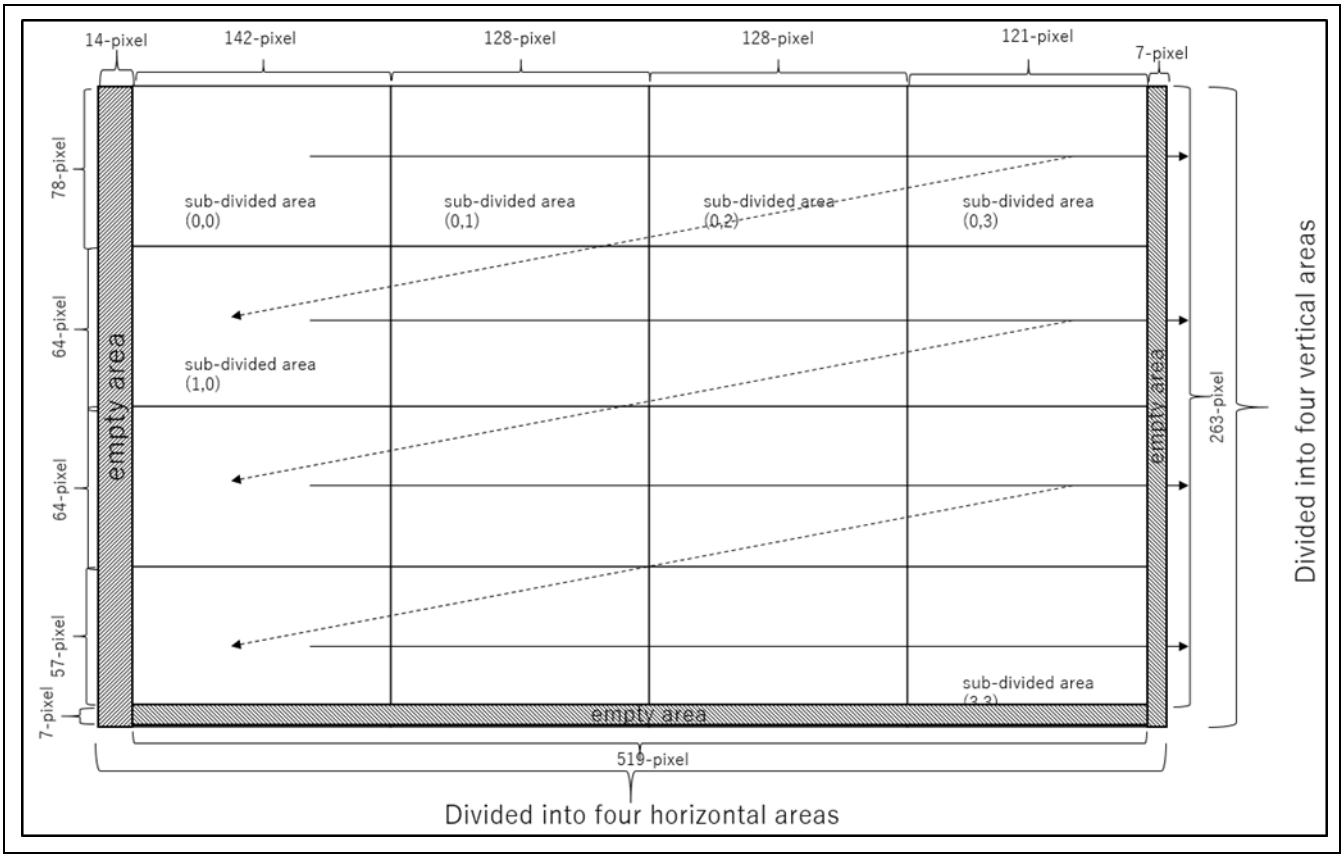



**Figure 56.52 Dividing into sub-areas – When the area touches the right edge and the lower edge of the output image (1-Byte/pixel)**

### (3) Inserting the instructions for synchronization and to start processing

The synchronization instruction (STRECT) needs to be inserted before the processing of each area starts. Also, the instruction to start processing (OUTNOR or OUTROT) needs to be inserted when drawing of the rows and columns configuring each sub-divided area is completed. For details of each instruction, see section 56.5.4, Display List.

The synchronization instruction (STRECT) should be inserted before the drawing of each sub-divided area starts. For example, organize the display list as follows when drawing sub-divided areas in the order (0,0), (0,1), and (0,2).

```
STRECT 0, 0
(Setting and drawing instructions for sub-divided area (0,0) )
STRECT 0, 1
(Setting and drawing instructions for sub-divided area (0,1) )
STRECT 0, 2
(Setting and drawing instructions for sub-divided area (0,2) )
```

**Figure 56.53 Example of Inserting the STRECT Instruction**

The OUTNOR and OUTROT instructions are respectively used for the output of a non-rotated image and rotated image. They should be executed with the following timing.

- Execute the OUTNOR instruction when drawing of a row in a sub-divided area is completed.
- Execute the OUTROT instruction when drawing of a column in a sub-divided area is completed.

After OUTNOR is executed on the row N, it is prohibited to execute TRI instructions which overwrites the row N before executing OUTNOR on the other rows or OUTROT on any column. Also, after OUTROT is executed on the column N, it is prohibited to execute TRI instructions which overwrites the column N before executing OUTROT on the other columns or OUTNOR on any row. Execute SYNCM instruction between OUTNOR/OUTROT and TRI instructions if these cases occur.

Figure 56.54 and Figure 56.55 show examples.

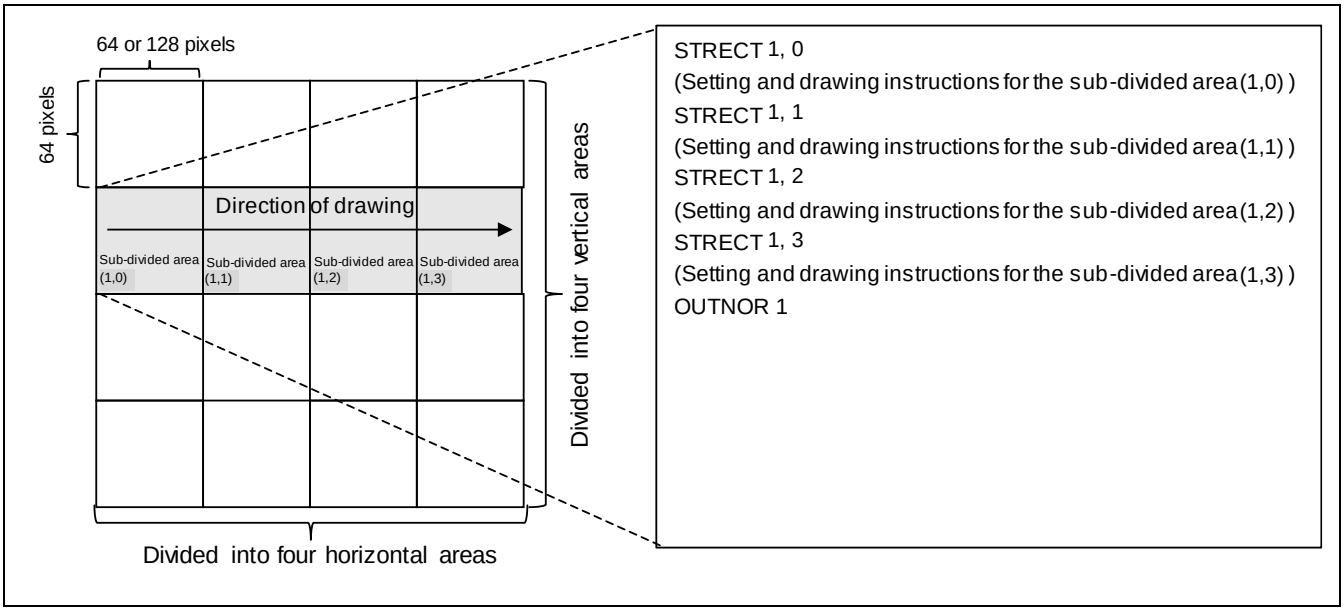



**Figure 56.54 Example of Executing the OUTNOR Instruction with 3x3 Averaging Filter**

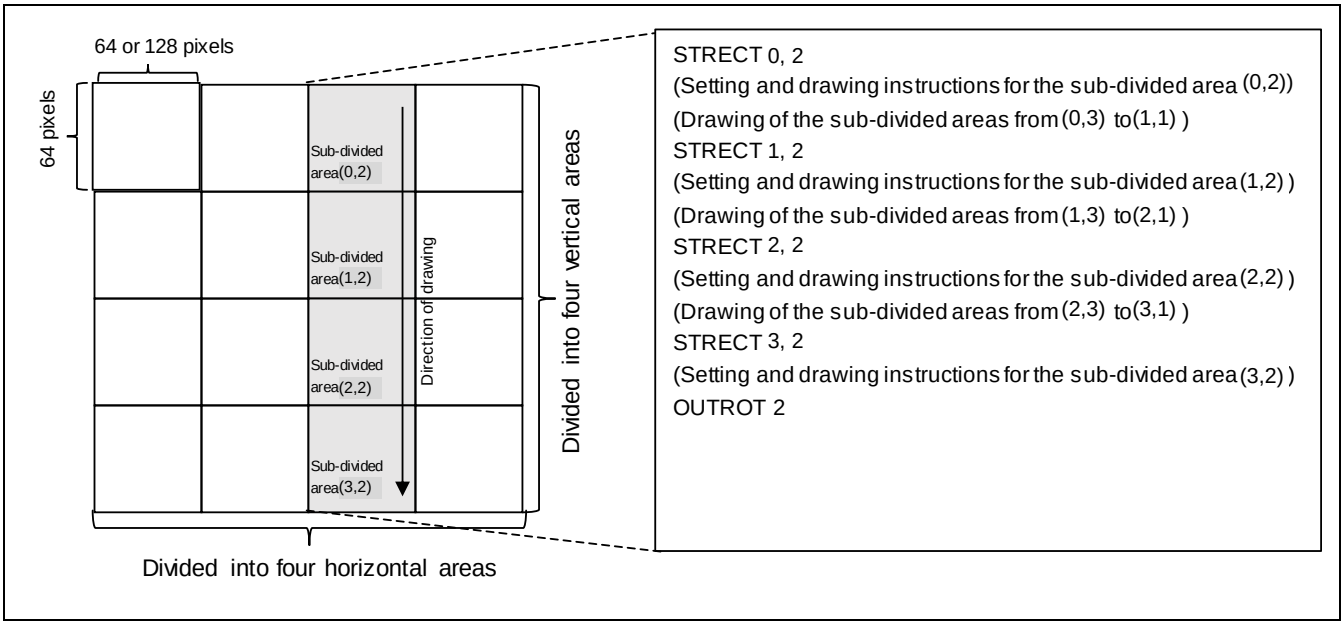



**Figure 56.55 Example of Executing the OUTROT Instruction with 3x3 Averaging Filter**

The OUTNOR and OUTROT instructions can be mixed in a single display list. Accordingly, output of the non-rotated and rotated images at the same time can be handled by executing a single display list.

Drawing of all the pixels in a sub-divided area may not necessarily be required, depending on how the processing area is divided up. In cases where this is not required, processing for such sub-divided areas can be skipped and the OUTNOR or OUTROT instruction can immediately be executed.

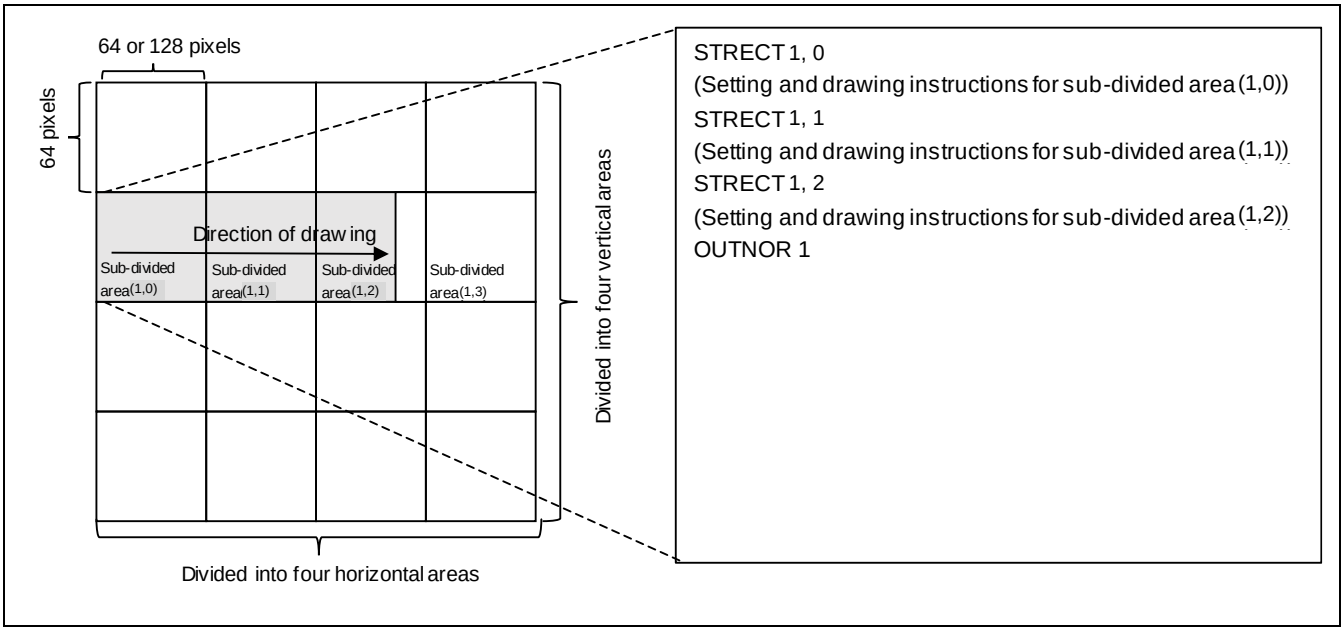



**Figure 56.56 Example of Executing the OUTNOR Instruction when Processing for a Sub-Divided Area is Skipped with 3x3 Averaging Filter**

The pixel values in a skipped sub-divided area are written to the external memory as undefined values.

### (4) Rotation

#### (a) Y plane rotation

If the YCM bit in CMRCR of the basic function block is 0, the input of this extended function block is regarded as Y plane and rotation is performed.

The Y plane rotation is done by transposing the pixels of each coordinate. An example is shown below.

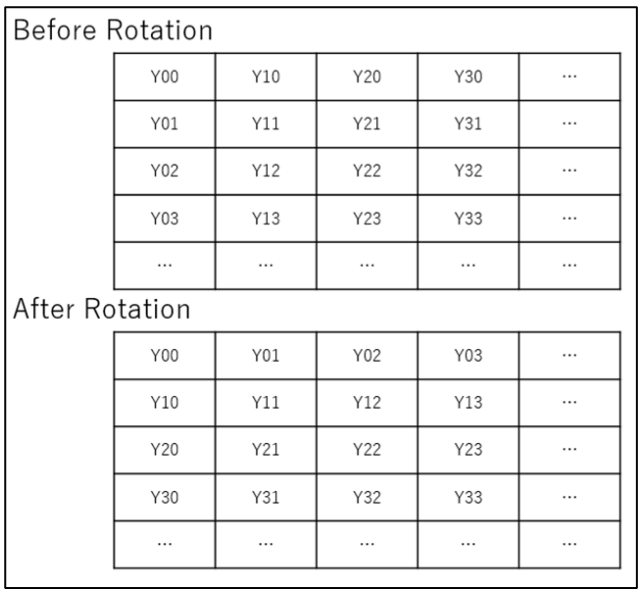



**Figure 56.57 Example of Y plane rotation**

#### (b) UV plane rotation (YUV422 format)

If the YCM bit in CMRCR of the basic function block is 1, and the RSUV420 bit in RSCR is 0, the input of this extended function block is regarded as UV plane in YUV422 format and rotation is performed.

The rotation of UV plane in YUV422 format can be selected from two kinds of algorithms.

If the RSUVFLT bit in RSCR is 0, transpose UV data in unit of 2x2 blocks. An example is shown in the figure below.

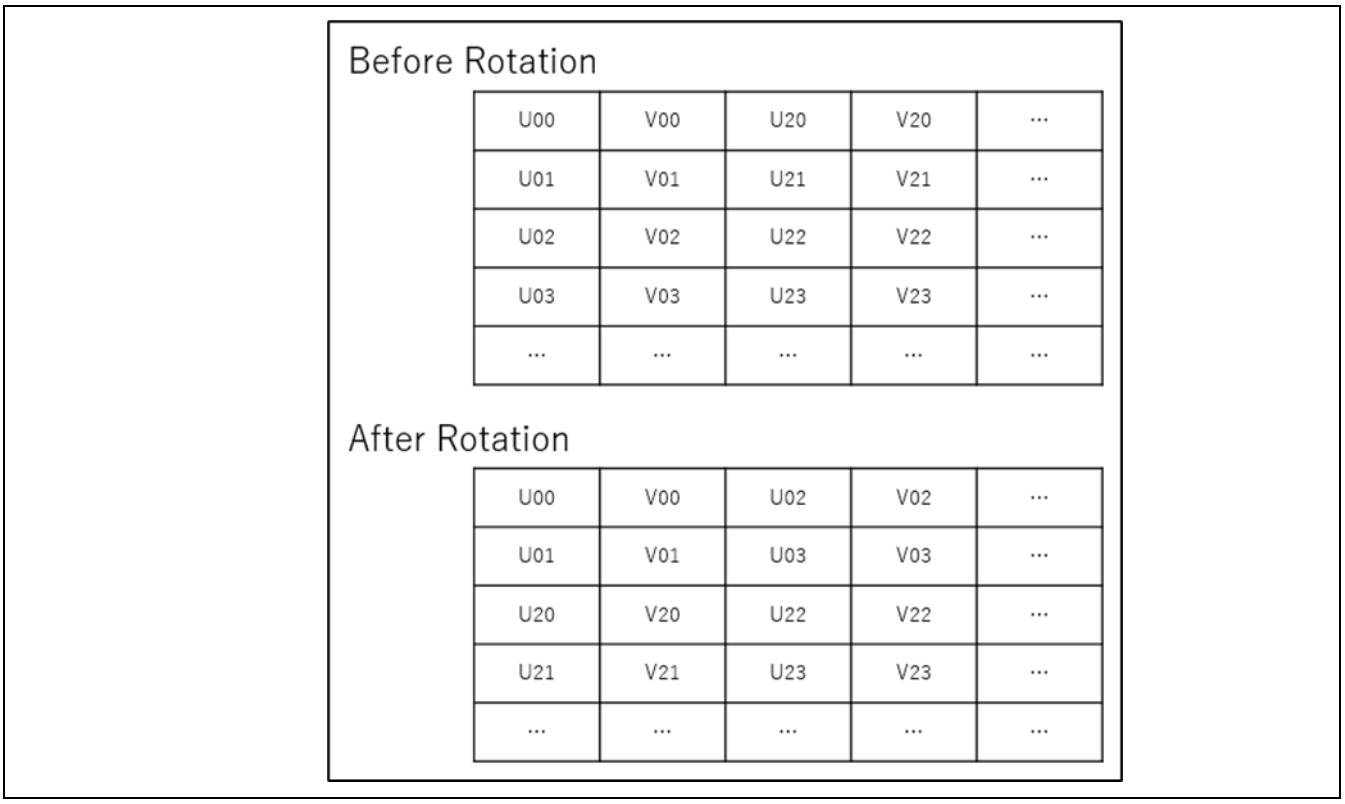



**Figure 56.58 Example of UV plane rotation in YUV422 format (RSUVFLT=0)**

When the RSUVFLT bit in RSCR is 1, transposition is performed while interpolating between pixel data. An example is shown in the figure below. When number line before rotation is odd, then interpolation process cannot be done for odd line. Therefore, please arrange the UV data with even line before rotation.

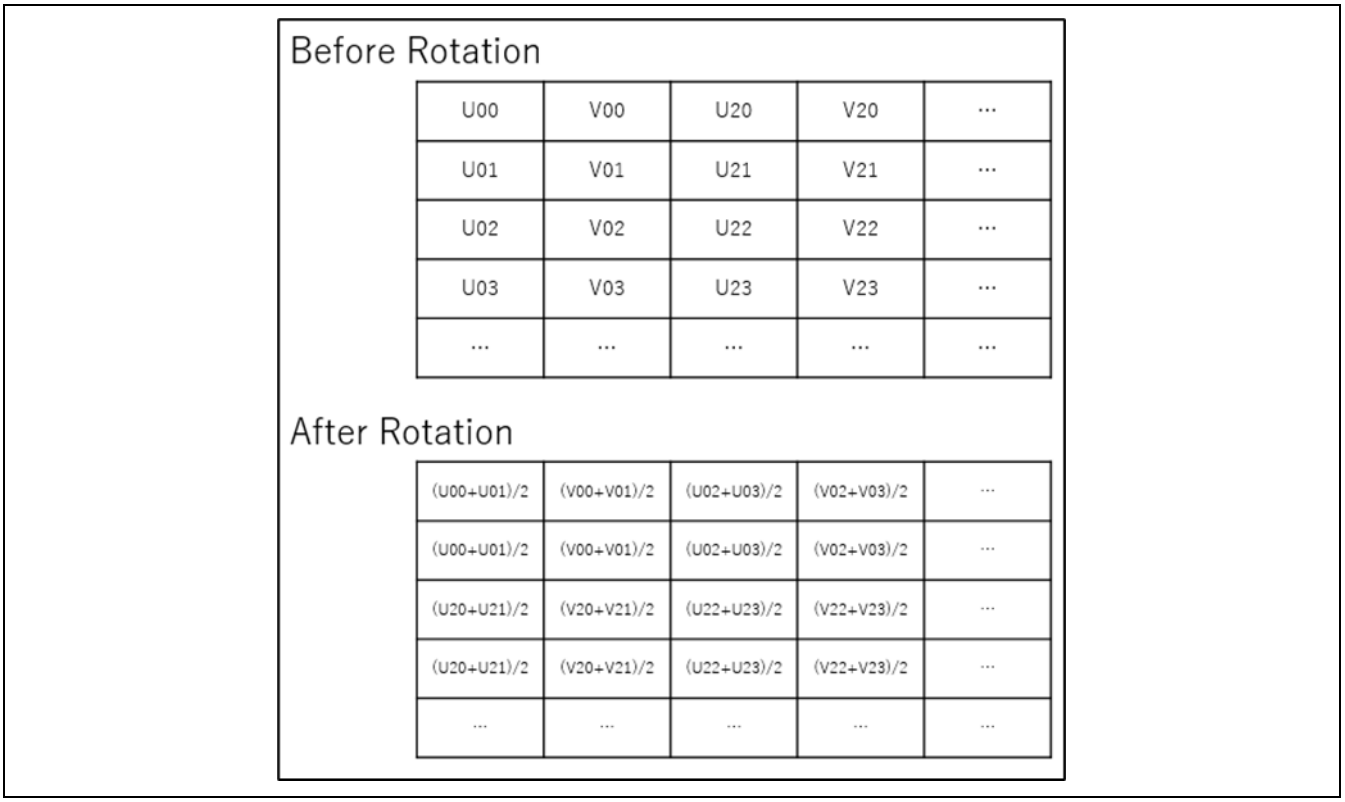



**Figure 56.59 Example of UV plane rotation in YUV422 format (RSUVFLT=1)**

### (c) UV plane rotation (YUV420 format)

If the YCM bit in CMRCR of the basic function block is 1, and the RSUV420 bit in RSCR is 1, the input of this extended function block is regarded as UV plane in YUV420 format and rotation is performed.

Rotation is done by assuming the adjacent U and V as 1 pixel and transposing it. An example is shown in the figure below.

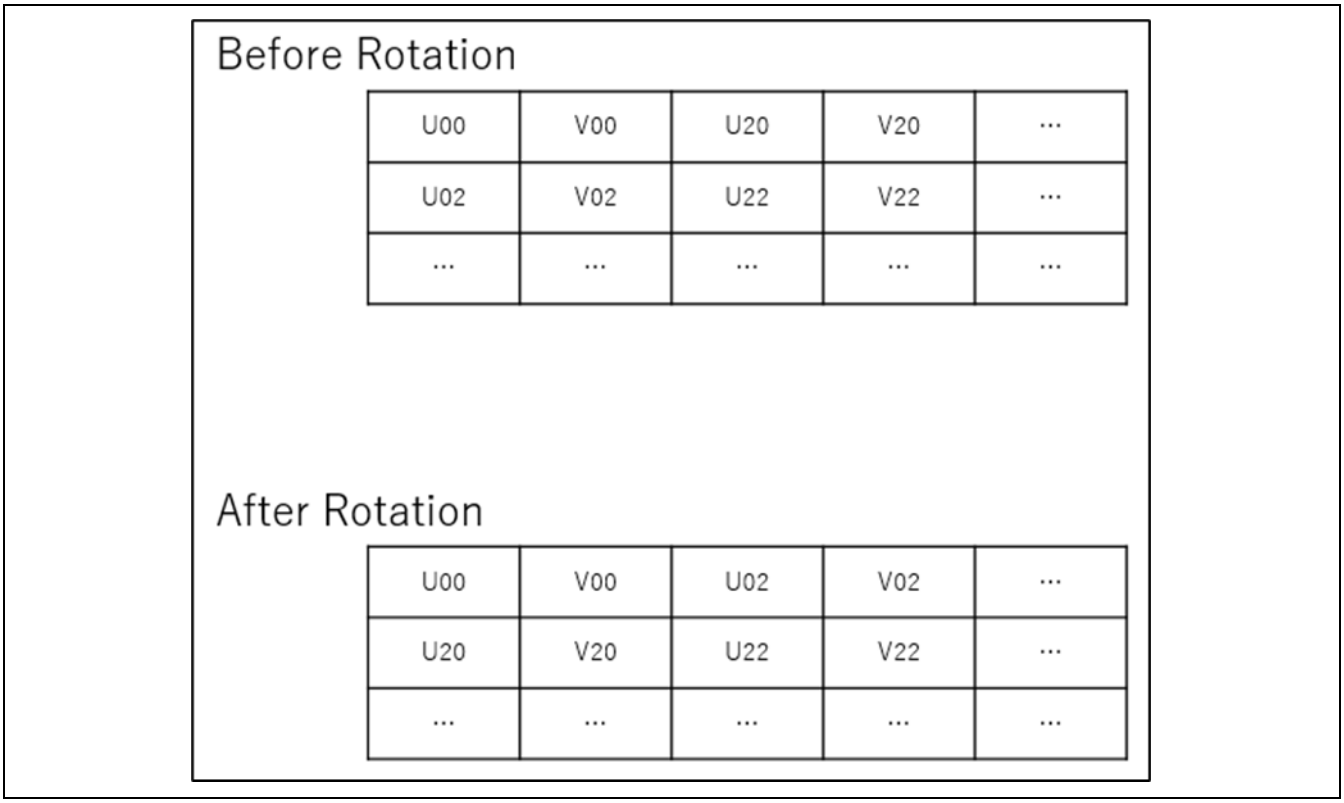



**Figure 56.60 Example of UV plane rotation in YUV420 format**

### 56.5.3.2 Using the Scaling-down Function

An image output by executing the OUTNOR and OUTROT instructions explained earlier can be recursively (up to three times) scaled down to 1/2, vertically and horizontally.

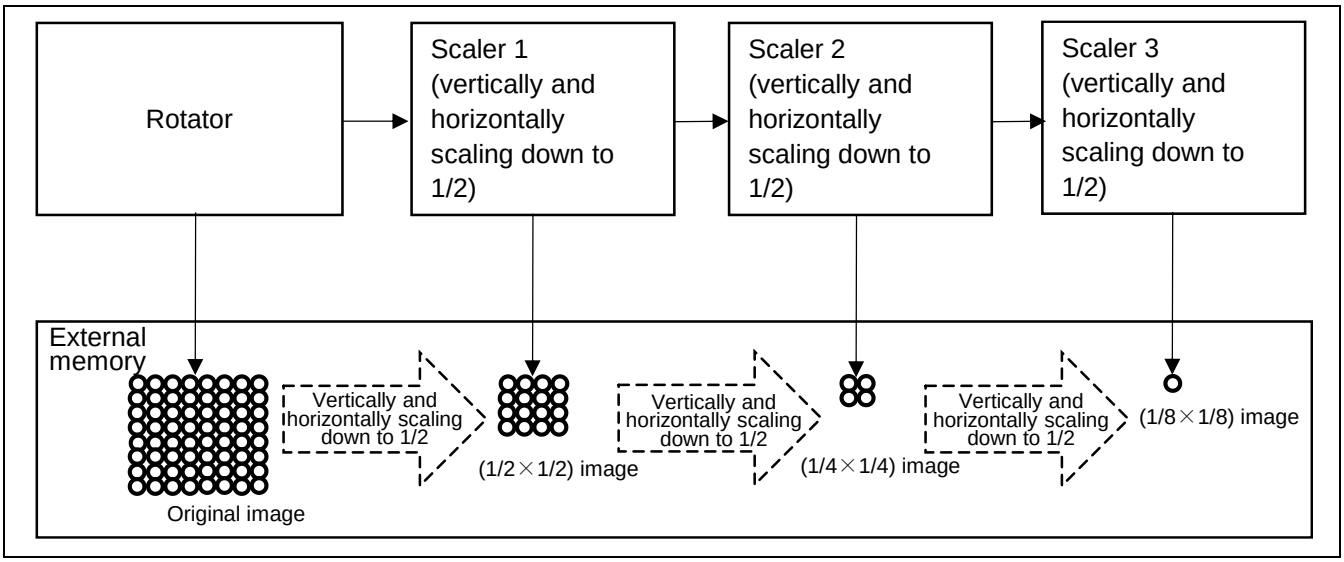



**Figure 56.61 Overview of Scaling Down in the Extended Function Block**

Figure 56.61 shows an overview of scaling down. The image processed by the rotator is output to the scaler and the image can be scaled down to 1/2 by each of three scalers.

### (1) Enabling or Disabling Output of the Scaled-Down Images

Output of the scaled down images can be enabled or disabled by the rotator and scaler control register (RSCR).
The NRnE bit can be used to enable or disable output of a non-rotated image. The RTnE bit can be used to enable or disable output of a rotated image.

Furthermore, set the base addresses, strides, and offsets from the base addresses for all outputs in the following registers: DSANRRn, DSTNRRn, DSARRn, DSTRRn, DSONRRn, and DSORRn. Refer to the descriptions of the register settings for details.

### (2) Setting for Scaling-Down Filtering

The scaling-down filtering is applied as part of the processing in each scaler. It is possible to select 2x2 averaging filter and 3x3 Gaussian filter as filtering method of Y plane. UV plane can only use the 2x2 averaging filter

When using the 2x2 averaging filter, clear the GFE bit of GFCR register to 0 and clear all the AFB8E bit, the AFB4E bit, and the AFB2E bit in RSCR2 register to 0.

When using the 3x3 Gaussian filter, set the GFE bit of GFCR register to 1 and set all the AFB8E bit, the AFB4E bit, and the AFB2E bit in RSCR2 register to 1.

Set the filtering method before starting to render by the TRI command, to which the desired filter is expected to be applied. It is prohibited to change the setting within one divided area.

The rendering method will change depending on the filtering method. For details, refer to 56.5.3.1.

#### 2x2 averaging filter

When selecting the 2x2 averaging filter, clear the GFE bit of GFCR register to 0 and clear all the AFB8E bit, the AFB4E bit, and the AFB2E bit in RSCR2 register to 0.

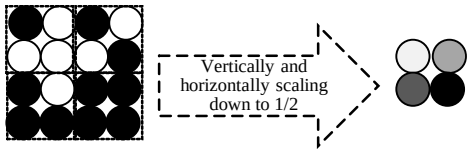



**Figure 56.62 Scaling-Down Filtering**

When the setting of the BPnE bit in RSCR is 0, the pixel values in the images after scaling down are determined by averaging the pixel values of the 2- x 2-pixel areas in the image before scaling down. When the setting of the BPnE bit in RSCR is 1, the pixel values in the image after scaling down are determined by adding the pixel values of the 2- x 2-pixel areas in the images before scaling down.

When the setting of the BPnE bit is 0, the rounding mode for use in averaging of the pixel values can be selected. When the RDnE bit in RSCR is 0, rounding down is selected, and when it is set to 1, rounding off to the nearest number is selected.

Regardless of the setting of BPnE, if the pixel format of the images before scaling down is N bpp, the pixel format of the images after scaling down becomes (N+2) bpp. ‘00’ is added to the LSB of the averaged pixel values when the setting of the BPnE bit is 0. Accordingly, set the format of input and output images and the size in bits of the logical right-shifting in the three scalers in accord with one of the combinations listed in Table 56.26 or Table 56.27. Setting of combinations other than those in the tables is prohibited.

**Table 56.26 Combination when the BPnE Bit in RSCR is Set to 0**

| Input Pixel Format *1 | Output Pixel Format *2 | Number of Digits of Logical Right-Shifting *3 |
|---|---|---|
| 8 bpp | 8 bpp | 2 |
| 10 bpp | 10 bpp | 2 |
| 12 bpp | 12 bpp | 2 |
| 14 bpp | 14 bpp | 2 |
| 16 bpp | 16 bpp | 2 |

**Table 56.27 Combination when the BPnE Bit in RSCR is Set to 1**

| Input Pixel Format *1 | Output Pixel Format *2 | Number of Digits of Logical Right-Shifting *3 |
|---|---|---|
| 8 bpp | 10 bpp | 0 |
| 10 bpp | 12 bpp | 0 |
| 12 bpp | 14 bpp | 0 |
| 14 bpp | 16 bpp | 0 |

Notes: 1. The following shows the registers for use in specifying the input pixel formats for the three scalers.
As settings for scaler 1, the values of the DY1416, Y12, and Y10 bits of CMRCR are used in Y processing and those of the DUV1416 and DUV bits of CMRCR are used in UV processing.
As settings for scaler 2, the values of the OFM2N bits of RSOFR are used in the case of an OUTNOR instruction and those of the OFM2R bits of RSOFR are used in the case of an OUTROT instruction.
As settings for scaler 3, the values of the OFM4N bits of RSOFR are used in the case of an OUTNOR instruction and those of the OFM4R bits of RSOFR are used in the case of an OUTROT instruction.

2. The following shows the registers for use in specifying the output pixel formats for the three scalers.
As settings for scaler 1, the values of the OFM2N bits of RSOFR are used in the case of an OUTNOR instruction and those of the OFM2R bits of RSOFR are used in the case of an OUTROT instruction.
As settings for scaler 2, the values of the OFM4N bits of RSOFR are used in the case of an OUTNOR instruction and those of the OFM4R bits of RSOFR are used in the case of an OUTROT instruction.
As settings for scaler 3, the values of the OFM8N bits of RSOFR are used in the case of an OUTNOR instruction and those of the OFM8R bits of RSOFR are used in the case of an OUTROT instruction.

3. The following shows the registers for use in specifying the number of bits of logical right-shifting in the three scalers.
As settings for scaler 1, the values of the SC2SHFVAL bits of RSCR are used.
As settings for scaler 2, the values of the SC4SHFVAL bits of RSCR are used.
As settings for scaler 3, the values of the SC8SHFVAL bits of RSCR are used.

**3x3 Gaussian filter**

When using the 3x3 Gaussian filter, set the BPnE bit in RSCR to 0 and AFBnE in RSCR2 to 1. Accordingly, set the format of input and output images and the size in bits of the logical right-shifting in the three scalers in accord with one of the combinations listed in Table 56.28 Setting of combinations other than those in the tables is prohibited.

**Table 56.28 Combination for input and output format when using Gaussian filter**

| Input Pixel Format *1 | Output Pixel Format *2 | Number of Digits of Logical Right-Shifting in Gaussian filter *3 | Number of Digits of Logical Right-Shifting *4 |
|---|---|---|---|
| 8 bpp | 8 bpp | 4 | 2 |
| 10 bpp | 10 bpp | 4 | 2 |
| 12 bpp | 12 bpp | 4 | 2 |
| 14 bpp | 14 bpp | 4 | 2 |
| 16 bpp | 16 bpp | 4 | 2 |
| 8 bpp | 10 bpp | 2 | 2 |
| 10 bpp | 12 bpp | 2 | 2 |
| 12 bpp | 14 bpp | 2 | 2 |
| 14 bpp | 16 bpp | 2 | 2 |

Notes: 1. The following shows the registers for use in specifying the input pixel formats for the three scalers.
As settings for scaler 1, the values of the DY1416, Y12, and Y10 bits of CMRCR are used in Y processing and those of the DUV1416 and DUV bits of CMRCR are used in UV processing.

As settings for scaler 2, the values of the OFM2N bits of RSOFR and GFM2N bits of GFOFR are used in the case of an OUTNOR instruction and those of the OFM2R bits of RSOFR and GFM2R bits of GFOFR are used in the case of an OUTROT instruction.

As settings for scaler 3, the values of the OFM4N bits of RSOFR and GFM4N bits of GFOFR are used in the case of an OUTNOR instruction and those of the OFM4R bits of RSOFR and GFM4R bits of GFOFR are used in the case of an OUTROT instruction.

2. The following shows the registers for use in specifying the output pixel formats for the three scalers.
As settings for scaler 1, the values of the OFM2N bits of RSOFR and GFM2N bits of GFOFR are used in the case of an OUTNOR instruction and those of the OFM2R bits of RSOFR and GFM2R bits of GFOFR are used in the case of an OUTROT instruction.

As settings for scaler 2, the values of the OFM4N bits of RSOFR and GFM4N bits of GFOFR are used in the case of an OUTNOR instruction and those of the OFM4R bits of RSOFR and GFM4R bits of GFOFR are used in the case of an OUTROT instruction.

As settings for scaler 3, the values of the OFM8N bits of RSOFR and GFM8N bits of GFOFR are used in the case of an OUTNOR instruction and those of the OFM8R bits of RSOFR and GFM8R bits of GFOFR are used in the case of an OUTROT instruction.

3. The following shows the registers for use in specifying the number of bits of logical right-shifting in the three scalers.
As settings for scaler 1, the values of the SC2GSHFVAL bits of GFCR are used.
As settings for scaler 2, the values of the SC4GSHFVAL bits of GFCR are used.
As settings for scaler 3, the values of the SC8GSHFVAL bits of GFCR are used.

4. The following shows the registers for use in specifying the number of bits of logical right-shifting in the three scalers.
As settings for scaler 1, the values of the SC2SHFVAL bits of RSCR are used.
As settings for scaler 2, the values of the SC4SHFVAL bits of RSCR are used.
As settings for scaler 3, the values of the SC8SHFVAL bits of RSCR are used.

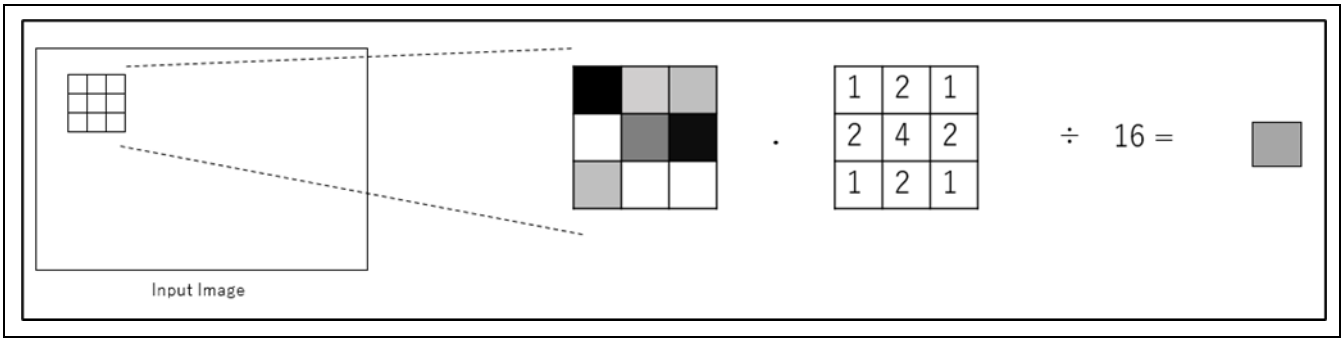



**Figure 56.63 Operation of 3x3 Gaussian filter.**

When using the 3x3 Gaussian filter, set the GFE bit of GFCR register to 1 and set all the AFB8E bit, the AFB4E bit, and the AFB2E bit in RSCR2 register to 1.
The 9 pixels around each coordinate of the input image are multiplied by the coefficient shown in the Figure 56.63, and the average value is taken as the output.

When processing the pixels that touches the upper, lower, left and right edges of the input image, one of the surrounding 9 pixels does not exist. Therefore, when the target area to be processed by OUTNOR or OUTROT touches the upper, lower, left or right edges of the input image, set the UP, DP, LP, or RP bit in the GFCR register to 1 depending on the contact side. Table 56.29 shows the setting condition of each bit.

**Table 56.29 Setting condition of UP, DP, LP, RP bit in UGFCR register**

| Bit name | Setting condition |
|---|---|
| UP | It touches the upper edge |
| DP | It touches the lower edge |
| LP | It touches the left edge |
| RP | It touches the right edge |

The output format of each Scaler follows the setting of GFOFR register and RSOFR register. If the calculation result will be saturated if it exceeds the value range of the set format. The setting of GFOFR register and RSOFR register should be the same. This setting applies to both output to external memory and output to the next Scaler stage.

After implementing 3x3 filter processing on all coordinates of each image input as a segmented area, scaling is taken place as in the Figure 56.64. Even pixels and even lines will be output after 3x3 filter processing in each scalar output 1/2, 1/4, and 1/8.

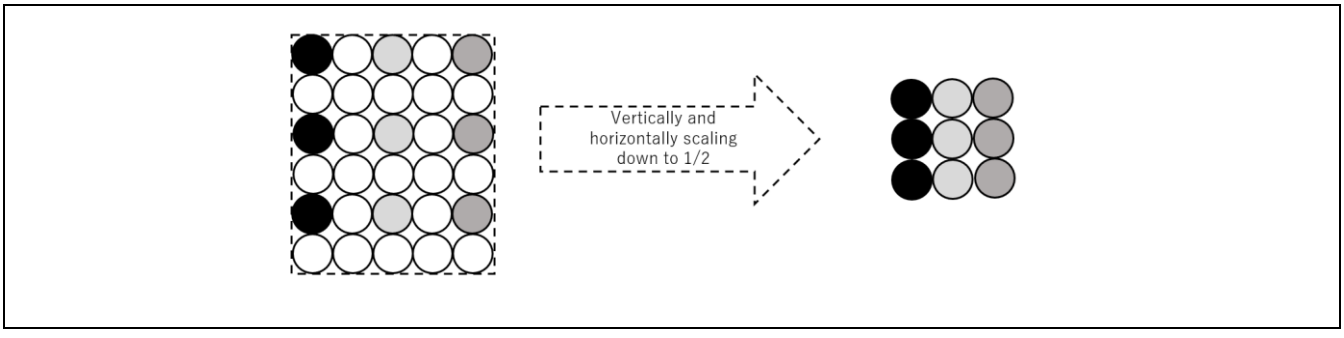



**Figure 56.64 Scaling-Down 3x3 Gaussian filter**

**(3) Settings for Logical Right-Shifting**

When logical right-shifting the pixel value after applying the 2x2 averaging filter, set SC2SHFVAL, SC4SHFVAL, SC8SHFVAL in the RSCR register.

Also, when logical right-shifting the pixel value after applying the 3x3 Gaussian filter, set SC2GSHFVAL, SC4GSFHVAL, SC8GSHFVAL in the GFCR register. This setting is also used for the division in the Gaussian filter calculation, and it is shifted 4 bits to the right by default. Note that the divisor of the division operation will also be changed if the setting is changed.

The shift down setting applies to both output to external memory and output to the next Scaler stage.

#### 56.5.3.3 Supported data format and Data Location in Memory

**(1) Supported data format**

This extended block supports processing of only Y plane or only UV plane of Semi-planar format. The simultaneous processing of Y/UV plane in Interleaved format or Semi-planar format is not supported. Therefore, also follow this constraint for processing in the basic function block.

The Y and UV planes support YUV422 format and YUV420 format. Switching YUV422/YUV420 in UV plane is performed by the RSUV420 bit of RSCR register.

The pixel accuracy supports 8-bpp, 10-bpp, 12-bpp, 14-bpp, 16-bpp for both input and output. The input accuracy of this extended block is set in the pixel format of the destination data in the basic function block. About the output accuracy, the non scaled-down image follows the destination image output format of the basic function block. Additionally, the scaled-down image group is set by the RSOFR register when using the 2x2 averaging filter, and by the GFOFR register when using the 3x3 Gaussian filter. Each scaling output can be set as non-rotated/rotated separately.

### (2) Data Location in Memory

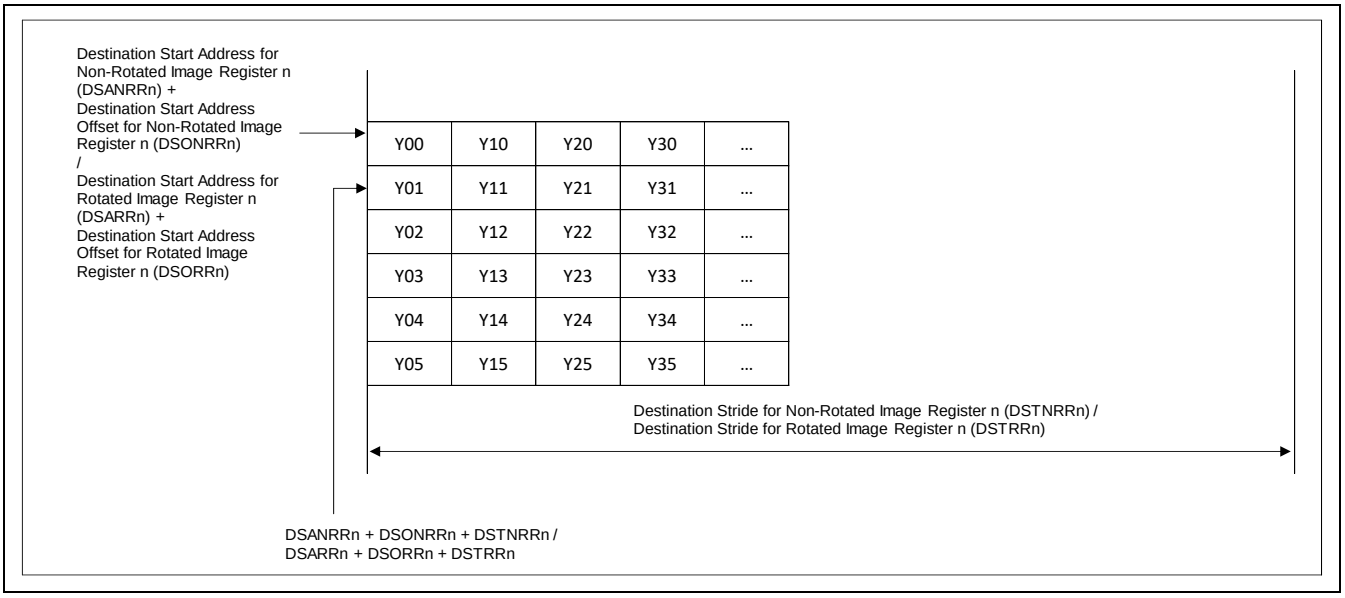



**Figure 56.65 Memory allocation for Y plane output in extended block**

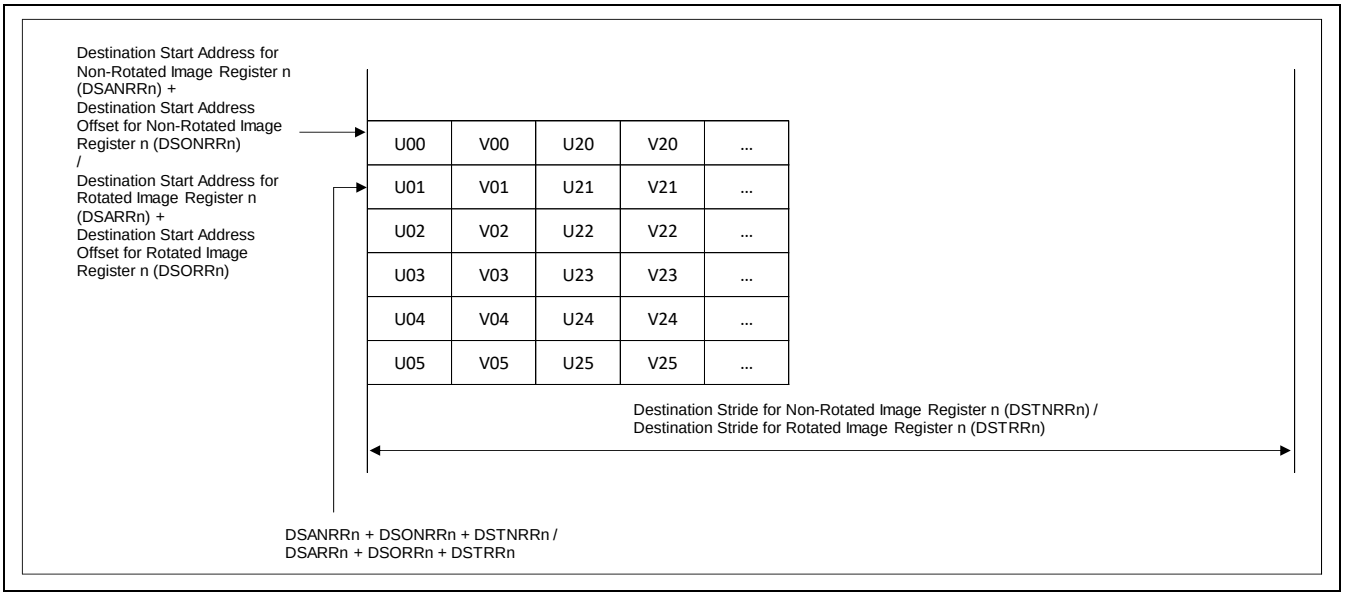



**Figure 56.66 Memory allocation for UV plane output in extended block**

Figure 56.65 shows the relationship between the memory allocation for Y plane output in this extended unit and the register setting, and Figure 56.66 shows the relationship between the memory allocation for UV plane output and the register setting. “n” indicates the setting register of each scaling-down ratio, n = 0, 1, 2, or 3.

The start address, to where output data is written, is obtained by adding DSANRRn and DSONRRn for the non-rotated image, and is obtained by adding DSARRn and DSORRn for the rotated image. This applies for both Y and UV planes. The memory stride is common for Y plane and UV plane, it is set by DSTNRRn for non-rotated image and by DSTRRn for rotated image.

The U and V order of the UV plane is the same as the output from the basic function unit. If you want to change the order, change the setting of the basic function unit.

#### 56.5.3.4 Usage Note

Assuming that the highest scaling ratio is 1/N, the size of the non-scaled image should be a multiple of N vertically and horizontally.

### 56.5.4 Display List

The use of instructions not listed in this section is prohibited.

This section only explains the instructions used to control the extended function blocks. For the instructions used to control the basic function block, see section 56.5, Display List.

Do not execute the instructions listed in this section in a channel which is not capable of rotation and scaling-down. Refer to Table 56.2 for the applicable channels. Additionally, do not execute the instructions when the setting of the RSE bit in SCR is 0.

#### 56.5.4.1 STRECT (STart RECTangle)

**(1) Function**

STRECT is used to specify the sub-divided areas of the image for drawing. The ROW field specifies the vertical size and the COLUMN field specifies the horizontal size.

Set the ROW and COLUMN fields to values from 0 to 3. Setting values outside this range is prohibited.

**(2) Instruction Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_0111 (H'87) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | ROW | | | | COLUMN | | | |

#### 56.5.4.2 OUTNOR (OUTput NORmally)

**(1) Function**

OUTNOR is used to specify the output of a non-rotated image when the drawing of a row in a sub-divided area is completed.

A row in the sub-divided area specified by the ROW field is output without being rotated. Set the ROW field to a value from 0 to 3. Setting values outside this range is prohibited.

**All output 8bbp:**

When the input image format from the basic function block is 8 bpp and the format for output from all three scalers of the extended function block is 8 bpp, set the XWIDTH field to **H'1FF**.

+ For right block (refer Figure 56.37) is equal or smaller than **512 pixels,** XWIDTH can be set to **H'1FF**.

+ For right block (refer Figure 56.37) is equal or smaller than **384 pixels,** XWIDTH can be set to **H'17F**.

+ For right block (refer Figure 56.37) is equal or smaller than **256 pixels,** XWIDTH can be set to **H'0FF**.

+ For right block (refer Figure 56.37) is equal or smaller than **128 pixels,** XWIDTH can be set to **H'07F**.

Always set the YWIDTH field to H'3F.

**Other:**

set XWIDTH to **H'0FF** in other cases.

+ For right block (refer Figure 56.37) is equal or smaller than **256 pixels,** XWIDTH can be set to **H'0FF.**

+ For right block (refer Figure 56.37) is equal or smaller than **192 pixels,** XWIDTH can be set to **H'0BF**.

+ For right block (refer Figure 56.37) is equal or smaller than **128 pixels,** XWIDTH can be set to **H'07F**

+ For right block (refer Figure 56.37) is equal or smaller than **64 pixels,** XWIDTH can be set to **H'03F**

Always set the YWIDTH field to **H'3F**.

After OUTNOR is executed on row N, the execution TRI instructions which overwrite row N before executing OUTNOR on the other rows or OUTROT on any column is prohibited. Execute a SYNCM instruction between the OUTNOR and TRI instructions if this kind of procedure is necessary.

**(2) Instruction Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1001 (H'89) | | | | | | | | ROW | | | | Reserved (all 0) | | | XWIDTH | | | | | | | | | YWIDTH | | | | | | | |

### 56.5.4.3 OUTROT (OUTput ROTationally)

**(1) Function**

OUTROT is used to specify the output of a rotated image when the drawing of a column in a sub-divided area is completed.

A column in the sub-divided area specified by the COLUMN field is output after being rotated. Set the COLUMN fields to values from 0 to 3. Setting values outside this range is prohibited.

**All output 8bbp:**

When the input image format from the basic function block is 8 bpp and the format for output from all three scalers of the extended function block is 8 bpp, set the XWIDTH field to **H'07F. S**et the YWIDTH field to **H'FF.**

+For bottom block (refer Figure 56.37) is equal or smaller than **256 lines,** YWIDTH can be set to **H'FF**.

+For bottom block (refer Figure 56.37) is equal or smaller than **192 lines,** YWIDTH can be set to **H'BF.**

+For bottom block (refer Figure 56.37) is equal or smaller than **128 lines,** YWIDTH can be set to **H'7F.**

+For bottom block (refer Figure 56.37) is equal or smaller than **64 lines,** YWIDTH can be set to **H'3F.**

**Other:**

Set XWIDTH field to **H'03F** in other cases.

Set the YWIDTH field to **H'FF.**

+For bottom block (refer Figure 56.37) is equal or smaller than **256 lines,** YWIDTH can be set to **H'FF.**

+For bottom block (refer Figure 56.37) is equal or smaller than **192 lines,** YWIDTH can be set to **H'BF.**

+ For bottom block (refer Figure 56.37) is equal or smaller than **128 lines,** YWIDTH can be set to **H'7F.**

+For bottom block (refer Figure 56.37) is equal or smaller than **64 lines,** YWIDTH can be set to **H'3F.**

After OUTROT is executed on column N, the execution of a TRI instruction which overwrites column N before the execution of OUTROT on another column or OUTNOR on any row is prohibited. Execute a SYNCM instruction between the OUTROT and TRI instructions if this kind of procedure is necessary.

**(2) Instruction Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1000_1011 (H'8B) | | | | | | | | COLUMN | | | | Reserved (all 0) | | | XWIDTH | | | | | | | | | YWIDTH | | | | | | | |

## 56.6 Usage Notes

### 56.6.1 Module Standby Mode

The module standby mode enables stoppage of the clock signals supplied to the IMR-LX6. Before placing the IMR-LX6 on module standby, first make sure that the IMR-LX6 is not operating and then stop supply of its clock signals. Do not attempt access to the registers of the IMR-LX6 while it is on module standby.

Be sure to apply a software reset to the clock pulse generator (CPG) before releasing the IMR-LX6 from module standby or to use the SWRST bit in CR to apply a software reset to this module after releasing it from module standby.

Refer to section 9, Module Standby, Software Reset, regarding the settings for module standby and software reset.

### 56.6.2 Software Reset by the Clock Pulse Generator (CPG)

A software reset applied to the IMR-LX6 by the CPG is equivalent to a power-on reset. Accordingly, operation cannot be guaranteed if the IMR-LX6 is reset while it is operating.

Do not attempt access to the registers of the IMR-LX6 while a software resetting is being applied to it.

### 56.6.3 Display list pre-fetch maximum data size information

IMR-LX6 has pre-fetch function of display list.

The maximum pre-fetch data size for IMR-LX6 is 1536-bytes. Therefore, DL must be placed in memory block so that 1536-bytes space is available for maximum pre-fetch after the TRAP/RET/GOSUB instruction.

Note: It is prohibited to compress the display List (DL), so place the DL including maximum pre-fetch data area only to the uncompressed region.

## 56.7 Synchronized system between modules

**(1)** Power must be being supplied to the A3IR domain and the modules must be released from reset state when image processing modules are to be synchronized. For the settings related to power control, refer to section 11, System Controller (SYSC). For controlling the clock signals and resetting, refer to section 9. Module Standby, Software Reset.

**(2)** Synchronization between modules by the WUP and SLP instructions must be started by a module that is not in operation.

**(3)** Do not transmit the WUP signal two or more consecutive times to the same module. Be sure to use the SLP instruction to suspend execution of the display list to allow the reception of a WUP signal from another module and then allow the transmitting module to transmit the subsequent WUP signal. Note, however, that since the SLP instruction is not available in the image signal processor (ISP) and VIN, the ISP or VIN might transmit the WUP signal to the IMR-LX6 two or more consecutive times. When an error is detected by reading the WOVF bit in SR, ensure that operation of this module is ended by a TRAP instruction and then stop the operation of the ISP or VIN.

**(4)** The ISP or VIN transmits the WUP signal regardless of whether or not the IMR-LX6 is activated. Accordingly, when synchronizing the operations between the ISP or VIN and this module, start by activating this module and then transmit the WUP signal form the ISP or VIN.

**(5)** When an error is detected by reading the WERR bit in SR before the RS bit in CR of this module has become 1, do not activate this module but stop the operation of the ISP or VIN. When an error is detected after the RS bit in CR has been set to 1, ensure that operation of this module is ended by a TRAP instruction and then stop the operation of the ISP or VIN.

## 56.8 Data compression

For the DATA compression, refer to 63. FCPR HWM for the details. The transmission size of the IMR-LX6 shall be 256-byte when using DATA compression. We can compress data of Rotator and Scalar only. The compression is possible for below combination only.

**Table 56.30 List of data compression support in IMR-LX6**

| Data precision Rotator and Scalar output Refer Table 56.26, Table 56.27, and Table 56.28 | Non-rotate non-scale output (RSCR.NR1E=1) | Non-rotate scaled down to 1/2 (RSCR.NR2E=1) | Rotated non-scaled data (RSCR.RT1E=1) | Rotated scaled down to 1/2 (RSCR.RT2E=1) |
|---|---|---|---|---|
| 1-byte/pixel (8 bpp) | ○<br>OUTNOR.XWIDTH= **H'1FF or H'0FF** | ○<br>OUTNOR.XWIDTH= **H'1FF** | x | x |
| 2-byte/pixel (10 bpp, 12 bpp, 14 bpp, 16 bpp) | ○<br>OUTNOR.XWIDTH= **H'0FF** | ○<br>OUTNOR.XWIDTH= **H'0FF** | ○<br>OUTROT.YWIDTH= **H'FF** | ○<br>OUTROT.YWIDTH= **H'FF** |

## 56.9 Safety Mechanism

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 56.9.1 Block diagram for safety mechanism.

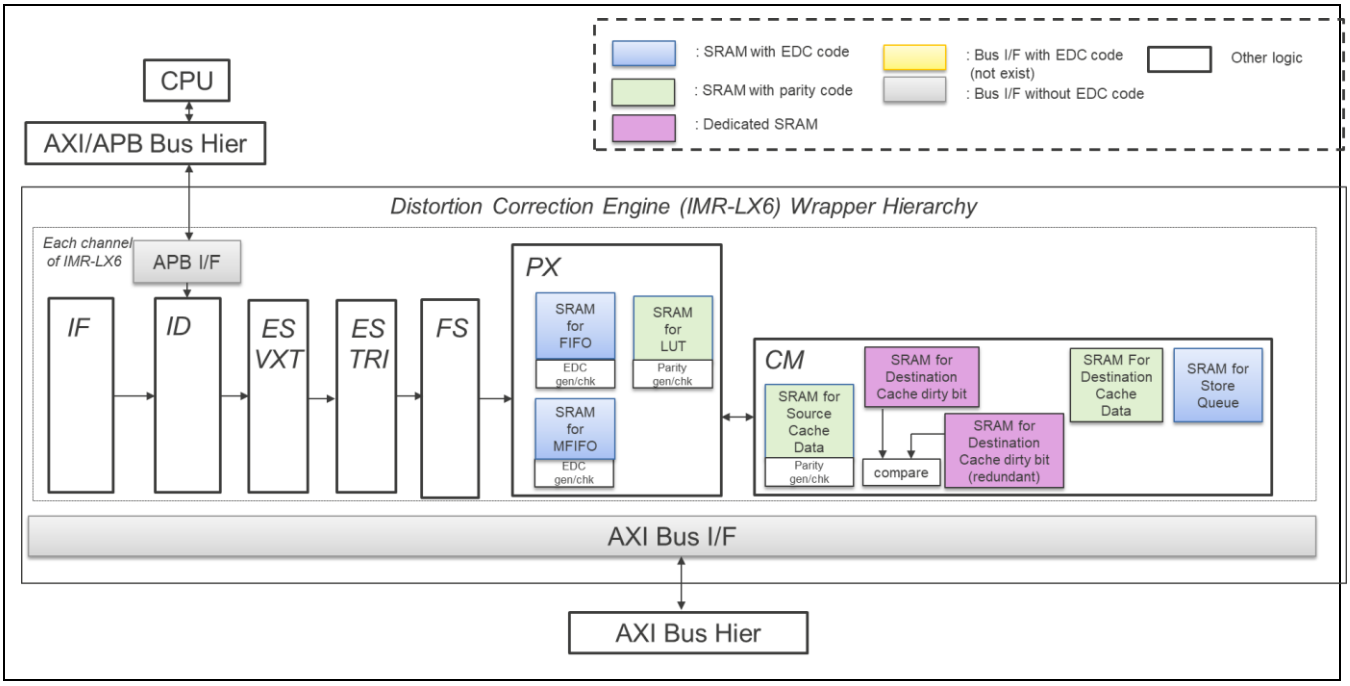



**Figure 56.67 IMR0/1 Block Diagram**

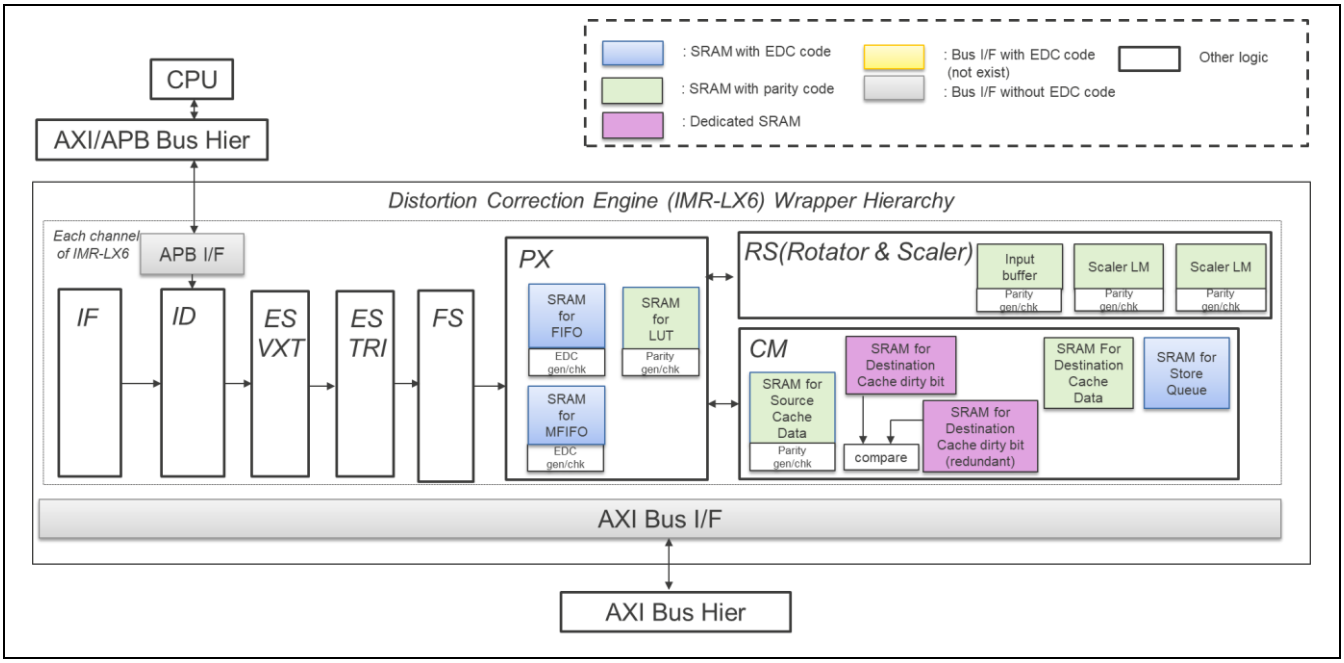



**Figure 56.68 IMR2/3 Block Diagram**

### 56.9.2 Hardware Description

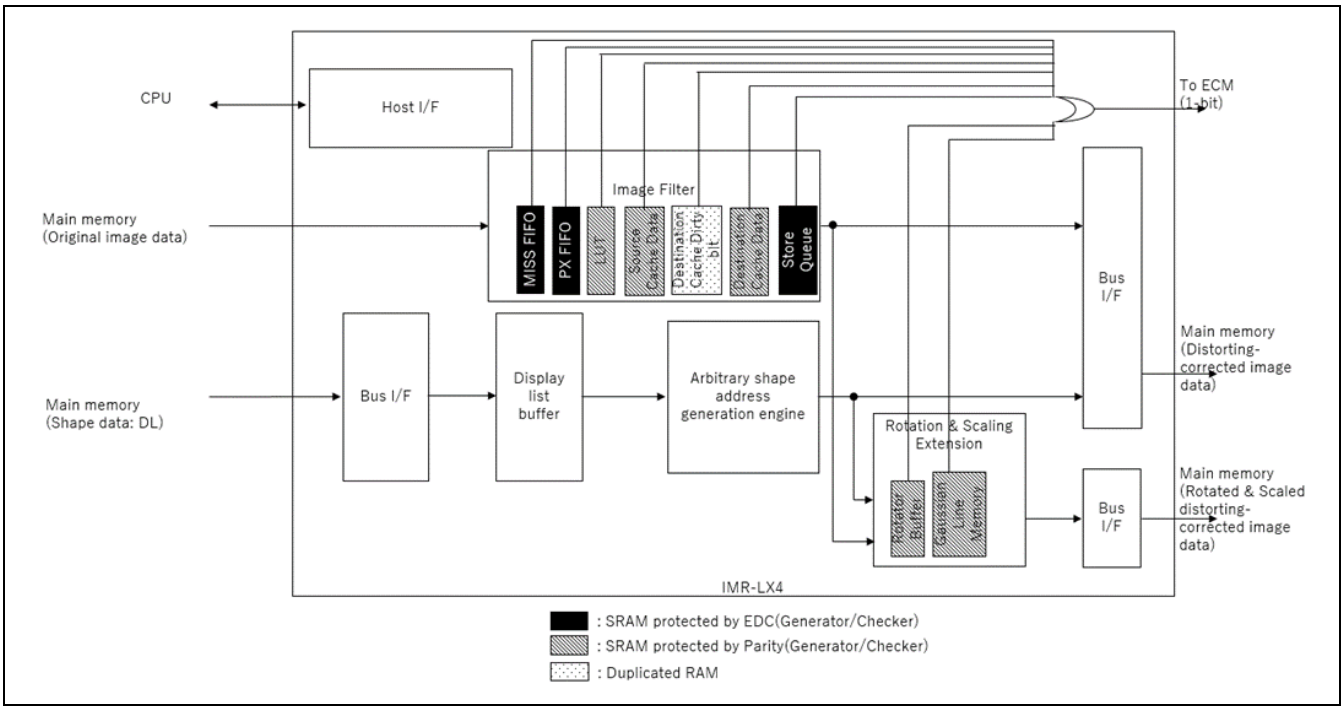



**Figure 56.69 Block diagram of SRAM protection for IMR-LX6**

#### 56.9.2.1 Parity for IMR-LX6

**(1) Overview**

Parity check is applied to the following internal RAMs of IMR-LX6. Parity check is always enabled.

- SRAM for LUT
- SRAM for Source Cache Data
- SRAM for Destination Cache Data
- SRAM for Rotator Buffer*1
- SRAM for Line Memory*1

Note *1: These RAMs are implemented if the channel of IMR-LX6 has Rotator & Scaler extension. Refer to Section 56.1 which channel of IMR-LX6 has the extension.

The parity code is generated and written to SRAMs describe above when data is written to each SRAM. And the data and the corresponding code are compared when it is read out. If the result of the comparison is mismatched, the corresponding bit in the EDC Status Register is set to 1.

The values of all bits including all protection method in the EDC Status Register is output to ECM as the error signal of IMR-LX6.

#### 56.9.2.2 EDC for IMR-LX6

**(1) Overview**

EDC is applied to the following internal RAMs of IMR-LX6. EDC check is always enabled.

- SRAM for PX FIFO
- SRAM for MISS FIFO
- SRAM for Store Queue

The EDC code is generated and written to SRAMs describe above when data is written to each SRAM. And the data and the corresponding code are compared when it is read out. If the result of the comparison is mismatched, the corresponding bit in the EDC Status Register is set to 1.

The values of all bits including all protection method in the EDC Status Register is output to ECM as the error signal of IMR-LX6.

#### 56.9.2.3 Hardware Redundancy for SRAM in IMR-LX6

**(1) Overview**

The following internal RAM of IMR-LX6 is duplicated. This safety mechanism is always enabled.

- SRAM for Destination Cache dirty bit

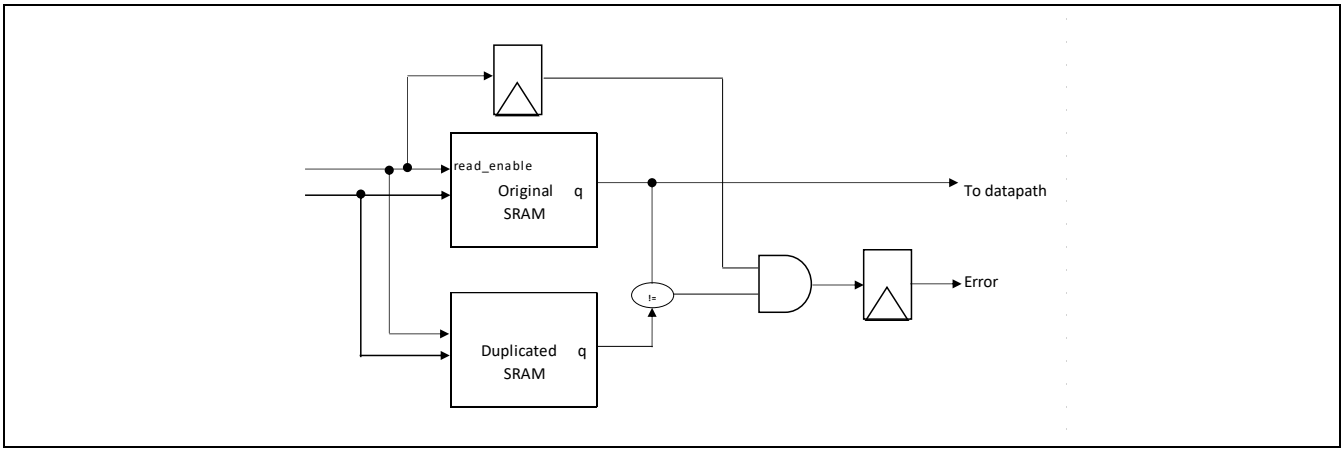



**Figure 56.70 Block diagram Hardware Redundancy for SRAM in IMR-LX6**

The SRAM described in above consists of two SRAM structure: the original SRAM and the duplicated SRAM. The same data is written to the original SRAM and the duplicated SRAM at the same cycle when the write request is issued to the SRAM. And the read data from both the original SRAM and the duplicated SRAM are compared when the SRAM is read out. If the result of the comparison is mismatched, the corresponding bit in the EDC Status Register is set to 1.

The values of all bits including all protection method in the EDC Status Register is output to ECM as the error signal of IMR-LX6.

### 56.9.3 Related registers

**Table 56.31 Register Configuration[*1]**

| Register Name | Abbreviation | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|---|
| EDC Control Register | EDCCR | R/W | (Base Address) + H'0280 | H'0000 0000 | 32 |

| Register Name | Abbreviation | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|---|
| EDC Status Register | EDCSR | R/WC1 | (Base Address) + H'0284 | H'0000 0000 | 32 |

Note *1: EDCSR is reset by CPG software reset in addition to power-on reset.

### 56.9.3.1 EDC Control Register (EDCCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | LUTEDCIJE | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 15 | - | All 0 | R | Reserved<br>These bits are read as 0. The write value should always be 0. |
| 14 | LUTEDCIJE | 0 | R/W | LUT EDC injection enable<br>0: LUT injection is disabled<br>1: LUT injection is enabled |
| 13 to 0 | - | All 0 | R | Reserved<br>These bits are read as 0. The write value should always be 0. |

### 56.9.3.2 EDC Status Register (EDCSR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | — | — | — | — | LMERR | ROTERR | DCERR2 | MFERR | SQERR | PXFIFOERR | DCERR | LUTERR | SCERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved<br>These bits are read as 0. The write value should always be 0. |
| 8 | LMERR | 0 | R/WC1 | Line Memory SRAM Parity checker status.<br>This bit is available if the channel of IMR-LX6 has the Rotator & Scaler extension. Otherwise the write value to this bit should always be 0.<br>Read:<br>0: Line Memory Parity error is not detected<br>1: Line Memory Parity error is detected<br>Write:<br>0: LMERR bit is not cleared<br>1: LMERR bit is cleared to 0 |

| | | | | |
|---|---|---|---|---|
| 7 | ROTERR | 0 | R/WC1 | Rotator Buffer SRAM Parity checker status.<br>This bit is available if the channel of IMR-LX6 has the Rotator & Scaler extension. Otherwise the write value to this bit should always be 0.<br>Read:<br>0: Rotator Buffer Parity error is not detected<br>1: Rotator Buffer Parity error is detected<br>Write:<br>0: ROTERR bit is not cleared<br>1: ROTERR bit is cleared to 0 |
| 6 | DCERR2 | 0 | R/WC1 | Destination Cache Data SRAM Parity checker status.<br>Read:<br>0: Destination Cache Data Parity error is not detected<br>1: Destination Cache Data Parity error is detected<br>Write:<br>0: DCERR2 bit is not cleared<br>1: DCERR2 bit is cleared |
| 5 | MFERR | 0 | R/WC1 | Miss FIFO SRAM EDC checker status.<br>Read:<br>0: Miss FIFO EDC error is not detected<br>1: Miss FIFO EDC error is detected<br>Write:<br>0: MFERR bit is not cleared<br>1: MFERR bit is cleared |
| 4 | SQERR | 0 | R/WC1 | Store Queue SRAM EDC checker status.<br>Read:<br>0: Store Queue EDC error is not detected<br>1: Store Queue EDC error is detected<br>Write:<br>0: SQERR bit is not cleared<br>1: SQERR bit is cleared |
| 3 | PXERR | 0 | R/WC1 | PX FIFO EDC checker status<br>Read:<br>0: PX FIFO EDC error is not detected<br>1: PX FIFO EDC error is detected<br>Write:<br>0: PXERR bit is not cleared<br>1: PXERR bit is cleared |
| 2 | DCERR | 0 | R/WC1 | Destination cache dirty bit SRAM checker status<br>Read:<br>0: Destination cache dirty bit error is not detected<br>1: Destination cache dirty bit error is detected<br>Write:<br>0: DCERR bit is not cleared<br>1: DCERR bit is cleared |

| | | | | |
|---|---|---|---|---|
| 1 | LUTERR | 0 | R/WC1 | LUT SRAM Parity checker status<br>Read:<br>0: LUT Parity error is not detected<br>1: LUT Parity error is detected<br>Write:<br>0: LUTERR bit is not cleared<br>1: LUTERR bit is cleared |
| 0 | SCERR | 0 | R/WC1 | Source Cache SRAM Parity checker status<br>Read:<br>0: Source Cache Parity error is not detected<br>1: Source Cache Parity error is detected<br>Write:<br>0: SCERR bit is not cleared<br>1: SCERR bit is cleared |

# 57.Video Encoding Processor for inter-device video transfer (iVCP1E)

## 57.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The iVCP1E is a video encoder which provides low-latency encoding capability on the basis of H.264/AVC video coding standard. Require software or the library to RENESAS for operating this module.

The iVCP1E has an AXI-bus interface and an APB-bus interface (Figure 57.1).

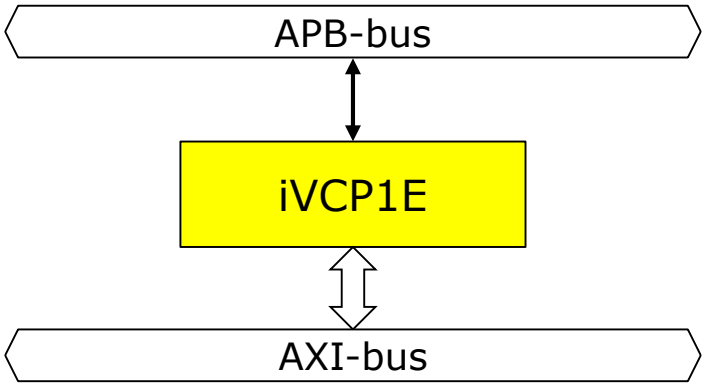



Figure 57.1 Overview of the iVCP1E

### 57.1.1 Features

The iVCP1E has the following features.

- Support for H.264/AVC with 8/10/12-bit depth
- Low-latency encoding with precise CBR (Constant Bit Rate) control
- Support 1920 x 1080 resolution
- Data handling on a picture-by-picture basis

**Table 57.1 Main Function of the iVCP1E**

| Item | Description |
|---|---|
| Encoding standard supported | H.264/AVC<br>Baseline Profile @Level 4.2 (The arbitrary slice order, the flexible macroblock order and the redundant slice aren't supported),<br>Constrained Baseline Profile @Level 4.2,<br>High Profile @Level 4.2,<br>Main Profile @Level 4.2,<br>Constrained High Profile @Level 4.2,<br>Progressive High Profile @Level 4.2,<br>High 10 Profile @Level 4.2,<br>High 10 Intra Profile @Level 4.2<br>High 4:2:2 Profile @Level 4.2<br>High 4:2:2 Intra Profile @Level 4.2<br>High 4:4:4 Predictive Profile @Level 4.2 (*1)<br>High 4:4:4 Intra Profile @Level 4.2 (*1)<br>CAVLC 4:4:4 Intra Profile @Level 4.2 (*1)<br>(*1) The 8-bit, 10-bit, 12-bit depth samples are supported. The YCbCr 4:4:4 is not supported. |
| Maximum pixel rate | 4:2:0/8bit:<br>1920 x 1080p x 120 fps, or 1280 x 720p x 240 fps<br>Others:<br>1920 x 1080p x 60 fps, or 1280 x 720p x 120 fps |
| Maximum bit rate | 200 Mbps @ CBR |
| Maximum bit per picture | 4 Mbit / picture |
| Picture size supported | Minimum size:<br>Horizontal 176 pixels x Vertical 144 lines<br>Maximum size:<br>Horizontal 1920 pixels x Vertical 1088 lines<br>Horizontal size is a multiple of 2 pixels, and vertical size is a multiple of 2 lines.<br>Additionally, 1080 x 1920 and 1088 x 1920 are supported. |
| Input color format | YCbCr 4:2:0, YCbCr 4:2:2 and YCbCr 4:0:0 are supported, in YCbCr 4:2:0, YCbCr 4:2:2 case, the following planar format is supported:<br>YCbCr 4:2:0 planar (Y, Cb and Cr plane)<br>YCbCr 4:2:0 semi-planar (Y and C plane) (*1)<br>YCbCr 4:2:2 planar (Y, Cb and Cr plane) (*2)<br>YCbCr 4:2:2 semi-planar (Y and C plane) (*1)(*2)<br>(*1) Cb/Cr and Cr/Cb interleaving order is supported in the semi-planar format.<br>(*2) YCbCr 4:2:2 is down-sampled to YCbCr 4:2:0 before encoding or YCbCr 4:2:2 encoding. |
| Encode color format | YCbCr 4:2:0 / 4:2:2 / 4:0:0 format |

| Item | Description |
|---|---|
| Input bit depth | 8bit, 10bit and 12bit<br><br>12bit pixels can be truncated to 10bit or 8bit pixels before encoding.<br>10bit pixels can also be truncated to 8bit pixels and encoded as 8bit pixels.<br>A single pixel of 10 or 12bit depth shall be placed in consecutive 2 bytes.<br>In case of chroma bit depth is 8bit, luma bit depth can be 8bit, 10bit or 12bit.<br>For others, they should be the same. |
| Encode bit depth | 8bit, 10bit or 12bit<br><br>Luma pixels and chroma pixels shall have an identical bit depth. |
| Picture order count type | Type 0, Type 1 and Type 2 |
| Bit stream format | Video elementary stream as follows:<br>H.264 NAL units which can include SPS, PPS, AUD, VCL and Filler.<br>SPS can include VUI which signals a low-latency indicator.<br>Function to insert the emulation prevention byte in the H.264 bit stream is supported. |
| Picture structure | Frame structure |

### 57.1.2 Register Configuration

Below table shows the register configuration related to EDC for SRAM which the host CPU can read and write directly. All registers are writable and readable by only 32-bit unit.

**Table 57.2 Register Configuration of iVCP1E**

| Register Name | Abbreviation | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| EDC control register | VP_EDC_CTRL | R/W | H'FEA0 03A0 | H'0000 0000 | 32 |
| EDC status register | VP_EDC_STA | R/W | H'FEA0 03A4 | H'0000 0000 | 32 |

### 57.1.3 Connected Module

Table 57.3 shows modules connected to the iVCP1E.

**Table 57.3 Connected Module**

| Connected Module | Function of the Connected Module |
|---|---|
| CPG | Reset and Clock Stop Signal Supplier |
| INTTOP | Interrupt Controller |
| ECM | Error Interrupt Controller |

## 57.2 Register Description

### 57.2.1 EDC Control Register (VP_EDC_CTRL)

This register controls the behavior of a SRAM EDC checking function and injecting dummy EDC error to test itself.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | FERRN | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | EDCD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | - | All 0 | R | Reserved<br>These bits are read as 0. The write value shall always be 0. |
| 24 | FERRN | 0 | R/W | This bit forces the EDC error notification signal (errreq_iv1es_ecc_p) to be asserted.<br>0: Does not assert EDC error notification signal forcibly.<br>1: Asserts EDC error notification signal forcibly. |
| 23 to 1 | - | All 0 | R | Reserved<br>These bits are read as 0. The write value shall always be 0. |
| 0 | EDCD | 0 | R/W | This bit specifies the disable flag of EDC checking function.<br>0: Enables SRAM-EDC checking function.<br>1: Disables SRAM-EDC checking function. |

### 57.2.2 EDC Error Status Register (VP_EDC_STA)

VP_EDC_STA indicates the status of a SRAM-EDC error. If EDC error occurs, the SRAM-EDC error status bit (EDC bit) is set to 1. iVCP1E issues the error notification signal to the external error controller if the VP_EDC_CTRL.EDCD is 0 and the EDC bit is 1.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | EDC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | - | All 0 | R | Reserved<br>These bits are read as 0. The write value shall always be 0. |
| 0 | EDC | 0 | R/W | This bit indicates SRAM-EDC error.<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it. |

## 57.3 Operation

The LSI supports module standby mode in which clock supply to the iVCP1E is stopped. Don't access iVCP1E registers during module standby mode.

### 57.3.1 Module Standby Mode

#### 57.3.1.1. Transition to Module Standby Mode

1. Confirm that iVCP1E is not encoding a picture.
2. Set MSTP16 bit to 1 in MSTPCR6 of the clock pulse generator (CPG) module.
3. Confirm that MSTPST16 bit to 1 in MSTPSR6 of CPG module.

For detail of registers of CPG, refer to section 8.

#### 57.3.1.2. Cancellation of Module Standby Mode

1. Set MSTP16 bit to 0 in MSTPCR6 of the clock pulse generator (CPG) module.
2. Confirm that MSTPST16 bit to 0 in MSTPSR6 of CPG module.
3. Set SRT16 bit to 1 in the Software Reset Register 6 (SRCR6) of CPG module.
4. Set SRTCLR16 to 1 in Software Reset Clearing Register 6 (SRSTCLR6) of CPG module.

For detail of registers of CPG, refer to section 8.

## 57.4 Usage Notes

### 57.4.1 Limitations on Software Reset, Module Standby and power-shutoff

Before executing the following operations for the iVCP1E, confirm that the iVCP1E is not encoding a picture. If the iVCP1E is encoding a picture, wait completion of encoding the picture before executing the following operations. Moreover, It's necessary to supply clock when execute Software Reset by CPG.

- Execute Software Reset by CPG
- Enter Module Standby (stop clock supply) by CPG
- Enter power-shutoff status (stop power supply) by SYSC

## 57.5 Safety Mechanism

### 57.5.1 Overview

EDC protection is applied to all internal SRAMs in iVCP1E.

### 57.5.2 Related Registers

Below table shows the registers related to EDC for SRAM.

**Table 57.4 Related Registers**

| Register Name | Abbreviation |
|---|---|
| EDC control register | VP_EDC_CTRL |
| EDC status register | VP_EDC_STA |

Behavior of notification signal can be turned on or off by VP_EDC_CTRL register. If VP_EDC_CTRL.EDCD is set to 0, the notification is enabled. If it is set to 1, the error notification signal is not asserted, and it is always negated.

When an EDC error occurs, the status register VP_EDC_STA.EDC becomes 1. If VP_EDC_STA.EDC is 1 and VP_EDC_CTRL.EDCD is 0, the error notification signal continues to be asserted. When software writes 0 to VP_EDC_STA.EDC, it is cleared and the error notification signal is also negated.

Writing 1 to VP_ EDC_CTRL.FERRN forces the error notification to be asserted. When 0 is written, the forced assertion is canceled. This bit can be used for checking connection of EDC error notification signals between iVCP1E and ECM without activating iVCP1E.

### 57.5.3 Hardware Description

iVCP1E checks all internal SRAMs for data corruption. iVCP1E generates a checksum (EDC) at the time of data writing to SRAM and stores it with data in SRAM. When iVCP1E read from the SRAM, it checks whether the read data matches the EDC. If they do not match, the notification signals to the error control module (ECM) is asserted.

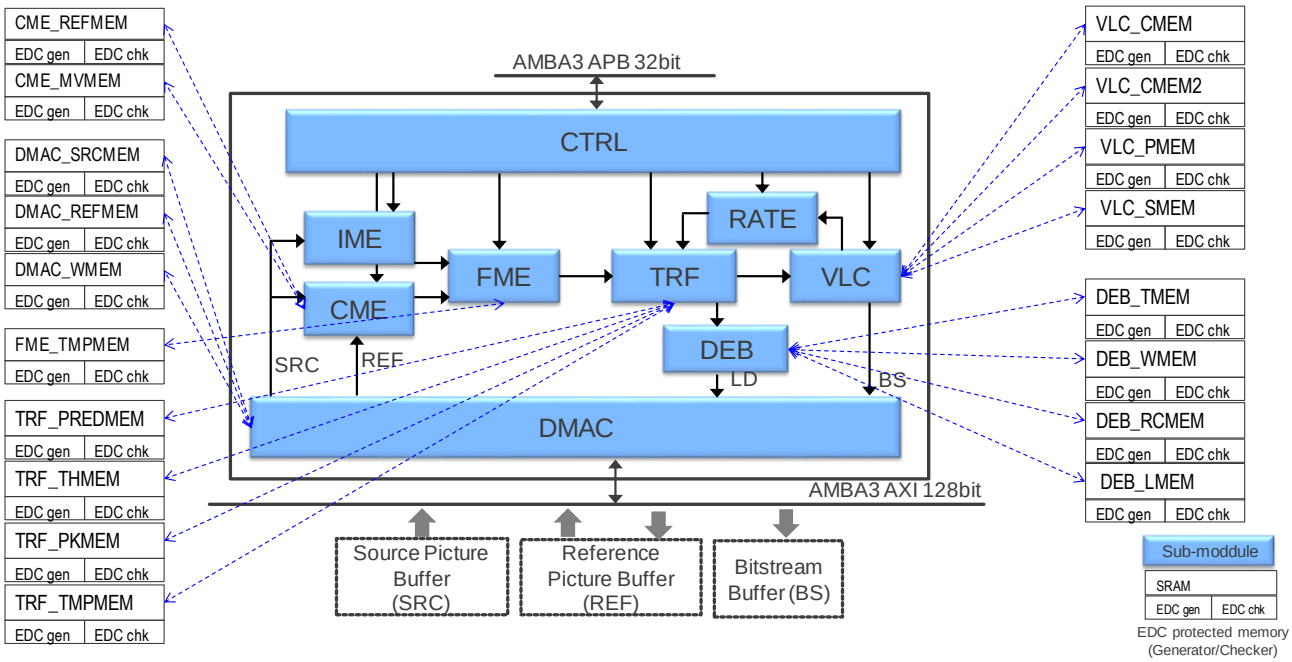



**Figure 57.2 EDC Protected SRAMs and Error Notification Signals**

# 58. Image Processing Unit (IMP-X7)

## 58.1 Overview

The image processing unit extended 7 (hereafter referred to as IMP-X7) captures image data from the DDR-SDRAM and processes images. The IMP-X7 is a collective term for the extended image processing unit incorporated in the fourth-generations R-Car series products.

### 58.1.1 Features

The IMP-X7 which includes the IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, IMP CNN, IMP DTA and VDSP acts as an accelerator for image processing at high speed. Co-operation between hardware and software reduces the load on the CPU and improves the efficiency of image processing. The number of cores in each IP depends on the product. For details, refer to Table 58.1.

**(1) IMP core**

The IMP core is capable of local parallel operations: multiple image-processing functions are run on a basis of 1 pixel/cycle or 2 pixels/cycle. High-speed pre-processing, post-processing, and histogram processing (two types can be processed simultaneously) are achieved by a pipelined architecture. The operating frequency is improved, and features are added compared to the IMP core included in the IMP-X6 of the third-generation R-Car series products.

**(2) CVe**

Core with a programmable parallel processing engine that is capable of processing operations in 32-bit integer or 32-bit floating point precision. The operating frequency is improved, and features are added compared to the CVe included in the IMP-X6 of the third-generation R-Car series products.

**(3) IMP DMAC**

A DMA controller which can improve the efficiency of accessing the DDR-SDRAM by fast copying of data to IMP-X7 internal memory and by synchronization with other IPs inside IMP-X7.

**(4) Slim-IMP DMAC**

A DMA controller in which usable functions are limited compared to the IMP DMAC.

**(5) IMP PSC**

A scaling engine which can generate reduced-size image data (up to 4 variations at once). The scaling ratios for horizontal and vertical can be set independently. Bilinear filtering is used as the image interpolation algorithm.

**(6) IMP CNN**

The IMP CNN module is a module enabling a fast execution of a Convolutional Neural Network with optional enabling of Rectified Linear Units (RELU) and pooling option for up to 16 input DMA and 32 output DMA channels with a simple instruction set.

**(7) IMP DTA**

Debug and Trace Agent supports debugging of the IMP-X7 sub system. To debug the synchronization between the different components, the Debug and Trace Agent can interrupt processing of each component or break all components.

**(8) VDSP**

Refer VDSP HWM for details.

### 58.1.2 Specifications of the IMP-X7 by Products

Table 58.1 shows the specifications of the IMP-X7 in the fourth-generation R-Car series products.

**Table 58.1 Specifications of the IMP-X7 by Products**

| Function Name | Specification | Fourth-Generation R-Car Series Products (Unit Name in each Product) |
|---|---|---|
| | | IMP-X7-V4M |
| AXI master | AXI Master 0 Bus width | 256 |
| | Operating frequency | SVD1_IRφ |
| | AXI Master 1 Bus width | 256 |
| | Operating frequency | SVD1_IRφ |
| AXI slave | Bus width | 64 |
| | Operating frequency | SVD1_IRφ |
| IMP-X7 internal memory [Scratchpad memory (SPM)] | Capacity | 2M bytes (256 Kbyte × 8 banks) SPMI |
| | Operating frequency | SVD1_IRφ |
| | Capacity | 2M bytes + 0.5 M bytes SPMC0 + SPMC1 |
| | Operating frequency | IMPAD1φ |
| IMP core | Number of cores | 2 |
| | Operating frequency | SVD1_IRφ |
| CVe | Number of clusters | 4* |
| | Operating frequency | SVD1_IRφ |
| IMP DMAC | Number of cores | 2 |
| | Operating frequency | SVD1_IRφ |
| IMP PSC | Number of cores | 1 |
| | Operating frequency | SVD1_IRφ |
| IMP CNN | Number of cores | 1 |
| | Operating frequency | IMPAD1φ |
| Slim-IMP DMAC | Number of cores | 2 |
| | Operating frequency | IMPAD1φ |
| IMPDTA | Number of cores | 1 |
| | Operating frequency | SVD1_IRφ |
| VDSP | Number of cores | 2 |
| | Operating frequency | IMPAD1φ |
| | AXI master | SVD1_IRφ /256-bit<br>SVD1_IRφ /128-bit |
| | AXI slave | SVD1_IRφ /64-bit |

Note: CVe clusters 1, 2 and 3 are light CVe with reduced area and reduced power.

### 58.1.3 Block Diagram

Figure 58.1 respectively show a block diagram of the IMP-X7 in the R-Car Gen4.

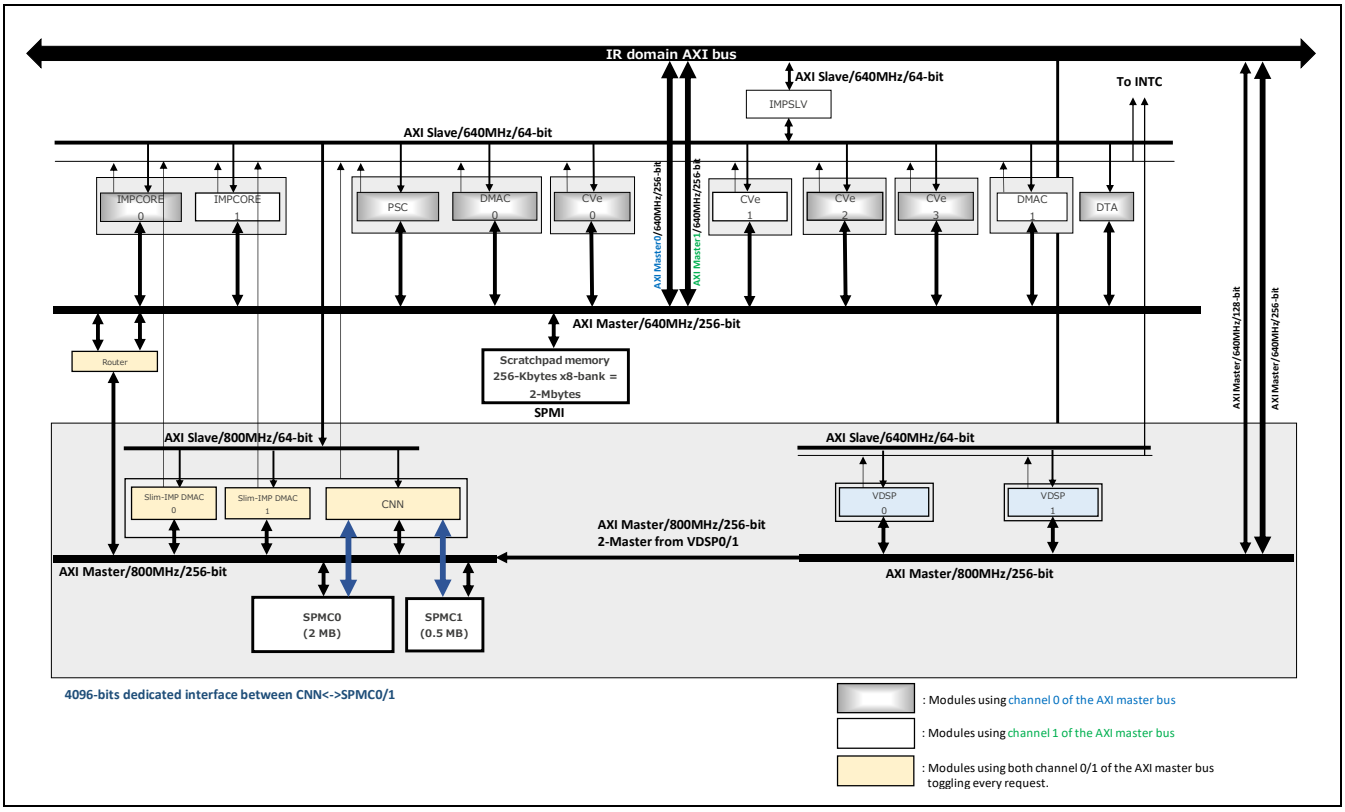



**Figure 58.1 Block Diagram of IMP-X7**

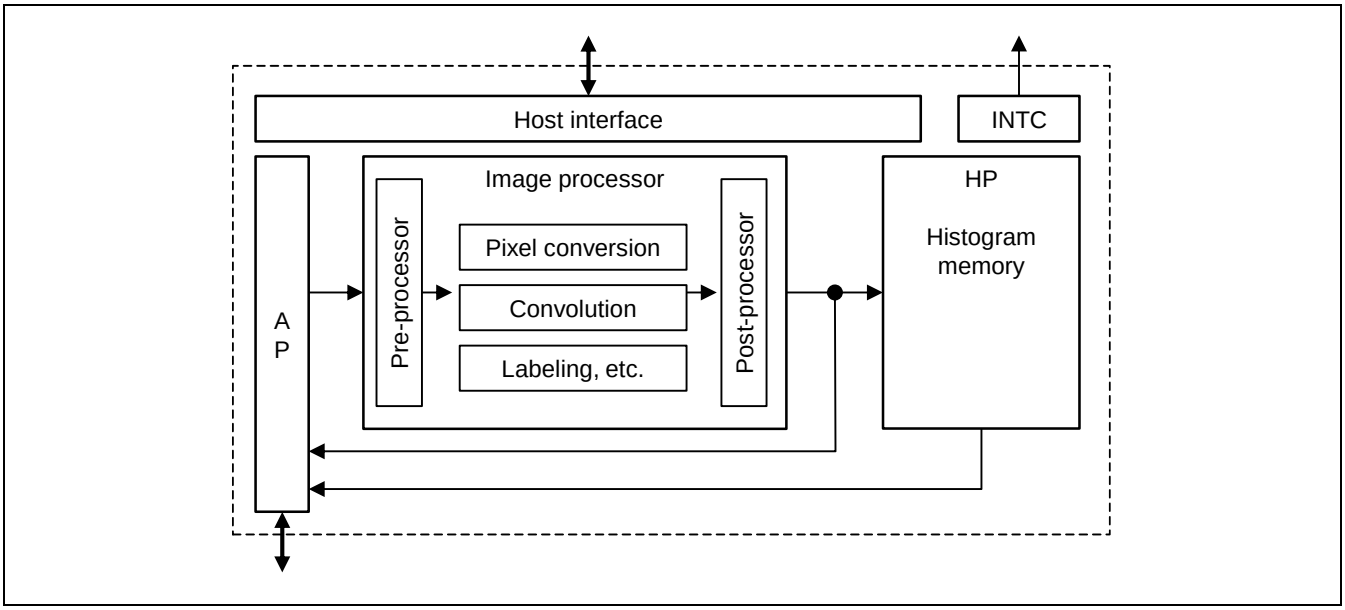



**Figure 58.2 Block Diagram of IMP core**

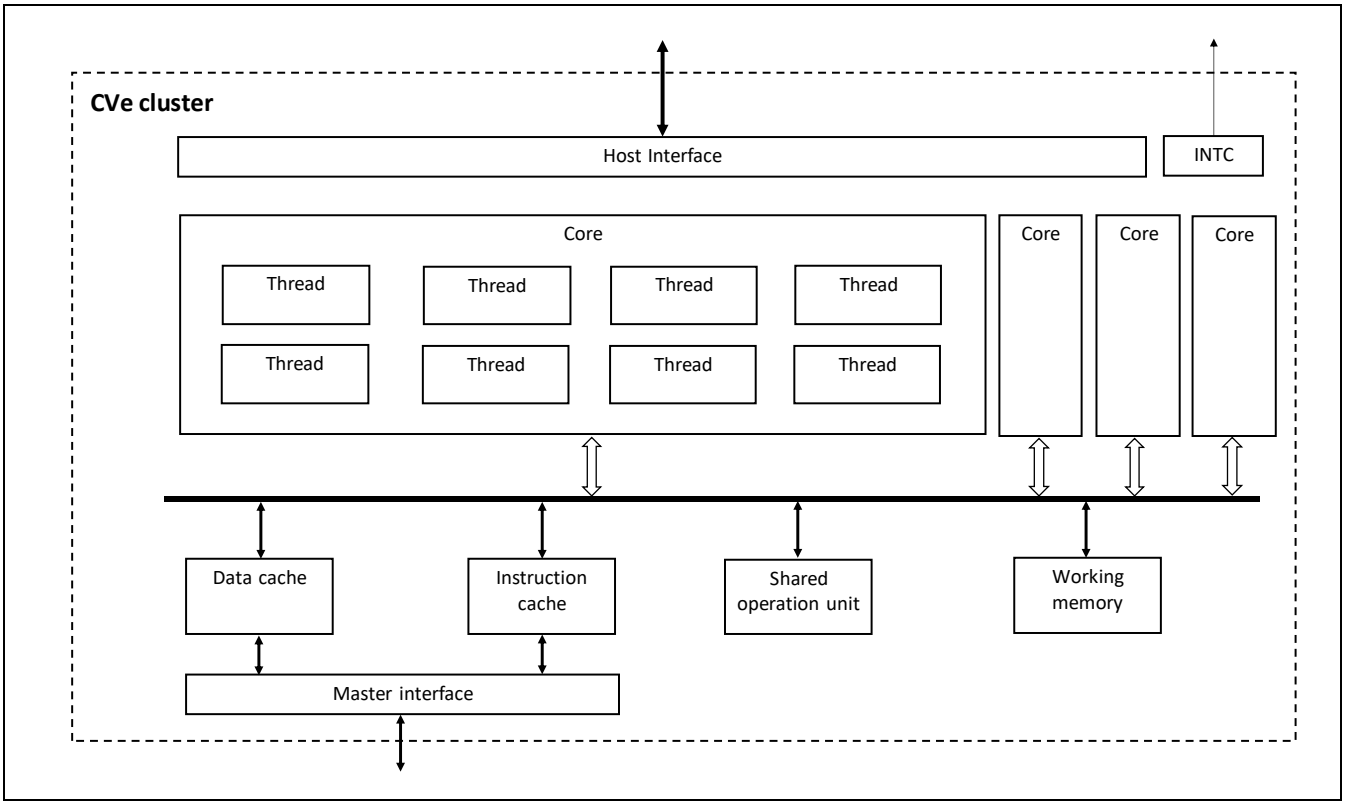



**Figure 58.3 Block Diagram of CVe**

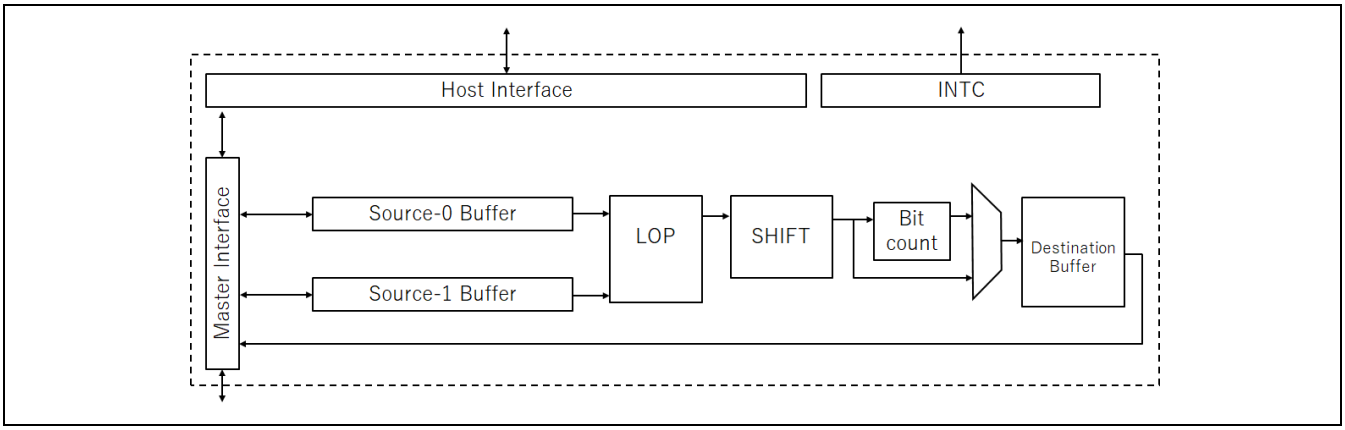



**Figure 58.4 Block Diagram of IMP DMAC**

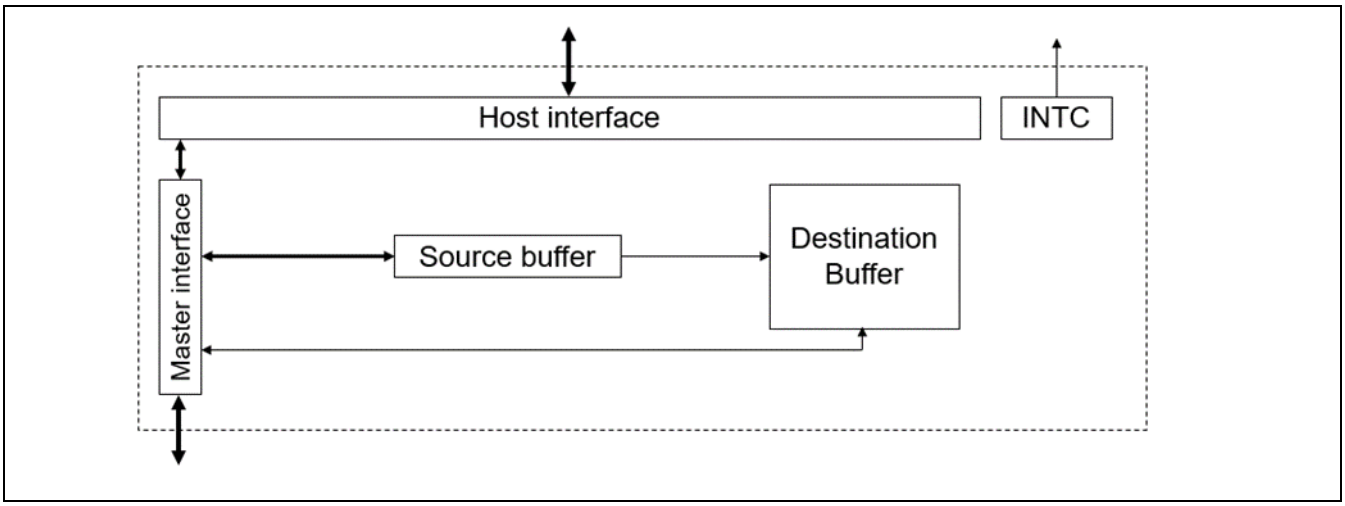



**Figure 58.5 Block Diagram of Slim-IMP DMAC**

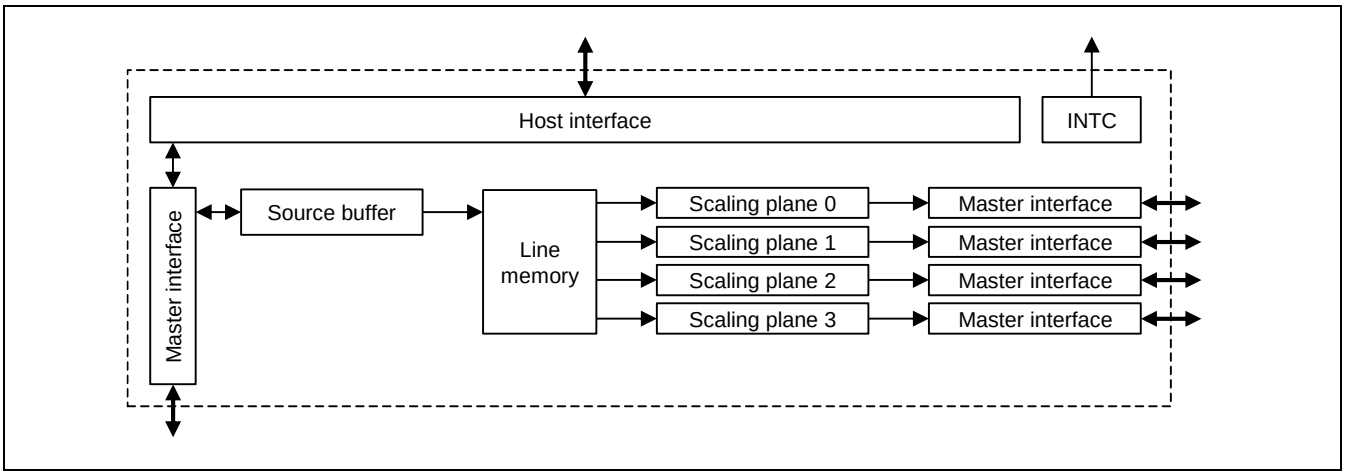



**Figure 58.6 Block Diagram of IMP PSC**

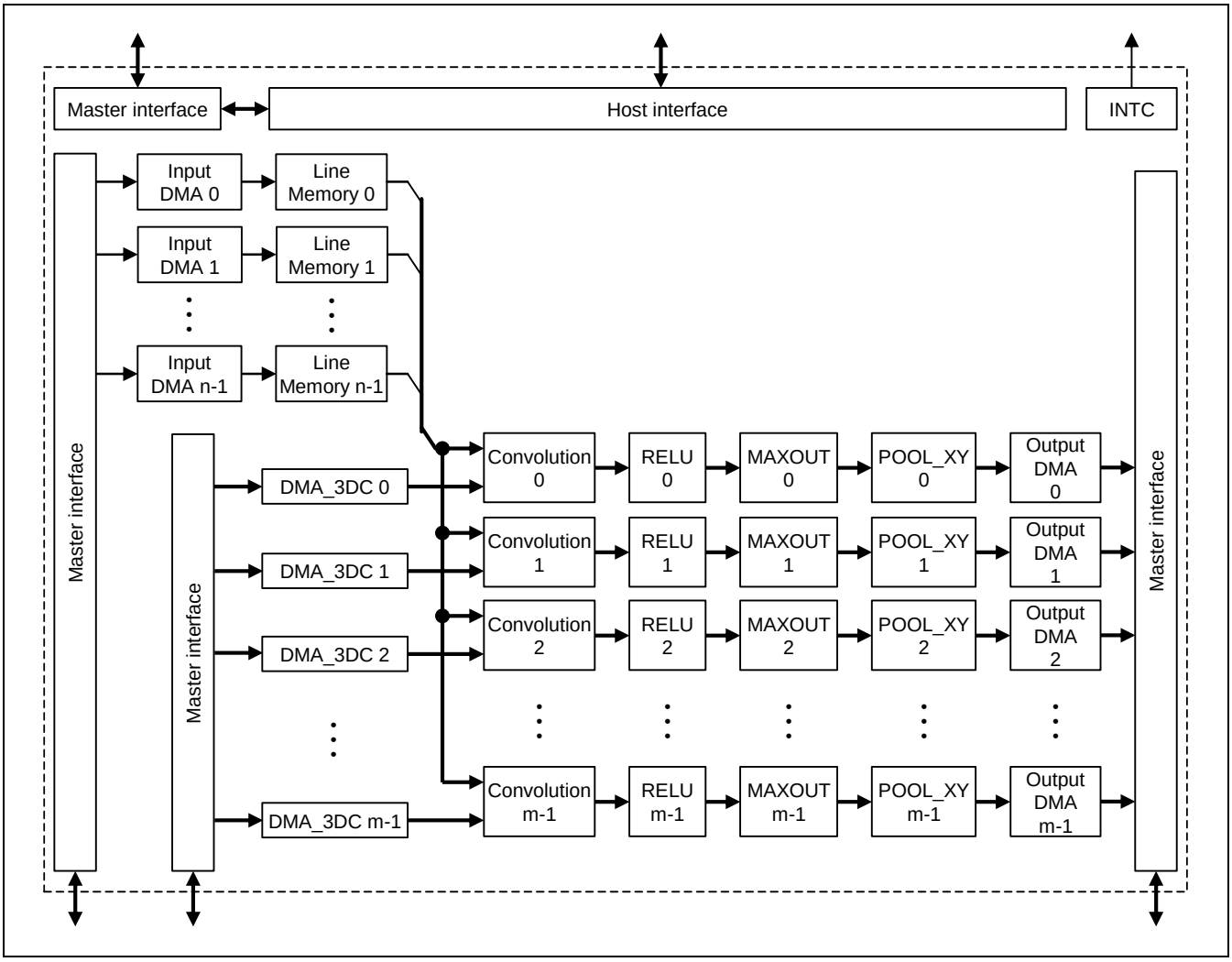



**Figure 58.7 Block Diagram of IMP CNN***

Note: * Number of input DMA channels n = 16, number of DMA3DC, Convolution, RELU, MAXOUT, POOL_XY and output DMA channels m = 32

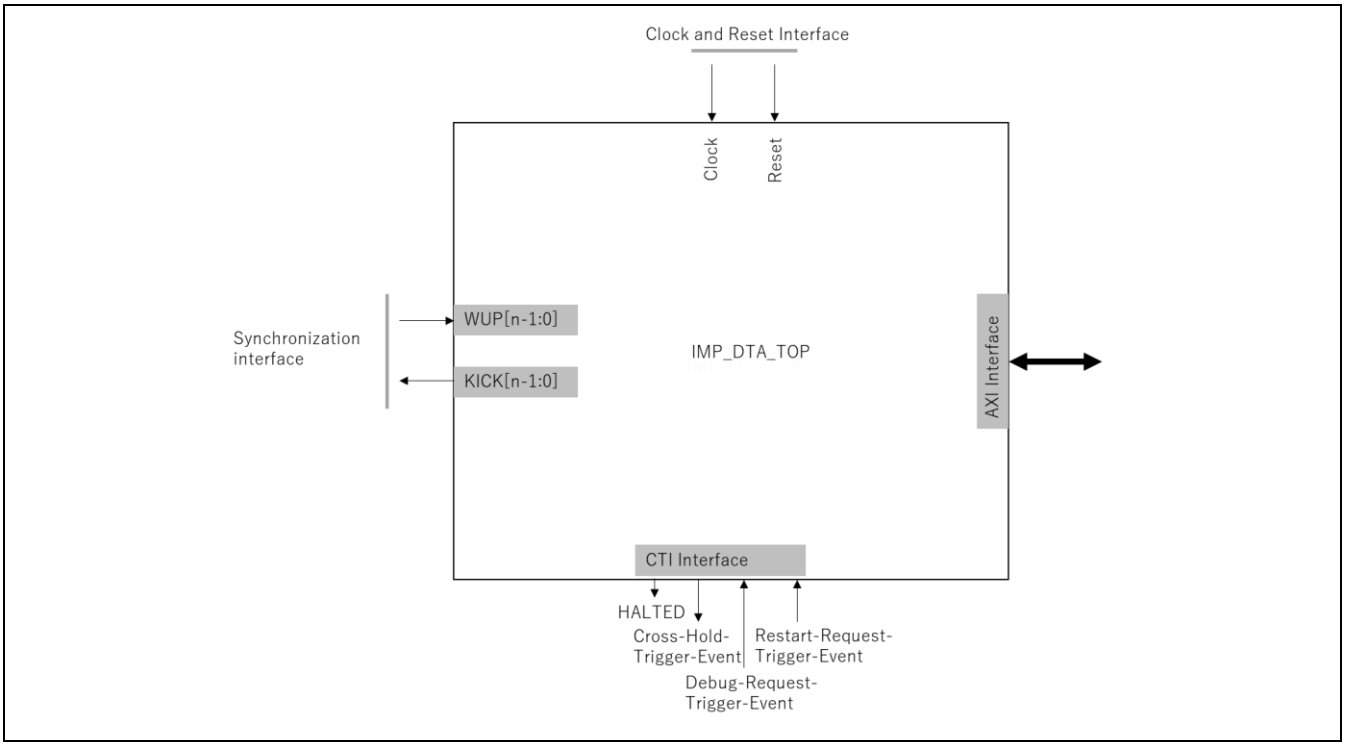



**Figure 58.8 Block Diagram of IMPDTA**

**Refer U39.VDSP HWM for Block diagram.**

The main features of each module in the IMP-X7 are shown below.

- IMP-X7:
  This is an image processing module that incorporates the IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, IMP CNN, IMPDTA, and processes images in parallel.
- AXI master (ch0/1):
  Performs arbitration of AXI master access from the IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, and IMP CNN. The AXI master of the IMP-X7 has two channels.
- AXI slave (IMPSLV):
  Performs routing for AXI slave access in the IMP-X7.
- Interrupt control logic:
  Controls output of various interrupt signals from each internal core. The grouping can be realized up to **7**.
- IMP-X7 internal memory (Scratchpad memory):
  By storing the intermediate processing data, e.g., the efficiency of bus access can be improved. Scratchpad memory or IMP-X7 internal memory is also named as SPMC0/1 Scratchpad memory dedicated to IMP CNN and Slim-IMP DMAC) and SPMI (Scratchpad memory dedicated to other IMP IPs like IMP core, CVe, IMP PSC, IMPDMAC).
- IMP core
  - Host interface:
    This is an interface module for AXI slave access. Handles slave access (in 4-byte units only) set via registers.
  - Interrupt control (INTC):
    Controls output of various interrupt signals.
  - Address processor (AP):
    Generates various addresses in accordance with register settings, provides data to the image processor through AXI master access, and writes the result of processing to the DDR-SDRAM through AXI master access.
  - Image processor*[1]:
    Processes image data loaded from the DDR-SDRAM to the IMP in accordance with register settings in the order of pre-processing → image processing (e.g., density conversion, kernel processing and labeling) → post-processing. The result of processing is sent to the HP or AP before being written to the DDR-SDRAM.
  - Histogram processor (HP)*[2]:
    Processes various types of histograms. Two cores of the histogram processing modules in the HP can execute two types of histogram processing simultaneously.
- CVe
  - Host interface:
    This is an interface module for AXI slave access. Handles AXI slave access (in 4-byte units only) set via registers.
  - Interrupt control (INTC):
    Controls output of various interrupt signals.
  - Thread (==Unified processor)
    This is a processor for loading, storing and operating pixel data in accordance with a designated program. 32-bit integer and 32-bit floating-point operations are supported.
  - Data cache:
    This is the caching logic for the data which is processed by Thread. And it is also the input/output interface of the DDR-SDRAM access.
  - Instruction cache:
    This is the caching logic for the instruction which is issued by Thread.
  - Shared operation unit:
    Arithmetic logical unit shared by all Threads.
  - Working memory:
    This is a storage unit which Thread can use to fast load/store intermediate processing data.

- IMP DMAC and Slim-IMP DMAC:
  This is a DMA controller that performs burst transfer to transfer data between the DDR-SDRAM and IMP-X7 internal memory for achieving efficient bus access.
  — Host interface:
  This is an interface module for AXI slave access. Handles AXI slave access (in 4-byte units only) set via registers.
  — Interrupt control (INTC):
  Controls output of various interrupt signals.
  — Logical Operation (LOP) *1:
  Logical operations using two pieces of input image data is possible.
  — Bit count*1:
  The number of 1s in image data can be counted in 8-, 16-, or 32-bit units.
  — SHIFT*1:
  Applying shifting, rounding, and saturation processing to the outputs from logical operations.

Note: 1. Slim-IMP DMAC does not support LOP, Bit count and SHIFT functions.

- IMP PSC:
  This is an image scaling engine. It can generate up to 4 reduced-size image data variations at once.
  — Host interface:
  This is an interface module for AXI slave access. Handles AXI slave access (in 4-byte units only) set via registers.
  — Interrupt control (INTC):
  Controls output of various interrupt signals.
  — Line memory:
  This is the storage unit to realize the bilinear filtering.
  — Scaling plane0-3:
  Due to the register settings, it can be generated 4 reduced-size image data in parallel.
- IMP CNN:
  — Host interface:
  This is an interface module for AXI slave access. Handles AXI slave access (in 4-byte units only) set via registers.
  — Interrupt control (INTC):
  Controls output of various interrupt signals.
  — Input DMA (DMAI):
  This module transfers data from the IMP-X7 internal memory (SPMC0) through the CNN <->SPMC0 dedicated interface to the line memory.
  — Line memory:
  Line memory stores the row-data to perform the next process.
  — Convolution:
  Convolution module is responsible for performing the convolution for up to 16 input DMA and 32 output DMA channels.
  — RELU:
  REctified Linear Unit (RELU) module is responsible for performing the activation function.
  — MAXOUT:
  MAXOUT operation is responsible for performing a selection operation by taking the maximum value out of 32 neighbored channels.
  — POOL_XY (POOL):
  POOL operation is responsible for performing a selection operation by taking the maximum value out of an XY area.

— Output DMAO:

This module transfers data from the output of the IMP CNN module to the IMP-X7 internal memory (SPMC0/1) through the CNN <->SPMC0/1 dedicated interface.

— DMA3DC:

The DMA3DC is responsible for the intermediate data transfer from IMP-X7 internal memory (SPMC1) to convolution module of IMP CNN through the CNN <->SPMC1 dedicated interface.

### 58.1.4 Connected Module

**Table 58.2 Connected Modules**

| Connected Module | Function of the Connected Module |
|---|---|
| AP-system core | Access to registers |
| Clock pulse generator (CPG) | Supply of clock signals |
| System controller (SYSC) | Power shutdown |
| Module standby | Supply and control of clock signals |
| Software reset | Supply and control of software reset signal |
| INTC-AP | Interrupt |
| External bus controller (DBSC4) | DDR-SDRAM access |
| Image renderer light extended 6 (IMR-LX6) | Distortion correction engine.<br>Drawing processor with a simple instruction system.<br>(IMP-X7 and IMR-LX6 can WUP/SLP) |
| Image signal processor (ISP) | Image pixel correction engine.<br>Correct the input image signal from the video input interface.<br>(IMR-LX6 and ISP are connected through IMP-X7) |

## 58.2 Operation

### 58.2.1 Image Processing (IMP Core)

Table 58.3 lists some of the image-processing functions supported by the hardware of the IMP core of the IMP-X7. All IMP cores inside IMP-X7 have the same functionality. Pipeline processing in the order of pre-processing → other image processing (those following post-processing in Table 58.3) → post-processing, for example, is possible. Detailed functions are specifiable for each of the processes.

**Table 58.3 Image-Processing Functions (Excerpt)**

| Feature | Details |
|---|---|
| Pre-processing (by the pre-processor) | Masking, density conversion, bit inversion, normalization, and differential absolute value |
| Post-processing (by the post-processor) | Absolute-value processing, normalization, density conversion, masking, and square-root operation |
| Pixel conversion*1 | Through mode, addition and subtraction of constants, output of maximum and minimum values, etc. |
| Arithmetic operations between images*1 | Addition, subtraction, multiplication, division, modulo, squared addition, subtraction, and output of the maximum and minimum values, etc. |
| Logical operations between images*1 | Logical multiplication (AND), logical addition (OR), exclusive OR, logical AND with a constant, logical OR with a constant, Logical AND bit count, etc. |
| Convolution*2 | 3 × 3, 1 × 9, 4 × 4, 1 × 16, 5 × 5, 1 × 25, or 7 × 7 kernel operation |
| Rank filtering | 3 × 3 kernel operation and 5 × 5 min./max. processing |
| Binary image matching filtering | The number of bits matching the template is output in 9 × 9 units. |
| Normalized correlation operation and cumulative difference operation | Normalized correlation operation and cumulative difference operation |
| Local parallel operation (normalized correlation/difference) | Extra cumulative difference operation of the normalized correlation operation is locally performed as 25 parallel operations. |
| Labeling | Labeling of binary images |
| Form conversion | Thinning and erosion |
| Binary-format filtering | 23 types of image processing for binary images are performed in a pipeline. |
| Histogram processing*2 | Cumulative density, extraction of maximum and minimum values, and projection |
| Others | Color-support arithmetic operation, Sobel-filter support, local difference operation, integral image, peripheral increment sign, edge code calculation, support for histograms of oriented gradients (HOG) feature calculation, etc. |

Notes: 1. Handling of two pixels per cycle is possible.
2. Handling of two pixels per cycle is possible but some restrictions apply.

### 58.2.2 CV engine (CVe)

The CVe is a programmable parallel processing engine. It is capable of processing operations in 32-bit integer or 32-bit floating point precision. For details, refer to the U- Chapter of the CV engine.

### 58.2.3 DMA Transfer (IMP DMAC)

The IMP DMAC is a DMA controller with a read/write function specific for the IMP-X7. Any IP can synchronize with the IMP DMAC via the IMP-X7 internal memory, the efficiency of access to the DDR-SDRAM can be improved.

**Table 58.4 IMP DMAC Functions**

| Feature | Details |
|---|---|
| Processing size | 8 bpp: 8192 × 8192 pixels |
| | 16 bpp: 4096 × 4096 pixels |
| | 32 bpp: 2048 × 2048 pixels |
| Access addressing | Linear addressing |
| | Tile addressing |
| | Linear–tile addressing interconversion*1 |
| Pixel operation*2 | 16 types of logical operations |
| | Bit count, Logic shift, Rounding, Saturation |

Notes: 1. A tile of 16 bytes × 4 lines which is supported by the IMR-LX6 can be processed.
2. Usage of the pixel operation function is prohibited during linear–tile addressing interconversion.

### 58.2.4 DMA Transfer (Slim-IMP DMAC)

The Slim-IMP DMAC is a DMA controller with a read/write function specific for the IMP-X7. By co-operating the IMP CNN processing with the Slim-IMP DMAC via the IMP-X7 internal memory, the efficiency of access to the DDR-SDRAM can be improved.

**Table 58.5 Slim-IMP DMAC Functions**

| Feature | Details |
|---|---|
| Processing size | 8 bpp: 8192 × 8192 pixels |
| | 16 bpp: 4096 × 4096 pixels |
| | 32 bpp: 2048 × 2048 pixels |
| Access addressing | Linear addressing |
| Pixel operation | Pixel operations are not supported |

### 58.2.5 Image scaling (IMP PSC)

The IMP PSC is a scaling engine to generate the reduced-size image data. It can generate up to 4 reduced-size image data variations at once. IMP PSC stores the generated image data to DDR-SDRAM or IMP-X7 internal memory.

**Table 58.6 IMP PSC Functions**

| Feature | Details |
|---|---|
| Image format | unsigned 8/10/12/16-bit |
| | signed 8/16-bit |
| Processing size | unsigned/signed 8-bit: 4096 × 4096 pixels |
| | others: 2048 × 4096 pixels |

| Feature | Details |
|---|---|
| Scaling ratio | Scaling ration is 0.5 to 1.0<br>- The scaling ratios for horizontal and vertical can be set independently.<br>- 20-bit fixed point expression, 1/1048576 step. |
| Image interpolation | Bilinear filtering |

### 58.2.6 Convolutional Neural Network (IMP CNN)

The IMP CNN is an engine to enable a fast execution of a Convolutional Neural Network. It supports up to 16 input DMA and 32 output DMA channels. Next to the convolution with a size of up to 5×5, optional a Rectified Linear Units (RELU) or pooling over output channels and x/y direction is supported. The IMP CNN supports the following features:

**Table 58.7 IMP CNN Functions**

| Function | Details |
|---|---|
| Image format | unsigned 8-bit<br>signed 8/16-bit |
| Processing size | Tile size up to 2048 line times 128 pixels |
| Input channel | Up to 16 channel (Input DMAI) |
| Output channel | Up to 32 channel (Output DMAO) |
| Convolution size | Up to 5 x 5 |
| Magnification setting | 1/5, 1/4, 1/3, 1/2, 1, 2, 3, 4, 5 |

## 58.3 Bus Access Latency Reduction

The internal bus of the IMP-X7 has the following features to reduce the latency for bus access to outside of the IMP-X7.

### 58.3.1 IMP-X7 internal memory (Scratchpad Memory)

An SRAM is included in the internal bus of the IMP-X7. The bus access latency can be reduced by making use of the IMP-X7 internal memory so that the IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, and IMP CNN perform processing efficiently. These are commonly named as SPMC0/1 (Scratchpad memory dedicated to IMP CNN and Slim-IMP DMAC) and SPMI (Scratchpad memory dedicated to other IMP IPs like IMP core, CVe, IMP PSC, IMPDMAC). Refer U34.IMP-X7_System chapter for the details of SPMI and SPMC.

## 58.4 Starting and Stopping Module Operation

Image processing starts to operate when mode settings are made in the corresponding registers of IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, and IMP CNN. While image data in an area of the image memory is being processed, the IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, and IMP CNN being used are busy and all accesses from the CPU, except the access for checking the status, is prohibited. On completion of the image processing, interrupt signals are output based on the pre-defined interrupt information in each unit, and each of the interrupt signals is converted by the interrupt control unit to be sent to the CPU to notify it of the completion of image processing. Every time the processing of image data is completed, the processing result of the executed image processing is reflected in the states of the registers and memory in each unit or each area of the image memory.

### 58.4.1 Power-On Reset

The IMP-X7 enters the power shutdown mode (a state in which the power supply is stopped) after a power-on reset. For the settings to recover the IMP-X7 from the power shutdown mode, refer to section 11, System Controller (SYSC). Figure 58.9 shows the flow of supplying power to the IMP-X7 after a power-on reset.

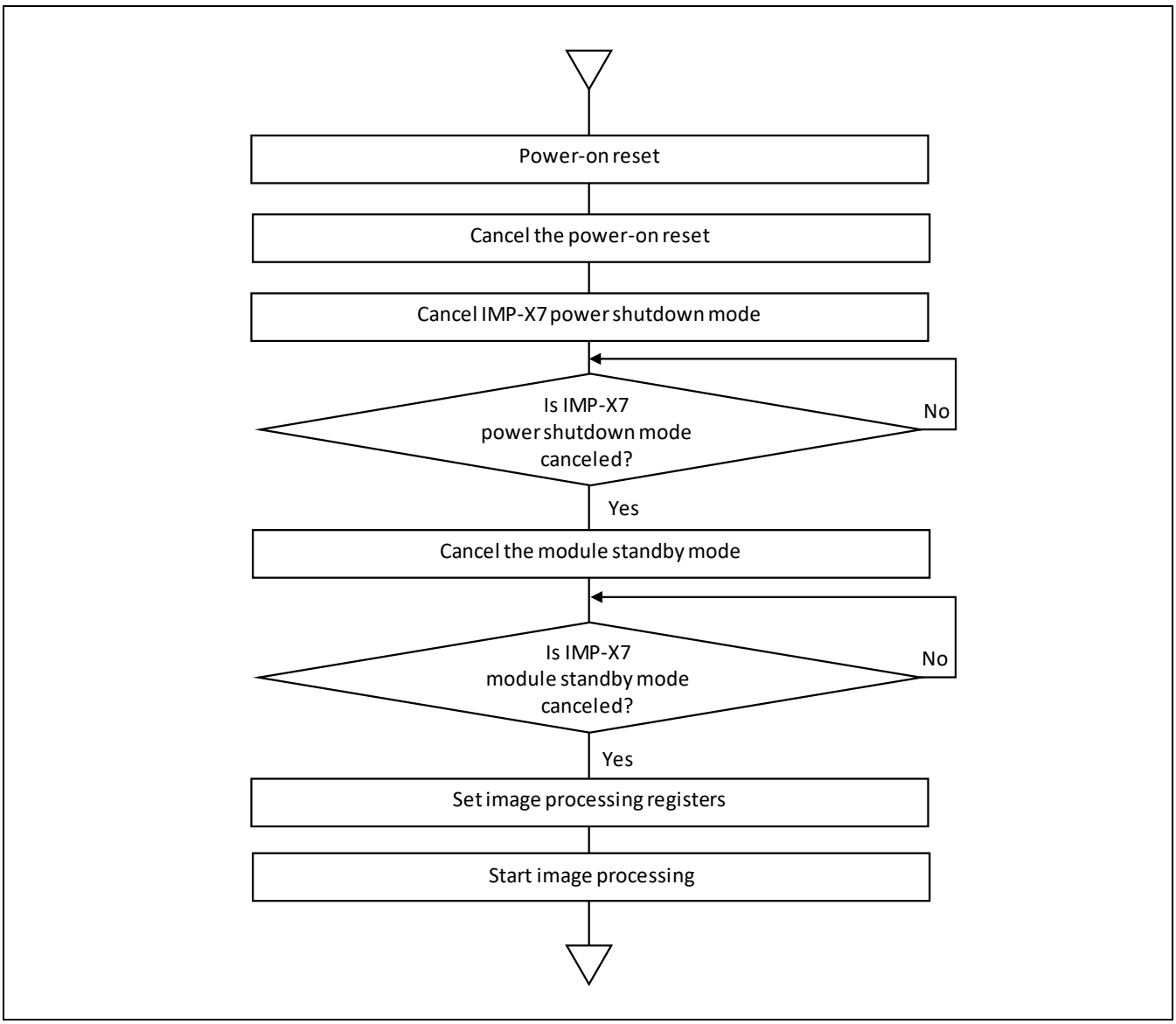



**Figure 58.9 Startup Flow of IMP-X7**

### 58.4.2 Low Power Consumption Modes

The IMP-X7 has the following three low power consumption modes.

**(1) Power shutdown mode**

Power to the IMP-X7 is shut down and clock supply to the IMP-X7 is stopped.

For details on transiting to power shutdown mode and recovering from it, refer to section 11, System Controller (SYSC).

**(2) Module Standby mode**

Clock supply to the IMP-X7 is stopped.

For details on stopping and restarting clock supply to the IMP-X7, refer to section 9, Module Standby, Software Reset.

## 58.5 Inter-Module Synchronization

Each module in the IMP-X7 has functionality for synchronizing the command list (CL) controllers between image processing modules.

The inter-module synchronization allows the IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, IMP CNN, VDSP and IMPDTA modules to synchronize their command list controllers by executing WUP and SLP instructions for the specified modules. For details refer to the U34.IMP-X7_System U-Chapter.

## 58.6 Address Map

Figure 58.10 show the address map of the IMP-X7 in the R-Car Gen4. Other modules can access 4 bytes to this area through the AXI bus. Below addresses are base address for registers of each module in IMP-X7 (IMP core, IMPDMAC, Slim-IMP DMAC, IMP PSC, IMP CNN, CVe, IMP DTA, IMP local bus registers). IMP-X7 Scratchpad memory (SPMI, SPMC0/1) address map is described in **U34.1.3 Address map of Scratchpad memory**. Refer to the U39.VDSP U-Chapter for VDSP.

| | |
|---|---|
| H'FF90 0000 | IMPSLV (24 Kbytes) |
| H'FF90 6000 | Reserved (1000 Kbytes) |
| H'FFA0 0000 | IMP core 0 (128 Kbytes) |
| H'FFA2 0000 | IMP core 1 (128 Kbytes) |
| H'FFA4 0000 | CVe cluster 0 (64 Kbytes) |
| H'FFA5 0000 | CVe cluster 1 (64 Kbytes) |
| H'FFA6 0000 | CVe cluster 2 64 Kbytes) |
| H'FFA7 0000 | CVe cluster 3 (64 Kbytes) |
| H'FFA8 0000 | IMP DMAC0 (16 Kbytes) |
| H'FFA8 4000 | IMP PSC (16 Kbytes) |
| H'FFA8 8000 | IMP DTA (16 Kbytes) |
| H'FFA8 C000 | SPMI Registers (16 Kbytes) |
| H'FFA9 0000 | Slim-IMP DMAC0 (16 Kbytes) |
| H'FFA9 4000 | Slim-IMP DMAC1 (16 Kbytes) |
| H'FFA9 8000 | Resereved (32 Kbytes) |
| H'FFAA 0000 | IMP CNN0 (64 Kbytes) |
| H'FFAB 0000 | SPMC Registers (16 Kbytes) |
| H'FFAB 4000 | Reserved (304 Kbytes) |
| H'FFAF FFFF | |

| | |
|---|---|
| H'FFB0 0000 | Reserved (128 Kbytes) |
| H'FFB4 0000 | Reserved (256 Kbytes) |
| H'FFB8 0000 | IMP DMAC1 (16 Kbytes) |
| H'FFB8 4000 | Reserved (496 Kbytes) |
| H'FFBF FFFF | |

**Figure 58.10 Address Map of IMP-X7**

## 58.7 Usage Notes

Shutting down the power supply, placing the modules in the module standby mode, and software reset by the clock pulse generator (CPG) is possible for the domains shown in Table 58.8 and Table 58.9. Table 58.8 lists the domains and the modules within each power domain and Table 58.9 lists the domain and the modules in CPG. For details on transition to and recovery from the power shutdown mode, refer to section 11, System Controller (SYSC). For details on the module standby mode and software reset from the CPG, refer to section 9, Module Standby, Software Reset.

### 58.7.1 Shutting Down the Power Supply

1. Do not attempt to access to the registers of the modules in a domain to which the supply of power has been shut down.
2. Before shutting down the power supply to a domain, ensure that no modules in the target domain are operating. Operation cannot be guaranteed if the power supply is shut down while any of the modules is operating.
3. Table 58.8 is power domain structure. When one of those power domains need to be Power-Down, which are located to the left the domain must be also Power-Down state and domains written right the domain must be in Power-ON state.
4. When one of those power domains need to be Power-ON, which are located to the right the domain must be also Power-ON state and domains which are written left the domain must be in Power-Down state.

**Table 58.8 Configuration of Power Domains**

| Power Domain (SYSC) | Module Description | Power Domain (SYSC) | Module Description | Power Domain (SYSC) | Module Description |
|---|---|---|---|---|---|
| **A1CN0** | CNN, Slim-IMP DMAC0/1 | **A2CN0** | High Speed Domain incl. CNN, CNN RAM0(SPMC), Slim-IMP DMAC0/1 and, VDSP 0/1 | **C4** | Image Recognition IP Including IMPSLV, SPMI and IMP DTA |
| **A1DSP0** | VDSP0 (DSP core0) | | | | |
| **A1DSP1** | VDSP1 (DSP core1) | | | | |
| | | **A2IMP01** | IMP core0/1 | | |
| | | **A2PSC** | IMPDMA0 | | |
| | | | IMPPSC | | |
| | | **A2DMA** | IMPDMA1 | | |
| | | **A2CV0** | CVe0 | | |
| | | **A2CV1** | CVe1 | | |
| | | **A2CV2** | CVe2 | | |
| | | **A2CV3** | CVe3 | | |

### 58.7.2 Module Standby Mode

1. Do not attempt to access to the registers of the modules in a domain that is in the module standby mode.
2. Before releasing a domain from the module standby mode, be sure to reset the target domain by a software reset from the clock pulse generator. Otherwise, reset the modules once they are released from the module standby mode.
3. Before placing a domain in the module standby mode, ensure that none of the modules in the target domain are operating. Operation cannot be guaranteed if a target domain is placed in the module standby mode while any of its modules is operating.
4. Table 58.9 shows the CPG domain structure for IMP-X7. Before placing any domain in the module standby mode, ensure that, all domains which are written left the domain has been shut down, or that they are in the module standby mode, or have been placed in the software reset state by the CPG, and all domains which are located to the right of the domain has not been shut down, or that they are not in the module standby mode, or not been placed in the software reset state by the CPG. Note that IMPSLV and SPMI need to place in module standby in one set. First place SPMI to module standby mode and last place IMPSLV to module standby mode.

   For example, if SPMC need to be placed in module standby mode then ensure that IMPCNN has been shut down, or it is in module standby mode, or it has been placed in the software reset state by the CPG and IMPSLV and SPMI must not be Power-Down state, or must not be in module standby mode, or must not be in software reset state by the CPG.
5. Before releasing any domain from the module standby mode, ensure that, all domains which are written left the domain has been shut down, or that they are in the module standby mode, or have been placed in the software reset state by the CPG, and all domains which are located to the right of the domain has not been shut down, or that they are not in the module standby mode, or not been placed in the software reset state by the CPG. Note that IMPSLV and SPMI need to release from module standby in one set. First release IMPSLV from module standby mode and last release SPMI from module standby mode.

   For example, if SPMC need to be released module standby mode then ensure that IMPCNN has been shut down, or it is in module standby mode, or it has been placed in the software reset state by the CPG and IMPSLV and SPMI must not be Power-Down state, or must not be in module standby mode, or must not be in software reset state by the CPG.

**Table 58.9 Configuration of CPG Domains**

| Domain (CPG) | Module Description | Domain (CPG) | Module Description | Domain (CPG) | Module Description |
|---|---|---|---|---|---|
| **IMPCNN** | IMP CNN<br>Slim-IMP DMAC0/1 | **SPMC** | High Speed Domain (SPMC) | **IMPSLV**<br>**SPMI** | **Image Recognition IP** Including (**IMPSLV and SPMI)** |
| | | **IMP0** | IMP core0 | | |
| | | **IMP1** | IMP core1 | | |
| | | **IMPDMA0** | IMPDMAC0 | | |
| | | **IMPPSC0** | IMPPSC0 | | |
| | | **OCV0** | CVe0 | | |
| | | **OCV1** | CVe1 | | |
| | | **OCV2** | CVe2 | | |
| | | **OCV3** | CVe3 | | |
| | | **IMPDMA1** | IMPDMAC1 | | |
| | | **IMP DTA** | IMP DTA | | |

### 58.7.3 Software Reset by the Clock Pulse Generator (CPG)

1. Do not attempt to access to the registers of the modules in a domain to which a software reset is being applied by the clock pulse generator.
2. Before applying a software reset to a target domain by using the CPG, ensure that none of the modules in the domain are operating. A software reset applied to a target domain by the CPG is equivalent to a power-on reset. Accordingly, operation cannot be guaranteed if a domain is reset while any of its modules is operating.
3. Ensure that software reset is released by reading the CPG register after reset is released before accessing the IMP modules.
4. Table 58.9 shows the CPG domain structure for IMP-X7. Before applying a software reset to the any domain by using the CPG, ensure that, all domains which are written left the domain has been shut down, or that they are in the module standby mode, or have been placed in the software reset state by the CPG, and all domains which are located to the right of the domain has not been shut down, or that they are not in the module standby mode, or not been placed in the software reset state by the CPG. Note that IMPSLV and SPMI need to reset in one set. First place SPMI to software reset mode and last place IMPSLV to software reset.

   For example, if SPMC need to be software reset by CPG then ensure that IMPCNN has been shut down, or it is in module standby mode, or it has been placed in the software reset state by the CPG and IMPSLV and SPMI must not be Power-Down state, or must not be in module standby mode, or must not be in software reset state by the CPG.
5. Before releasing a software reset to the any domain by using the CPG, ensure that, all domains which are written left the domain has been shut down, or that they are in the module standby mode, or have been placed in the software reset state by the CPG, and all domains which are located to the right of the domain has not been shut down, or that they are not in the module standby mode, or not been placed in the software reset state by the CPG. Note that IMPSLV and SPMI need to be released from software reset in one set. First release IMPSLV from software reset mode and last release SPMI from software reset mode.

   For example, if SPMC need to be released module standby mode then ensure that IMPCNN has been shut down, or it is in module standby mode, or it has been placed in the software reset state by the CPG and IMPSLV and SPMI must not be Power-Down state, or must not be in module standby mode, or must not be in software reset state by the CPG.

# 60. Video Signal Processor (VSP2)

## 60.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The VSP2 is the successor IP of Renesas' VSP-IP series which supports image processing such as image blending, interleaver, interface to Display Unit (DU), interface of ISP streaming IP and Display Output Compare (DISCOM).

Description of Display Output Compare (DISCOM) is shown in the section 60.5.2.

### 60.1.1 Features

This LSI incorporates different types of VSP2 modules (VSPX VSPD) as image processing systems. Table 60.1 shows the number of VSP for each product.

The VSPX read data from external memory, and send the data to ISP via Streaming IP.

The VSPD supports image processing such as image blending, interface to Display Output Compare (DISCOM), and output image data to Display Unit (DU) with or without writing back the image data to memory.

**Table 60.1 VSP2 configuration**

| Module Name | Number of channels |
|---|---|
| VSPX | 1 |
| VSPD | 1 |

Each VSPs features are shown in from section 60.1.1.1 to 60.1.1.2

#### 60.1.1.1 Features of VSPX

- Supports reading data from external memory to ISP via Streaming IP
  - Supports RAW8, RAW10, RAW12, RAW14, RAW16, RAW20 and RAW24 for RPF0
  - Supports RAW8, RAW10, RAW12 for RPF1
  - Supports up to 2 exposures (2 raw data)
  - Supports reading ISP config data from memory to ISP.
  - Supports up to 5120 pixels x 4096 lines
- Supports wake up assertion from VIN
- Up to 16 VINs
- Performance of VSPX
  - 720 Mpix/s process rate per 1 set of VSPX.
  - Refer to 60.4.7.2 for more detail

#### 60.1.1.2 Features of VSPD

- Supports various data formats and conversion
  - Supports YCbCr444/422/420, RGB, α RGB, α plane
  - Supports reading high pixel bitdepth (10 bits or 12 bits per each component Y, Cb, Cr, R, G, B) (*1)
  - Color space conversion and changes to the number of colors by dithering
  - Color keying
  - Supports combination between pixel alpha and global alpha
  - Supports generating pre multiplied alpha
- Video processing
  - Blending of five picture layers and raster operations (ROPs)
- Direct connection to display module
  - Supporting 6144 pixels in horizontal direction
  - Writing back image data which is transferred to Display Unit (DU) to memory
- Supports DISCOM function
  - Support 2 set of DISCOM per 1 VSPD
  - Support same size with VSPD
  - CRC calculation area is configlable
  - Support detection of unmatching the expectation CRC value
  - Support detection of matching image data over multiple frames for detecting image frozen
  - Support both RGB and YCbCr color space

(*1) For supporting 10 bits / 12 bits in each module, refer to section 60.1.2.12

### 60.1.2 Block Diagram

#### 60.1.2.1 Block Diagram of Each VSP

Figure 60.1 to Figure 60.2 shows the configuration of VSPX and VSPD sub modules.

All the registers and functions which doesn't appears in the each VSP2 unit's block diagram should be treated as un-implemented functions in each VSP2 unit. Do not set any registers related to the un-implemented function without any statement.

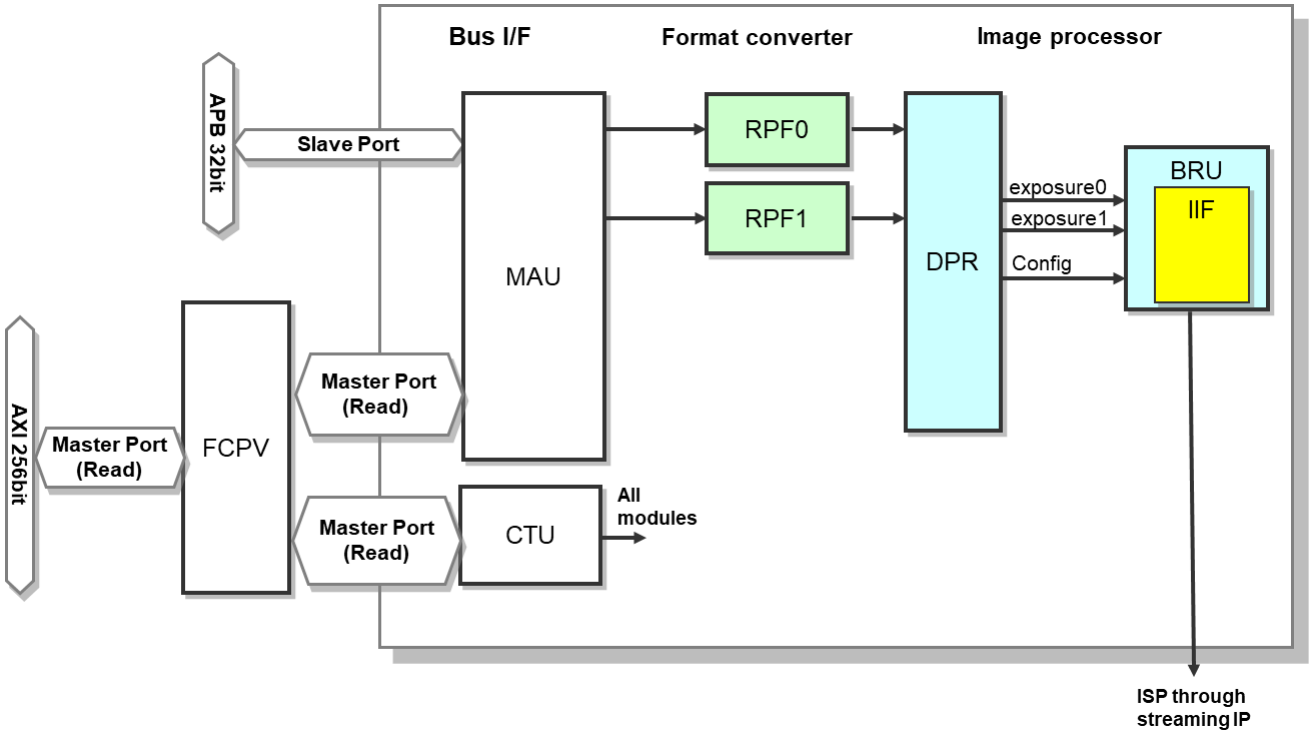



**Figure 60.1 VSPX Module Configuration**

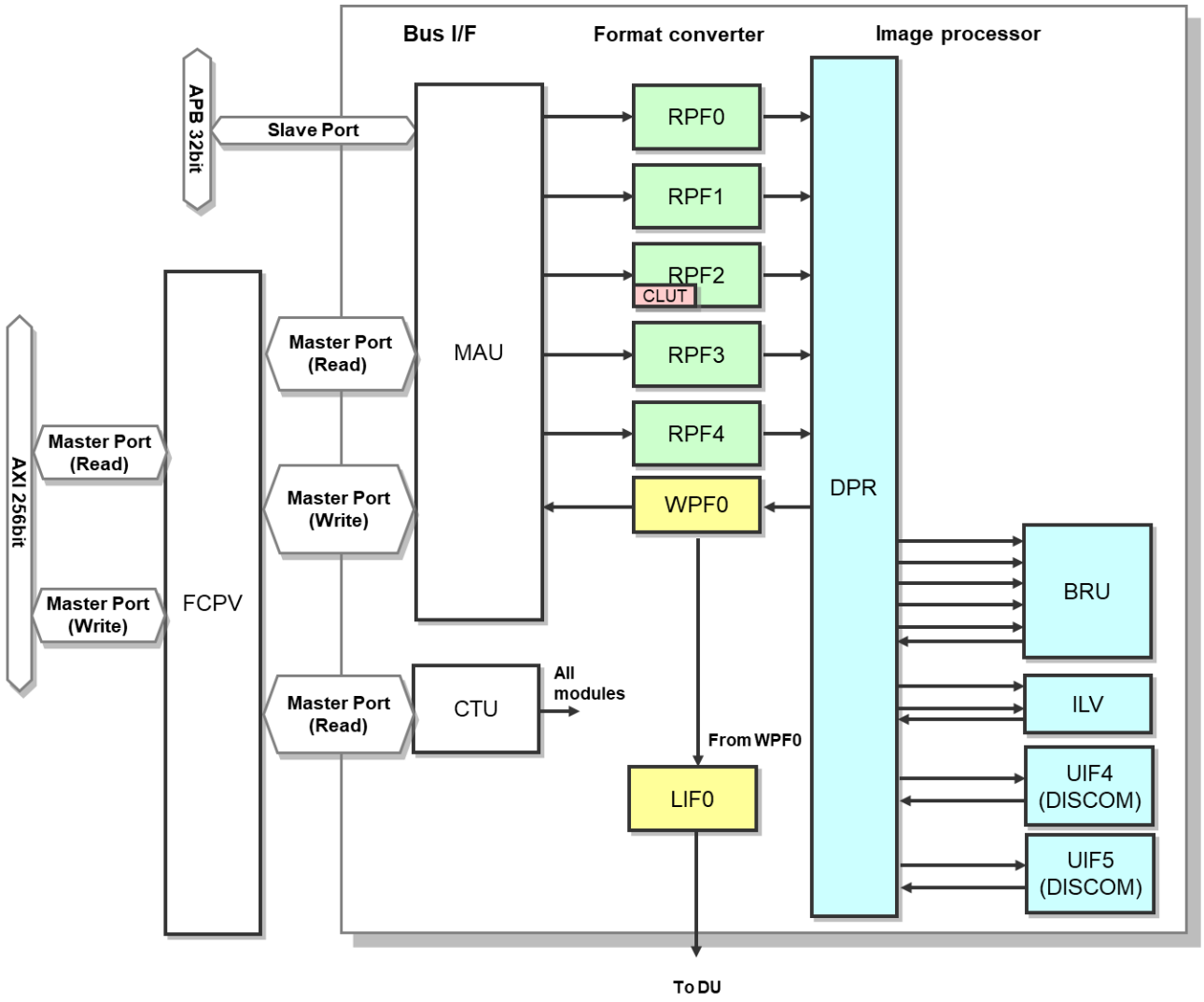



**Figure 60.2 VSPD Module Configuration**

The following gives an overview of each block function shown in Figure 60.1 to Figure 60.2.

The operation of these functions is determined by the register setting explained in section 60.2. Refer to section 60.2 for the relationship between the register setting and the functional behavior.

#### 60.1.2.2 Memory Access Unit (MAU)

The VSP2 applies processing to the image data stored in the external memory and writes the resultant data back to the external memory. The data transfer between the external memory and VSP2 necessary for this operation is done by the MAU, which works as the bus master, according to the register settings. The MAU executes this data transfer between the external memory and VSP2.

#### 60.1.2.3 Command Transfer Unit (CTU)

The VSP2 can directly read register parameters for image processing by display lists stored in external memory. The CTU module is a bus interface and controls display lists when the CTU reads display lists as a bus master.

#### 60.1.2.4 Read Pixel Formatter (RPF)

The RPF reads image data from the external memory through the MAU, unpacks data according to the specified format, converts the color space, converts the number of colors, executes color keying, ROP operation, Multiply-alpha and OSD processing, and outputs the resultant data to the DPR. The RPF has an input format unpacking unit, an index color expanding unit (OSD-CLUT), a 1-bit mask generator, a raster operation unit (ROP unit), a color keying unit, a color space converter and multiply-alpha. Figure 60.3 show the processing flow in the RPF.

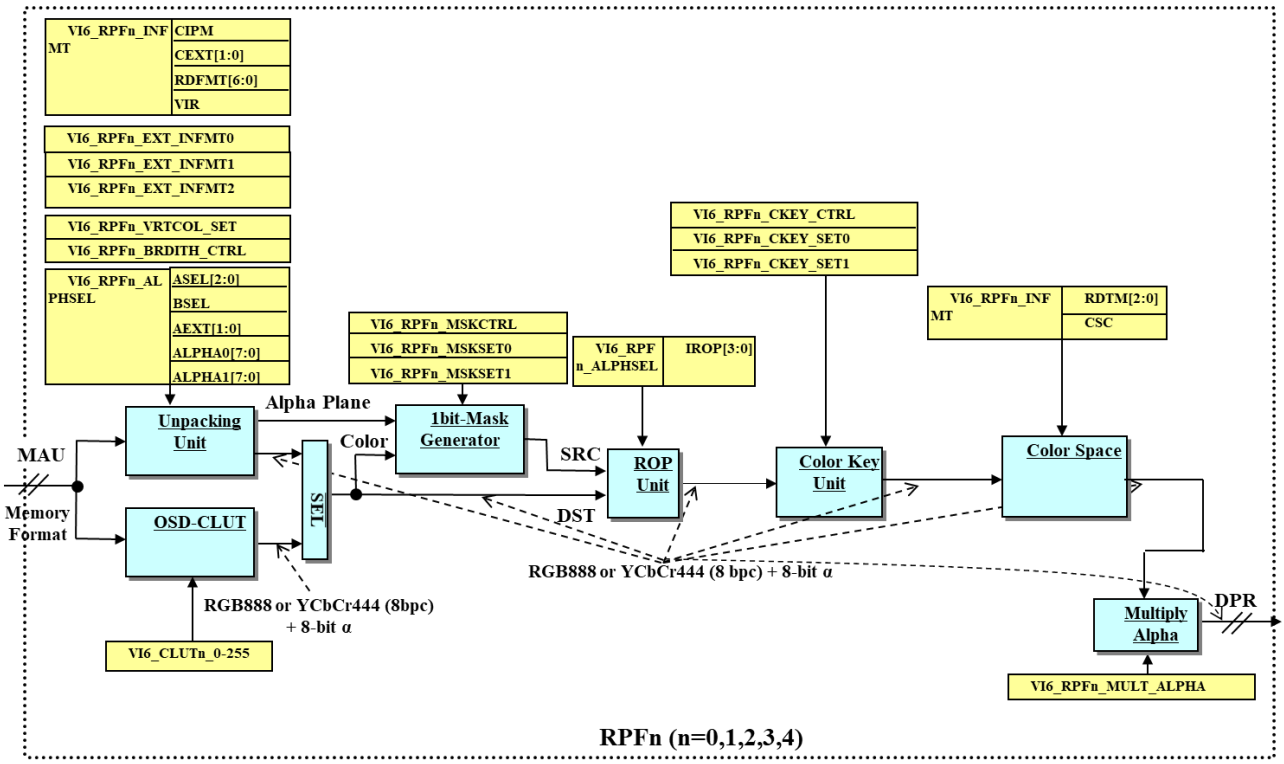



**Figure 60.3 RPF Processing Flow**

The input format unpacking unit and index color expanding unit (OSD-CLUT) expand the image data input from the MAU into the image format for internal processing (YCbCr444 8bpc or RGB888 8bpc), and the 1-bit mask generator generates a 1-bit image mask from the image data expanded through the unpacking unit and OSD-CLUT. Alternately, a 1-bit image mask can be generated from the alpha plane that is different from the picture plane read through the MAU.

The raster operation unit (ROP unit) executes raster operation between the data from the 1-bit mask generator and the image data expanded from the input format, and the color keying unit applies color replacement and specifies the transparent color for the image data input from the ROP unit. The color space converter converts the color space (RGB-YCbCr) of the image data input from the color keying unit as necessary. The multiply-alpha unit multiplies pixel-alpha by specified alpha or/and multiplies image data by pixel-alpha or/and specified alpha.

The VSP2 provides maximum two RPF modules (RPF0 to RPF4). The implementation of OSD-CLUT depends on the configuration of each VSP.

### 60.1.2.5 Data Path Router (DPR)

The DPR controls the data paths among RPFs, function modules, and WPFs. The DPR selects one of the images input from RPFs, outputs it to a function module (BRU, IIF, UIF or ILV), and selects one of WPFs as the destination where the image data processed in the function module will be output. Before output to the WPF, the output from each function module can be input to another function module, which enables multiple image processing functions to be executed continuously without involving the external memory.

### 60.1.2.6 Blend ROP Unit (BRU)

The BRU is a module connected to the DPR, which executes the image blending processing and ROP operation. The BRU has five blend/ROP operation units (blend/ROP unit m, m = A to E), an ROP operation unit, a blend/ROP input switch (SEL) for selecting the input to these operation units, and a divider for normalization (div unit).

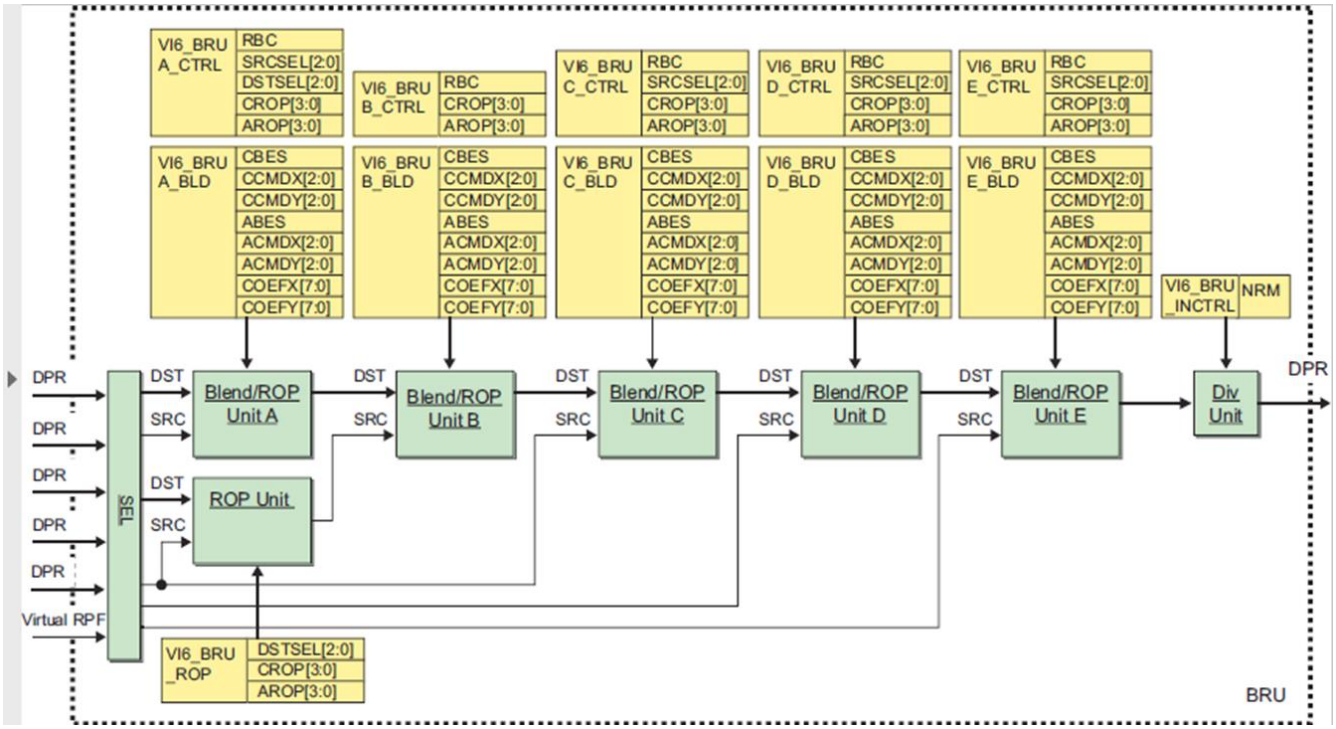



**Figure 60.4 BRU Processing Flow**

Each of the five blend-ROP operation units (blend/ROP unit m) receives the output from the SEL, blend/ROP Unit m, or ROP unit, and executes blending or raster operation (ROP) of images. The ROP unit receives the output from the SEL

and executes 2-input raster operation (ROP2). By combining the ROP operation in the blend/ROP unit m and ROP2 operation in the ROP unit, 256-type 3-input raster operation (ROP3) can be implemented.

The divider for normalization (div unit) divides the pixel value by the α value.

BRU has an IIF function. IIF is only implemented for VSPX. Refer to 60.1.2.9 for IIF function.

### 60.1.2.7 Write Pixel Formatter (WPF)

The WPF is an output module that receives 32bits image data (YCbCr444 or RGB888 + 8-bitα) from the DPR, converts the color space, number of colors, and format of the data, and outputs the results of VSP2 image processing to external memory through the MAU. The WPF is mainly configured from a color space converter and an output format converter (the packing unit).

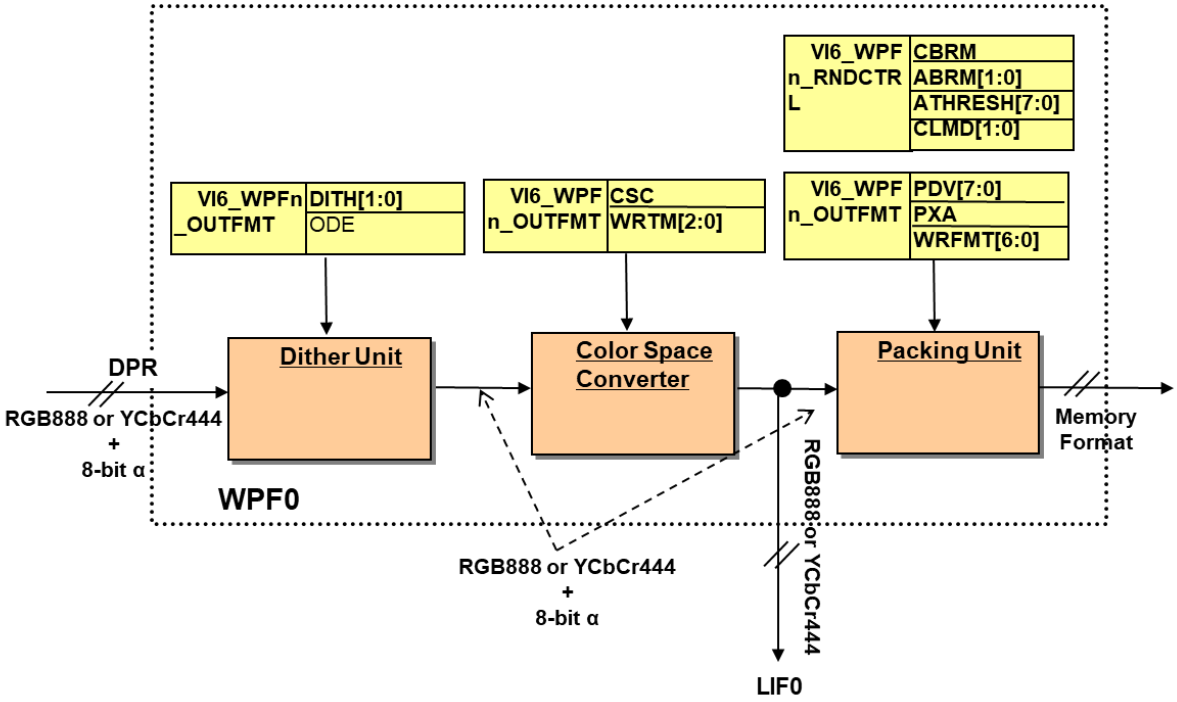



**Figure 60.5 WPF Processing Flow**

The color space converter converts the color space between RGB and YCbCr, and the packing unit converts the format into the picture plane storing format. The VSP2 provides two WPF0. The WPF has the output to LIF module after the color space conversion for transferring the video data to the display module. Image data to LIF is 24bits (YCbCr444 or RGB888) format, and its' color space is after color space conversion.

### 60.1.2.8 Display Unit InterFace (LIF)

The LIF module is used for transferring image data to the display module. The input port of the LIF module is connected to WPF, and the output port of the LIF module is connected to DU (Display Unit). Data flow in LIF is shown in Figure 60.28.

### 60.1.2.9 ISP InterFace (IIF)

The IIF concatenates raw data of up to 2 exposures synchrornously in pixel unit then send this data to streaming IP. The active exposure is read from external memory through RPF0 to RPF1 with format RGB332 (for RAW8 of 1-byte), RGB565 (for RAW8 of 2-byte container, RAW10, RAW12, RAW14, RAW16) or ARGB8888 (for RAW20, RAW24).The IIF also send ISP config data frome external memory through RPF0. IIF is a function of BRU, as shown in Figure 60.1, and works by setting 1 to VI6_DPR_BRU_ROUTE.IIFSEL. Refer to section 60.3.8 for detailes.

### 60.1.2.10 User Logic InterFace (UIF)

The UIF module is used for transferring image data to the User Logic module (DISCOM).

The UIF is a module connected to the DPR and can transfer image data to the User Logic module and can receive image data from the User Logic module after the User Logic module execute its image processing.

### 60.1.2.11 Detail Function

**Table 60.2 Detail Function of VSP2**

**Image Data Transfer Function**

| | | | |
|---|---|---|---|
| Bus interface | Protocol | | AXI 256 bits through FCPV |
| | Data alignment | Conversion method | Byte, Word, LW, or LLW data swapping |
| | | Channel | Data alignment can be specified separately for each input/output channel |
| Image memory | Input | Address setting | 1-byte units |
| | Output | Address setting | 1-byte units |
| Tile transfer mode | | | Supported by RPF0 to RPF4 |
| YCbCr memory storage format | | | Interleaved, planar, or semi-planar |

**Display list/Extended Display List Transfer Function**

| | | | |
|---|---|---|---|
| Bus interface | Protocol | | AXI 256 bits through FCPV |
| | Data alignment | Conversion method | Byte, Word, LW data swapping |
| | | | Data alignment can be specified separately for display list and each entry of extended display list |
| Data memory | Load | Address setting | 8-byte units |
| | Store | - | Not available |

**Read Pixel Formatter (RPF)**

| | | | |
|---|---|---|---|
| Number of channels | | | Five channels (RPF0 to RPF4 for VSPD)<br>Two channels (RPF0 to RPF1 for VSPX) |
| Input color bit-depth | | Color | 2 ~ 8bit / 10bit / 12bit |
| | | Alpha | 1/2/4/8 bit |
| Operation bit-depth | | Color | 8bit |
| | | Alpha | 8bit |
| Image format | Input | RGB | RGB888, RGB565, RGB666,<br>αRGB8888, αRGB4444, αRGB1555,<br>RGB10, RGB10A2, A2RGB10<br>α plane (8 bpp, 1 bpp) |

**Read Pixel Formatter (RPF)**

| | | | |
|---|---|---|---|
| | | YCbCr | YCbCr4:4:4 (8bpc) Planar/Semi planar/Interleaved<br>YCbCr4:2:2 (8/10/12 bpc) Planar/Semi planar/Interleaved (*)<br>YCbCr4:2:0 (8 bpc) Planar/Semi planar/Interleaved<br>YCbCr4:2:0 (10/12 bpc) Planar/Semi planar (*)<br>α plane (8 bpp, 1 bpp)<br>(*): bpc of Y and Cb/Cr can be equal or different for Planar/Semi planar and shall be equal for Interleaved. |
| | | RAW | RAW8 1byte, RAW8 2byte, RAW10, RAW12, RAW14, RAW16, RAW20, RAW24 |
| | | Maximum size | 8190 × 8190 pixels [VSPD]<br>5120 × 4096 pixels [VSPX]<br>The internal data path modules have separate restrictions on the maximum image size. For details, refer to section 60.4.3 |
| | | Minimum size | 128 x 96 pixel [VSPD]<br>128 × 32 pixel [VSPX]<br>The internal data path modules have separate restrictions on the minimum image size. For details, refer to section 60.4.3. |
| | | Size setting unit | YCbCr420: 2-pixel units both horizontally and vertically.<br>YCbCr422: 2-pixel units horizontally and 1-pixel units vertically.<br>In other formats, the size can be set in 1-pixel units. |
| Color keying | Color replacement | Compared data | RGB or Y (8 bpc) |
| | | Replaced data | αRGB or αYCbCr (8 bpc for R, G, B, Y, Cb, Cr, α) |
| | | Comparison Mode | Matched color mode |
| | | Input source | RPF0 to RPF4 |
| | Transparent color | Compared data | RGB or Y (8 bpc) |
| | | Replaced data | α 8bpp |
| | | Comparison Mode | Matched color mode, Luma threshold mode |
| | | Input source | RPF0 to RPF4 |
| OSD/CLUT | Format | Memory storage format | 8 bpp |
| | | Image format | α RGB32 or α YCbCr32 |
| | Input channel | | RPF2 |

**Read Pixel Formatter (RPF)**

| | | | |
|---|---|---|---|
| Raster operation | ROP2 (within input channels) | Operator | 16 types (OpenGL2.0 is supported) |
| | | Sources of operation | ROP2 operation between the 1-bpp α plane and RGB/YCbCr data in RPF0 to RPF4.<br>Note that 1-bpp α is converted to αRGB or αYCbCr4:4:4. (8 bpc) |
| Color space conversion | RGB to YCbCr | Conversion expression | RGB (0, 255) to BT601 (16, 235/240)<br>RGB (0, 255) to BT709 (16, 235/240)<br>RGB (0, 255) to BT601 (0, 255)<br>RGB (16, 235) to BT709 (16, 235/240) |
| | | Target of conversion | RPF0 to RPF4 |
| | YCbCr to RGB | Conversion expression | BT.601 (16, 235/240) to RGB (0, 255)<br>BT.709 (16, 235/240) to RGB (0, 255)<br>BT.601 (0, 255) to RGB (0, 255)<br>BT.709 (16, 235/240) to RGB (16, 235) |
| | | Target of conversion | RPF0 to RPF4 |
| Changing number of colors | Bit extension | Extended bits | Input bitdepth (2bpc ~ 6bpc) to operation bitdepth 8bpc |
| | | Target format | RGB666, RGB565, RGB555, RGB444, or RGB332 |
| | | Bit reduction method | Padded with 0.<br>Copied from the most significant bits. |
| | | Target of conversion | RPF0 to RPF4 |
| | Bit reduction | Reduced bits | Input bitdepth 10/12 bpc to operation bitdepth 8bpc |
| | | Target format | RGB10bpc,<br>YCbCr 4:2:2 10/12 bpc Semiplanar/Planar/Interleaved,<br>YCbCr 4:2:0 10/12 bpc Semiplanar/Planar |
| | | Rounding method | Truncation or Rounding-off or dithering |
| | | Dithering method | Ordered dither |
| | | Dithering target component | R/G/B/Y |
| | YCbCr444 generation | Vertical | CbCr copying |
| | | Horizontal | copying or interpolation. |
| α bit count conversion | Bit extension | Extended bits | Input bitdepth 1bpc to operation bitdepth 8bpc |
| | | Method | Padded with 0.<br>Copied from the most significant bits. |
| Multiply-alpha function | Fade-alpha | | RPF0 to RPF4 |
| | | | Available for straight pixel and pre-multiplied pixel. |
| | Generate pre-multiplied alpha | | RPF0 to RPF4 |
| Virtual display | RPF0 to RPF4 | Color format | αRGB8888 or αYCbCr4:4:4 single-color |
| | | Display size | Same as the size of the input channel |
| | Virtual RPF | Color format | αRGB8888 or αYCbCr4:4:4 single-color |
| | | Display size | Maximum: 8190 × 8190 pixels<br>Minimum: 4 × 4 pixels |

**Write Pixel Formatter (WPF)**

| | | | |
|---|---|---|---|
| Number of channels | | | One channel (WPF0) |
| Image format | Output | RGB | RGB332, RGB444, RGB565, RGB666, RGB888, αRGB8666, αRGB8888, αRGB4444, αRGB1555 |
| | | YCbCr | YCbCr4:4:4, YCbCr4:2:2, YCbCr4:2:0 |
| | | Maximum size | 8190 × 8190 pixels<br>The internal data path modules have separate restrictions on the maximum image size. For details, refer to section 60.4.4 |
| | | Minimum size | 1 × 1 pixel<br>The internal data path modules have separate restrictions on the minimum image size. For details, refer to section 60.4.4 |
| | | Size setting unit | YCbCr420: 2-pixel units both horizontally and vertically.<br>YCbCr422: 2-pixel units horizontally and 1-pixel units vertically.<br>In other formats, the size can be set in 1-pixel units. |
| Color space conversion | RGB to YCbCr | Conversion expression | RGB (0, 255) to BT.601 (16, 235/240)<br>RGB (0, 255) to BT.709 (16, 235/240)<br>RGB (0, 255) to BT.601 (0, 255)<br>RGB (16, 235) to BT.709 (16, 235/240) |
| | | Target of conversion | WPF0 |
| | YCbCr to RGB | Conversion expression | BT.601 (16, 235/240) to RGB (0, 255)<br>BT.709 (16, 235/240) to RGB (0, 255)<br>BT.601 (0, 255) to RGB (0, 255)<br>BT.709 (16, 235/240) to RGB (16, 235) |
| | | Target of conversion | WPF0 |
| Changing number of colors | Output | Reducing RGB color depth | Dithering, lower-order bit truncation, or rounding |
| | | YCbCr422/420 | CbCr skipping or CbCr vertical skipping and horizontal skipping |
| α bit count conversion | Output | Bit reduction | Truncation, rounding, or comparison with threshold (for 1 bpp) |
| **Image Compositing** | | | |
| α blending | Input α value selection | RGB | Pixel α, fixed α value, α plane, or<br>1-bit α converted from the color specified for pixels |

**Image Compositing**

| | | | |
|---|---|---|---|
| | | YCbCr | α plane, fixed α value, or<br>1-bit α converted from the color specified for pixels |
| | α blending expression | Layer A: Upper layer<br>Layer B: Lower layer | xA + yB, xA - yB<br>Coefficients x and y should be selected from the following.<br>Fixed α value,<br>(α for layer A), (1 - α for layer A),<br>(α for layer B), (1 - α for layer B) |
| | Output α value selection | RGB | Fixed α value,<br>x (α for layer A) + y (α for layer B),<br>x (α for layer A) – y (α for layer B)<br>Coefficients x and y should be selected from the following.<br>Fixed α value,<br>(α for layer A), (1 - α for layer A),<br>(α for layer B), (1 - α for layer B) |
| | Blending layers | Number of layers | Five layers selected from RPF0 to RPF4 and video processing function output, and virtual RPF; six layers in total |
| | | Order of layers | The order of six layers selected from RPF0 to RPF4, virtual RPF, and video processing function output can be changed as desired. |
| | α plane | Format | 8 bpp or 1 bpp (α value can be specified through register) |
| | | Input source | RPF0 to RPF4 |
| | Fixed α value | Format | 8 bpp |
| | | Input source | RPF0 to RPF4, virtual RPF, or video processing function output |
| Raster operation | ROP2 (between input channels) | Operator | 16 types (OpenGL2.0 is supported) |
| | | Sources of operation | RPF0 to RPF4, virtual RPF, and video processing function output |
| | | Operation control | RGB/YCbCr and α are operated separately. |
| | ROP3 (between input channels) | Operator | 256-type ROP3 is available by combining ROP2 operations |
| | | Sources of operation | RPF0 to RPF4, virtual RPF, and video processing function output |
| | | Operation control | RGB/YCbCr and α are operated separately. |

**Direct connection to Display module**

| | | |
|---|---|---|
| Display I/F | Destination | DU |
| | Direct connection | Can transfer data without going through external memory to DU |
| Line padding | Padding cycles | 1 to 32 cycles |
| | Padding pattern | Arbitrarily setting up to 32 cycles |

| **Interface with ISP** | | |
|---|---|---|
| Exposures | Number of exposure | 1 to 2 exposures reading through RPF0 to RPF1<br>(active exposure shall be set in order RPF0 to RPF1) |
| | Format | RAW8 1byte, RAW8 2bytes, RAW10, RAW12, RAW14, RAW16, RAW20, RAW24 |
| Config data for ISP | | ISP config data can be read through RPF0. |
| Wake up | Features | VSPX can wait for reading image data of specified lines output from VIN without S/W intervention. |
| | Supported VIN channels | 16 channels (VIN0 to VIN15) |
| | | Independent wake-up counter per channel. |
| | | Can elect which channel wake-up is recognized. |
| | Outstanding of wake-up | Up to 15 outstanding of wake-up. |
| Size | Maximum size | 5120 pixels x 4096 lines |
| | Minimum size | 128 pixels x 32 lines |
| | Other | All exposures shall have same size |

#### 60.1.2.12 Supporting 12 Bits Processing

Table 60.3 shows which module supports 12 bits processing for VSPD.

**Table 60.3 VSPD sub modules which supports 12 bits processing**

| Module | 12 bits processing |
|---|---|
| RPF | √(*1) |
| WPF | |
| BRU | |
| DISCOM | |
| ILV | |
| LIF | |

(*1) VSPD doesn't support RPF 12 bits processing. But RPF supports 10 bits / 12 bits input pixel format, and data is rounded from 12 bits / 10 bits to 8 bits at the RPF input point.

### 60.1.3 External Pins

VSP2 doesn't have external pins.

### 60.1.4 Register Configuration

#### (1) Base Address

Below are the base addresses of each VSP unit. VSPX has 32Kbyte address space. VSPD has 28Kbyte address space.<

VSPD0 H'FEA2 0000
VSPX0 H'FEDD 0000

### (2) Register/Table Address

Table 60.4 shows the VSP2 register configuration (represents in relative address).

**Table 60.4 VSP2 Register Configuration**

| Register Name | Abbr. | R/W | Offset Address | Initial value | Access Size |
|---|---|---|---|---|---|
| VSP2 Start Registers 0 | VI6_CMD0 | R/W*1 | H'0000 | H'0000 0000 | 32 |
| Clock Control Register0 | VI6_CLK_CTRL0 | R/W | H'0010 | H'0000 0000 | 32 |
| Clock Control Register1 | VI6_CLK_CTRL1 | R/W | H'0014 | H'0000 0000 | 32 |
| Dynamic Clock Stop Control Register | VI6_CLK_DCSWT | R/W | H'0018 | H'0000 0000 | 32 |
| Dynamic Clock Stop Disable Register0 | VI6_CLK_DCSM0 | R/W | H'001c | H'0000 0000 | 32 |
| Dynamic Clock Stop Disable Register1 | VI6_CLK_DCSM1 | R/W | H'0020 | H'0000 0000 | 32 |
| Software Reset Register | VI6_SRESET | R/W | H'0028 | H'0000 0000 | 32 |
| Module Reset Enable Register0 | VI6_MRESET_ENB0 | R/W | H'002c | H'0000 0000 | 32 |
| Module Reset Enable Register1 | VI6_MRESET_ENB1 | R/W | H'0030 | H'0000 0000 | 32 |
| Module Reset Issuing Register | VI6_MRESET | R/W | H'0034 | H'0000 0000 | 32 |
| Operating Status Register | VI6_STATUS | R | H'0038 | H'0000 0000 | 32 |
| WPF0 Interrupt Enable Registers | VI6_WPF0_IRQ_ENB | R/W | H'0048 | H'0000 0000 | 32 |
| WPF0 Interrupt Status Registers | VI6_WPF0_IRQ_STA | R/W | H'004c | H'0000 0000 | 32 |
| Display0 Interrupt Enable Register | VI6_DISP0_IRQ_ENB | R/W | H'0078 | H'0000 0000 | 32 |
| Display0 Interrupt Status Register | VI6_DISP0_IRQ_STA | R/W | H'007c | H'0000 0000 | 32 |
| Wake-up0 Interrupt Enable Registers | VI6_WUP0_IRQ_ENB | R/W | H'00EC | H'0000 0000 | 32 |
| Wake-up0 Interrupt Status Registers | VI6_WUP0_IRQ_STA | R/W | H'00F0 | H'0000 0000 | 32 |
| Display List Control Register | VI6_DL_CTRL | R/W | H'0100 | H'0000 0000 | 32 |
| Display List-0 Header Address Register | VI6_DL_HDR_ADDR0 | R/W | H'0104 | H'0000 0000 | 32 |
| Display List0 Data Swapping Register | VI6_DL_SWAP0 | R/W | H'0114 | H'0000 0000 | 32 |
| Extended Display List Control Register | VI6_DL_EXT_CTRL | R/W | H'011c | H'0000 0000 | 32 |
| Display List Body Size Register-0 | VI6_DL_BODY_SIZE0 | R/W | H'0120 | H'0000 0000 | 32 |
| Display List-0 Header Reference Address Register | VI6_DL_HDR_REF_ADDR0 | R | H'0130 | H'0000 0000 | 32 |
| Display List VIN Wake Up Counter0 Register | VI6_DL_VIN_WUPCNT0 | R/W | H'015C | H'0000 0000 | 32 |
| Display List VIN Wake Up Counter1 Register | VI6_DL_VIN_WUPCNT1 | R/W | H'0160 | H'0000 0000 | 32 |
| Display List VIN Wake Up Enable Register | VI6_DL_VIN_WUPEN0 | R/W | H'0164 | H'0000 0000 | 32 |
| RPFn Basic Read Size Registers | VI6_RPFn_SRC_BSIZE | R/W | H'0300 + H'0100*n | H'0000 0000 | 32 |
| RPFn Extended Read Size Registers | VI6_RPFn_SRC_ESIZE | R/W | H'0304 + H'0100*n | H'0000 0000 | 32 |
| RPFn Input Format Registers | VI6_RPFn_INFMT | R/W | H'0308 + H'0100*n | H'0000 0000 | 32 |
| RPFn Data Swapping Registers | VI6_RPFn_DSWAP | R/W | H'030c + H'0100*n | H'0000 0000 | 32 |

| | | | | | |
|---|---|---|---|---|---|
| RPFn Display Location Registers | VI6_RPFn_LOC | R/W | H'0310 + H'0100*n | H'0000 0000 | 32 |
| RPFn a Plane Selection Control Registers | VI6_RPFn_ALPH_SEL | R/W | H'0314 + H'0100*n | H'0000 0000 | 32 |
| RPFn Virtual Plane Color Information Registers | VI6_RPFn_VRTCOL_SET | R/W | H'0318 + H'0100*n | H'0000 0000 | 32 |
| RPFn Mask Control Registers | VI6_RPFn_MSKCTRL | R/W | H'031c + H'0100*n | H'0000 0000 | 32 |
| RPFn IROP-SRC Input Value Registers 0 | VI6_RPFn_MSKSET0 | R/W | H'0320 + H'0100*n | H'0000 0000 | 32 |
| RPFn IROP-SRC Input Value Registers 1 | VI6_RPFn_MSKSET1 | R/W | H'0324 + H'0100*n | H'0000 0000 | 32 |
| RPFn Color Keying Control Registers | VI6_RPFn_CKEY_CTRL | R/W | H'0328 + H'0100*n | H'0000 0000 | 32 |
| RPFn Color Keying Color Setting Registers-0 | VI6_RPFn_CKEY_SET0 | R/W | H'032c + H'0100*n | H'0000 0000 | 32 |
| RPFn Color Keying Color Setting Registers-1 | VI6_RPFn_CKEY_SET1 | R/W | H'0330 + H'0100*n | H'0000 0000 | 32 |
| RPFn Source Picture Memory Stride Setting Registers | VI6_RPFn_SRCM_PSTRIDE | R/W | H'0334 + H'0100*n | H'0000 0000 | 32 |
| RPFn Source α Memory Stride Setting Registers | VI6_RPFn_SRCM_ASTRIDE | R/W | H'0338 + H'0100*n | H'0000 0000 | 32 |
| RPFn Source Y/RGB Address Registers | VI6_RPFn_SRCM_ADDR_Y | R/W | H'033c + H'0100*n | H'0000 0000 | 32 |
| RPFn Source Chroma Address Registers 0 | VI6_RPFn_SRCM_ADDR_C0 | R/W | H'0340 + H'0100*n | H'0000 0000 | 32 |
| RPFn Source Chroma Address Registers 1 | VI6_RPFn_SRCM_ADDR_C1 | R/W | H'0344 + H'0100*n | H'0000 0000 | 32 |
| RPFn Source α Address Registers | VI6_RPFn_SRCM_ADDR_AI | R/W | H'0348 + H'0100*n | H'0000 0000 | 32 |
| RPFn Bus Access Control Registers | VI6_RPFn_BAC | R/W | H'0350 + H'0100*n | H'0000 0000 | 32 |
| RPFn Multiple Alpha Control | VI6_RPFn_MULT_ALPH | R/W | H'036C+ H'0100*n | H'0000 0000 | 32 |
| RPFn Extended Input Format Registers 0 | VI6_RPFn_EXT_INFMT0 | R/W | H'0370 + H'0100*n | H'0000 0000 | 32 |
| RPFn Extended Input Format Registers 1 | VI6_RPFn_EXT_INFMT1 | R/W | H'0374 + H'0100*n | H'0000 0000 | 32 |
| RPFn Extended Input Format Registers 2 | VI6_RPFn_EXT_INFMT2 | R/W | H'0378 + H'0100*n | H'0000 0000 | 32 |
| RPFn Bit Reduction And Dither Control | VI6_RPFn_BRDITH_CTRL | R/W | H'03E0 + H'0100*n | H'0000 0000 | 32 |
| IIF Control Registers (For only VSPX) | VI6_IIF_CTRL | W | H'0608 | H'0000 0000 | 32 |
| WPF0-Source-RPF Registers | VI6_WPF0_SRCRPF | R/W | H'1000 | H'0000 0000 | 32 |
| WPF0 Horizontal Input Size Clipping Registers | VI6_WPF0_HSZCLIP | R/W | H'1004 | H'0000 0000 | 32 |
| WPF0 Vertical Input Size Clipping Registers | VI6_WPF0_VSZCLIP | R/W | H'1008 | H'0000 0000 | 32 |
| WPF0 Output Format Registers | VI6_WPF0_OUTFMT | R/W | H'100c | H'0000 0000 | 32 |
| WPF0 Data Swapping Registers | VI6_WPF0_DSWAP | R/W | H'1010 | H'0000 0000 | 32 |
| WPF0 Rounding Control Registers | VI6_WPF0_RNDCTRL | R/W | H'1014 | H'0000 0000 | 32 |
| WPF0 Destination Y Plane Memory Stride Registers | VI6_WPF0_DSTM_STRIDE_Y | R/W | H'101c | H'0000 0000 | 32 |
| WPF0 Destination C Plane Memory Stride Registers | VI6_WPF0_DSTM_STRIDE_C | R/W | H'1020 | H'0000 0000 | 32 |
| WPF0 Destination Y/RGB Address Registers | VI6_WPF0_DSTM_ADDR_Y | R/W | H'1024 | H'0000 0000 | 32 |
| WPF0 Destination Chroma Address Registers 0 | VI6_WPF0_DSTM_ADDR_C0 | R/W | H'1028 | H'0000 0000 | 32 |
| WPF0 Destination Chroma Address Registers 1 | VI6_WPF0_DSTM_ADDR_C1 | R/W | H'102c | H'0000 0000 | 32 |
| WPF0 LIF Write Back Control Registers | VI6_WPF0_WRBCK_CTRL | R/W | H'1034 | H'0000 0000 | 32 |
| Registers User Logic 4 | Registers User Logic 4 *2 | - | H'1c00 to H'1cfc | - | 32 |
| Registers User Logic 5 | Registers User Logic 5 *2 | - | H'1d00 to H'1dfc | - | 32 |
| RPF0 Routing Register | VI6_DPR_RPF0_ROUTE | R/W | H'2000 | H'0000 0000 | 32 |

| | | | | | |
|---|---|---|---|---|---|
| RPF1 Routing Register | VI6_DPR_RPF1_ROUTE | R/W | H'2004 | H'0000 0000 | 32 |
| RPF2 Routing Register | VI6_DPR_RPF2_ROUTE | R/W | H'2008 | H'0000 0000 | 32 |
| RPF3 Routing Register | VI6_DPR_RPF3_ROUTE | R/W | H'200c | H'0000 0000 | 32 |
| RPF4 Routing Register | VI6_DPR_RPF4_ROUTE | R/W | H'2010 | H'0000 0000 | 32 |
| WPF0 Timing Control Register | VI6_DPR_WPF0_FPORCH | R/W | H'2014 | H'0000 0000 | 32 |
| BRU Routing Register | VI6_DPR_BRU_ROUTE | R/W | H'204c | H'0000 0000 | 32 |
| ILV Routing Register | VI6_DPR_ILV _ROUTE | R/W | H'2050 | H'0000 0000 | 32 |
| UIF4 Routing Register | VI6_DPR_UIF4_ROUTE | R/W | H'2074 | H'0000 0000 | 32 |
| UIF5 Routing Register | VI6_DPR_UIF5_ROUTE | R/ | H'2078 | H'0000 0000 | 32 |
| BRU Input Control Register | VI6_BRU_INCTRL | R/W | H'2c00 | H'0000 0000 | 32 |
| Size Register of BRU Input Virtual RPF | VI6_BRU_VIRRPF_SIZE | R/W | H'2c04 | H'0000 0000 | 32 |
| Display Location Register of BRU Input Virtual RPF | VI6_BRU_VIRRPF_LOC | R/W | H'2c08 | H'0000 0000 | 32 |
| Color Information Register of BRU Input Virtual RPF | VI6_BRU_VIRRPF_COL | R/W | H'2c0c | H'0000 0000 | 32 |
| BRU Control Registers A | VI6_BRUA_CTRL | R/W | H'2c10 | H'0000 0000 | 32 |
| BRU Blend Control Registers A | VI6_BRUA_BLD | R/W | H'2c14 | H'0000 0000 | 32 |
| BRU Control Registers B | VI6_BRUB_CTRL | R/W | H'2c18 | H'0000 0000 | 32 |
| BRU Blend Control Registers B | VI6_BRUB_BLD | R/W | H'2c1c | H'0000 0000 | 32 |
| BRU Control Registers C | VI6_BRUC_CTRL | R/W | H'2c20 | H'0000 0000 | 32 |
| BRU Blend Control Registers C | VI6_BRUC_BLD | R/W | H'2c24 | H'0000 0000 | 32 |
| BRU Control Registers D | VI6_BRUD_CTRL | R/W | H'2c28 | H'0000 0000 | 32 |
| BRU Blend Control Registers D | VI6_BRUD_BLD | R/W | H'2c2c | H'0000 0000 | 32 |
| BRU Raster Operation Control Register | VI6_BRU_ROP | R/W | H'2c30 | H'0000 0000 | 32 |
| BRU Control Registers E | VI6_BRUE_CTRL | R/W | H'2c34 | H'0000 0000 | 32 |
| BRU Blend Control Registers E | VI6_BRUE_BLD | R/W | H'2c38 | H'0000 0000 | 32 |
| ILV Control Register | VI6_ILV_CTRL | R/W | H'3800 | H'0000 0000 | 32 |
| LIF0 Control Registers | VI6_LIF0_CTRL | R/W | H'3b00 | H'0000 0000 | 32 |
| LIF0 Clock Stop Buffer Control Register | VI6_LIF0_CSBTH | R/W | H'3b04 | H'0000 0000 | 32 |
| LIF0 Padding Line Cycle Register | VI6_LIF0_PADLN_CYC | R/W | H'3b30 | H'0000 0000 | 32 |
| LIF0 Padding Line Pattern Register | VI6_LIF0_PADLN_PT | R/W | H'3b34 | H'0000 0000 | 32 |
| LIF0 Padding Line Value Register | VI6_LIF0_PADLN_VAL | R/W | H'3b38 | H'0000 0000 | 32 |
| LIF0 Padding Line Size Register | VI6_LIF0_PADLN_SIZE | R/W | H'3b3C | H'0000 0000 | 32 |
| IP Version Register | VI6_IP_VERSION | R | H'3f00 | (*3) | 32 |
| RPF n OSD-CLUT Table | VI6_CLUTn_TBL0 to VI6_CLUTn_TBL255 | R/W | H'4000 + H'0400*n to H'43fc + H'0400*n | Undefined | 32 |

(*1) Read only bits or write only bits or read/write bits are mixed in the registers.

(*2) DISCOM registers is explained in the section 60.5.2

(*3) Initial value of IP version registers depend on product and Modules. See section 60.2.12.1 for more detail.

### 60.1.5 Connected Module

Table 60.5 shows modules connected to the VSP2. Table 60.6 shows channel number of DU which each VSPD/LIF is connected.

**Table 60.5 Connected Module**

| Module | Connected Module | Description |
|---|---|---|
| VSPX | FCPVX | FCPV for VSPX |
| | VIN | Video Input Module |
| | ISP | Image signal processor |
| VSPD | FCPVD | FCPV for VSPD |
| | VIN | Video Input Module |
| | DU | Display Unit |
| | TX-LINK | DSI/CSI2 TX LINK |

**Table 60.6 Correspondence between ch number of VSPD/LIF and ch number of DU**

| VSPD ch | LIF ch | DU ch |
|---|---|---|
| VSPD0 | LIF0 | DU0 |

Bus access of VSP2 is carried out through FCPV (Frame Compression IP for VSP) module in R-Car. So, refer to also FCP User's Manual.

## 60.2 Register Description

### 60.2.1 Notational Conventions for Registers and Bit Fields

This document uses the following notational conventions for the VSP2 registers and bit fields.

1. The names of registers and bits are written in uppercase.
2. A bit or bit field in a register is indicated as [register name.bit name]. For example, the STRCMD bit in the VI6_CMD0 register is indicated as VI6_CMD0.STRCMD.
3. Lowercase "n" in a register name or a bit name indicates an integer and the range of value n is defined when necessary. For example, VI6_RPFn_SRC_BSIZE.BHSIZE (n = 0, 1, 2, 3, 4) indicates the BHSIZE bit in five registers VI6_RPF0_SRC_BSIZE.BHSIZE, VI6_RPF1_SRC_BSIZE.BHSIZE, VI6_RPF2_SRC_BSIZE.BHSIZE, VI6_RPF3_SRC_BSIZE.BHSIZE and VI6_RPF4_SRC_BSIZE.BHSIZE. For RPFn and UIFn, when the range of value n is not defined, RPFn (n = 0, 1, 2, 3, 4) and UIFn (n = 4, 5) are assumed.
4. In each subsection for register description in section 60.2, when only a bit name is written without showing its register name, the bit is in the register described in that subsection.
5. A wildcard (*) indicates any characters in a name and represents all registers or bits that match the specified first part of a name. For example, when there are two registers VI6_RPF_SRC_BSIZE and VI6_RPF_SRC_ESIZE, VI6_RPF_SRC_* indicates both registers.

### 60.2.2 Restrictions on Access to Registers and Lookup Tables

The VSP2 has control registers and lookup tables. All VSP2 registers are writable and readable by only 32 bits unit. To write partial bits in each register, read-modify-write is needed. When accessing the addresses where these registers and lookup tables are allocated, the following restrictions should be satisfied. If any restriction is violated, the VSP2 will not operate correctly.

1. For the read-only bits and reserved bits in all VSP2 registers, writing 1 is prohibited unless otherwise specified.
2. Addresses undefined in section 60.1.4 are reserved areas and write access is prohibited in these areas. Addresses undefined in section 60.5.2.4 for UIF4/UIF5 (DISCOM) are reserved areas and write access is prohibited in these areas.
3. For all registers and lookup tables, except VI6_CMD0, VI6_SRESET, VI6_*IRQ* and two plane registers such as VI6_DL_HDR_ADDR0, VI6_DL_BODY_SIZE0, modifying register values during operation of the module is prohibited. Modify registers while the corresponding module is stopped. For the operating status of the target module, refer to section 60.3.2.

**Table 60.7 Correspondence between Modules and Register Names**

| Module Name | Register Name |
|---|---|
| RPFn (n=0 to 4) | VI6_RPFn_* |
| WPF0 | VI6_WPF0_* |
| DPR | VI6_DPR_* |
| UIFn (n=4, 5) | VI6_UIFn_* |
| BRU | VI6_BRU_* |
| ILV | VI6_ILV_* |
| LIF0 | VI6_LIF0_* |

### 60.2.3 Register Classification

The VSP2 registers are arranged in the following order; the general control registers to display list control registers control operation of the entire VSP2, and the other registers control each image processing and specify parameters for the processing. The functions of the registers are described in this order starting from section 60.2.4.

1. General control registers
2. Display list control registers (VI6_DL_*)
3. RPF control registers (VI6_RPFn_* (n = 0, 1, 2, 3, 4))
4. WPF control registers (VI6_WPF0_*)
5. DPR control registers (VI6_DPR_*)
6. BRU control registers (VI6_BRU_*)
7. ILV control register (VI6_ILV_CTRL)
8. LIF control registers (VI6_LIF0_*)
9. VSP Internal Registers
10. IIF control registers (VI6_IIF_CTRL)

The VSP2 has five RPF channels and the register configuration is the same for all of RPF0 to RPF4. However, some bit fields have restrictions in certain RPFs. These restrictions are included in the description of the corresponding bit fields and registers. The restrictions are included in the description of the corresponding bit fields and registers.

For each register address, refer to section 60.1.4

### 60.2.4 General Control Registers

#### 60.2.4.1 VSP2 Start Registers (VI6_CMD0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | UPDHDR | – | – | – | STRCMD |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5, 3 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 4 | UPDHDR | 0 | R/W | Reserved state of updating DLH address.<br>UPDHDR can be negated by software reset (VI6_SRESET.SRSTn).<br>[Write] 0: NOP<br>1: Don't set 1 to this bit<br>[Read] 0: Not reserved to update DLHeader address<br>1: Reserved to update DLHeader address<br>Set 0 to this bit when STRCMD bit is set to 1. |
| 0 | STRCMD | 0 | R/W | Start reservation of each WPF<br>VSP2 has two output channels (WPFs). Two WPFs can be started at the same time.<br>VI6_CMD0.STRCMD controls WPF0. Writing 1 to this bit starts WPF0 in VSP2. Set this bit for activation only after all register settings in each output channel have been completed.<br>If WPF0 is idle, WPF0 starts right after this bit is set to 1.<br>If WPF0 is active, writing 1 to this bit reserves starting WPF0 operation. Wait to set 1 to VI6_CMD0.STRCMD until this bit is read as 0.<br>VI6_CMD0.STRCMD can be negated by software reset (VI6_SRESET.SRST0).<br>[Write] 0: Start reservation of WPF0 is canceled.<br>1: Start reservation of WPF0 is set.<br>[Read] 0: Starting VSP is not reserved.<br>1: Starting VSP is reserved. |

The basic concept of image processing operation started by activating the VSP2 is shown in Figure 60.6. The actual data input/output is executed by the MAU, which is the bus interface module, as described in section 60.1.2, but conceptually the RPF works as the data entry point to the VSP2 and the WPF works as the data exit point. To process images through the VSP2, the RPF (entrance) and WPF (exit) should be connected and a data path from the RPF to the WPF should be formed.

To connect RPFn to WPF0, specify RPFn as the source RPF for WPF0 in VI6_WPF0_SRCRPF, which is a register for WPF0 (refer to section 60.2.7.1). This setting determines that RPFn will be started when WPF0 is started through VI6_CMD0.

A data path to execute desired image processing should then be formed between the RPF (entrance) and WPF (exit). To form a data path, connect the necessary function modules in the VSP2 between RPF and WPF. This function is provided by the DPR; specify the information for each module connection in data path routing registers VI6_DPR_*_ROUTE (refer to section 60.2.8).

After a data path is formed (RPFn → WPF0) as described above, starting output module WPF0 in the VSP2 through VI6_CMD0 starts all function modules connected to WPF0 and the desired image processing is executed. According to this design concept, starting a WPF module means starting the VSP2.

There are two types of data path configuration in the VSP2; one is "a single input module to a single output module" as shown in Figure 60.6 (A), and the other is "multiple input modules to a single output module" as shown in Figure 60.6 (B).

Figure 60.6 (A) shows an example of a configuration where modules with single input and single output are implemented through the DPR.

Figure 60.6 (B) shows another configuration example where the module with multiple input and single output is implemented through the DPR.

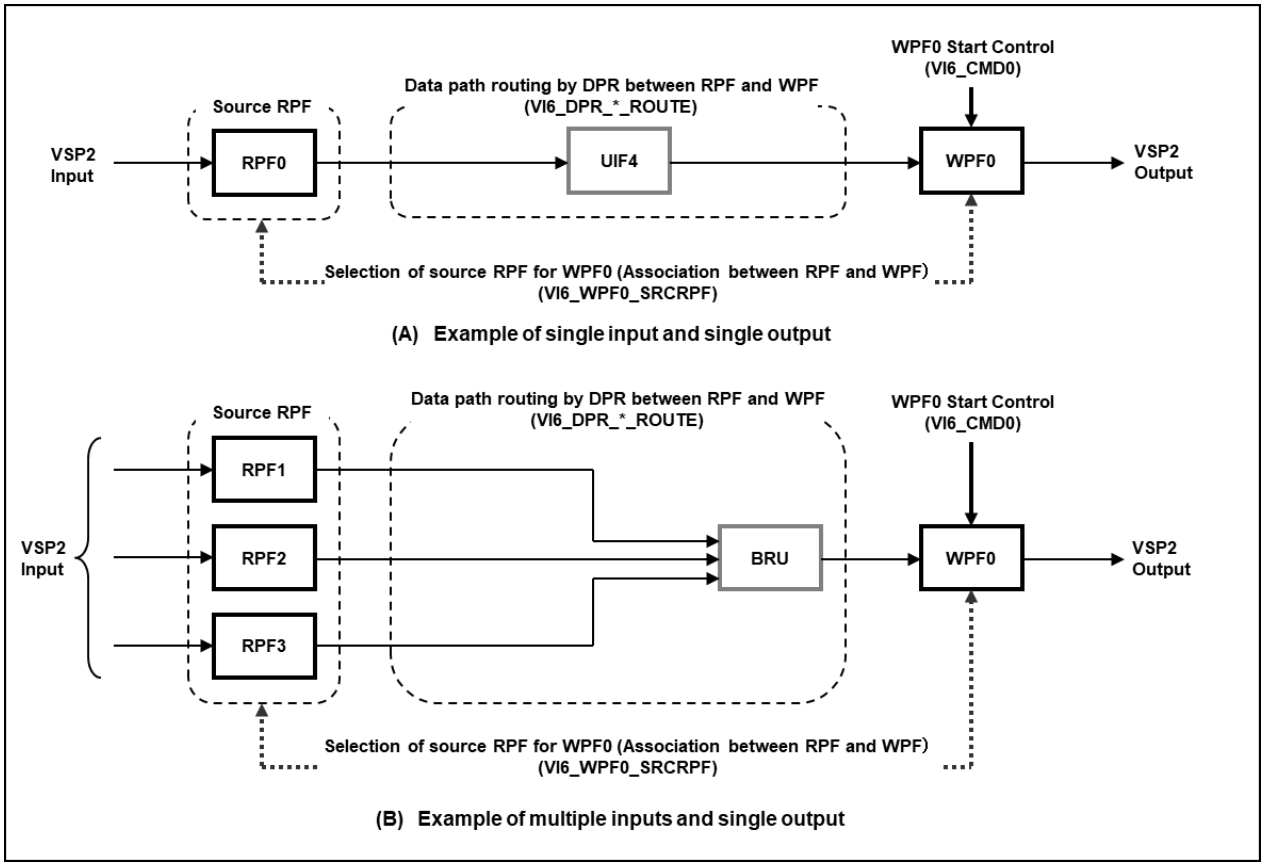



**Figure 60.6 Basic Concept of VSP2 Startup**

### 60.2.4.2 Clock Control Register 0 (VI6_CLK_CTRL0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | GCS0 | – | – | – | – | – | – | – | – | – | – | – | GCS1 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | GCS2 | | | | – | – | – | GCS3 | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 31 to 29,<br>27 to 17,<br>15 to 12,<br>7 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 | GCS0 | 0 | R/W | Clock Control Setting0 |
| 16 | GCS1 | 0 | R/W | Clock Control Setting1 |
| 11 to 8 | GCS2 | 0 | R/W | Clock Control Setting2 |
| 4 to 0 | GCS3 | 0 | R/W | Clock Control Setting3 |

VSP2 can stop its operating clock for reducing power consumption.

To enable clock stop function, set following registers:

- VI6_CLK_CTRL0 = H'0000 0000
- VI6_CLK_CTRL1 = H'0000 0000
- VI6_CLK_DCSWT = H'0000 0808
- VI6_CLK_DCSM0 = H'0000 0000
- VI6_CLK_DCSM1 = H'0000 0000

To disable clock stop function, set following registers:

- VI6_CLK_CTRL0 = H'1001 0F1F
- VI6_CLK_CTRL1 = H'FFFF FFFF
- VI6_CLK_DCSWT = H'0033 0808
- VI6_CLK_DCSM0 = H'1FFF 0F1F
- VI6_CLK_DCSM1 = H'FF10 FFFF

When the clock gating is enabled, the supply of the clock signal is stopped for the module which is not operated. If the bit field of this register is changed from 0 to 1, the supply of the clock signal for the module is immediately started. If the bit field of the register is changed from 1 to 0, the clock supply is immediately stopped. Therefore, please do not change this register field from 1 to 0 in operation because the VSP2 may be stalled.

### 60.2.4.3 Clock Control Register 1 (VI6_CLK_CTRL1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GCS4 | | | | | | | | GCS6 | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | GCS5 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | GCS4 | 0 | R/W | Clock Control Setting4 |
| 23 to 16 | GCS6 | 0 | R/W | Clock Control Setting6 |
| 15 to 0 | GCS5 | 0 | R/W | Clock Control Setting5 |

See section 60.2.4.2 for detail.

### 60.2.4.4 Dynamic Clock Stop Control Register (VI6_CLK_DCSWT)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | DCC0 [1:0] | | – | – | DCC1 | DCC2 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CSTPW [7:0] | | | | | | | | CSTRW [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 22, 19 to 18 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 21, 20 | DCC0 | 0 | R/W | Dynamic Clock Control Setting0 |
| 17 | DCC1 | 0 | R/W | Dynamic Clock Control Setting1 |
| 16 | DCC2 | 0 | R/W | Dynamic Clock Control Setting2 |
| 15 to 8 | CSTPW [7:0] | 0 | R/W | Dynamic Clock Stop Control 1<br>Always specify 8. |

| | | | | |
|---|---|---|---|---|
| 7 to 0 | CSTRW [7:0] | 0 | R/W | Dynamic Clock Stop Control 2<br>Always specify 8. |

See section 60.2.4.2 for detail.

### 60.2.4.5 Dynamic Clock Stop Disable Register 0 (VI6_CLK_DCSM0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | DCD0 | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | DCD1 | | | | – | – | – | DCD2 | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29,<br>15 to 12,<br>7 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 16 | DCD0 | 0 | R/W | Dynamic Clock Stop Disable Setting0 |
| 11 to 8 | DCD1 | 0 | R/W | Dynamic Clock Stop Disable Setting1 |
| 4 to 0 | DCD2 | 0 | R/W | Dynamic Clock Stop Disable Setting2 |

See section 60.2.4.2 for detail.

### 60.2.4.6 Dynamic Clock Stop Disable Register 1 (VI6_CLK_DCSM1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCD3 | | | | | | | | – | – | – | DCD5 | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCD4 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 23 to 21, 19 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 31 to 24 | DCD3 | 0 | R/W | Dynamic Clock Stop Disable Setting3 |
| 20 | DCD5 | 0 | R/W | Dynamic Clock Stop Disable Setting5 |
| 15 to 0 | DCD4 | 0 | R/W | Dynamic Clock Stop Disable Setting4 |

See section 60.2.4.2 for detail.

### 60.2.4.7 Software Reset Register (VI6_SRESET)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | SRST0 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 0 | SRST0 | 0 | R/W | WPF0 Software Reset<br>Writing 1 to this bit aborts the current processing in WPF0 (the partially-completed image undergoing processing is output). The period until this software reset processing is completed depends on the bus state.<br>When this reset processing is completed, the VI6_WPF0_IRQ_STA.FRE interrupt source bit is set to 1; when the FRE interrupt is enabled, the FRE end interrupt is output to notify the end of the reset processing.<br>This bit is always read as 0.<br>0: NOP<br>1: WPF0 software reset (*1) |

(*1) Applying a software reset to each WPF has the following restrictions.

1. The end of software reset processing is notified through the FRE bit in VI6_WPF0_IRQ_STA, but the software reset issued while WPF is stopped is ignored as NOP. As it takes a while until the reset is actually issued after the reset bit is set, the VSP2 may complete operation before the reset is actually issued. In this case, no interrupt is output for the software reset that is issued after the VSP2 completes operation.
2. If a software reset is issued during downloading of a display list, the downloading processing is not aborted. After the end of downloading that is in progress when a software reset is issued, a frame end interrupt is output.
3. When software reset is issued to VSP, issue software reset to also FCPV. See Figure 60.36 for VSPD software reset sequence. See FCP User's Manual for VSPX software reset sequence.

### 60.2.4.8 Module Reset Enable Register 0 (VI6_MRESET_ENB0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | MRSTE0 | | – | – | – | – | – | – | – | – | – | – | – | MRSET5 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | MRSTE1 | | | | – | – | – | MRSTE2 | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30, 27 to 17, 15 to 12, 7 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29 to 28 | MRSTE0 | 0 | R/W | Module Reset Enable 0 |
| 16 | MRSTE5 | 0 | R/W | Module Reset Enable 5 |
| 11 to 8 | MRSTE1 | 0 | R/W | Module Reset Enable 1 |

| | | | | |
|---|---|---|---|---|
| 4 to 0 | MRSTE2 | 0 | R/W | Module Reset Enable 2 |

This register is for purpose of h/w debugging.

### 60.2.4.9 Module Reset Enable Register 1 (VI6_MRESET_ENB1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MRSTE3 | | | | | | | | – | – | – | MRSTE6 | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MRSTE4 | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 23 to 21, 19 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 31 to 24 | MRSTE3 | 0 | R/W | Module Reset Enable 3 |
| 20 | MRSTE6 | 0 | R/W | Module Reset Enable 6 |
| 15 to 0 | MRSTE4 | 0 | R/W | Module Reset Enable 4 |

This register is for purpose of h/w debugging.

### 60.2.4.10 Module Reset Issuing Register (VI6_MRESET)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | MRST |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | MRST | 0 | R/W | Module Reset Assertion<br>This register is for purpose of h/w debugging. |

This register is for purpose of h/w debugging.

### 60.2.4.11 Operating Status Register (VI6_STATUS)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | - | FLDST0 | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | - | SYS0_ACT | – | – | – | – | – | – | – | DBG |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29, 27 to 9,<br>7 to 1 | – | All 0 | R | Reserved<br>These bits are read as 0 or 1. Do not care about this value.<br>Write access is prohibited. |
| 28 | FLDST0 | 0 | R | Field status of previous frame of WPF0.<br>0: Previous finished frame is TOP field.<br>1: Previous finished frame is BOT field.<br>This bit can be referred in case of AUTO-FLD or AUTO-DISP.<br>This bit is changed at the timing of VI6_WPF0_IRQ_STA.FRE. |
| 8 | SYS0_ACT | 0 | R | WPF0 Operating Status<br>Each bit indicates the operating or stopped state of control channel n (WPF0).<br>0: WPF0 is stopped.<br>1: WPF0 is operating. |

| | | | | |
|---|---|---|---|---|
| 0 | DBG | 0 | R | This register is for purpose of h/w debugging. |

### 60.2.4.12 WPF0 Interrupt Enable Registers (VI6_WPF0_IRQ_ENB)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | UNDE |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | DFEE | FREE |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

Each bit controls the interrupt enable of the corresponding interrupt source.

0: Interrupt Disabled

1: Interrupt Enabled

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17, 15 to 2 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | UNDE | 0 | R/W | Interrupt Enable for WPF0 Underrun in case of DU connection. |
| 1 | DFEE | 0 | R/W | Interrupt Enable for WPF0 Display List Frame End |
| 0 | FREE | 0 | R/W | Interrupt Enable for WPF0 Frame End |

Each bit in VI6_WPF0_IRQ_STA is set to 1 when the corresponding interrupt source is generated. VI6_WPF0_IRQ_ENB specifies whether to output an interrupt signal for the generated source. When an interrupt is disabled in this register, no interrupt signal is generated even when the corresponding bit in VI6_WPF0_IRQ_STA is set to 1. When an interrupt is enabled in this register, an interrupt signal is output when the corresponding bit in VI6_WPF0_IRQ_STA is set to 1.

### 60.2.4.13 WPF0 Interrupt Status Registers (VI6_WPF0_IRQ_STA)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | UND |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | DFE | FRE |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

The read value from each bit is the status of the interrupt source, and the write access to each bit controls the interrupt status.

[Read Access] Interrupt Status

0 : No interrupt

1 : Interrupt activated

[Write Access] Interrupt Clear

0 : The interrupt status is cleared to 0

1 : Hold the interrupt status value

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17, 15 to to 2 | – | All 0 | R | Reserved<br>These bits are read as 0 or 1. Do not care about this value. The write value should always be 0. |
| 16 | UND | 0 | R/W | Interrupt Status and Clear for WPF0 Underrun in case of DU connection.<br>This interrupt source bit is set to 1 when data underrun occurs in case of DU connection.<br>Timing that this bit is set to 1 is at end of 1 frame. |
| 1 | DFE | 0 | R/W | Interrupt Status and Clear for WPF0 Display List Frame End<br>This interrupt source bit is set to 1 when VSP2 completes one-frame processing while the current frame in enable value stored in the display list header is 1 (refer to section 60.3.3.2).<br>When display lists are not used, this bit is not used. In this case, clear VI6_WPF0_IRQ_ENB.DFEE to 0 to mask the interrupt generation by this interrupt source.<br>This bit can hold the most recent of two times of interrupt status at the maximum. When writing 0 to this bit, the oldest interrupt status is cleared. And the status bit becomes 0 after all interrupt statuses are cleared. |
| 0 | FRE | 0 | R/W | Interrupt Status and Clear for WPF0 Frame End<br>This interrupt source bit is set to 1 when VSP2 completes one-frame processing. This bit is also set to 1 when one-frame processing using a display list is completed. |

| | | | | |
|---|---|---|---|---|
| | | | | The interrupt status is set to 1 by any of the following conditions.<br>a) Processing one frame is finished normally<br>b) Software reset is issued during VSP is processing.<br>c) One frame's data from DU is displayed while VSP couldn't transfer one frame data to DU. (in case of Linked with DU)<br>This bit can hold the most recent of two times of interrupt status at the maximum. When writing 0 to this bit, the oldest interrupt status is cleared. And the status bit becomes 0 after all interrupt statuses are cleared. |

VI6_WPF0_IRQ_STA indicates the state of the interrupt sources generated in the VSP2. Whether to output a VSP2 interrupt when an interrupt source is generated and the corresponding bit is set to 1 is determined by the corresponding bit setting in VI6_WPF0_IRQ_ENB. While an interrupt is disabled in VI6_WPF0_IRQ_ENB, the VSP2 does not output an interrupt signal even when an interrupt source is generated, but the source flag in this register is set to 1.

Note that the interrupt source bits in this register cannot be cleared by write access using a display list.

### 60.2.4.14 Display-0 Interrupt Enable Register (VI6_DISP0_IRQ_ENB)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | DSTE | – | – | MAEE | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R/W | R | R | R | R | R |

Each bit controls the interrupt enable of the corresponding interrupt source.

0 : Interrupt Disabled

1 : Interrupt Enabled

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9,<br>7 to 6,<br>4 to 0 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | DSTE | 0 | R/W | Interrupt Enable for Display Start |
| 5 | MAEE | 0 | R/W | Interrupt Enable for Display Read Data End |

### 60.2.4.15 Display-0 Interrupt Status Register (VI6_DISP0_IRQ_STA)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | DST | – | – | MAE | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R/W | R | R | R | R | R |

The read value from each bit is the status of the interrupt source, and the write access to each bit controls the interrupt status.

[Read Access] Interrupt Status

0: No interrupt

1: Interrupt activated

[Write Access] Interrupt Clear

0: The interrupt status is cleared to 0

1: Hold the interrupt status value

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9,<br>7 to 6,<br>4 to 0 | – | All 0 | R | Reserved<br>These bits are always read as 0 or 1. The write value should always be 0. |
| 8 | DST | 0 | R/W | Interrupt Status and Clear for Display Start<br>This bit is set to 1 when LIF module transfers the first data to the display module at the beginning of each frame. The timing depends on the output buffer status of LIF module. |
| 5 | MAE | 0 | R/W | Interrupt Status and Clear for Display Read Data End<br>This bit is set to 1 when all RPFs transfer the last data of the frame to LIF module. The RPF module which is not used by LIF module does not affect this bit. |

### 60.2.4.16 Wake-up0 Interrupt Enable Registers (VI6_WUP0_IRQ_ENB)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | VWER15E | VWER14E | VWER13E | VWER12E | VWER11E | VWER10E | VWER9E | VWER8E | VWER7E | VWER6E | VWER5E | VWER4E | VWER3E | VWER2E | VWER1E | VWER0E |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Each bit controls the interrupt enable of the corresponding interrupt source.

0: Interrupt Disabled

1: Interrupt Enabled

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | VWER15E | 0 | R/W | Interrupt Enable for VINn Wake up Error.<br>(VWERnE: n= 0 to 15) |
| 14 | VWER14E | 0 | R/W | |
| 13 | VWER13E | 0 | R/W | |
| 12 | VWER12E | 0 | R/W | |
| 11 | VWER11E | 0 | R/W | |
| 10 | VWER10E | 0 | R/W | |
| 9 | VWER9E | 0 | R/W | |
| 8 | VWER8E | 0 | R/W | |
| 7 | VWER7E | 0 | R/W | |
| 6 | VWER6E | 0 | R/W | |
| 5 | VWER5E | 0 | R/W | |
| 4 | VWER4E | 0 | R/W | |
| 3 | VWER3E | 0 | R/W | |
| 2 | VWER2E | 0 | R/W | |
| 1 | VWER1E | 0 | R/W | |
| 0 | VWER0E | 0 | R/W | |

Each bit in VI6_WUP0_IRQ_STA is set to 1 when the corresponding interrupt source is generated. VI6_WUP0_IRQ_ENB specifies whether to output an interrupt signal for the generated source. When an interrupt is disabled in this register, no interrupt signal is generated even when the corresponding bit in VI6_WUP0_IRQ_STA is set to 1. When an interrupt is enabled in this register, an interrupt signal is output when the corresponding bit in VI6_WUP0_IRQ_STA is set to 1.

### 60.2.4.17 Wake-up0 Interrupt Status Registers (VI6_WUP0_IRQ_STA)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VWER15 | VWER14 | VWER13 | VWER12 | VWER11 | VWER10 | VWER9 | VWER8 | VWER7 | VWER6 | VWER5 | VWER4 | VWER3 | VWER2 | VWER1 | VWER0 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The read value from each bit is the status of the interrupt source, and the write access to each bit controls the interrupt status.

[Read Access] Interrupt Status

0 : No interrupt

1 : Interrupt activated

[Write Access] Interrupt Clear

0 : The interrupt status is cleared to 0

1 : Hold the interrupt status value

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | VWER15 | 0 | R/W | Interrupt Status and Clear for VINn Wake up Error<br>(VWERn: n=0 to 15)<br>VSPX can hold 15 outstanding WUPs from VINn . This interrupt source bit is set to 1 when WUP pulse asserts from VINn when wake up counter is 15. While this bit is 1, relevant wake-up counter (VI6_DL_VIN_WUPCNT0.WUPCNTn [3:0] / VI6_DL_VIN_WUPCNT1.WUPCNTn [3:0]) doesn't decrease by VSPX start (Lock of wake-up counter decrement). |
| 14 | VWER14 | 0 | R/W | |
| 13 | VWER13 | 0 | R/W | |
| 12 | VWER12 | 0 | R/W | |
| 11 | VWER11 | 0 | R/W | |
| 10 | VWER10 | 0 | R/W | |
| 9 | VWER9 | 0 | R/W | |
| 8 | VWER8 | 0 | R/W | |
| 7 | VWER7 | 0 | R/W | |
| 6 | VWER6 | 0 | R/W | |
| 5 | VWER5 | 0 | R/W | |
| 4 | VWER4 | 0 | R/W | |
| 3 | VWER3 | 0 | R/W | |
| 2 | VWER2 | 0 | R/W | |
| 1 | VWER1 | 0 | R/W | |

| | | | | |
|---|---|---|---|---|
| 0 | VWER0 | 0 | R/W | |

### 60.2.5 Display List Control Registers

#### 60.2.5.1 Display List Control Register (VI6_DL_CTRL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AR_WAIT [15:0] | | | | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | DC2 | – | – | – | DC1 | – | – | – | DC0 | RLM0 | CFM0 | NH0 | DLE0 |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R | R | R | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | AR_WAIT [15:0] | All 0 | R/W | Display List Control Setting<br>Always specify 256. |
| 15 to 13, 11 to 9, 7 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 | DC2 | 0 | R/W | This bit doesn't affect anything to VSP (NOP) |
| 8 | DC1 | 0 | R/W | This bit doesn't affect anything to VSP (NOP) |
| 4 | DC0 | 0 | R/W | This bit doesn't affect anything to VSP (NOP) |
| 3 | RLM0 | 0 | R/W | Loading two plane Registers Mode for WPF0.<br>0: Reserved status (VI6_CMD0.UPDHDR) is accepted by next frame auto start.<br>Two plane registers are downloaded by next_frame_start also.<br>1: Reserved status (VI6_CMD0.UPDHDR) is not accepted by next frame auto start.<br>pNextHeader in previous frame's DLH is used for loading DisplayList.<br>Two plane registers are not downloaded by next_frame_start also. |
| 2 | CFM0 | 0 | R/W | Continuous Frame Mode for Header-less Display List for WPF0<br>This bit determines whether the next frame is automatically started or not. When the updated flag of the display list, VI6_DL_BODY_SIZE0.UPD0, is not updated, the display list of the next frame is not transferred and the same register values are |

| | | | | |
|---|---|---|---|---|
| | | | | used for the next frame. When the value of VI6_DL_BODY_SIZE0.UPD0 is updated, the new display list is transferred.<br>0: Stopped at the end of every frame<br>1: The next frame is automatically started |
| 1 | NH0 | 0 | R/W | Header-less Display List Mode<br>This bit is used for specifying the header-less display list mode. In case of header-less mode, the number of the display lists is 1. The address of the display body is set in VI6_DL_HDR_ADDR0 register, and the body size is set in VI6_DL_BODY_SIZE0 register.<br>When this bit is changed, make sure that VSP2 is stopped. And also make sure the following before starting VSP2.<br>- Header Address (VI6_DL_HDR_ADDR0)<br>- Body Size (VI6_DL_BODY_SIZE0) in case of header-less mode<br>0: Use Display List Header (Normal DL Mode)<br>1: Don't use Display List Header (Header-less Mode)<br>Note0: When DLE0 bit is 0, set 0 to NH0 bit. |
| 0 | DLE0 | 0 | R/W | Display List Enable/Disable for WPF0<br>Enables or disables the WPF0 display list function. When the display list function is enabled through this bit, all WPF processing channels work in display list mode.<br>When using display lists, note the restrictions in section 60.2.2.<br>0: The display list function is disabled<br>1: The display list function is enabled |

### 60.2.5.2 Display List-0 Data Swapping Register (VI6_DL_SWAP0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | LWS | WDS | BTS |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 30 to 3 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | LWS | 0 | R/W | Display List Data Swapping in long word Units<br>The effect of this bit setting is defined in Table 60.8. |

| | | | | |
|---|---|---|---|---|
| | | | | 0: Data swapping in long word (32-bit) units is disabled<br>1: Data swapping in long word (32-bit) units is enabled |
| 1 | WDS | 0 | R/W | Display List Data Swapping in Word Units<br>The effect of this bit setting is defined in Table 60.8.<br>0: Data swapping in word (16-bit) units is disabled<br>1: Data swapping in word (16-bit) units is enabled |
| 0 | BTS | 0 | R/W | Display List Data Swapping in Byte Units<br>The effect of this bit setting is defined in Table 60.8.<br>0: Data swapping in byte (8-bit) units is disabled<br>1: Data swapping in byte (8-bit) units is enabled |

Table 60.8 shows the data order before and after swapping according to the long word, word, and byte swapping settings.

When data order in memory for each format is the same as Table 60.34, set 3'b111 to {LWS, WDS, BTS}. If data order is not the same as the definition, change data order within 8byte unit by these bits as shown in Table 60.8.

**Table 60.8 Changing data order according to display list swap register**

| Data order in memory | | | | | | | | | *_LWS | *_WDS | *_BTS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Byte address | 8n+0 | 8n+1 | 8n+2 | 8n+3 | 8n+4 | 8n+5 | 8n+6 | 8n+7 | | | |
| Data | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 1 | 1 |
| | 1 | 0 | 3 | 2 | 5 | 4 | 7 | 6 | 1 | 1 | 0 |
| | 2 | 3 | 0 | 1 | 6 | 7 | 4 | 5 | 1 | 0 | 1 |
| | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 1 | 0 | 0 |
| | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 0 | 1 | 1 |
| | 5 | 4 | 7 | 6 | 1 | 0 | 3 | 2 | 0 | 1 | 0 |
| | 6 | 7 | 4 | 5 | 2 | 3 | 0 | 1 | 0 | 0 | 1 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 0 |

⇩

| Data order defined in Table 60.34 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte address | 8n+0 | 8n+1 | 8n+2 | 8n+3 | 8n+4 | 8n+5 | 8n+6 | 8n+7 |
| Data | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

### 60.2.5.3 Display List-0 Header Address Register (VI6_DL_HDR_ADDR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DL_HDR_PTR-0 [31:16] | | | | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DL_HDR_PTR-0 [15:0] | | | | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DL_HDR_PTR-0 [31:0] | All 0 | R/W | Display List-0 Header Address<br>These bits specify the address of the display list header to be read for display list-0 in 16-byte units (the lower-order four bits are read-only). When WPF0 is first started in display list mode, the display list header is loaded from the address specified in this register. After loading of the header is completed, the register value of the display list address is updated to the next header address stored in the loaded header to prepare for loading of the next display list header. After that, this header address updating is repeated.<br>When VI6_DL_CTRL.NH0 is 0, writing header address to this register reserves updating two plane registers for the next frame. Two plane registers is VI6_DL_HDR_ADDR0.<br>A value from H'0000 0000 to H'FFFF FFF0 can be specified. |

### 60.2.5.4 Extended Display List Control Register (VI6_DL_EXT_CTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | NWE |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | POLINT [5:0] | | | | | | – | – | DLPRI | EXPRI | – | – | – | EXT |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17, 15, 14, 7, 6, 3 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | NWE | 0 | R/W | No Wait for Polling<br>When this bit is set to 1, the polling condition for extended display lists is always assumed to be true. |
| 13 to 8 | POLINT [5:0] | All 0 | R/W | Extended Display List Command Control<br>Always specify 2. |
| 5 | DLPRI | 0 | R/W | Display List Control 0<br>Always specify 1. |
| 4 | EXPRI | 0 | R/W | Display List Control 1<br>Always specify 0. |
| 0 | EXT | 0 | R/W | Extended Display List for WPF0 |

| | | | | |
|---|---|---|---|---|
| | | | | Enables or disables the extended display list function. When extended display lists are used, the display list header size is 96 bytes; when they are not used, the header size is 80 bytes.<br>0: No extended display lists are used<br>1: Extended display lists are used<br>[Note]<br>When using extended display lists, be sure to also use normal display list mode (VI6_DL_CTRL.DLEn); executing only extended display lists is not possible.<br>When the header-less display list mode is activated, this bit should be set to 0. The extended display list cannot be used with the header-less display list mode.<br>Extended Display List is to realized AUTO-FLD or AUTO-DISP. |

### 60.2.5.5 Display List Body Size Register-n (VI6_DL_BODY_SIZE0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | UPD0 | – | – | – | – | – | – | – | BS0[16] |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BS0[15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25,<br>23 to 17 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 24 | UPD0 | 0 | R/W | Update Flag<br>This bit controls the download of the display list at the next downloading timing in case that the header-less display list mode is used. Set 1 to this bit when display list need to be downloaded for next frame. This bit is automatically cleared to 0 after VI6_DL_HDR_ADDR0 and VI6_DL_BODY_SIZE0.BS0 is downloaded in H/W side. When this bit is set to 1, the value of VI6_DL_HDR_ADDR0 and VI6_DL_BODY_SIZE0.BS0 should not be changed.<br>0: Updating display list for the next frame is not reserved<br>1: Updating display list for the next frame is reserved |
| 16 to 0 | BS0[16:0] | 0 | R/W | Header-less Display List Body Size<br>These bits are used for specifying the body size of the display list in case of header-less display list mode. The unit of the size is byte. The value should be set in multiples of 8. |

See section 60.3.6.2 for detail.

### 60.2.5.6 Display List-0 Header Reference Address Register (VI6_DL_HDR_REF_ADDR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DL_HDR_REF_PTR-0 [31:16] | | | | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DL_HDR_REF_PTR-0 [15:0] | | | | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DL_HDR_REF_PTR-0 [31:0] | All 0 | R | Display List-0 reference Header Address<br>Following value is read out from the VI6_DL_HDR_REF_ADDR0 for each period.<br>(1) When H/W is reading display list from external memory, header address of the display list referred by VSP2-H/W is read out.<br>(2) When H/W is not reading display list from external memory, the value of VI6_DL_HDR_ADDR0 is read out.<br>For details, refer to section 60.3.6 |

### 60.2.5.7 Display List VIN Wake Up Counter 0 Register (VI6_DL_VIN_WUPCNT0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUPCNT7 [3:0] | | | | WUPCNT6 [3:0] | | | | WUPCNT5 [3:0] | | | | WUPCNT4 [3:0] | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUPCNT3 [3:0] | | | | WUPCNT2 [3:0] | | | | WUPCNT1 [3:0] | | | | WUPCNT0 [3:0] | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | WUPCNT7 [3:0] | 0 | R/W | |

| | | | | |
|---|---|---|---|---|
| 27 to 24 | WUPCNT6 [3:0] | 0 | R/W | Wake Up counter for VINn (WUPCNTn: n=0 to 7)<br><br>[Read Access]<br>Wake-up counter is read out.<br>[Write Access]<br>4'h0 : The counter is cleared as 0.<br>4'hf : Hold the counter value.<br>others : Wrtting other values are prohibitted.<br>The counter increments when wake-up signal from VIN{n} asserts. The counter decrements by all three following conditions:<br>(1) The counter is equall to or bigger than 1.<br>(2) Processing of frame, which wait_wup in display list header is 1, is started.<br>(3) VI6_DL_VIN_WUPEN0.WE_Vn is 1<br>When this counter is 15 and wake-up signal from VINn asserts, VI6_WUP0_IRQ_STA.VWERn asserts.<br>These bits doen't decrement by VSPX start until VI6_WUP0_IRQ_ST.VWERn is cleared as 0 after VI6_WUP0_IRQ_STA.VWERn asserts except writting 4'h0 to these bits.<br>Therefore, set 0 to VI6_WUP0_IRQ_STA.VWERn for releasing the lock of WUPCNTn [3:0] before activating VSPX.<br>Set 0 to also these bits before VINn starts working with VSPX. Set H'F to the other bits to hold the value.<br>Ex. Set 32'hFFFF F0F0 to initialze VIN0 and VIN2 wakeup counter. The value of the wake-up counters of VIN1, VIN3 to VIN7 are retained. |
| 23 to 20 | WUPCNT5 [3:0] | 0 | R/W | |
| 19 to 16 | WUPCNT4 [3:0] | 0 | R/W | |
| 15 to 12 | WUPCNT3 [3:0] | 0 | R/W | |
| 11 to 8 | WUPCNT2 [3:0] | 0 | R/W | |
| 7 to 4 | WUPCNT1 [3:0] | 0 | R/W | |
| 3 to 0 | WUPCNT0 [3:0] | 0 | R/W | |

### 60.2.5.8 Display List VIN Wake Up Counter 1 Register (VI6_DL_VIN_WUPCNT1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUPCNT15 [3:0] | | | | WUPCNT14 [3:0] | | | | WUPCNT13 [3:0] | | | | WUPCNT12 [3:0] | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUPCNT11 [3:0] | | | | WUPCNT10 [3:0] | | | | WUPCNT9 [3:0] | | | | WUPCNT8 [3:0] | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | WUPCNT15 [3:0] | 0 | R/W | |
| 27 to 24 | WUPCNT14 [3:0] | 0 | R/W | |

| | | | | |
|---|---|---|---|---|
| 23 to 20 | WUPCNT13 [3:0] | 0 | R/W | Wake Up counter for VINn (WUPCNTn: n=8 to 15)<br><br>[Read Access]<br>Wake-up counter is read out.<br>[Write Access]<br>4'h0 : The counter is cleared as 0.<br>4'hf : Hold the counter value.<br>others : Wrtting other values are prohibitted.<br>Refer to description of VI6_DL_VIN_WUPCNT0 |
| 19 to 16 | WUPCNT12 [3:0] | 0 | R/W | |
| 15 to 12 | WUPCNT11 [3:0] | 0 | R/W | |
| 11 to 8 | WUPCNT10 [3:0] | 0 | R/W | |
| 7 to 4 | WUPCNT9 [3:0] | 0 | R/W | |
| 3 to 0 | WUPCNT8 [3:0] | 0 | R/W | |

### 60.2.5.9 Display List VIN Wake Up Enable Register (VI6_DL_VIN_WUPEN0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WE15 | WE14 | WE13 | WE12 | WE11 | WE10 | WE9 | WE8 | WE7 | WE6 | WE5 | WE4 | WE3 | WE2 | WE1 | WE0 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | WE15 | 0 | R/W | Enable waiting wake up from VINn<br>(WEn: n= 0 to 15)<br>When this bit is set to 1, the VSPX waits for the wake-up counter which indicates the number of outstanding to reach 1 or higher before starting processing.<br>Refer to description of VI6_DL_VIN_WUPCNT0. |
| 14 | WE14 | 0 | R/W | |
| 13 | WE13 | 0 | R/W | |
| 12 | WE12 | 0 | R/W | |
| 11 | WE11 | 0 | R/W | |
| 10 | WE10 | 0 | R/W | |
| 9 | WE9 | 0 | R/W | |
| 8 | WE8 | 0 | R/W | |
| 7 | WE7 | 0 | R/W | |
| 6 | WE6 | 0 | R/W | |
| 5 | WE5 | 0 | R/W | |
| 4 | WE4 | 0 | R/W | |
| 3 | WE3 | 0 | R/W | |

| | | | | |
|---|---|---|---|---|
| 2 | WE2 | 0 | R/W | |
| 1 | WE1 | 0 | R/W | |
| 0 | WE0 | 0 | R/W | |

### 60.2.6 RPF Control Registers

#### 60.2.6.1 RPFn Basic Read Size Registers (VI6_RPFn_SRC_BSIZE)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | BHSIZE [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | BVSIZE [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29,<br>15 to 13 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 16 | BHSIZE<br>[12:0] | 0 | R/W | Horizontal Size of RPF Basic Read Area<br>These bits specify the horizontal size of the basic area to be read from the external RAM by the RPFn. When the input format is YCbCr4:2:2 or YCbCr4:2:0, specify the size in 2-pixel units. A value from 1 to 8190 can be specified.<br>Specify a value equal to or smaller than the extended read size (VI6_RPFn_SRC_ESIZE.EHSIZE). |
| 12 to 0 | BVSIZE<br>[12:0] | All 0 | R/W | Vertical Size of RPF Basic Read Area<br>These bits specify the vertical size of the basic area to be read from the external RAM by the RPFn. When the input format is YCbCr4:2:0, specify the size in 2-pixel units. A value from 1 to 8190 can be specified.<br>Specify a value equal to or smaller than the extended read size (VI6_RPFn_SRC_ESIZE.EVSIZE). |

Figure 60.7 shows the relationship between the basic read size and extended read size. The RPF reads data from the source memory area specified by the basic read size. The RPF repeats reading the basic read area in the horizontal and vertical directions up to the extended read size and sends the read data to the processing modules in the VSP2.

For basic read size reading, the reading start address, called the RPFn source image storing address, should be specified in VI6_RPFn_SRCM_ADDR_*. In the memory area where the basic read area image is stored, the distance

(number of bytes) between addresses for lines n and n + 1 of two-dimensional image data, called the memory stride, should be specified in VI6_RPFn_SRCM_PSTRIDE.

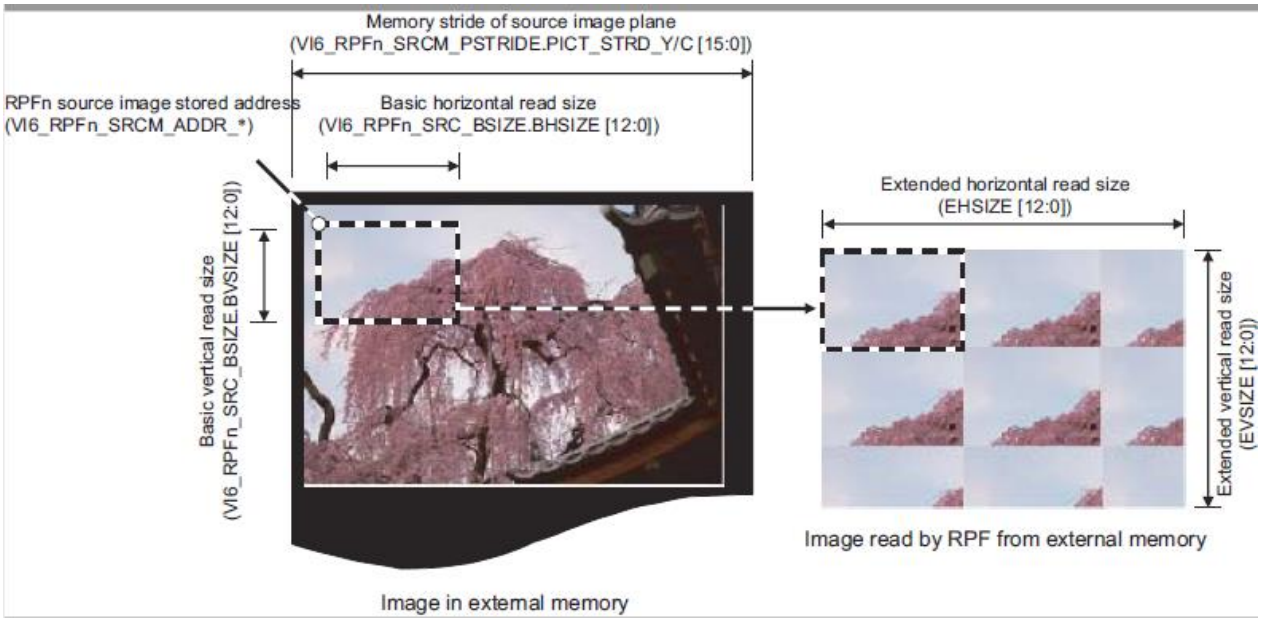



**Figure 60.7 Relationship between Basic Read Size and Extended Read Size**

Refer also to the following sections.

- Section 60.2.6.2, RPFn Extended Read Size Registers (VI6_RPFn_SRC_ESIZE)
- Section 60.2.6.13, RPFn Source Picture Memory Stride Setting Registers (VI6_RPFn_SRCM_PSTRIDE)
- Section 60.2.6.15, RPFn Source Y/RGB Address Registers (VI6_RPFn_SRCM_ADDR_Y)
- Section 60.2.6.16, RPFn Source Chroma Address Registers 0 (VI6_RPFn_SRCM_ADDR_C0)
- Section 60.2.6.17, RPFn Source Chroma Address Registers 1 (VI6_RPFn_SRCM_ADDR_C1)

### 60.2.6.2 RPFn Extended Read Size Registers (VI6_RPFn_SRC_ESIZE)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | EHSIZE [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | EVSIZE [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29, | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always |

| | | | | |
|---|---|---|---|---|
| 15 to 13 | | | | be 0. |
| 28 to 16 | EHSIZE [12:0] | All 0 | R/W | RPF Extended Horizontal Read Size<br>These bits specify the horizontal size of the extended read area to which the RPFn reads data from the external RAM. As shown in Figure 60.7<br>, the basic read area image is repeatedly placed in the extended read area; in the EHSIZE bits, specify a value not smaller than the horizontal size of the basic read area. When the input format is YCbCr4:2:2 or YCbCr4:2:0, specify the size in 2-pixel units (an even value).<br>A value from 1 to 8190 can be specified. |
| 12 to 0 | EVSIZE [12:0] | All 0 | R/W | RPF Extended Vertical Read Size<br>These bits specify the vertical size of the extended read area to which the RPFn reads data from the external RAM. As shown in Figure 60.7, the basic read area image is repeatedly placed in the extended read area; in the EVSIZE bits, specify a value not smaller than the vertical size of the basic read area. When the input format is YCbCr4:2:0, specify the size in 2-pixel units (an even value).<br>A value from 1 to 8190 can be specified. |

VI6_RPFn_SRC_ESIZE specifies the extended size for RPFn. The extended horizontal and vertical sizes should be equal to or greater than the basic sizes specified in VI6_RPFn_SRC_BSIZE. The RPF internal data processing described later and image processing described in section 60.2.8 and later sections are all applied to the image in the extended read size shown on the right side in Figure 60.7.

### 60.2.6.3 RPFn Input Format Registers (VI6_RPFn_INFMT)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | VIR | – | – | – | – | – | – | – | – | – | – | – | CIPM |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPYCS | SPUVS | CEXT [1:0] | | RDTM [2:0] | | | CSC | – | RDFMT [6:0] | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29,<br>27 to 17, | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 7 | | | | |
| 28 | VIR | 0 | R/W | Virtual Input Enable<br>Enables or disables the virtual input function of the RPFn. The image to be processed by the RPFn is usually read from the external memory by the MAU. Instead of this input, the virtual input function generates a single-color image within the RPFn and sends it to the modules in VSP2. When the virtual input function is enabled, the fixed value specified in VI6_RPFn_VRTCOL_SET is used as the input to the RPFn.<br>While the virtual input function is enabled, data is not read from the external memory; that is, the α plane is not read and the IROP calculation thus cannot be executed. In this case, set VI6_RPFn_ALPH_SEL.ASEL to 4. Neither the color space conversion through CSC nor the color keying described in section 60.2.6.11 can be used.<br>0: RPFn uses general input.<br>1: RPFn uses virtual input. |
| 16 | CIPM | 0 | R/W | Horizontal Chrominance Interpolation Method Setting<br>Image data is processed in the YCbCr444 format inside VSP2 in case of YCbCr color space. When the chrominance format of the input image is YCbCr422 or YCbCr420, data is up-sampled as shown in Figure 60.8 for internal processing. This bit specifies the method of up-sampling for this purpose.<br>0: The nearest-neighbor method is used for horizontal chrominance interpolation.<br>1: The bilinear method is used for horizontal chrominance interpolation. |
| 15 | SPYCS | 0 | R/W | RPF Input Mode Setting 1<br>When the input format is YUY2, set this bit to 1 and set the RDFMT bits to 71 (0x47).<br>When the input format is YVYU, set this bit and the SPUVS bit to 1 and set the RDFMT bits to 71 (0x47).<br>In other cases, set this bit to 0. |
| 14 | SPUVS | 0 | R/W | RPF Input Mode Setting 2<br>When the input format is NV61, set this bit to 1 and set the RDFMT bits to 65 (0x41).<br>When the input format is NV21, set this bit to 1 and set the RDFMT bits to 66 (0x42).<br>When the input format is YVYU, set this bit and the SPYCS bit to 1 and set the RDFMT bits to 71 (0x47).<br>In other cases, set this bit to 0. |
| 13, 12 | CEXT [1:0] | All 0 | R/W | Lower-Bit Color Data Extension Method Setting<br>When an RGB input format where each color component is expressed in less than 8 bits are selected from Table 60.10 through the RDFMT bits, VSP2 internally extends each color component to 8 bits before using the data. These bits select this extension method.<br>b'00: Lower-order bits of color data are extended with 0.<br>b'01: Upper-order bits of color data are copied to the lower -order bits.<br>b'10: Lower-order bits of color data are extended with 0. The maximum value is limited to 0xFF. |

| | | | | |
|---|---|---|---|---|
| | | | | b'11: Setting prohibited |
| 11 to 9 | RDTM [2:0] | All 0 | R/W | CSC Conversion Expression Setting<br>These bits select the expression used for color space conversion. The conversion direction is RGB → YCbCr when RGB is selected through the RDFMT bits; the direction is YCbCr → RGB when YCbCr is selected.<br>0: BT.601 YCbCr [16,235/240] ↔ RGB [0,255]<br>1: BT.601 YCbCr [0,255] ↔ RGB [0,255]<br>2: BT.709 YCbCr [16,235/240] ↔ RGB [0,255]<br>3: BT.709 YCbCr [16,235/240] ↔ RGB [16,235]<br>4 to 7: Setting prohibited |
| 8 | CSC | 0 | R/W | Color Space Conversion Enable<br>Enables or disables color space conversion between YCbCr and RGB to be executed in RPFn. The characteristics of color space conversion are determined by the RDTM bit setting. (*1)<br>When using the virtual input (VIR = 1), specify 0.<br>0: Color space conversion is disabled.<br>1: Color space conversion is enabled. |
| 6 to 0 | RDFMT [6:0] | All 0 | R/W | RPF Input Image Format Setting<br>These bits select the format of the image input from the external RAM to the RPFn. Select a value corresponding to the desired format from those shown in Table 60.10, Table 60.11, and Table 60.12.<br>When the virtual input function is used (VIR = 1), the color information for the virtual input should be specified in VI6_RPFn_VRTCOL_SET. If this color information is in the RGB format, set the RDFMT bits to 19. If the color information is in the YCbCr format, set these bits to 64.<br>Set 0x00 (RGB332) to this bit when reading RAW8 1byte for VSPX.<br>Set 0x06 (RGB565) to this bit when reading RAW8 2byte, RAW10, RAW12, RAW14, RAW16 for VSPX.<br>Set 0x13 (ARGB8888) to this bit when reading RAW20, RAW24 for VSPX.<br>[Note 1] Number of input pixels<br>When YCbCr4:2:2 is selected through the RDFMT bits, the horizontal size of the input image should be specified in 2-pixel units. When YCbCr4:2:0 is selected, the vertical and horizontal sizes should be specified in 2-pixel units. Observe these restrictions when specifying the image size in VI6_RPFn_SRC_BSIZE and VI6_RPFn_SRC_ESIZE.<br>[Note 2] OSD-CLUT Setting<br>When the RDFMT bits are set to 0x3F, RGB color data should be stored in the OSD-CLUT. When these bits are set to 0x7f, YCbCr color data should be stored in the OSD-CLUT. Note that while the target WPF is operating, the RPF is also operating and the OSD-CLUT cannot be read or written to. |

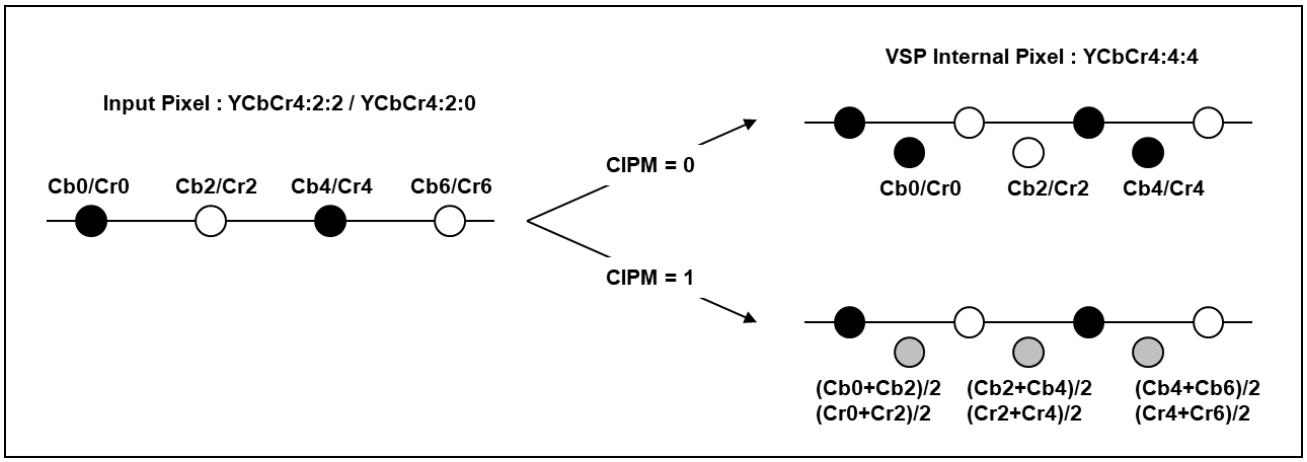



**Figure 60.8 Chrominance Interpolation Methods Selectable through CIPM Setting**

(*[1]) Note on color space settings

The color space for the image output from the RPF to VSP2 internal modules is determined by the combination of the color space for the image input to the RPF, which is selected through the VI6_RPFn_INFMT.RDFMT setting, and the enabled or disabled state of the color space conversion function, which is selected through the VI6_RPFn_INFMT.CSC setting (Table 60.9). For example, when the image input to the RPF is in the YCbCr format, the RPF outputs data to VSP2 internal modules in the YCbCr format if color space conversion is disabled through the CSC bit, and the RPF outputs data in the RGB format if color space conversion is enabled. When the image input to the RPF is in the RGB format, the relationship between the output format and the color space conversion setting is the opposite of the YCbCr case. For some VSP2 internal modules, the YCbCr format is recommended for image processing because of the characteristics of the processing, or the same color space needs to be specified between multiple RPF outputs. In these cases, set VI6_RPFn_INFMT.RDFMT and VI6_RPFn_INFMT.CSC appropriately so that the RPFs can output the required color space according to the color space conditions described above.

**Table 60.9 RPFn Input Color Space and Output Color Space**

| RPFn Input Color Space (VI6_RPFn_INFMT. RDFMT) | | Color Space Conversion Setting (VI6_RPFn_INFMT.CSC) | | RPFn Output Color Space |
|---|---|---|---|---|
| RGB | (0x00 to 0x3f)*1 | Disabled | (0) *1 | RGB |
| | | Enabled | (1) *1 | YCbCr |
| YCbCr | (0x40 to 0x7f)*1 | Disabled | (0) *1 | YCbCr |
| | | Enabled | (1) *1 | RGB |

*1 Value specified in the register

A color space conversion function equivalent to that in the RPFn is also provided by the WPF. As shown in Table 60.9, the color space (YCbCr or RGB) output from the RPF becomes the input format for the WPF. Here, the color space of the output image obtained by the color space conversion function of the WPF must match the color space of the format specified through VI6_WPF0_OUTFMT.WRFMT.

Figure 60.9 shows the relationship between the input/output format and color space. The input color space for the RPF is determined when the input image format for the RPF is specified through the RDFMT bits. The color space for the image output from the RPF to subsequent VSP2 internal modules depends on the combination of the RPF input format and CSC (color space conversion function) enabled or disabled state in the RPF as shown in Table 60.9 and Figure 60.9. The user should first determine whether the image processing in the VSP2 is done in YCbCr or RGB, and then specify the RPF input format and CSC enabled or disabled state to obtain the desired color space. The color space of the RPF output image is also that of the WPF input image; the color space of the data output from the WPF to the outside of VSP2 depends on the combination of the WPF input color space and the enabled or disabled state of the CSC implemented in the WPF as shown in Figure 60.9. The color space of the WPF output image must match that of the WPF output format (determined by VI6_WPF0_OUTFMT.WRFMT). For example, in the flow shown in Figure 60.9, YCbCr should not be specified as the WPF output format regardless of the fact that the color space of the WPF output image is in RGB format.

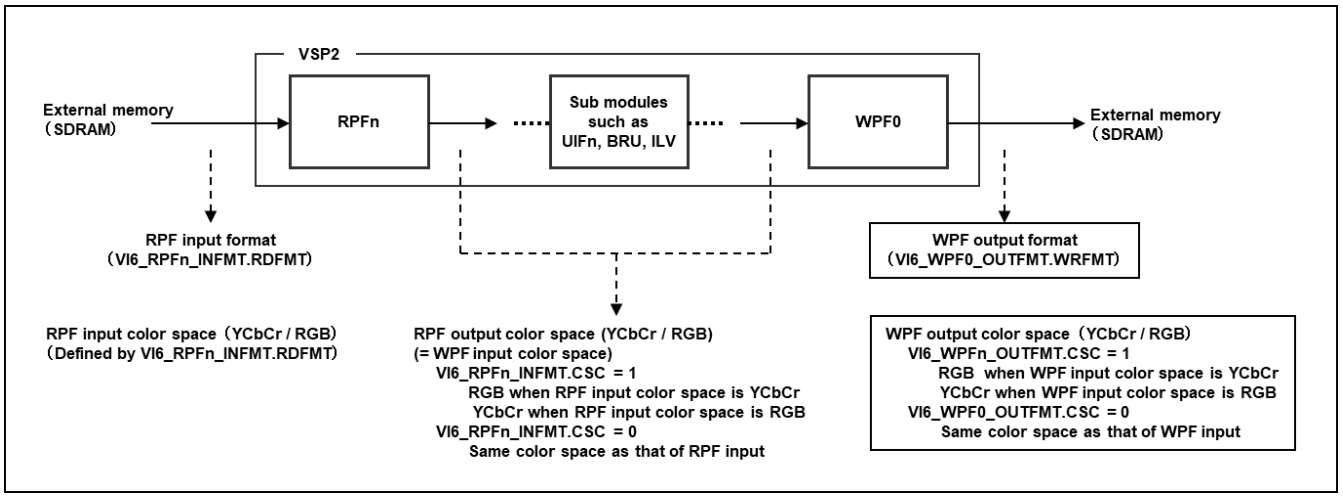



**Figure 60.9 Relationship between Input/output Format and Color Space**

## Table 60.10 Packed Formats for RPF Input

| RDFMT [6:0] | Bit per pixel | Phase | upper row – address / bottom row – bit field | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | n | | | | | | | | n+1 | | | | | | | | n+2 | | | | | | | | n+3 | | | | | | | |
| | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x00 | 8 | – | R0 | R0 | R0 | G0 | G0 | G0 | B0 | B0 | R1 | R1 | R1 | G1 | G1 | G1 | B1 | B1 | R2 | R2 | R2 | G2 | G2 | G2 | B2 | B2 | R3 | R3 | R3 | G3 | G3 | G3 | B3 | B3 |
| 0x01 | 12 | – | | | | | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | | | | | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 |
| 0x02 | | – | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | | | | | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | | | | |
| 0x03 | – | – | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | |
| 0x04 | 15 | – | | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 |
| 0x05 | | | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | |
| 0x06 | 16 | – | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 |
| 0x07 | 18 | – | A | A | A | A | A | A | A | A | | | | | | | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x08 | | | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | | | | | | | A | A | A | A | A | A | A | A |
| 0x09 | | | | | | | | | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | A | A | A | A | A | A | A | A |
| 0x0A | | – | A | A | A | A | A | A | A | A | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | | | | | | |
| 0x0B | | | A | A | A | A | A | A | A | A | | | R0 | R0 | R0 | R0 | R0 | R0 | | | G0 | G0 | G0 | G0 | G0 | G0 | | | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x0C | | | | | R0 | R0 | R0 | R0 | R0 | R0 | | | G0 | G0 | G0 | G0 | G0 | G0 | | | B0 | B0 | B0 | B0 | B0 | B0 | A | A | A | A | A | A | A | A |
| 0x0D | | | A | A | A | A | A | A | A | A | R0 | R0 | R0 | R0 | R0 | R0 | | | G0 | G0 | G0 | G0 | G0 | G0 | | | B0 | B0 | B0 | B0 | B0 | B0 | | |
| 0x0E | | | R0 | R0 | R0 | R0 | R0 | R0 | | | G0 | G0 | G0 | G0 | G0 | G0 | | | B0 | B0 | B0 | B0 | B0 | B0 | | | A | A | A | A | A | A | A | A |
| 0x0F | | 0 | | | | | | | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | | | | | | | R1 | R1 |
| | | 1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | B1 | | | | | | | R2 | R2 | R2 | R2 | R2 | R2 | G2 | G2 | G2 | G2 |
| | | 2 | G2 | G2 | B2 | B2 | B2 | B2 | B2 | B2 | | | | | | | R3 | R3 | R3 | R3 | R3 | R3 | G3 | G3 | G3 | G3 | G3 | G3 | B3 | B3 | B3 | B3 | B3 | B3 |
| 0x10 | | 0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | | | | | | | R1 | R1 | R1 | R1 | R1 | R1 | G1 | G1 |
| | | 1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | B1 | | | | | | | R2 | R2 | R2 | R2 | R2 | R2 | G2 | G2 | G2 | G2 | G2 | G2 | B2 | B2 | B2 | B2 |
| | | 2 | B2 | B2 | | | | | | | R3 | R3 | R3 | R3 | R3 | R3 | G3 | G3 | G3 | G3 | G3 | G3 | B3 | B3 | B3 | B3 | B3 | B3 | | | | | | |
| 0x11 | | 0 | | | R0 | R0 | R0 | R0 | R0 | R0 | | | G0 | G0 | G0 | G0 | G0 | G0 | | | B0 | B0 | B0 | B0 | B0 | B0 | | | R1 | R1 | R1 | R1 | R1 | R1 |
| | | 1 | | | G1 | G1 | G1 | G1 | G1 | G1 | | | B1 | B1 | B1 | B1 | B1 | B1 | | | R2 | R2 | R2 | R2 | R2 | R2 | | | G2 | G2 | G2 | G2 | G2 | G2 |
| | | 2 | | | B2 | B2 | B2 | B2 | B2 | B2 | | | R3 | R3 | R3 | R3 | R3 | R3 | | | G3 | G3 | G3 | G3 | G3 | G3 | | | B3 | B3 | B3 | B3 | B3 | B3 |
| 0x12 | | 0 | R0 | R0 | R0 | R0 | R0 | R0 | | | G0 | G0 | G0 | G0 | G0 | G0 | | | B0 | B0 | B0 | B0 | B0 | B0 | | | R1 | R1 | R1 | R1 | R1 | R1 | | |
| | | 1 | G1 | G1 | G1 | G1 | G1 | G1 | | | B1 | B1 | B1 | B1 | B1 | B1 | | | R2 | R2 | R2 | R2 | R2 | R2 | | | G2 | G2 | G2 | G2 | G2 | G2 | | |
| | | 2 | B2 | B2 | B2 | B2 | B2 | B2 | | | R3 | R3 | R3 | R3 | R3 | R3 | | | G3 | G3 | G3 | G3 | G3 | G3 | | | B3 | B3 | B3 | B3 | B3 | B3 | | |
| 0x13 | 24 | – | A | A | A | A | A | A | A | A | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x14 | | | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | A | A | A | A | A | A | A | A |
| 0x15 | | 0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | R1 | R1 | R1 | R1 | R1 | R1 | R1 | R1 |
| | | 1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | B1 | B1 | B1 | R2 | R2 | R2 | R2 | R2 | R2 | R2 | R2 | G2 | G2 | G2 | G2 | G2 | G2 | G2 | G2 |
| | | 2 | B2 | B2 | B2 | B2 | B2 | B2 | B2 | B2 | R3 | R3 | R3 | R3 | R3 | R3 | R3 | R3 | G3 | G3 | G3 | G3 | G3 | G3 | G3 | G3 | B3 | B3 | B3 | B3 | B3 | B3 | B3 | B3 |
| 0x16 | 18 | – | | | | | | | | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | | | | | | | | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x17 | | | | | | | | | | | | | | | | | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x18 | 24 | 0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | B1 | B1 | B1 | B1 | B1 | B1 | B1 | B1 |
| | | 1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 | R1 | R1 | R1 | R1 | B2 | B2 | B2 | B2 | B2 | B2 | B2 | B2 | G2 | G2 | G2 | G2 | G2 | G2 | G2 | G2 |
| | | 2 | R2 | R2 | R2 | R2 | R2 | R2 | R2 | R2 | B3 | B3 | B3 | B3 | B3 | B3 | B3 | B3 | G3 | G3 | G3 | G3 | G3 | G3 | G3 | G3 | R3 | R3 | R3 | R3 | R3 | R3 | R3 | R3 |
| 0x19 | 12 | – | A | A | A | A | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | A | A | A | A | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 |
| 0x1A | | – | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | A | A | A | A | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | A | A | A | A |
| 0x1B | 15 | – | A | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | A | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 |
| 0x1C | | – | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | A | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | A |
| 0x1D | 12 | – | A | A | A | A | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | A | A | A | A | B1 | B1 | B1 | B1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 |
| 0x1E | | – | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | A | A | A | A | B1 | B1 | B1 | B1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 | A | A | A | A |
| 0x1F | 15 | – | A | B0 | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | R0 | A | B1 | B1 | B1 | B1 | B1 | G1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 | R1 |
| 0x20 | | – | B0 | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | R0 | A | B1 | B1 | B1 | B1 | B1 | G1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 | R1 | A |
| 0x21 | 18 | 0 | | | B0 | B0 | B0 | B0 | B0 | B0 | | | G0 | G0 | G0 | G0 | G0 | G0 | | | R0 | R0 | R0 | R0 | R0 | R0 | | | B1 | B1 | B1 | B1 | B1 | B1 |
| | | 1 | | | G1 | G1 | G1 | G1 | G1 | G1 | | | R1 | R1 | R1 | R1 | R1 | R1 | | | B2 | B2 | B2 | B2 | B2 | B2 | | | G2 | G2 | G2 | G2 | G2 | G2 |
| | | 2 | | | R2 | R2 | R2 | R2 | R2 | R2 | | | B3 | B3 | B3 | B3 | B3 | B3 | | | G3 | G3 | G3 | G3 | G3 | G3 | | | R3 | R3 | R3 | R3 | R3 | R3 |
| 0x22 | 24 | – | A | A | A | A | A | A | A | A | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 |
| 0x23 | 16 | – | | | | | | | | | | | | | | | | | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 |
| 0x24 ～ 0x2F | – | – | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | |
| 0x30 | (*1) | (*1) | (*1) | | | | | | | | (*1) | | | | | | | | (*1) | | | | | | | | (*1) | | | | | | | |
| 0x31 ～ 0x3E | – | – | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | |
| 0x3F (*2) | – | – | RGB_CLUT_DATA0 | | | | | | | | RGB_CLUT_DATA1 | | | | | | | | RGB_CLUT_DATA2 | | | | | | | | RGB_CLUT_DATA3 | | | | | | | |

(*1) When the RDFMT [6:0] bits are set to 0x30, refer to Table 60.11 for the additional settings and the packed formats

(*2) When the RDFMT [6:0] bits are set to 0x3F, it is necessary to set up the OSD-CLUT in the RGB color space.

**Table 60.11 Packed formats of RGB10, RGB10A2, A2RGB10**

| Format name | upper row - address / bottom row - bit field | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | n | | | | | | | | n+1 | | | | | | | | n+2 | | | | | | | | n+3 | | | | | | | |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RGB10 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | | |
| RGB10A2 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | A0 | A0 |
| A2RGB10 | A0 | A0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 |

To specify input format shown in Table 60.11, set 0x30 to VI6_RPFn_INFMT.RDFMT [6:0] and set the following registers additionally.

[1] RGB10

VI6_RPFn_EXT_INFMT0 [31:0] = 0x00000003

VI6_RPFn_EXT_INFMT1 [31:0] = 0x000A1400

VI6_RPFn_EXT_INFMT2 [31:0] = 0x0A0A0A00

[2] RGB10A2

VI6_RPFn_EXT_INFMT0 [31:0] = 0x00000003

VI6_RPFn_EXT_INFMT1 [31:0] = 0x000A141E

VI6_RPFn_EXT_INFMT2 [31:0] = 0x0A0A0A02

[3] A2RGB10

VI6_RPFn_EXT_INFMT0 [31:0] = 0x00000003

VI6_RPFn_EXT_INFMT1 [31:0] = 0x020C1600

VI6_RPFn_EXT_INFMT2 [31:0] = 0x0A0A0A02

**Table 60.12 Packed YCbCr Formats for RPF Input**

| RDFMT [6:0] | Reference | |
|---|---|---|
| 0x40 | YCbCr4:4:4 semi-planar | Figure 60.10*6 |

| | | |
|---|---|---|
| 0x41 | YCbCr4:2:2 semi-planar (NV16, NV61*1) *9 | |
| 0x42 | YCbCr4:2:0 semi-planar (NV12, NV21*1) *7 *9 | |
| 0x43 to 0x45 | Reserved | – |
| 0x46 | YCbCr4:4:4 interleaved | Figure 60.11*6 |
| 0x47 | YCbCr4:2:2 interleaved type 0 (UYVY, YUY2*2, YVYU*3) *9 | |
| 0x48 | YCbCr4:2:2 interleaved type 1 *9 | |
| 0x49 | YCbCr4:2:0 interleaved*8 | |
| 0x4a | YCbCr4:4:4 planar | Figure 60.12*6 |
| 0x4b | YCbCr4:2:2 planar (YV16) *9 | |
| 0x4c | YCbCr4:2:0 planar (YV12, I420) *7 *9 | |
| 0x4d to 0x7e | Reserved | – |
| 0x7f | YCBCR_CLUT_DATA*4 *5 | – |

*1 When the input format is NV61 or NV21, set the SPUVS bit to 1.

*2 When the input format is YUY2, set the SPYCS bit to 1.

*3 When the input format is YVYU, set the SPUVS bit to 1 and SPYCS bit to 1.

*4 When the RDFMT [6:0] bits are set to 0x7F, it is necessary to set up the OSD-CLUT in the YCbCr color space.

*5 The Monochrome format can be implemented through the OSD-CLUT.

*6 Figure 60.13 shows the definition of memory address for each pixel in Figure 60.10 to Figure 60.12.

*7 Each line of C plane is read twice, so byte/pixel of YCbCr420 is 2 byte/pixel (same as YCbCr422).

*8 Each line of plane is read twice, so byte/pixel of YCbCr420ITL is 3 byte/pixel (same as YCbCr444ITL).

*9 Only input format YUV422/YUV420 semi-planar, YUV422 interleaved, YUV422/YUV420 planar support 10bpc, 12bpc for Y/Cb/Cr by setting register VI6_RPFn_EXT_INFMT0.

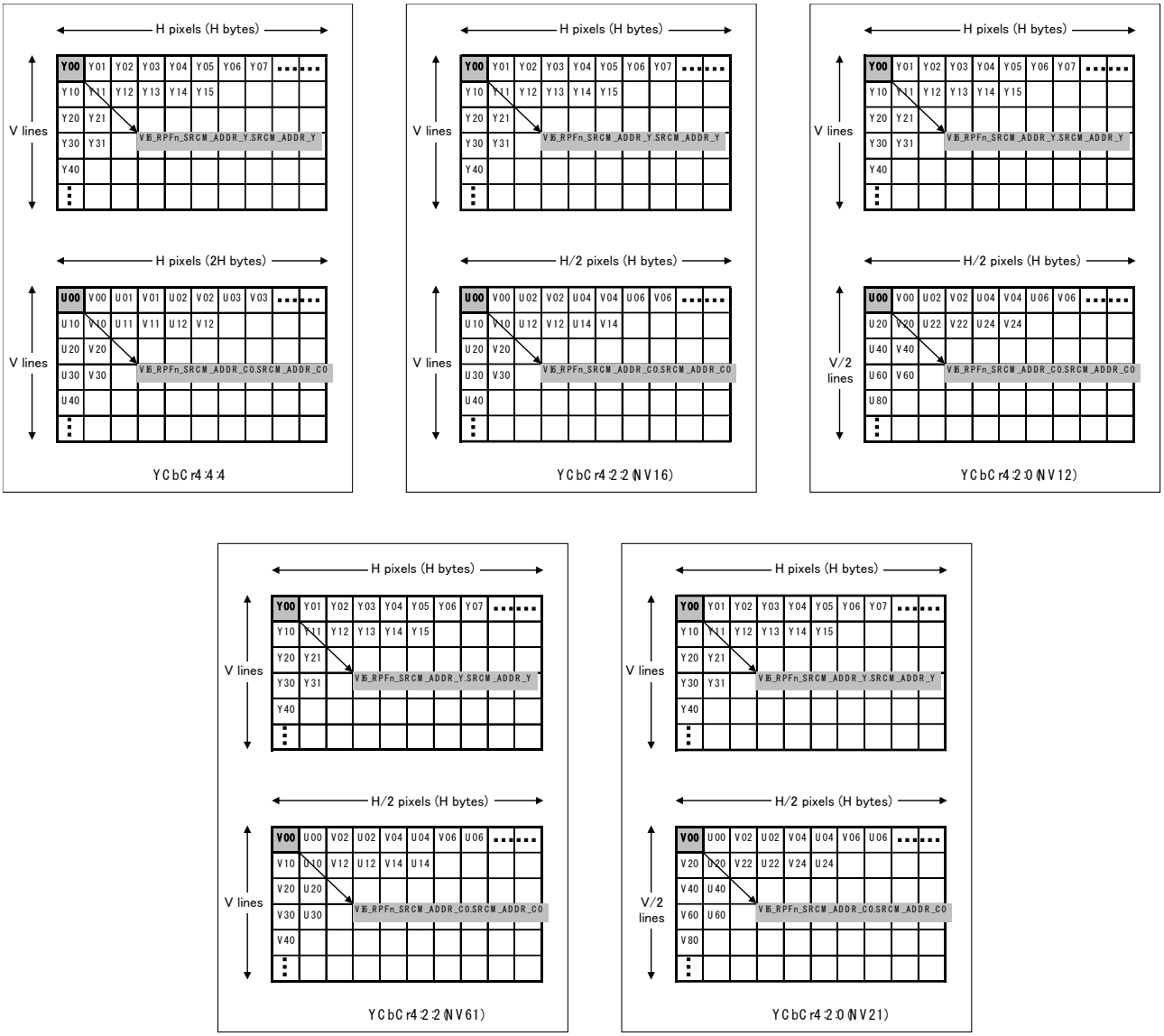



**Figure 60.10 YCbCr Semi-Planar Formats (*1)**

(*1) This figure is for 8bpc. When bits per component of Y/Cb/Cr is 10bpc/12bpc the pixel memory area shall be 2bytes as defined in Figure 60.14.

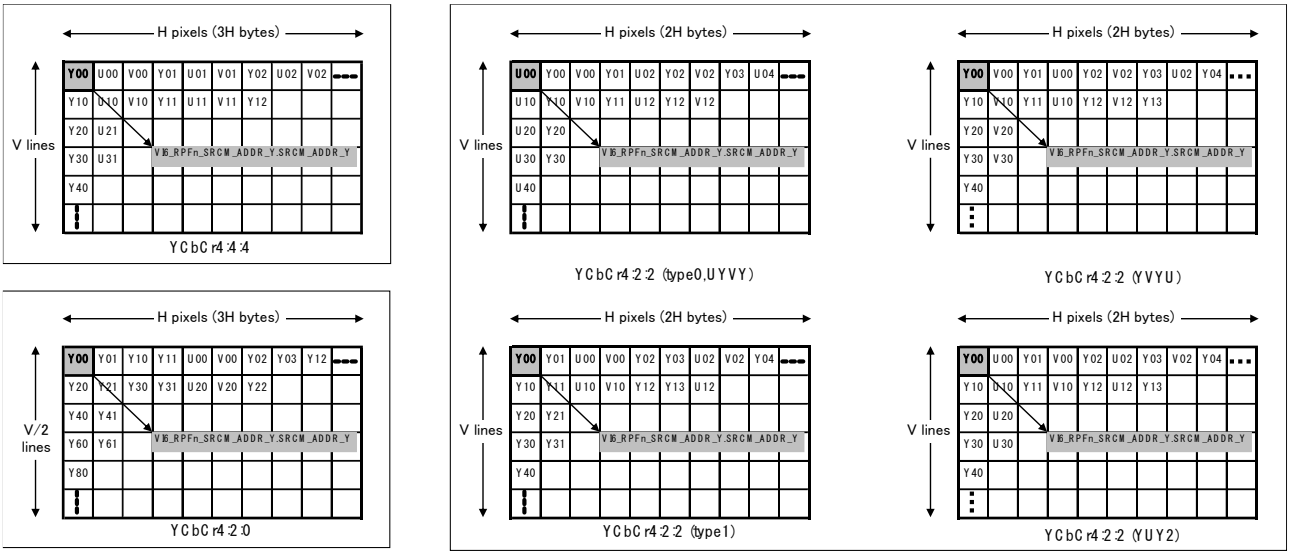



**Figure 60.11 YCbCr Interleaved Formats (*1)**

(*1) This figure is for 8bpc. When bits per component of Y/Cb/Cr is 10bpc/12bpc the pixel memory area shall be 2bytes as defined in Figure 60.14.

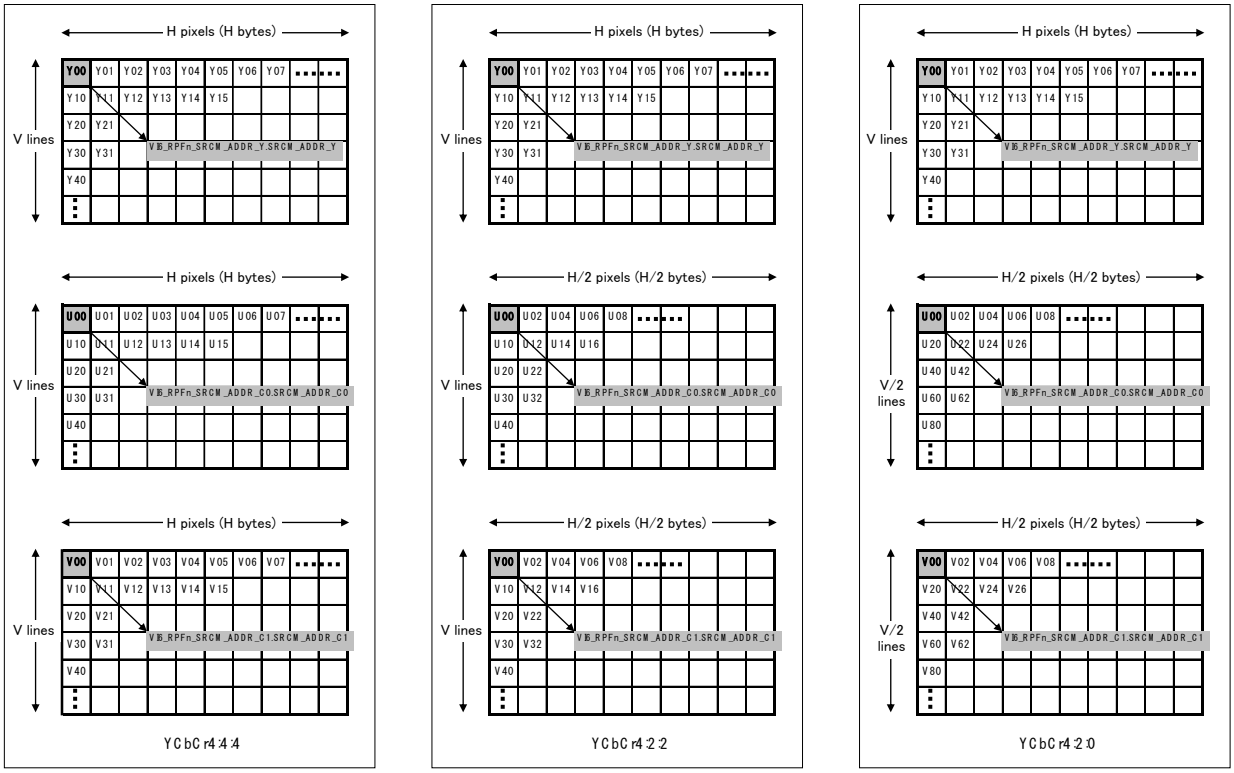



**Figure 60.12 YCbCr Planar Formats(*1)**

(*1) This figure is for 8bpc. When bits per component of Y/Cb/Cr is 10bpc/12bpc the pixel memory area shall be 2bytes as defined in Figure 60.14.

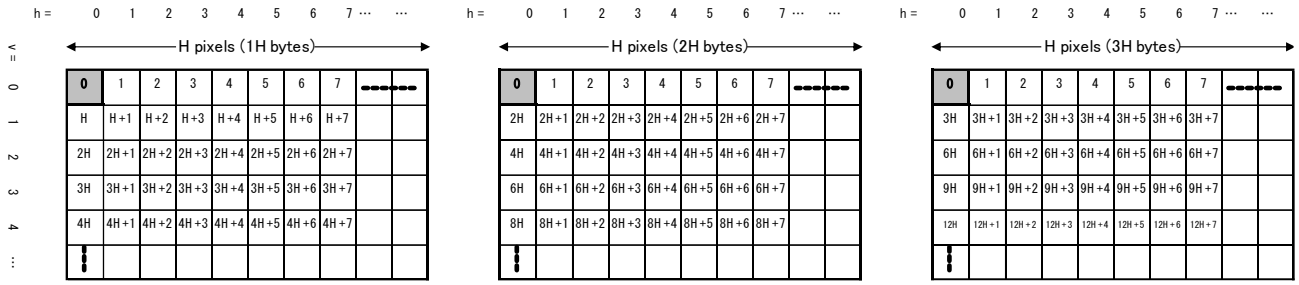



**Figure 60.13 Memory address for the corresponding pixel's position**

Legend of Figure 60.13

- H is horizontal image size in pixel unit.
- 1H, 2H and 3H is corresponding to the case 1 pixel has 1byte, 2bytes, 3bytes of data respectively.

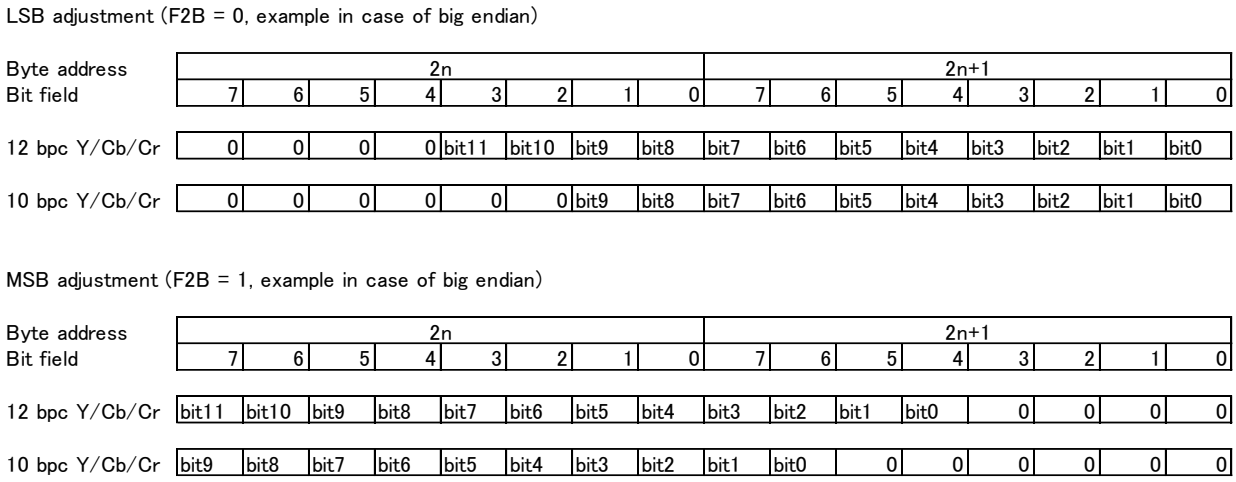



**Figure 60.14 Pixel bit alignment of Y/Cb/Cr 10bpc, 12bpc**

[OSD-CLUT Setting]

To set up the OSD-CLUT, store the values to replace each of R (Cr), G (Y), and B (Cb) components in the OSD-CLUT allocated as shown in Table 60.13. The OSD-CLUT space has 256 32-bit entries, which should be accessed in 32 bits in the same way as for register access. The address for each entry is obtained from the start address of each space (Table 60.13) + entry number ´ 4. For example, the OSD-CLUT address of entry 7 for RPF2 is (offset address) + 0x481C. Figure 60.15 shows the OSD-CLUT format.

**Table 60.13 OSD-CLUT Space for Each RPF**

| RPF | Address Area for CLUT Space |
|---|---|
| RPF0 | (offset address) + 0x4000 to (offset address) + 0x43FF |
| RPF1 | (offset address) + 0x4400 to (offset address) + 0x47FF |

| | |
|---|---|
| RPF2 | (offset address) + 0x4800 to (offset address) + 0x4BFF |

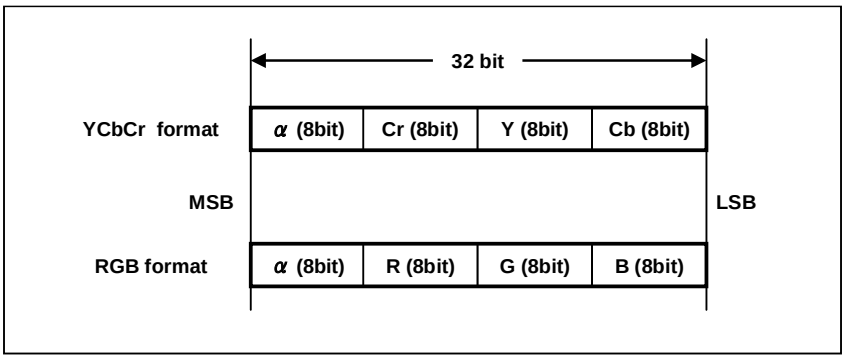



**Figure 60.15 OSD-CLUT Format**

According to the specified 256-entry replacement table (OSD-CLUT), input data is replaced as shown in Figure 60.16. Input data is 8 bits but output data is 32 bits including α and RGB or YCbCr components.

When writing to an entry in the OSD-CLUT, the host CPU writes all components at the same time. Therefore, specify replacement data for all components in the OSD-CLUT at the same time as shown in Figure 60.15.

Note that write access to the OSD-CLUT space is prohibited while the RPF is operating.

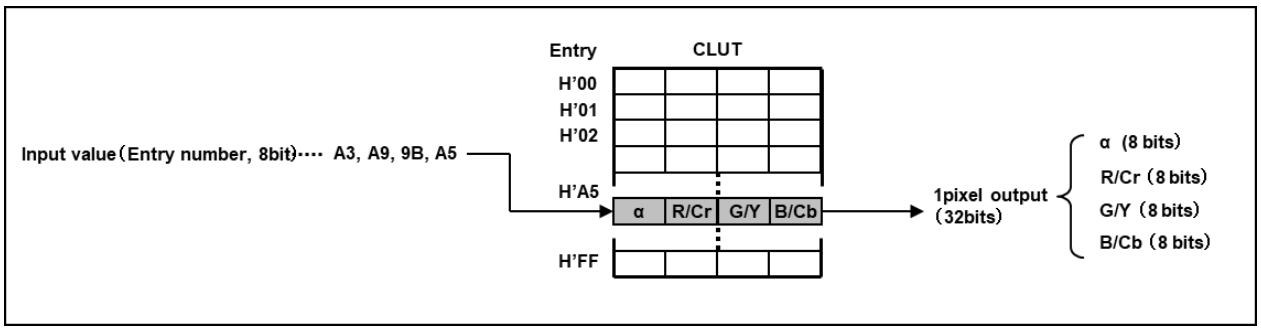



**Figure 60.16 Replacement of Input Data with Pixel Values Specified in OSD-CLUT**

### 60.2.6.4 RPFn Data Swapping Registers (VI6_RPFn_DSWAP)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | C_LLS | C_LWS | C_WDS | C_BTS | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | A_LLS | A_LWS | A_WDS | A_BTS | – | – | – | – | P_LLS | P_LWS | P_WDS | P_BTS |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28, 23 to 12, 7 to 4 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 | C_LLS | 0 | R/W | Extended Chroma Plane Data Swapping in LONG LWORD Units (64-bit)<br>This register is available when input data format is YUV Plannar or Semi-Plannar and input bit-depth is 2bypc (IPBD_Y != 0 or IPBD_C != 0), specifies data swapping of color different planes (CbCr, U, V).<br>0: Data swapping in LONG LWORD (64-bit) units is disabled<br>1: Data swapping in LONG LWORD (64-bit) units is enabled |
| 26 | C_LWS | 0 | R/W | Extended Chroma Plane Data Swapping in long word Units (32-bit)<br>This register is available when input data format is YUV Plannar or Semi-Plannar and input bit-depth is 2bypc (IPBD_Y != 0 or IPBD_C != 0), specifies data swapping of color different planes (CbCr, U, V) .<br>0: Data swapping in long word (32-bit) units is disabled<br>1: Data swapping in long word (32-bit) units is enabled |
| 25 | C_WDS | 0 | R/W | Extended Chroma Plane Data Swapping in word Units (16-bit)<br>This register is available when input data format is YUV Plannar or Semi-Plannar and input bit-depth is 2bypc (IPBD_Y != 0 or IPBD_C != 0), specifies data swapping of color different planes (CbCr, U, V) .<br>0: Data swapping in word (16-bit) units is disabled<br>1: Data swapping in word (16-bit) units is enabled |
| 24 | C_BTS | 0 | R/W | Extended Chroma Plane Data Swapping in byte Units<br>This register is available when input data format is YUV Plannar or Semi-Plannar and input bit-depth is 2bypc (IPBD_Y != 0 or IPBD_C != 0), specifies data swapping of color different planes (CbCr, U, V) .<br>0: Data swapping in byte (8-bit) units is disabled<br>1: Data swapping in byte (8-bit) units is enabled |
| 11 | A_LLS | 0 | R/W | α Plane Data Swapping in LONG LWORD Units<br>The effect of this bit setting is defined in Table 60.14.<br>0: Data swapping in LONG LWORD (64-bit) units is disabled<br>1: Data swapping in LONG LWORD (64-bit) units is enabled |
| 10 | A_LWS | 0 | R/W | α Plane Data Swapping in long word Units<br>The effect of this bit setting is defined in Table 60.14.<br>0: Data swapping in long word (32-bit) units is disabled<br>1: Data swapping in long word (32-bit) units is enabled |
| 9 | A_WDS | 0 | R/W | α Plane Data Swapping in Word Units<br>The effect of this bit setting is defined in Table 60.14.<br>0: Data swapping in word (16-bit) units is disabled<br>1: Data swapping in word (16-bit) units is enabled |
| 8 | A_BTS | 0 | R/W | α Plane Data Swapping in Byte Units<br>The effect of this bit setting is defined in Table 60.14.<br>0: Data swapping in byte (8-bit) units is disabled |

| | | | | |
|---|---|---|---|---|
| | | | | 1: Data swapping in byte (8-bit) units is enabled |
| 3 | P_LLS | 0 | R/W | Picture Plane Data Swapping in LONG LWORD Units<br>The effect of this bit setting is defined in Table 60.14.<br>0: Data swapping in LONG LWORD (64-bit) units is disabled<br>1: Data swapping in LONG LWORD (64-bit) units is enabled<br>This register is available for only Luma plane (Y) in the case input data format is YUV Plannar or Semi-Plannar and input bit-depth is 2bypc (IPBD_Y != 0 or IPBD_C != 0), and available for both Luma plane (Y) and Chroma plane (CbCr/U/V) in other cases. |
| 2 | P_LWS | 0 | R/W | Picture Plane Data Swapping in long word Units<br>The effect of this bit setting is defined in Table 60.14.<br>0: Data swapping in long word (32-bit) units is disabled<br>1: Data swapping in long word (32-bit) units is enabled<br>This register is available for only Luma plane (Y) in the case input data format is YUV Plannar or Semi-Plannar and input bit-depth is 2bypc (IPBD_Y != 0 or IPBD_C != 0), and available for both Luma plane (Y) and Chroma plane (CbCr/U/V) in other cases. |
| 1 | P_WDS | 0 | R/W | Picture Plane Data Swapping in Word Units<br>The effect of this bit setting is defined in Table 60.14.<br>0: Data swapping in word (16-bit) units is disabled<br>1: Data swapping in word (16-bit) units is enabled<br>This register is available for only Luma plane (Y) in the case input data format is YUV Plannar or Semi-Plannar and input bit-depth is 2bypc (IPBD_Y != 0 or IPBD_C != 0), and available for both Luma plane (Y) and Chroma plane (CbCr/U/V) in other cases. |
| 0 | P_BTS | 0 | R/W | Picture Plane Data Swapping in Byte Units<br>The effect of this bit setting is defined in Table 60.14.<br>0: Data swapping in byte (8-bit) units is disabled<br>1: Data swapping in byte (8-bit) units is enabled<br>This register is available for only Luma plane (Y) in the case input data format is YUV Plannar or Semi-Plannar and input bit-depth is 2bypc (IPBD_Y != 0 or IPBD_C != 0), and available for both Luma plane (Y) and Chroma plane (CbCr/U/V) in other cases. |

When the virtual input function of the RPFn is used (VI6_RPFn_INFMT.VIR = 1), this register setting is ignored. Swapping of RPF input data can be specified separately for the α plane and picture plane.

Table 60.14 shows the data order before and after swapping according to the long long word, long word, word, and byte swapping settings.

When data order in memory for each format is the same as Table 60.10 for RGB format and Figure 60.10 to Figure 60.12 for YCbCr format, set 4'b1111 to {*_LLS, *_LWS, *_WDS, *_BTS}. If data order is not the same as the definition, change data order within 16byte unit by these bits as shown in Table 60.14.

**Table 60.14 Changing data order according to swap register**

| Data order in memory | | | | | | | | | | | | | | | | | *_LLS | *_LWS | *_WDS | *_BTS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Byte address | 16n +0 | 16n+ 1 | 16n +2 | 16n +3 | 16n +4 | 16n +5 | 16n +6 | 16n +7 | 16n +8 | 16n +9 | 16n +10 | 16n +11 | 16n +12 | 16n +13 | 16n +14 | 16n +15 | | | | |
| Data | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 1 | 1 | 1 | 1 |
| | 1 | 0 | 3 | 2 | 5 | 4 | 7 | 6 | 9 | 8 | 11 | 10 | 13 | 12 | 15 | 14 | 1 | 1 | 1 | 0 |
| | 2 | 3 | 0 | 1 | 6 | 7 | 4 | 5 | 10 | 11 | 8 | 9 | 14 | 15 | 12 | 13 | 1 | 1 | 0 | 1 |
| | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 11 | 10 | 9 | 8 | 15 | 14 | 13 | 12 | 1 | 1 | 0 | 0 |
| | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 12 | 13 | 14 | 15 | 8 | 9 | 10 | 11 | 1 | 0 | 1 | 1 |

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 5 | 4 | 7 | 6 | 1 | 0 | 3 | 2 | 13 | 12 | 15 | 14 | 9 | 8 | 11 | 10 | 1 | 0 | 1 | 0 |
| | 6 | 7 | 4 | 5 | 2 | 3 | 0 | 1 | 14 | 15 | 12 | 13 | 10 | 11 | 8 | 9 | 1 | 0 | 0 | 1 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 1 | 0 | 0 | 0 |
| | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 1 | 1 |
| | 9 | 8 | 11 | 10 | 13 | 12 | 15 | 14 | 1 | 0 | 3 | 2 | 5 | 4 | 7 | 6 | 0 | 1 | 1 | 0 |
| | 10 | 11 | 8 | 9 | 14 | 15 | 12 | 13 | 2 | 3 | 0 | 1 | 6 | 7 | 4 | 5 | 0 | 1 | 0 | 1 |
| | 11 | 10 | 9 | 8 | 15 | 14 | 13 | 12 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 0 | 1 | 0 | 0 |
| | 12 | 13 | 14 | 15 | 8 | 9 | 10 | 11 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 0 | 0 | 1 | 1 |
| | 13 | 12 | 15 | 14 | 9 | 8 | 11 | 10 | 5 | 4 | 7 | 6 | 1 | 0 | 3 | 2 | 0 | 0 | 1 | 0 |
| | 14 | 15 | 12 | 13 | 10 | 11 | 8 | 9 | 6 | 7 | 4 | 5 | 2 | 3 | 0 | 1 | 0 | 0 | 0 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |

| Data order defined in Table 60.10 for RGB format and Figure 60.10 to Figure 60.12 for YCbCr format | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Byte address | 16n +0 | 16n+ 1 | 16n +2 | 16n +3 | 16n +4 | 16n +5 | 16n +6 | 16n +7 | 16n +8 | 16n +9 | 16n +10 | 16n +11 | 16n +12 | 16n +13 | 16n +14 | 16n +15 |
| Data | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

### 60.2.6.5 RPFn Display Location Registers (VI6_RPFn_LOC)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | HCOORD [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | VCOORD [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29, 15 to 13 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 16 | HCOORD [12:0] | All 0 | R/W | Horizontal Coordinate of Sublayer Display Location on Master Layer<br>These bits specify the left-end location of the sublayer displayed by the RPFn and the subsequent module connected through the DPR. Specify the horizontal coordinate of the location in pixel units with the left-end pixel of the master layer set at coordinate 0. When the RPFn is the master layer, set these bits to 0.<br>If the sublayer extends beyond the master layer according to the HCOORD setting, the extended section is cut off at the right end of the master layer. Even in this case, however, a bus transfer that is unnecessary for output image generation is executed since the whole sublayer data is read from the external memory. Appropriate coordinate setting is required so that the sublayer does not extend beyond the right end of the master layer.<br>A value from 0 to 8189 can be specified. |
| 12 to 0 | VCOORD [12:0] | All 0 | R/W | Vertical Coordinate of Sublayer Display Location on Master Layer<br>These bits specify the top-end location of the sublayer displayed by the RPFn and the subsequent module connected through the |

| | | | | |
|---|---|---|---|---|
| | | | | DPR. Specify the vertical coordinate of the location in pixel units with the top-end pixel of the master layer set at coordinate 0. When the RPFn is the master layer, set these bits to 0.<br>If the sublayer extends beyond the master layer according to the VCOORD setting, the extended section is cut off at the bottom end of the master layer. Even in this case, however, a bus transfer that is unnecessary for output image generation is executed since the whole sublayer data is read from the external memory. Appropriate coordinate setting is required so that the sublayer does not extend beyond the bottom end of the master layer.<br>A value from 0 to 8189 can be specified. |

Figure 60.17 shows an example of RPF1 and RPF2 offsets with respect to master layer RPF0. Although this figure only shows sublayers RPF1 and RPF2, specify offsets for all RPFs other than the master layer in the same way as shown in this example.

Whether an RPFn is the master layer or a sublayer is determined through the selection of the source RPF for WPF0 (the VI6_WPF0_SRCRPF setting). For details, refer to section 60.2.7.1.

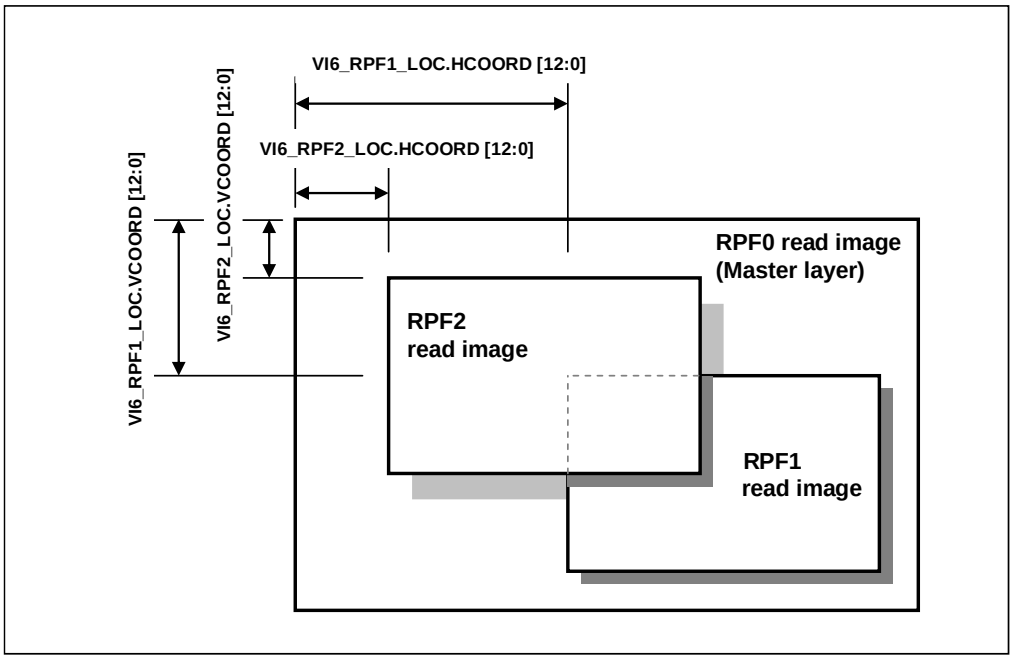



**Figure 60.17 RPF1 and RPF2 Offsets from Master Layer**

### 60.2.6.6 RPFn A Plane Selection Control Registers (VI6_RPFn_ALPH_SEL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | ASEL [2:0] | | | IROP [3:0] | | | | BSEL | – | – | – | AEXT [1:0] | | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ALPHA1 [7:0] | | | | | | | | ALPHA0 [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31,<br>22 to 20,<br>17, 16 | – | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 30 to 28 | ASEL [2:0] | All 0 | R/W | α Format and Processing Method Select<br>These bits select how to handle the α value to be used. The RPF handles two types of α value; 8-bit and 1-bit values. When a 1-bit α value is used, VSP2 assumes that the 1-bpp α value for each pixel is stored in the order from MSB to LSB in each byte (big endian).<br>The α value is used as either transparency information or mask information. Transparency information is included in the α plane read from external memory when the ASEL bits are set to 1 or 3 and in the α value stored in the packed RGB bit field when these bits are set to 0 or 2. The α value as transparency information is sent as the destination value to the IROP as shown in Figure 60.18 and then output to the subsequent modules. The output α value is used, for example, for blending in the BRU.<br>The α value as mask information is used for IROP operation in the RPF. The mask information is included in the α plane read from external RAM when the ASEL bits are set to 0 or 2 and the source value is used in IROP operation (IROP setting other than 0, 5, 10, or 15). This α value is sent as the source value to the IROP as shown in Figure 60.18.<br>Note that the α value selected through the ASEL bits has a lower priority than the VI6_RPFn_CKEY_SET*.AP* value replaced through the color keying function. When the color keying function is used, the α value may be replaced with the VI6_RPFn_CKEY_SET*.AP* value regardless of the ASEL bit setting.<br>When using the virtual input function (VI6_RPFn_INFMT.VIR = 1), specify 4.<br>0: 1, 4, or 8-bit packed α + plane α (IROP ! = 0, 5, 10, 15)<br>The α bit field in 1, 4, or 8-bit packed α is handled as transparency information. Be sure to specify the packed format that includes α through VI6_RPFn_INFMT.RDFMT.<br>When VI6_RPFn_MSKCTRL.MSK_EN is 0 and the IROP bit value is not 0, 5, 10, or 15, the α plane should be read as mask information. Specify the number of α data bits (BSEL) stored in the α plane and the α plane read start address (VI6_RPFn_SRCM_ADDR_AI). When the IROP bits are set to 0, 5, 10, or 15, the α plane is not read.<br>1: 8-bit plane α |

| | | | | |
|---|---|---|---|---|
| | | | | The 8-bit α plane is read from external RAM as transparency information. When the packed RGB format has a bit field for α, the information in the α bit field is discarded. The α plane read start address (VI6_RPFn_SRCM_ADDR_AI) should be specified. The α value goes through the 8-bit transparent α generator shown in Figure 60.18 without change.<br>When VI6_RPFn_MSKCTRL.MSK_EN is 0, IROP operation cannot be executed; set the IROP bits to 0 in this case. When VI6_RPFn_MSKCTRL.MSK_EN is 1. IROP operation can be executed.<br>2: 1-bit packed α + plane α (IROP ! = 0, 5, 10, 15)<br>The 1-bit packed α input is converted by the 8-bit transparent α generator shown in Figure 60.18<br>according to the ALPHA0/1 setting into the 8-bit α value as transparency information. Select the packed input format that includes a 1-bit α field.<br>When VI6_RPFn_MSKCTRL.MSK_EN is 0 and the IROP value is not 0, 5, 10, or 15, the α plane should be read as mask information. Specify the number of α data bits (BSEL) stored in the α plane and the α plane read start address (VI6_RPFn_SRCM_ADDR_AI). When the IROP bits are set to 0, 5, 10, or 15, the α plane is not read.<br>3: 1-bit plane α<br>The 1-bit α plane is read from external RAM and converted by the 8-bit transparent α generator shown in Figure 60.18<br>according to the ALPHA0/1 setting into the 8-bit α value as transparency information. When the packed RGB format has a bit field for α, the information in the α bit field is discarded. The α plane read start address (VI6_RPFn_SRCM_ADDR_AI) should be specified.<br>When VI6_RPFn_MSKCTRL.MSK_EN is 0, IROP operation cannot be executed; set the IROP bits to 0 in this case. When VI6_RPFn_MSKCTRL.MSK_EN is 1. IROP operation can be executed.<br>4: Fixed α<br>The fixed α value (VI6_RPFn_VRTCOL_SET.LAYA value) is output from the RPF. IROP operation cannot be executed; set the IROP bits to 0 in this case.<br>5 to 7: Setting prohibited. |
| 27 to 24 | IROP [3:0] | All 0 | R/W | IROP Operation Setting<br>These bits specify the operator to be executed in the IROP operation unit shown in Figure 60.18. The source (S) for the IROP operation is the pixel data and α data specified in the VI6_RPFn_MSKSET0 or VI6_RPFn_MSKSET1 IROP input value register, which is selected according to the value (0 or 1) generated by the 1-bit mask generator. The destination (D) is the image data (RGB/YCbCr) and 8-bit α data output from the unpack/OSD processor. IROP operation is applied both for the image data and α data between the source and destination data.<br>If these bits are set to the operation that involves the source (S) (IROP setting other than 0, 5, 10, or 15) while VI6_RPFn_MSKCTRL.MSK_EN is 0, the α plane is read from the external RAM to be used for the α value for IROP operation; specify the α plane read start address (VI6_RPFn_SRCM_ADDR_A).<br>When the virtual input function is used (VI6_RPFn_INFMT.VIR = 1), IROP operation is not available; set these bits to b'0000.<br>b'0000: NOP(D) |

| | | | | |
|---|---|---|---|---|
| | | | | b'0001: AND(S & D)<br>b'0010: AND_REVERSE(S & ~D)<br>b'0011: COPY(S)<br>b'0100: AND_INVERTED(~S & D)<br>b'0101: CLEAR(0)<br>b'0110: XOR(S ^ D)<br>b'0111: OR(S \| D)<br>b'1000: NOR(~(S \| D))<br>b'1001: EQUIV(~(S ^ D))<br>b'1010: INVERT(~D)<br>b'1011: OR_REVERSE(S \| ~D)<br>b'1100: COPY_INVERTED(~S)<br>b'1101: OR_INVERTED(~S \| D)<br>b'1110: NAND(~(S & D))<br>b'1111: SET(all 1) |
| 23 | BSEL | 0 | R/W | α Bit Count Conversion Selection for 1-Bit Mask Generator<br>Specifies the number of bits in the α plane to be read as mask information from the external RAM. The α value in mask information is used for the source (S) in IROP. When α plane data is 8 bits, it is converted to one bit through the 1-bit mask generator shown in Figure 60.18.<br>Note that this bit setting is valid when the ASEL bits are set to 0 or 2 and VI6_RPFn_MSKCTRL.MSK_EN is set to 0. In other cases, this bit setting has no effect.<br>0: 8-bit α is converted to 1-bit α through the 1-bit mask generator.<br>When the 8-bit α value input to the RPF is not 0, it is converted to b'1; when the value is 0, it is converted to b'0.<br>1: α value goes through the 1-bit mask generator.<br>The 1-bit α value input to the RPF is output through the 1-bit mask generator without change. |
| 19, 18 | AEXT[1:0] | All 0 | R/W | Lower-Bit α Value Extension Method Set<br>These bits specify the method for extending the input α data to 8 bits through the unpack processing.<br>b'00: The lower-order bits of α value are extended with 0.<br>b'01: The upper-order bits of α value are copied to the lower-order bits.<br>b'10: The lower-order bits of α value are extended with 0. The maximum value is limited to 0xFF.<br>b'11: Setting prohibited |
| 15 to 8 | ALPHA1[7:0] | All 0 | R/W | 8-Bit α Value Output when 1-Bit α Value is 1<br>These bits specify the 8-bit α value to be output when 1-bit α data is input and the α value input to the 8-bit transparent α generator shown in Figure 60.18 is b'1. This setting is valid when the ASEL bits are set to b'010 or b'011.<br>A value from 0 to 255 can be specified. |
| 7 to 0 | ALPHA0 [7:0] | All 0 | R/W | 8-Bit α Value Output when 1-Bit α Value is 0<br>These bits specify the 8-bit α value to be output when 1-bit α data is input and the α value input to the 8-bit transparent α generator shown in Figure 60.18 is b'0. This setting is valid when the ASEL bits are set to b'010 or b'011.<br>A value from 0 to 255 can be specified. |

Figure 60.18 shows the relationship between the α selector, IROP operation unit, color keying unit, and related registers. The IROP operation unit receives two inputs, source and destination. The image data input from the external memory is processed through the unpack processor and 8-bit transparent α generator and then input to the IROP operation unit as destination data. The α plane data input from the external memory is sent to the 8-bit transparent α generator when the ASEL bits are set to 1 or 3, or sent to the 1-bit mask α generator when the ASEL bits are set to 0 or 2. For the pixel data and 8-bit α value on the source side of the IROP operation unit, either the VI6_RPFn_MSKSET0 value or VI6_RPFn_MSKSET1 values will be selected according to the 1-bit α value output by the 1-bit mask α generator.

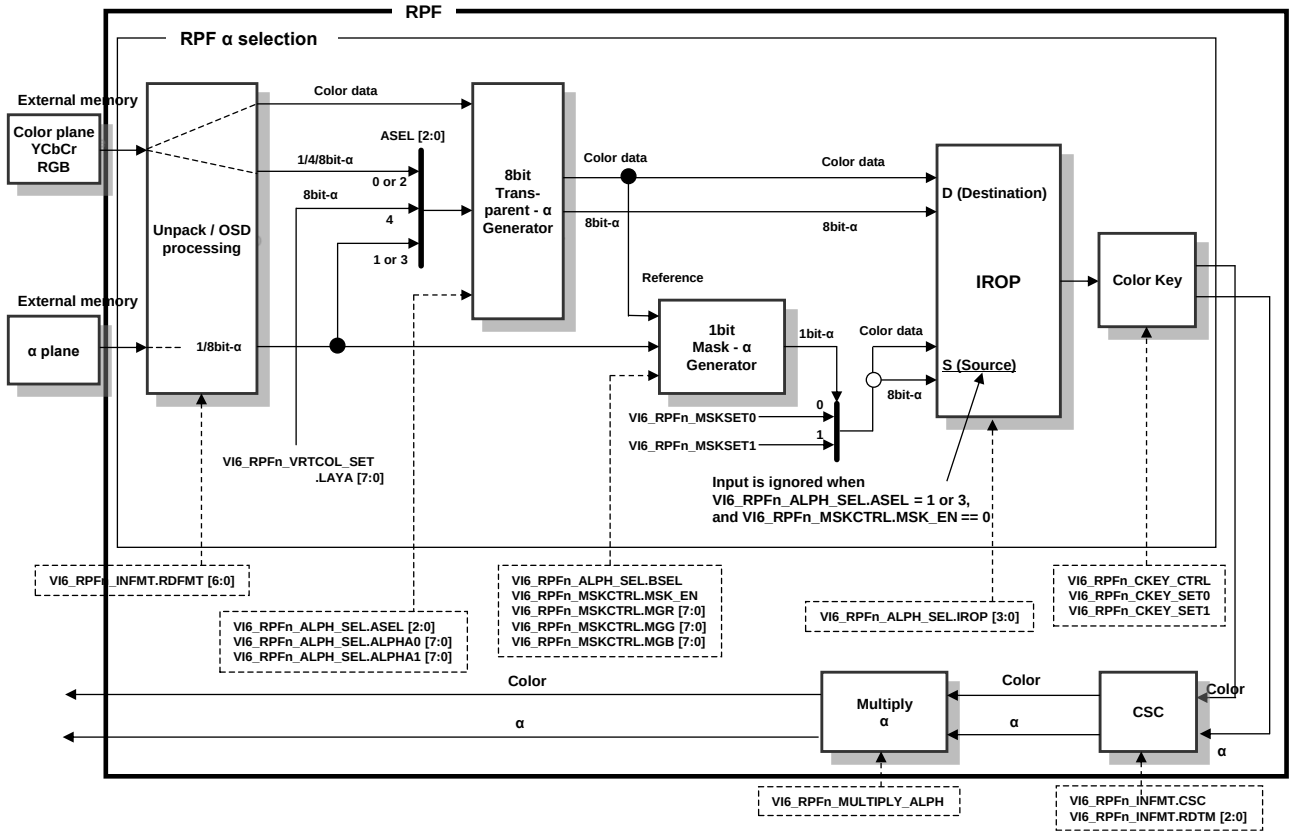



**Figure 60.18 Configuration of alpha Selector and IROP Operation Unit in RPF**

The following describes the function of each block shown in Figure 60.18. Read the following while referring to the figure as necessary.

Unpack/OSD processor:

Unpacks each component and α value of the image data according to the packed format specified in VI6_RPFn_INFMT.RDFMT.

8-bit transparent α generator:

Converts the input α value into 8-bit α when the input α is four bits or one bit.

When VI6_RPFn_ALPH_SEL.ASEL are set to 0 (8-, 4-, or 1-bit packed α), this generator outputs the input α value without change if the α bit field in the packed α data is 8 bits; if the α bit field is less than 8 bits, it is converted to an 8-bit α value by extending the LSB side according to the VI6_RPFn_ALPH_SEL.AEXT setting.

When VI6_RPFn_ALPH_SEL.ASEL are set to 1 (8-bit plane α) or 4 (fixed α), this generator outputs the input 8-bit plane α without change. If a packed α value is included in RGB data, it is discarded.

When VI6_RPFn_ALPH_SEL.ASEL are set to 2 (1-bit packed α) or 3 (8-bit α generated from 1-bit plane α), an 8-bit α value is generated by using the VI6_RPFn_ALPH_SEL.ALPHA0 [7:0] value when the input 1-bit α value is 0 or by using the VI6_RPFn_ALPH_SEL.ALPHA1 [7:0] value when the input 1-bit α value is 1. When VI6_RPFn_ALPH_SEL.ASEL is set to 3, a packed α value that is included in RGB data is discarded.

1-bit mask α generator:

Generates 1-bit α data from the input 8-bit α data or pixel data. When the input α data is one bit, this generator outputs it without change.

When VI6_RPFn_ALPH_SEL.ASEL are set to 0 (8-, 4-, or 1-bit packed α) or 2 (plane α) and VI6_RPFn_MSKCTRL.MSK_EN is set to 0, the α plane read from the external memory to be used in IROP is converted to 1-bit α data when necessary. When the α plane data read from the external RAM is 8 bits (BSEL = 0), if the value is 0, a 1-bit α value of b'0 is generated; if the value is not 0, a 1-bit α value of b'1 is generated. When the α plane data is one bit (BSEL = 1), this generator outputs it without change.

When the value of the 1-bit α generated by the 1-bit mask α generator is b'0, the 8-bit α and pixel data specified in VI6_RPFn_MSKSET0 are output as the source. When the generated 1-bit α value is b'1, the 8-bit α and pixel data specified in VI6_RPFn_MSKSET1 are output as the source.

As shown in Figure 60.18, when VI6_RPFn_ALPH_SEL.ASEL are set to 1 (8-bit plane α) or 3 (1-bit plane α), the α plane read from the external RAM is sent to the 8-bit transparent α generator as transparency information. When VI6_RPFn_MSKCTRL.MSK_EN is set to 0, the 1-bit α for masking is generated according to the input α plane (refer to section 60.2.6.8), but the 1-bit mask α generator does not refer to the input α plane because it is input to the 8-bit transparent α generator as transparency information. Accordingly, the 1-bit mask α generator does not generate a 1-bit α for masking and the data on the source side becomes invalid; that is, IROP operation cannot be executed. Set the IROP bits to 0 in this case. In contrast, when VI6_RPFn_MSKCTRL.MSK_EN is set to 1, the 1-bit mask α generator creates α data for masking according to the pixel data instead of the input α plane data, and IROP operation can be executed in this case.

IROP operation unit

Executes ROP operation according to the opcode specified in VI6_RPFn_ALPH_SEL.IROP. For ROP operation (other than NOP), valid values should be input both for the source and destination. As described in the above (description of the 1-bit mask α generator), when VI6_RPFn_ALPH_SEL.ASEL are set to 1 or 3 and VI6_RPFn_MSKCTRL.MSK_EN is set to 0, the source data for the IROP operation unit is treated as invalid; set VI6_RPFn_ALPH_SEL.IROP to 0 (NOP). When VI6_RPFn_ALPH_SEL.ASEL are set to 4, a fixed α value is output from the RPF and IROP operation is not available. In the same way as the above case, set VI6_RPFn_ALPH_SEL.IROP to 0 (NOP).

To specify a valid source value for the IROP operation unit and execute IROP operation (specify an opcode other than NOP in the IROP bits), specify register values as shown in Table 60.15. Where the source input state is indicated as "Valid" in the table, IROP operation can be executed. In the cases where IROP operation is not available, set the IROP bits to 0 (NOP).

**Table 60.15 Source Input State in IROP Operation Unit**

| VI6_RPFn_ALPH_SEL.ASEL [2:0] | | VI6_RPFn_MSKCTRL.MSK_EN | | |
|---|---|---|---|---|
| | | 0<br>(Source data is generated according to input α plane) | | 1<br>(Source data is generated according to the destination-side pixel data) |
| b'000 | (1-, 4-, or 8-bit packed α + plane α ) | Valid | (α plane input) | Valid |
| b'001 | (8-bit α plane) | Invalid | (IROP operation is not available; α plane is output to the subsequent modules behind RPF) | Valid |
| b'010 | (8-bit α generated from 1-bit packed α + plane α) | Valid | (α plane input) | Valid |
| b'011 | (8-bit α generated from 1-bit plane α) | Invalid | (IROP operation is not available; α plane is output to the subsequent modules behind RPF) | Valid |
| b'100 | (Fixed α) | Invalid (IROP operation is not available; fixed α is output to the subsequent modules behind RPF) | | |

For the handling of the α values shown in Figure 60.18 and Table 60.15, the relationship between the RPF input format and RPF output α value is shown in Table 60.16. Where only bit names are shown in the table, the bits are in VI6_RPFn_ALPH_SEL described in this section.

**Table 60.16 α Value Selected and Output according to ASEL Bits in Each Input Format**
*Fixed value 0xFF is output because packed α is not included in YCbCr.

| ASEL Setting | | α Value Output for Each Input Format | | |
|---|---|---|---|---|
| | | RGB | YCbCr | OSD-CLUT |
| b'000 | (8-, 4-, or 1-bit packed α is input) | 1- ,4-, or 8-bit pixel α | 0xFF* | α value in CLUT |
| b'001 | (8-bit plane α is input) | 8-bit α plane | 8-bit α plane | 8-bit α plane |
| b'010 | (8-bit α is generated from the 1-bit packed α input) | ALPHA0 or ALPHA1 setting | 0xFF* | 0xFF |
| b'011 | (8-bit α is generated from the 1-bit plane α input) | ALPHA0 or ALPHA1 setting | ALPHA0 or ALPHA1 setting | ALPHA0 or ALPHA1 setting |
| b'100 | (Fixed α is output) | VI6_RPFn_VRTCOL_SET.LAYA setting | | |

### 60.2.6.7 RPFn Virtual Plane Color Information Registers (VI6_RPFn_VRTCOL_SET)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LAYA [7:0] | | | | | | | | LAYR [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | LAYG [7:0] | | | | | | | | LAYB [7:0] | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | LAYA [7:0] | All 0 | R/W | Virtual-Input Fixed α Value<br>These bits specify the fixed α value generated in the RPFn when the virtual input function is enabled through the VI6_RPFn_INFMT.VIR setting.<br>When the virtual input function is disabled (VI6_RPFn_INFMT.VIR = 0), these bits are used to specify the fixed α value to be output from the RPF while VI6_RPFn_ALPH_SEL.ASEL are set to 4. A value from 0 to 255 can be specified. |
| 23 to 16 | LAYR [7:0] | All 0 | R/W | Virtual-Input Fixed R/Cr Component Value<br>These bits specify the fixed R or Cr value generated in the RPFn when the virtual input function is enabled through the VI6_RPFn_INFMT.VIR setting. When VI6_RPFn_INFMT.RDFMT is set to 19, the value specified in these bits is used as the R value. When VI6_RPFn_INFMT.RDFMT is set to 64, the value specified in these bits is used as the Cr value. A value from 0 to 255 can be specified. |
| 15 to 8 | LAYG [7:0] | All 0 | R/W | Virtual-Input Fixed G/Y Component Value<br>These bits specify the fixed G or Y value generated in the RPFn when the virtual input function is enabled through the VI6_RPFn_INFMT.VIR setting. When VI6_RPFn_INFMT.RDFMT is set to 19, the value specified in these bits is used as the G value. When VI6_RPFn_INFMT.RDFMT is set to 64, the value specified in these bits is used as the Y value. A value from 0 to 255 can be specified. |
| 7 to 0 | LAYB [7:0] | All 0 | R/W | Virtual-Input Fixed B/Cb Component Value<br>These bits specify the fixed B or Cb value generated in the RPFn when the virtual input function is enabled through the VI6_RPFn_INFMT.VIR setting. When VI6_RPFn_INFMT.RDFMT is set to 19, the value specified in these bits is used as the B value. When VI6_RPFn_INFMT.RDFMT is set to 64, the value specified in these bits is used as the Cb value. A value from 0 to 255 can be specified. |

### 60.2.6.8 RPFn Mask Control Registers (VI6_RPFn_MSKCTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | MSK_EN | MGR [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MGG [7:0] | | | | | | | | MGB [7:0] | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 24 | MSK_EN | 0 | R/W | Mask Generation Specification<br>Specifies the method of α value generation in the 1-bit mask α generator shown in Figure 60.18.<br>0: A 1-bit mask value is generated according to the input α plane value. When the input α is in the 1-bit format (VI6_RPFn_ALPH_SEL.BSEL = 1), the 1-bit mask value is output without change. When the input α is in the 8-bit format (VI6_RPFn_ALPH_SEL.BSEL = 0), the 1-bit mask value is 0 if the α value is 0x00; otherwise, the 1-bit mask value is 1.<br>1: The R/Cr, G/Y, and B/Cb components of the image input to the destination side of the IROP operation unit are compared with the values specified in the MGR, MGG, and MGB bits, respectively. When all values match, 1 is output as the 1-bit mask value, and in other cases, 0 is output. When the generated 1-bit mask data is not used, set VI6_RPFn_ALPH_SEL.IROP to 0. |
| 23 to 16 | MGR [7:0] | All 0 | R/W | R/Cr Comparison Value for 1-Bit α Generation<br>These bits specify the R/Cr value to be compared for 1-bit α generation by using the pixel data on the destination side. When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify an R value for comparison. When YCbCr is specified, specify a Cr value for comparison.<br>This setting is ignored when the MSK_EN bit is set 0.<br>A value from 0 to 255 can be specified. |
| 15 to 8 | MGG [7:0] | All 0 | R/W | G/Y Comparison Value for 1-Bit α Generation<br>These bits specify the G/Y value to be compared for 1-bit α generation by using the pixel data on the destination side. When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify a G value for comparison. When YCbCr is specified, specify a Y value for comparison.<br>This setting is ignored when MSK_EN is set to 0.<br>A value from 0 to 255 can be specified. |
| 7 to 0 | MGB [7:0] | All 0 | R/W | B/Cb Comparison Value for 1-Bit α Generation<br>These bits specify the B/Cb value to be compared for 1-bit α generation by using the pixel data on the destination side. When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify a B value for comparison. When YCbCr is specified, specify a Cb value for comparison.<br>This setting is ignored when MSK_EN is set to 0.<br>A value from 0 to 255 can be specified. |

### 60.2.6.9 RPFn IROP-SRC Input Value Registers 0 (VI6_RPFn_MSKSET0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSA0 [7:0] | | | | | | | | MSR0 [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSG0 [7:0] | | | | | | | | MSB0 [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | MSA0 [7:0] | All 0 | R/W | IROP-Source Input α Value when 1-Bit α is 0<br>These bits specify the 8-bit α value to be input as the source to the IROP operation unit when the internal 1-bit α value generated through the 1-bit mask generator is 0 (Figure 60.18). A value from 0 to 255 can be specified. |
| 23 to 16 | MSR0 [7:0] | All 0 | R/W | IROP-Source Input R/Cr Value when 1-Bit α is 0<br>These bits specify the R/Cr value to be input as the source to the IROP operation unit when the internal 1-bit α value generated through the 1-bit mask generator is 0 (Figure 60.18).<br>When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify an R component value. When YCbCr is specified, specify a Cr component value. A value from 0 to 255 can be specified. |
| 15 to 8 | MSG0 [7:0] | All 0 | R/W | IROP-Source Input G/Y Value when 1-Bit α is 0<br>These bits specify the G/Y value to be input as the source to the IROP operation unit when the internal 1-bit α value generated through the 1-bit mask generator is 0 (Figure 60.18).<br>When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify a G component value. When YCbCr is specified, specify a Y component value. A value from 0 to 255 can be specified. |
| 7 to 0 | MSB0 [7:0] | All 0 | R/W | IROP-Source Input B/Cb Value when 1-Bit α is 0<br>These bits specify the B/Cb value to be input as the source to the IROP operation unit when the internal 1-bit α value generated through the 1-bit mask generator is 0 (Figure 60.18).<br>When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify a B component value. When YCbCr is specified, specify a Cb component value. A value from 0 to 255 can be specified. |

### 60.2.6.10 RPFn IROP-SRC Input Value Registers 1 (VI6_RPFn_MSKSET1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSA1 [7:0] | | | | | | | | MSR1 [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSG1 [7:0] | | | | | | | | MSB1 [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | MSA1 [7:0] | All 0 | R/W | IROP-Source Input α Value when 1-Bit α is 1<br>These bits specify the 8-bit α value to be input as the source to the IROP operation unit when the internal 1-bit α value generated through the 1-bit mask generator is 1. A value from 0 to 255 can be specified (Figure 60.18). |
| 23 to 16 | MSR1 [7:0] | All 0 | R/W | IROP-Source Input R/Cr Value when 1-Bit α is 1<br>These bits specify the R/Cr value to be input as the source to the IROP operation unit when the internal 1-bit α value generated through the 1-bit mask generator is 1 (Figure 60.18).<br>When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify an R component value. When YCbCr is specified, specify a Cr component value. A value from 0 to 255 can be specified. |
| 15 to 8 | MSG1 [7:0] | All 0 | R/W | IROP-Source Input G/Y Value when 1-Bit α is 1<br>These bits specify the G/Y value to be input as the source to the IROP operation unit when the internal 1-bit α value generated through the 1-bit mask generator is 1 (Figure 60.18).<br>When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify a G component value. When YCbCr is specified, specify a Y component value. A value from 0 to 255 can be specified. |
| 7 to 0 | MSB1 [7:0] | All 0 | R/W | IROP-Source Input B/Cb Value when 1-Bit α is 1<br>These bits specify the B/Cb value to be input as the source to the IROP operation unit when the internal 1-bit α value generated through the 1-bit mask generator is 1 (Figure 60.18).<br>When RGB is specified as the RPFn input format through VI6_RPFn_INFMT.RDFMT, specify a B component value. When YCbCr is specified, specify a Cb component value. A value from 0 to 255 can be specified. |

### 60.2.6.11 RPFn Color Keying Control Registers (VI6_RPFn_CKEY_CTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | – | – | – | – | – | – | – | LTH | – | – | – | CV | – | – | SAPE1 | SAPE0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9, 7 to 5, 3, 2 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | LTH | 0 | R/W | Transparent color-luma Threshold Mode Enable/Disable<br>This bit enables or disables transparent color-luma threshold mode for the color keying module.<br>In transparent color-luma threshold mode, color information 0 (GY0) specified in VI6_RPFn_CKEY_SET0 register is compared with the input luma values.<br>When the input data is in YCbCr format, and if input Y value is equal to or smaller than specified color information (VI6_RPFn_CKEY_SET0.GY0), the input α value is replaced with the value specified in VI6_RPFn_CKEY_SET0.AP0. This bit is available only when the input data is in YCbCr format. When the input data is in an RGB format, set this bit to 0.<br>This bit setting is valid only when the CV bit is set to 0; it is ignored when the CV bit is set to 1. When using the virtual input function (VI6_RPFn_INFMT.VIR = 1), set this bit to 0.<br>0: Luma threshold mode is disabled (Matched color mode)<br>1: Luma threshold mode is enabled |
| 4 | CV | 0 | R/W | Color Replacement Control<br>This bit controls the color replacement function in the color keying module shown in Figure 60.3. When an RGB format is specified as the color space of the RPFn input data through VI6_RPFn_INFMT.RDFMT, and if all components of an input pixel match the color components specified in VI6_RPFn_CKEY_SET0, the color replacement function replaces the values of the input α and all RGB components with the α and color components specified in VI6_RPFn_CKEY_SET1. When a YCbCr format is specified as the color space of the RPFn input data through VI6_RPFn_INFMT.RDFMT, only the Y data is compared; if the luminance component of an input pixel matches the value specified in VI6_RPFn_CKEY_SET0.GY0, the color replacement function replaces the values of the input α and all YCbCr components with the α and color components specified in VI6_RPFn_CKEY_SET1.<br>When the CV bit is set to 1, the color replacement function is enabled. When using the virtual input function (VI6_RPFn_INFMT.VIR = 1), set this bit to 0.<br>0: Color replacement function is disabled (transparent color mode).<br>1: Color replacement function is enabled. |
| 1 | SAPE1 | 0 | R/W | Comparison Color Data Setting 1 Enable/Disable<br>This bit enables or disables comparison color data setting 1 in the |

| | | | | |
|---|---|---|---|---|
| | | | | transparent color mode for the color keying module. This bit setting is valid when the CV bit and the LTH bit are set to 0; it is ignored when the CV bit or the LTH bit is set to 1.<br>In transparent color mode, color information 1 (VI6_RPFn_CKEY_SET1.R1/GY1/B1) specified in VI6_RPFn_CKEY_SET1 is compared with the input component values. When the input data is in an RGB format, and if all input values match the specified color information, the input α value is replaced with the value specified in VI6_RPFn_CKEY_SET1.AP. When the input data is in YCbCr format, only the Y data is compared.<br>When using the virtual input function (VI6_RPFn_INFMT.VIR = 1), set this bit to 0.<br>0: Comparison color data setting 1 is disabled.<br>1: Comparison color data setting 1 is enabled. |
| 0 | SAPE0 | 0 | R/W | Comparison Color Data Setting 0 Enable/Disable<br>This bit enables or disables comparison color data setting 0 in the transparent color mode for the color keying module. This bit setting is valid when the CV bit and the LTH bit are set to 0; it is ignored when the CV bit or the LTH bit is set to 1.<br>In transparent color mode, color information 0 (VI6_RPFn_CKEY_SET0.R0/GY0/B0) specified in VI6_RPFn_CKEY_SET0 is compared with the input component values. When the input data is in an RGB format, and if all input values match the specified color information, the input α value is replaced with the value specified in VI6_RPFn_CKEY_SET0.AP. When the input data is in YCbCr format, only the Y data is compared.<br>When using the virtual input function (VI6_RPFn_INFMT.VIR = 1), clear this bit to 0.<br>0: Comparison color data setting 0 is disabled.<br>1: Comparison color data setting 0 is enabled. |

### 60.2.6.12 RPFn Color Keying Color Setting Registers-m (VI6_RPFn_CKEY_SETm : M=0, 1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | APm [7:0] | | | | | | | | Rm [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GYm [7:0] | | | | | | | | Bm [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 31 to 24 | APm [7:0] | All 0 | R/W | α Data in Color Keying Color Information-m<br>• In transparent color-matched color mode for color keying<br>When the input data matches color setting-m (Rm, GYm, and Bm) for transparent color comparison in the color keying module, the input α value is replaced with the value specified in these bits. Specify the α value to replace the input value.<br>• In transparent color-luma threshold mode for color keying<br>When the input data is in YCbCr format, and if input Y value is equal to or smaller than GY0, the input α value is replaced with the value specified in this bit field (AP0).<br>Specify the α value in this bit field (AP0) to replace the input value.<br>AP1 is not used in this mode. Set AP1 to 0 value.<br>• In color replacement mode for color keying<br>These bits are not used in this mode. Clear them to 0.<br>α value replacement through these bits in transparent color mode for color keying takes priority over the α value selected through the VI6_RPFn_ALPH_SEL.ASEL setting.<br>A value from 0 to 255 can be specified. |
| 23 to 16 | Rm [7:0] | All 0 | R/W | R*1/Cr Component Data in Color Keying Color Information-m<br>• In transparent color-matched color mode for color keying<br>Specify the R component value for comparison enabled through the VI6_RPFn_CKEY_CTRL.SAPEm setting. When the RPFn input is in YCbCr format, the color keying module does not compare the Cr component, and the setting of these bits is ignored.<br>• In transparent color-luma threshold mode for color keying<br>The color keying module does not refer this bit field, and the setting of these bits is ignored.<br>• In color replacement mode for color keying<br>Specify the R component value to be compared with the input data in the color replacement function of the color keying module.<br>A value from 0 to 255 can be specified. |
| 15 to 8 | GYm [7:0] | All 0 | R/W | G*1/Y Component Data in Color Keying Color Information-m<br>• In transparent color-matched color mode for color keying<br>Specify the G/Y component value for comparison enabled through the VI6_RPFn_CKEY_CTRL.SAPEm setting.<br>• In transparent color-luma threshold mode for color keying<br>Specify the Y component value in the GY0 to be compared.<br>• In color replacement mode for color keying<br>Specify the G/Y component value to be compared with the input data in the color replacement function of the color keying module.<br>A value from 0 to 255 can be specified. |
| 7 to 0 | Bm [7:0] | All 0 | R/W | B*1/Cb Component Data in Color Keying Color Information-m<br>• In transparent color-matched color mode for color keying<br>Specify the B component value for comparison enabled through the VI6_RPFn_CKEY_CTRL.SAPEm setting. When the RPFn input is in YCbCr format, the color keying module does not compare the Cb component, and the setting of these bits is ignored.<br>• In transparent color-luma threshold mode for color keying<br>The color keying module does not refer this bit field, and the setting of these bits is ignored. |

| | | | | |
|---|---|---|---|---|
| | | | | • In color replacement mode for color keying<br>Specify the B component value to be compared with the input data in the color replacement function of the color keying module.<br>A value from 0 to 255 can be specified. |

(*1) When comparison data is specified in an RGB format, if a packed format is selected for RGB input and each of the RGB components is not 8 bits, the lower-order bits of input data are extended as specified through VI6_RPFn_INFMT.CEXT before comparison. The RGB components to be compared with the input should also be extended in the same way and the extended values should be specified in this register.

### 60.2.6.13 RPFn Source Picture Memory Stride Setting Registers (VI6_RPFn_SRCM_PSTRIDE)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PICT_STRD_Y [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PICT_STRD_C [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | PICT_STRD_Y [15:0] | All 0 | R/W | Memory Stride of Source Picture Y/RGB Plane<br>These bits specify in 1-byte units the memory stride of the source picture Y/RGB plane read by the RPFn.<br>A value from 0 to 65535 can be specified. Refer to Figure 60.19 for settings. |
| 15 to 0 | PICT_STRD_C [15:0] | All 0 | R/W | Memory Stride of Source Picture C Plane<br>These bits specify in 1-byte units the memory stride of the source picture C plane read by the RPFn. When an RGB-format picture is read, these bits do not need to be set.<br>A value from 0 to 65535 can be specified. Refer to Figure 60.19 for settings.<br>In the YCbCr planar format, this setting is used as the memory stride of the Cb and Cr planes. |

This register specifies the memory stride of the picture planes in the source area as shown in Figure 60.19. The memory stride of the α plane should be specified through VI6_RPFn_SRCM_ASTRIDE.ALPH_STRD. When the RPF input is in an RGB format, only the RGB plane is read; when the input is in YCbCr format, the Y and C planes are read as shown in Figure 60.19. When the α plane is used, the α plane is also read. According to the image format and the necessity of the α

plane, specify the necessary addresses where the source image is stored (VI6_RPFn_SRCM_ADDR_Y, VI6_RPFn_SRCM_ADDR_C, and VI6_RPFn_SRCM_ADDR_AI).

Whether the α plane needs to be read is determined according to the α plane selection method and IROP operation type. For details, refer to section 60.2.6.6.

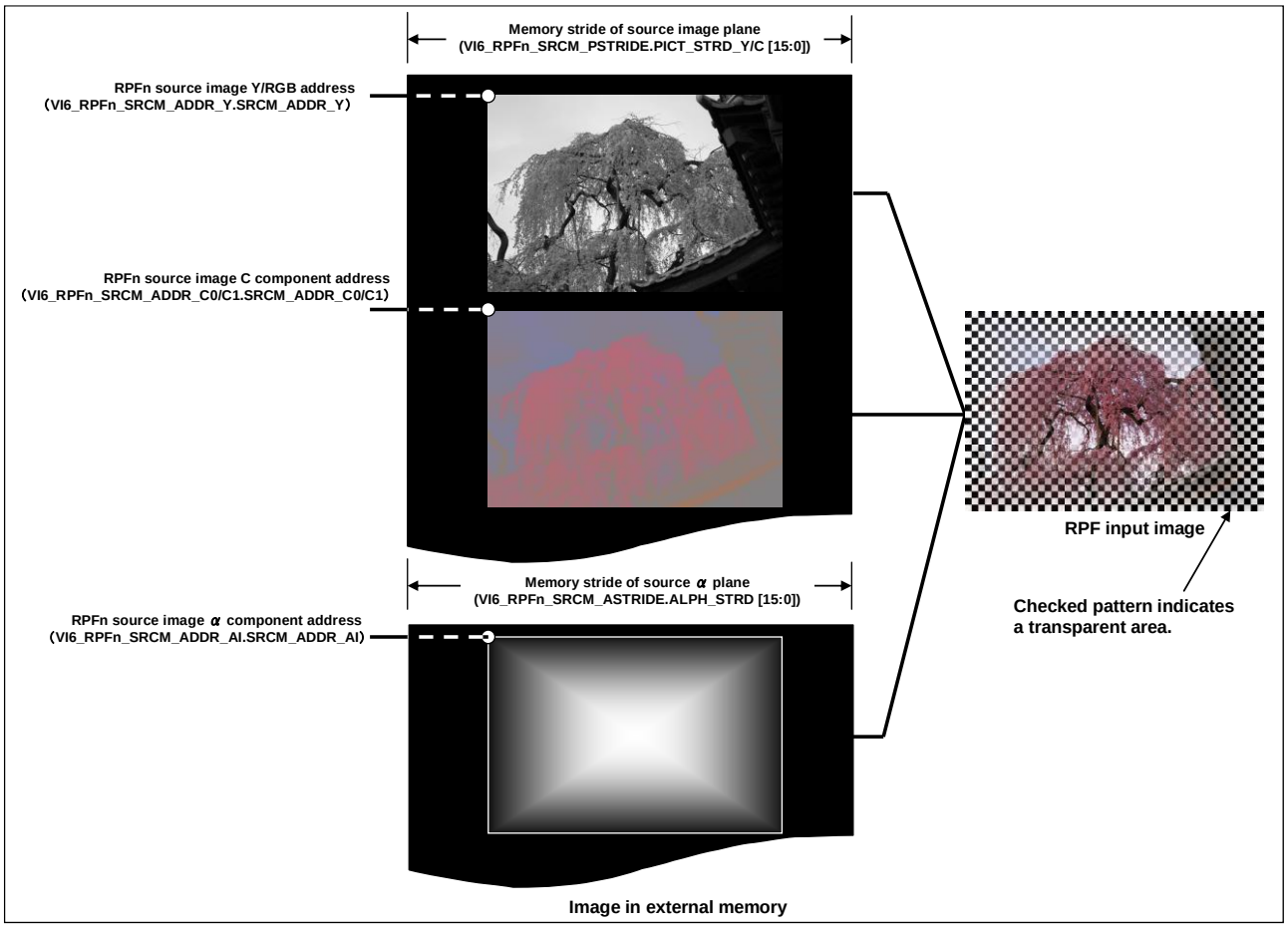



**Figure 60.19 Reading an Image from RPFn Source Area**

### 60.2.6.14 RPFn Source A Memory Stride Setting Registers (VI6_RPFn_SRCM_ASTRIDE)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ALPH_STRD [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 0 | ALPH_STRD [15:0] | All 0 | R/W | Memory Stride of Source α Plane<br>These bits specify in 1-byte units the memory stride of the source α plane read by the RPFn. A value from 0 to 65535 can be specified. Refer to Figure 60.19 for settings. |

### 60.2.6.15 RPFn Source Y/RGB Address Registers (VI6_RPFn_SRCM_ADDR_Y)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRCM_ADDR_Y [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRCM_ADDR_Y [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRCM_ ADDR_Y [31:0] | All 0 | R/W | Source Image Y/RGB Plane Storing Address<br>These bits specify in 1-byte units the start address of the source image Y plane and packed RGB plane read by the RPFn.<br>A value from H'0000 0000 to H'FFFF FFFF can be specified. Refer to Figure 60.19 for settings. |

### 60.2.6.16 RPFn Source Chroma Address Registers 0 (VI6_RPFn_SRCM_ADDR_C0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRCM_ADDR_C0 [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRCM_ADDR_C0 [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRCM_ ADDR_C0 [31:0] | All 0 | R/W | Source Image C Plane Storing Address 0<br>These bits specify in 1-byte units the start address of the source image C plane read by the RPFn. Here, the C plane indicates the combined CbCr plane when a semi-planar format is selected from the packed YCbCr formats shown in Table 60.12 or the Cb plane when a planar format is selected. When an interleaved format is selected or the RPF input is in an RGB format, this setting is not used.<br>A value from H'0000 0000 to H'FFFF FFFF can be specified. Refer to Figure 60.19 for settings. |

### 60.2.6.17 RPFn Source Chroma Address Registers 1 (VI6_RPFn_SRCM_ADDR_C1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRCM_ADDR_C1 [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRCM_ADDR_C1 [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRCM_ ADDR_C1 [31:0] | All 0 | R/W | Source Image C Plane Storing Address 1<br>These bits specify in 1-byte units the start address of the Cr plane when a planar YCbCr format shown in Table 60.12 is read by the RPFn.<br>This setting is not used when the RPF input is in YCbCr format that is not a planar format or in an RGB format.<br>A value from H'0000 0000 to H'FFFF FFFF can be specified. Refer to Figure 60.19 for settings. |

### 60.2.6.18 RPFn Source A Address Registers (VI6_RPFn_SRCM_ADDR_AI)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRCM_ADDR_AI [31:16] | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRCM_ADDR_AI [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRCM_ ADDR_AI [31:0] | All 0 | R/W | Source Image α Plane Storing Address<br>These bits specify in 1-byte units the start address of the α plane of the source image read by the RPFn. Specify in the same way as the start address of the picture plane. When the α plane is not read from the source area, these bits do not need to be set.<br>A value from H'0000 0000 to H'FFFF FFFF can be specified. Refer to Figure 60.19 for settings. |

### 60.2.6.19 RPFn Bus Access Control Registers (VI6_RPFn_BAC)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | B512 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17, 15 to 0 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | B512 | 0 | R/W | This register is for purpose of h/w debugging. |

### 60.2.6.20 RPFn Multiple Alpha Control (VI6_RPFn_MULT_ALPH)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | A_MMD | – | – | P_MMD[1:0] | | ALPHA_RATIO[7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13, 11, 10 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 | A_MMD | 0 | R/W | 0: Alpha data go through multiply-alpha unit.<br>1: Multiply-alpha unit multiplies alpha data by specified alpha (ALPHA_RATIO [7:0].<br>When output format from CSC unit is YCbCr, set 0 to this bit. |
| 9 to 8 | P_MMD[1:0] | All 0 | R/W | 0: Image data go through multiply-alpha unit.<br>1: Multiply-alpha unit multiplies image data by specified alpha (ALPHA_RATIO [7:0]).<br>2: Multiply-alpha unit multiplies image data by alpha data.<br>3: Multiply-alpha unit multiplies image data by alpha data and specified alpha (ALPHA_RATIO [7:0].<br>When output format from CSC unit is YCbCr, set 0 to this bit. |
| 7 to 0 | ALPHA_RATIO [7:0] | All 0 | R/W | Multiply alpha value |

ALPin: Input Alpha data to Multiply-alpha unit
PIXin(R): Input R data to Multiply-alpha unit
PIXin(G): Input G data to Multiply-alpha unit
PIXin(B): Input B data to Multiply-alpha unit
ALPout: Output Alpha data from Multiply-alpha unit
PIXout(R): Output R data from Multiply-alpha unit
PIXout(G): Output G data from Multiply-alpha unit
PIXout(B): Output B data from Multiply-alpha unit

**Table 60.17 Expression of output alpha data from Multiply-alpha unit**

| A_MMD | ALPHA_RATIO[7:0] | Expression |
|---|---|---|
| 0 | Don't care | ALPout = ALPin |
| 1 | Not 255 | ALPout = ALPin * ALPHA_RATIO / 256 |
| | 255 | ALPout = ALPin |

**Table 60.18 Expression of output pixel data from Multiply-alpha unit**

| P_MMD[1:0] | ALPHA_RATIO[7:0] | ALPin | Expression |
|---|---|---|---|
| 0 | Don't care | Don't care | PIXout(R) = PIXin(R)<br>PIXout(G) = PIXin(G)<br>PIXout(B) = PIXin(B) |
| 1 | Not 255 | Don't care | PIXout(R) = PIXin(R) * ALPHA_RATIO / 256<br>PIXout(G) = PIXin(G) * ALPHA_RATIO / 256<br>PIXout(B) = PIXin(B) * ALPHA_RATIO / 256 |
| | 255 | Don't care | PIXout(R) = PIXin(R)<br>PIXout(G) = PIXin(G)<br>PIXout(B) = PIXin(B) |
| 2 | Don't care | Not 255 | PIXout(R) = PIXin(R) * ALPin / 256<br>PIXout(G) = PIXin(G) * ALPin / 256<br>PIXout(B) = PIXin(B) * ALPin / 256 |
| | Don't care | 255 | PIXout(R) = PIXin(R)<br>PIXout(G) = PIXin(G)<br>PIXout(B) = PIXin(B) |
| 3 | Not 255 | Not 255 | PIXout(R) = PIXin(R) * ALPin * ALPHA_RATIO / 256 / 256<br>PIXout(G) = PIXin(G) * ALPin * ALPHA_RATIO / 256 / 256<br>PIXout(B) = PIXin(B) * ALPin * ALPHA_RATIO / 256 / 256 |
| | 255 | Not 255 | PIXout(R) = PIXin(R) * ALPin / 256<br>PIXout(G) = PIXin(G) * ALPin / 256<br>PIXout(B) = PIXin(B) * ALPin / 256 |
| | Not 255 | 255 | PIXout(R) = PIXin(R) * ALPHA_RATIO / 256<br>PIXout(G) = PIXin(G) * ALPHA_RATIO / 256<br>PIXout(B) = PIXin(B) * ALPHA_RATIO / 256 |
| | 255 | 255 | PIXout(R) = PIXin(R)<br>PIXout(G) = PIXin(G)<br>PIXout(B) = PIXin(B) |

### 60.2.6.21 RPFn Extended Input Format Registers 0 (VI6_RPFn_EXT_INFMT0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | F2B | – | IPBD_Y [2:0] | | | – | IPBD_C [2:0] | | | – | BYPP_M1 [2:0] | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13, 11, 7, 3 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 | F2B | 0 | R/W | Pixel bit alignment of 2 bytes/component for YUV data.<br>This register available when IPBD_Y != 0 or IPBD_C != 0.<br>0: LSB adjust format<br>1: MSB adjust format<br>Refer to Figure 60.14 for clear. |
| 10 to 8 | IPBD_Y [2:0] | 0 | R/W | Input bit-depth for Y component:<br>0: 8-bit<br>1: 10-bit<br>2: 12-bit<br>3~7: reserved<br>For YUV 4:2:2 Interleaved set the same value to IPBD_C[2:0] and IPBD_Y[2:0] |
| 6 to 4 | IPBD_C [2:0] | 0 | R/W | Input bit-depth for chroma (Cb/Cr) component:<br>0: 8-bit<br>1: 10-bit<br>2: 12-bit<br>3~7: reserved<br>For YUV 4:2:2 Interleaved set the same value to IPBD_C[2:0] and IPBD_Y[2:0] |
| 2 to 0 | BYPP_M1 [2:0] | 0 | R/W | Byte per pixel minus 1 for RGB packed data format RGB10, RGB10A2, A2RGB10.<br>Set 3 when RDFMT[6:0] is 0x30.<br>Set 0 when RDFMT[6:0] is not 0x30. |

### 60.2.6.22 RPFn Extended Input Format Registers 1 (VI6_RPFn_EXT_INFMT1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | CPOS0 [5:0] | | | | | | – | – | CPOS1 [5:0] | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | CPOS2 [5:0] | | | | | | – | – | CPOS3 [5:0] | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30, 23, 22, 15, 14, 7, 6 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29 to 24 | CPOS0 [5:0] | 0 | R/W | Bit position of color component register 0 |
| 21 to 16 | CPOS1 [5:0] | 0 | R/W | Bit position of color component register 1 |
| 13 to 8 | CPOS2 [5:0] | 0 | R/W | Bit position of color component register 2 |
| 5 to 0 | CPOS3 [5:0] | 0 | R/W | Bit position of color component register 3 |

Specify bit position of color components in a pixel of RGB packed data format RGB10, RGB10A2, A2RGB10. Set this register referring to Table 60.11

### 60.2.6.23 RPFn Extended Input Format Registers 2 (VI6_RPFn_EXT_INFMT2)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | CLEN0 [4:0] | | | | | – | – | – | CLEN1 [4:0] | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | CLEN2 [4:0] | | | | | – | – | – | CLEN3 [4:0] | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 , 23 to 21, 15 to 13, 7 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 24 | CLEN0 [4:0] | 0 | R/W | Bit width of color components register 0 |
| 20 to 16 | CLEN1 [4:0] | 0 | R/W | Bit width of color components register 1 |
| 12 to 8 | CLEN2 [4:0] | 0 | R/W | Bit width of color components register 2 |
| 4 to 0 | CLEN3 [4:0] | 0 | R/W | Bit width of color components register 3 |

Specify bit width of color components in a pixel of RGB packed data format RGB10, RGB10A2, A2RGB10. Set this register referring to Table 60.11.

### 60.2.6.24 RPFn Bit Reduction and DITHer ConTRoL (VI6_RPFn_BRDITH_CTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | ODE | – | – | – | – | – | – | – | CBRM |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9, 7 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 8 | ODE | 0 | R/W | Ordered Dither (mode A) Enable/Disable for when reducing from 10 bits (for Y, R, G, B) or 12 bits (Y) to 8bits after reading YCbCr/RGB format data from memory<br>0: Ordered dither (mode A) is disabled.<br>1: Ordered dither (mode A) is enabled.<br>Refer to Table 60.19 for usage. |
| 0 | CBRM | 0 | R/W | Bit count Reduction Method selection for component, Y, Cb, Cr, R, G, B when reducing from 10 bits (for Y, Cb, Cr, R, G, B) or 12bits (for Y, Cbr, Cr) to 8bits after reading YCbCr/RGB format data from memory<br>0: Bit count conversion: The lower-order bits are truncated<br>1: Bit count conversion: Rounding (rounding off)<br>Refer to Table 60.19 for usage. |

Table 60.19 Usage of RPF bit count reduction registers

| Setting of VI6_RPFn_BRDITH_CTRL | | Operation of bit count reduction from 10bpc/12bpc to 8bpc | |
|---|---|---|---|
| ODE | CBRM | For R/G/B/Y | For Cb/Cr |
| 0 | 0 | The lower-order bits are truncated | The lower-order bits are truncated |
| 0 | 1 | Rounding off | Rounding off |
| 1 | 0 | Ordered dither (mode A) | The lower-order bits are truncated |
| 1 | 1 | Ordered dither (mode A) | Rounding off |

### 60.2.7 WPF Control Registers

#### 60.2.7.1 WPF0-Source-RPF Registers (VI6_WPF0_SRCRPF)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | VIR_ACT [1:0] | | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | RPF4_ACT [1:0] | | RPF3_ACT [1:0] | | RPF2_ACT [1:0] | | RPF1_ACT [1:0] | | RPF0_ACT [1:0] | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30, 27, 26, 23 to 10 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29, | VIR_ACT | All 0 | R/W | Virtual RPF Start Enable in BRU |

| | | | | |
|---|---|---|---|---|
| 28 | [1:0] | | | These bits enable start of the virtual RPF in the BRU as the source RPF for the WPF0 when the WPF0 is started. For details of the virtual RPF, refer to the following.<br>Section 60.2.9.1, BRU Input Control Register (VI6_BRU_INCTRL)<br>Section 60.2.9.2, Size Register of BRU Input Virtual RPF (VI6_BRU_VIRRPF_SIZE)<br>Section 60.2.9.3, Display Location Register of BRU Input Virtual RPF (VI6_BRU_VIRRPF_LOC)<br>Section 60.2.9.4, Color Information Register of BRU Input Virtual RPF (VI6_BRU_VIRRPF_COL)Note that the virtual RPF is in the BRU as shown in Figure 60.25 and there are no register bits for DPR setting related to the virtual RPF.<br>0: The virtual RPF in the BRU is not started.<br>1: The virtual RPF in the BRU is started as a sublayer source RPF for the WPF0.<br>2: The virtual RPF in the BRU is started as the master-layer source RPF for the WPF0.<br>3: Setting prohibited |
| 9, 8 | RPF4_ACT [1:0] | All 0 | R/W | RPFn Start Enable (RPFn_ACT, n=0 to 4)<br>These bits enable start of RPFn as the source RPF for the WPF0 when the WPF0 is started. When RPFn is not used, set the VI6_DPR_RPFn_ROUTE.RT_RPFn bits to D'63. When ILV is used and image size of ILV output is equal to output size of the WPF0, set the RPFn_ACT, which ILVL (ILV left) is connected, to 2.<br>0: RPFn is not started.<br>1: RPFn is started as a sublayer source RPF for the WPF0.<br>2: RPFn is started as the master-layer source RPF for the WPF0.<br>3: Setting prohibited |
| 7, 6 | RPF3_ACT [1:0] | All 0 | R/W | |
| 5, 4 | RPF2_ACT [1:0] | All 0 | R/W | |
| 3, 2 | RPF1_ACT [1:0] | All 0 | R/W | |
| 1, 0 | RPF0_ACT [1:0] | All 0 | R/W | |

When the WPF0 is started through the VSP2 start register (VI6_CMD0), the RPF and virtual RPF in the BRU specified as the source RPF in this register are also started to supply data to the VSP2 internal modules.

Note the following when specifying the source RPF.

(1) When blending, ROP or IIF operation is applied to multiple images through the BRU, multiple source RPFs are necessary. When multiple source RPFs are used, images should be classified into a master layer and sublayers; assign one of the source RPFs as the master-layer source RPF and other RPFs as sublayer source RPFs. Do not assign all RPFs as sublayer source RPFs (VI6_WPF0_SRCRPF = 0x00000015) or two or more RPFs as the master-layer source RPF (VI6_WPF0_SRCRPF = 0x0000025A) (such settings are prohibited).
(2) When the BRU is not used, there should be only one source RPF for one WPF. In this case, the source RPF should be assigned as the master-layer source RPF.

### 60.2.7.2 WPF0 Horizontal Input Size Clipping Registers (VI6_WPF0_HSZCLIP)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | HCEN | – | – | – | – | HCL_OFST [7:0] | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | HCL_SIZE [12:0] | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29,<br>27 to 24,<br>15 to 13 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 | HCEN | 0 | R/W | Horizontal Size Clipping Enable/Disable<br>Enables or disables clipping of the horizontal size of the WPF0 input image.<br>0: Horizontal size clipping is disabled<br>1: Horizontal size clipping is enabled |
| 23 to 16 | HCL_OFST [7:0] | All 0 | R/W | Horizontal Size Clipping Offset Value Setting<br>These bits specify the offset size (pixels) from the left end of the image in horizontal size clipping when the HCEN bit is 1 (Figure 60.20). The left side of the image input to the WPF is cut off for the size specified in these bits. When the HCEN bit is 0, this setting is ignored.<br>For WPF0, specify the left-side offset size (pixels) for the image.<br>A value from 0 to 255 can be specified. (HCL_OFST + HCL_SIZE) should not exceed the horizontal size of the WPF input. If the setting shown in the bottom example in Figure 60.20 is made, VSP2 does not operate correctly. |
| 12 to 0 | HCL_SIZE [12:0] | All 0 | R/W | Horizontal Clipping Size Setting<br>When the HCEN bit is 1, these bits specify the clipping size for horizontal clipping processing. Through this processing, the area of the horizontal size specified through the HCL_SIZE bits starting from the offset position specified through the HCL_OFST bits is determined as the valid image area. Accordingly, the right-side pixels beyond the (HCL_OFST + HCL_SIZE) size in the WPF0 input image are discarded. When the HCEN bit is 0, this setting is ignored.<br>For WPF0, specify the horizontal clipping size for the image. |

| | | | | |
|---|---|---|---|---|
| | | | | A value from 1 to 8190 can be specified. (HCL_OFST + HCL_SIZE) should not exceed the horizontal size of the WPF input. If the setting shown in the bottom example in Figure 60.20 is made, VSP2 does not operate correctly. |

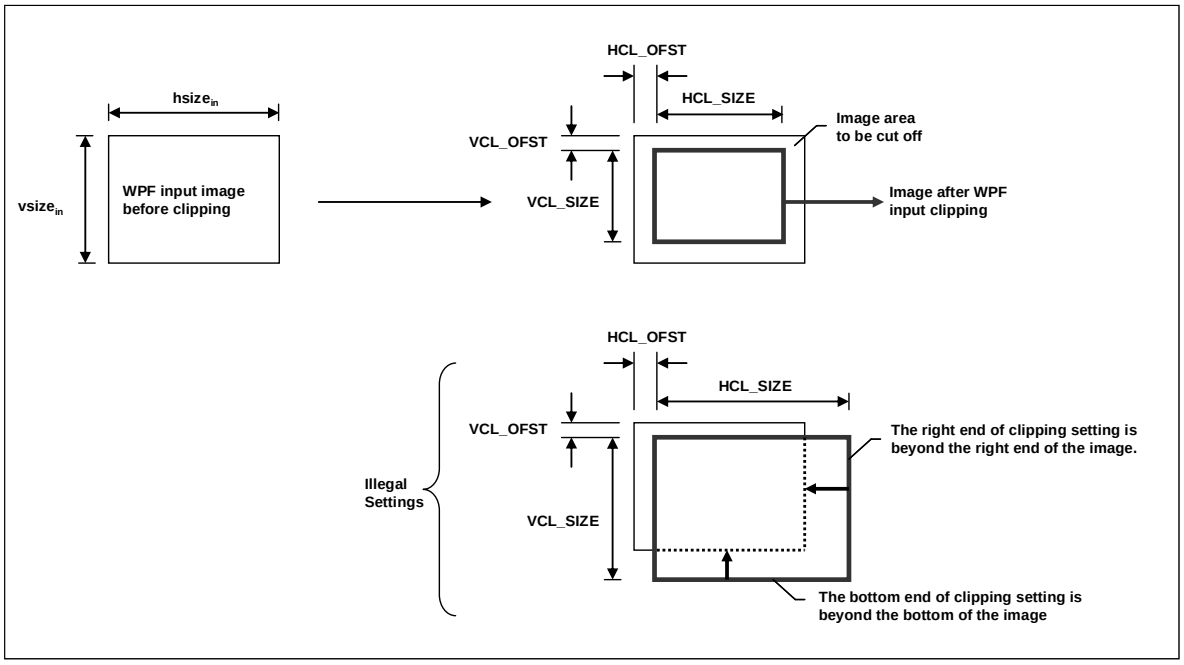



**Figure 60.20 Image Clipping in WPF Input Section**

### 60.2.7.3 WPF0 Vertical Input Size Clipping Registers (VI6_WPF0_VSZCLIP)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | VCEN | – | – | – | – | VCL_OFST [7:0] | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | VCL_SIZE [12:0] | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 31 to 29,<br>27 to 24,<br>15 to 13 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 | VCEN | 0 | R/W | Vertical Size Clipping Enable/Disable<br>Enables or disables clipping of the vertical size of the WPF0 input image.<br>0: Vertical size clipping is disabled<br>1: Vertical size clipping is enabled |
| 23 to 16 | VCL_OFST [7:0] | All 0 | R/W | Vertical Size Clipping Offset Value Setting<br>These bits specify the offset size (pixels) from the top end of the image in vertical size clipping when the VCEN bit is 1 (Figure 60.20). The top of the image input to the WPF is cut off for the size specified in these bits. When the VCEN bit is 0, this setting is ignored.<br>For WPF0, specify the top offset size (pixels) for the image.<br>A value from 0 to 255 can be specified. (VCL_OFST + VCL_SIZE) should not exceed the vertical size of the WPF input. If the setting shown in the bottom example in Figure 60.20 is made, VSP2 does not operate correctly. |
| 12 to 0 | VCL_SIZE [12:0] | All 0 | R/W | Vertical Clipping Size Setting<br>When the VCEN bit is 1, these bits specify the clipping size for vertical clipping processing. Through this processing, the area of the vertical size specified through the VCL_SIZE bits starting from the offset position specified through the VCL_OFST bits is determined as the valid image area. Accordingly, the bottom pixels beyond the (VCL_OFST + VCL_SIZE) size in the WPF0 input image are discarded. When the VCEN bit is 0, this setting is ignored.<br>For WPF0, specify the vertical clipping size for the image.<br>A value from 1 to 8190 can be specified. (VCL_OFST + VCL_SIZE) should not exceed the vertical size of the WPF input. If the setting shown in the bottom example in Figure 60.20 is made, VSP2 does not operate correctly. |

### 60.2.7.4 WPF0 Output Format Registers (VI6_WPF0_OUTFMT)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PDV [7:0] | | | | | | | | PXA | ODE | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPYCS | SPUVS | DITH [1:0] | | WRTM [2:0] | | | CSC | – | WRFMT [6:0] | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 21 to 16, 7 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 31 to 24 | PDV [7:0] | All 0 | R/W | PAD Value in Output Packed Data<br>These bits specify the value to be stored in the bit field indicated as PAD or P in the output formats shown in Table 60.20. To store this value in PAD, specify 0 in the PXA bit. A value from 0 to 255 can be specified. |
| 23 | PXA | 0 | R/W | PAD Data Select<br>Selects the value to be stored in the bit field indicated as PAD or P in the packed RGB output formats shown in Table 60.20. Both the value specified in the PDV bits and the α data input from the DPR to WPF are 8 bits, but some of the PAD and P bit fields shown in Table 60.20 are four bits or one bit. When the target bit field is not 8 bits, the number of bits in the PDV value and the α data input from the DPR to WPF is reduced according to the VI6_WPF0_RNDCTRL.ABRM setting. For bit count reduction, refer to Figure 60.21 and the description of VI6_WPF0_RNDCTRL.ABRM.<br>0: The value specified in the PDV bits is stored in the PAD shown in Table 60.20.<br>1: The α value output from DPR in pixel units is stored in the PAD shown in Table 60.20. |
| 22 | ODE | 0 | R/W | Ordered Dither (mode A) Enable/Disable<br>0: Ordered dither (mode A) is disabled.<br>1: Ordered dither (mode A) is enabled.<br>When the output format specified through the WRFMT bits is YCbCr, specify 0 in this bit. And when VI6_WPF0_OUTFMT.CSC is set to 1, specify 0 in this bit even in the case that the output format specified through the WRFMT bits is RGB.<br>Ordered dither is available only for 18bpp. So, when ODE bit is equal to 1, set WRFMT at 18bpp format.<br>When ODE bit is equal to 0, WPF dither method is specified by DITH[1:0] in the register<br>Ordered dither (mode A) is recommended rather than ordered dither (mode B) in case of 18bpp. |
| 15 | SPYCS | 0 | R/W | WPF Output Mode Setting 1<br>When the output format is YUY2, set this bit to 1 and set the WRFMT bits to 71 (0x47).<br>When the output format is YVYU, set this bit and the SPUVS bit to 1 and set the WRFMT bits to 71 (0x47).<br>In other cases, set this bit to 0. |
| 14 | SPUVS | 0 | R/W | WPF Output Mode Setting 2<br>When the output format is NV61, set this bit to 1 and set the WRFMT bits to 65 (0x41).<br>When the output format is NV21, set this bit to 1 and set the WRFMT bits to 66 (0x42).<br>When the output format is YVYU, set this bit and the SPYCS bit to 1 and set the WRFMT bits to 71 (0x47).<br>In other cases, set this bit to 0. |
| 13, 12 | DITH [1:0] | All 0 | R/W | Ordered Dither (mode B) Enable/Disable<br>When the output format specified through the WRFMT bits is RGB with 18 bpp |

| | | | | |
|---|---|---|---|---|
| | | | | (260000 colors) or less, the color reduction processing is applied to match the number of colors. The color reduction processing may generate the artifacts of pseudo gradation, which can be suppressed through dithering. The DITH bits enable or disable dithering during color reduction.<br>When the output format specified through the WRFMT bits is YCbCr, specify 0 in these bits.<br>And when VI6_WPF0_OUTFMT.CSC is set to 1, specify 0 in these<br>bits even in the case that the output format specified through the<br>WRFMT bits is RGB.<br>0: Dithering (mode B) is disabled<br>3: Dithering (mode B) is enabled<br>1, 2: Setting prohibited<br>When ODE bit in the register is 1, set this bit to 0. |
| 11 to 9 | WRTM [2:0] | All 0 | R/W | CSC Conversion Expression Setting<br>These bits select the expression for color space conversion. The conversion direction is RGB to YCbCr when the format specified in the WRFMT bits is YCbCr, or YCbCr to RGB when the format specified in the WRFMT bits is RGB.<br>0: BT.601 YCbCr [16,235/240] ↔ RGB [0,255]<br>1: BT.601 YCbCr [0,255] ↔ RGB [0,255]<br>2: BT.709 YCbCr [16,235/240] ↔ RGB [0,255]<br>3: BT.709 YCbCr [16,235/240] ↔ RGB [16,235]<br>4 to 7: Setting prohibited |
| 8 | CSC | 0 | R/W | Color Space Conversion Setting<br>Enables or disables YCbCr ↔ RGB color space conversion to be executed in the WPF0. The characteristics of color space conversion are determined by the WRTM setting.<br>There are some points to be noted about the relationship between the CSC setting and output format (WRFMT). For details refer to (*1) in section 60.2.6.3 RPFn Input Format Registers (VI6_RPFn_INFMT).<br>0: Color space is not converted.<br>1: Color space is converted. |
| 6 to 0 | WRFMT [6:0] | All 0 | R/W | WPF Output Image Format Setting<br>These bits select the format of the image output from the WPF0 to the external memory from among those listed in Table 60.20 and Table 60.21.<br>[Note 1] Number of output pixels<br>When YCbCr4:2:2 is specified through WRFMT, the horizontal size of the output image should be a multiple of 2 pixels. When YCbCr4:2:0 is specified, the vertical and horizontal sizes of the output image should be multiples of 2 pixels. Specify an appropriate data flow of the source RPF → DPR → target WPF so that the size of the image input to the target WPF satisfies the above restrictions. In particular, when the data flow includes a module or a function that modifies (up-scales, down-scales, or clips) the image size, take special care about the module or function settings.<br>[Note 2] Output lines in YCbCr4:2:0<br>In the YCbCr4:2:0 output format, the number of chrominance lines in the vertical direction is one-half the number of luminance lines. For this reason, the WPF outputs only even-numbered chrominance lines (lines 0, 2, 4, 6, ...)<br>[Note 3] Down sampling of CbCr in horizontal direction in YCbCr4:2:0 or YCbCr4:2:2<br>In the YCbCr4:2:2 or YCbCr4:2:0 output format, method of down sample of Cb/Cr in horizontal direction is average of neighbor two pixels. |

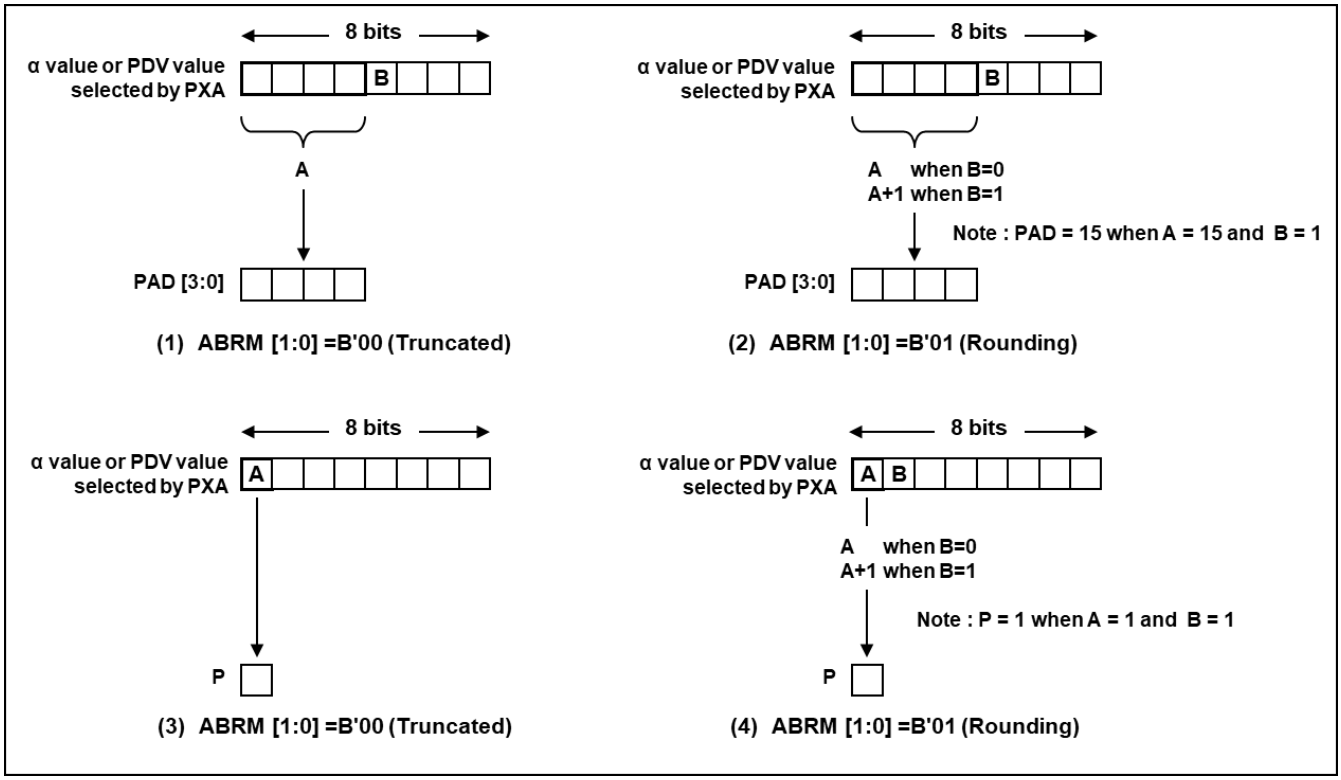



**Figure 60.21 Selection of PAD Value and Reduction of Bit Count through PXA Setting**

**Table 60.20 Packed RGB Formats for WPF Output**

| WRFMT [6:0] | Bit per pixel | Phase | upper row = address / bottom row = bit field | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | n | | | | | | | | n+1 | | | | | | | | n+2 | | | | | | | | n+3 | | | | | | | |
| | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x00 | 8 | – | R0 | R0 | R0 | G0 | G0 | G0 | B0 | B0 | R1 | R1 | R1 | G1 | G1 | G1 | B1 | B1 | R2 | R2 | R2 | G2 | G2 | G2 | B2 | B2 | R3 | R3 | R3 | G3 | G3 | G3 | B3 | B3 |
| 0x01 | 12 | – | 0 | 0 | 0 | 0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | 0 | 0 | 0 | 0 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 |
| 0x02 | | | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | 0 | 0 | 0 | 0 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | 0 | 0 | 0 | 0 |
| 0x03 | – | – | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | |
| 0x04 | 15 | – | 0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | 0 | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 |
| 0x05 | | | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | 0 | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | 0 |
| 0x06 | 16 | – | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 |
| 0x07 | 18 | – | PAD | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x08 | | | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 | 0 | 0 | 0 | 0 | PAD | | | | | | | |
| 0x09 | | | 0 | 0 | 0 | 0 | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | PAD | | | | | | | |
| 0x0A | | | PAD | | | | | | | | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0B | | | PAD | | | | | | | | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | 0 | 0 | G0 | G0 | G0 | G0 | G0 | G0 | 0 | 0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x0C | | | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | 0 | 0 | G0 | G0 | G0 | G0 | G0 | G0 | 0 | 0 | B0 | B0 | B0 | B0 | B0 | B0 | PAD | | | | | | | |
| 0x0D | | | PAD | | | | | | | | R0 | R0 | R0 | R0 | R0 | R0 | 0 | 0 | G0 | G0 | G0 | G0 | G0 | G0 | 0 | 0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 |
| 0x0E | | | R0 | R0 | R0 | R0 | R0 | R0 | 0 | 0 | G0 | G0 | G0 | G0 | G0 | G0 | 0 | 0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 | PAD | | | | | | | |
| 0x0F | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 | 0 | 0 | 0 | 0 | R1 | R1 |
| | | 1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | B1 | 0 | 0 | 0 | 0 | 0 | 0 | R2 | R2 | R2 | R2 | R2 | R2 | G2 | G2 | G2 | G2 |
| | | 2 | G2 | G2 | B2 | B2 | B2 | B2 | B2 | B2 | 0 | 0 | 0 | 0 | 0 | 0 | R3 | R3 | R3 | R3 | R3 | R3 | G3 | G3 | G3 | G3 | G3 | G3 | B3 | B3 | B3 | B3 | B3 | B3 |
| 0x10 | | 0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 | 0 | 0 | 0 | 0 | R1 | R1 | R1 | R1 | R1 | R1 | G1 | G1 |
| | | 1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | B1 | 0 | 0 | 0 | 0 | 0 | 0 | R2 | R2 | R2 | R2 | R2 | R2 | G2 | G2 | G2 | G2 | G2 | G2 | B2 | B2 | B2 | B2 |
| | | 2 | B2 | B2 | 0 | 0 | 0 | 0 | 0 | 0 | R3 | R3 | R3 | R3 | R3 | R3 | G3 | G3 | G3 | G3 | G3 | G3 | B3 | B3 | B3 | B3 | B3 | B3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x11 | | 0 | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | 0 | 0 | G0 | G0 | G0 | G0 | G0 | G0 | 0 | 0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 | R1 | R1 | R1 | R1 | R1 | R1 |
| | | 1 | 0 | 0 | G1 | G1 | G1 | G1 | G1 | G1 | 0 | 0 | B1 | B1 | B1 | B1 | B1 | B1 | 0 | 0 | R2 | R2 | R2 | R2 | R2 | R2 | 0 | 0 | G2 | G2 | G2 | G2 | G2 | G2 |
| | | 2 | 0 | 0 | B2 | B2 | B2 | B2 | B2 | B2 | 0 | 0 | R3 | R3 | R3 | R3 | R3 | R3 | 0 | 0 | G3 | G3 | G3 | G3 | G3 | G3 | 0 | 0 | B3 | B3 | B3 | B3 | B3 | B3 |
| 0x12 | | 0 | R0 | R0 | R0 | R0 | R0 | R0 | 0 | 0 | G0 | G0 | G0 | G0 | G0 | G0 | 0 | 0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 | R1 | R1 | R1 | R1 | R1 | R1 | 0 | 0 |
| | | 1 | G1 | G1 | G1 | G1 | G1 | G1 | 0 | 0 | B1 | B1 | B1 | B1 | B1 | B1 | 0 | 0 | R2 | R2 | R2 | R2 | R2 | R2 | 0 | 0 | G2 | G2 | G2 | G2 | G2 | G2 | 0 | 0 |
| | | 2 | B2 | B2 | B2 | B2 | B2 | B2 | 0 | 0 | R3 | R3 | R3 | R3 | R3 | R3 | 0 | 0 | G3 | G3 | G3 | G3 | G3 | G3 | 0 | 0 | B3 | B3 | B3 | B3 | B3 | B3 | 0 | 0 |
| 0x13 | 24 | – | PAD | | | | | | | | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x14 | | | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | PAD | | | | | | | |
| 0x15 | | 0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | R1 | R1 | R1 | R1 | R1 | R1 | R1 | R1 |
| | | 1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | B1 | B1 | B1 | R2 | R2 | R2 | R2 | R2 | R2 | R2 | R2 | G2 | G2 | G2 | G2 | G2 | G2 | G2 | G2 |
| | | 2 | B2 | B2 | B2 | B2 | B2 | B2 | B2 | B2 | R3 | R3 | R3 | R3 | R3 | R3 | R3 | R3 | G3 | G3 | G3 | G3 | G3 | G3 | G3 | G3 | B3 | B3 | B3 | B3 | B3 | B3 | B3 | B3 |
| 0x16 | 18 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x17 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | B0 |
| 0x18 | 24 | 0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | B1 | B1 | B1 | B1 | B1 | B1 | B1 | B1 |
| | | 1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 | R1 | R1 | R1 | R1 | B2 | B2 | B2 | B2 | B2 | B2 | B2 | B2 | G2 | G2 | G2 | G2 | G2 | G2 | G2 | G2 |
| | | 2 | R2 | R2 | R2 | R2 | R2 | R2 | R2 | R2 | B3 | B3 | B3 | B3 | B3 | B3 | B3 | B3 | G3 | G3 | G3 | G3 | G3 | G3 | G3 | G3 | R3 | R3 | R3 | R3 | R3 | R3 | R3 | R3 |
| 0x19 | 12 | – | PAD | | | | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | PAD | | | | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 |
| 0x1A | | – | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | PAD | | | | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | PAD | | | |
| 0x1B | 15 | – | P | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | P | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 |
| 0x1C | | – | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 | P | R1 | R1 | R1 | R1 | R1 | G1 | G1 | G1 | G1 | G1 | B1 | B1 | B1 | B1 | B1 | P |
| 0x1D | 12 | – | PAD | | | | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | PAD | | | | B1 | B1 | B1 | B1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 |
| 0x1E | | – | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | PAD | | | | B1 | B1 | B1 | B1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 | PAD | | | |
| 0x1F | 15 | – | P | B0 | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | R0 | P | B1 | B1 | B1 | B1 | B1 | G1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 | R1 |
| 0x20 | | – | B0 | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | R0 | P | B1 | B1 | B1 | B1 | B1 | G1 | G1 | G1 | G1 | G1 | R1 | R1 | R1 | R1 | R1 | P |
| 0x21 | 18 | 0 | 0 | 0 | B0 | B0 | B0 | B0 | B0 | B0 | 0 | 0 | G0 | G0 | G0 | G0 | G0 | G0 | 0 | 0 | R0 | R0 | R0 | R0 | R0 | R0 | 0 | 0 | B1 | B1 | B1 | B1 | B1 | B1 |
| | | 1 | 0 | 0 | G1 | G1 | G1 | G1 | G1 | G1 | 0 | 0 | R1 | R1 | R1 | R1 | R1 | R1 | 0 | 0 | B2 | B2 | B2 | B2 | B2 | B2 | 0 | 0 | G2 | G2 | G2 | G2 | G2 | G2 |
| | | 2 | 0 | 0 | R2 | R2 | R2 | R2 | R2 | R2 | 0 | 0 | B3 | B3 | B3 | B3 | B3 | B3 | 0 | 0 | G3 | G3 | G3 | G3 | G3 | G3 | 0 | 0 | R3 | R3 | R3 | R3 | R3 | R3 |
| 0x22 | 24 | – | PAD | | | | | | | | B0 | B0 | B0 | B0 | B0 | B0 | B0 | B0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | G0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 | R0 |
| 0x23 | 16 | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R0 | R0 | R0 | R0 | R0 | G0 | G0 | G0 | G0 | G0 | G0 | B0 | B0 | B0 | B0 | B0 |
| 0x24 ～ 0x3F | – | – | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | | Reserved | | | | | | | |

**Table 60.21 Packed YCbCr Formats for WPF Output**

| WRFMT [6:0] | Reference | |
|---|---|---|
| 0x40 | YCbCr4:4:4 semi-planar | Figure 60.10*4 |
| 0x41 | YCbCr4:2:2 semi-planar (NV16, NV61*1) | |
| 0x42 | YCbCr4:2:0 semi-planar (NV12, NV21*1) | |
| 0x43 to 0x45 | Reserved | — |
| 0x46 | YCbCr4:4:4 interleaved | Figure 60.11*4 |
| 0x47 | YCbCr4:2:2 interleaved type 0 (UYVY, YUY2*2, YVYU*3) | |
| 0x48 | YCbCr4:2:2 interleaved type 1 | |
| 0x49 | YCbCr4:2:0 interleaved *5 | |
| 0x4a | YCbCr4:4:4 planar | Figure 60.12*4 |
| 0x4b | YCbCr4:2:2 planar (YV16) | |
| 0x4c | YCbCr4:2:0 planar (YV12, I420) | |
| 0x4d to 0x7f | Reserved | — |

*1 When the output format is NV61 or NV21, set SPUVS (bit 14) to 1.

*2 When the output format is YUY2, set SPYCS (bit 15) to 1.

*3 When the output format is YVYU, set SPUVS (bit 14) to 1 and SPYCS (bit 15) to 1.

*4 Figure 60.13 shows the definition of memory address for each pixel in Figure 60.10 to Figure 60.12.

*5 Each line of plane is written twice, so byte/pixel of YCbCr420ITL is 3 byte/pixel (same as YCbCr444ITL).

For details of each YCbCr format, refer to Figure 60.10, Figure 60.11, Figure 60.12, and Figure 60.13. In these figures, registers for the RPF are indicated; read them as registers for the WPF as follows.

(RPF registers in the figures) ® (Corresponding WPF registers)

VI6_RPFn_SRCM_ADDR_Y.SRCM_ADDR_Y → VI6_WPF0_DSTM_ADDR_Y.DSTM_ADDR_Y

VI6_RPFn_SRCM_ADDR_C0.SRCM_ADDR_C0 → VI6_WPF0_DSTM_ADDR_C0.DSTM_ADDR_C0

VI6_RPFn_SRCM_ADDR_C1.SRCM_ADDR_C1 → VI6_WPF0_DSTM_ADDR_C1.DSTM_ADDR_C1

### 60.2.7.5 WPF0 Data Swapping Registers (VI6_WPF0_DSWAP)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | P_LLS | P_LWS | P_WDS | P_BTS |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 3 | P_LLS | 0 | R/W | WPF Output Data Swapping in LONG LWORD Units<br>The effect of this bit setting is the same as data swapping in the RPF; refer to Table 60.14.<br>0: Data swapping in LONG LWORD (64-bit) units is disabled<br>1: Data swapping in LONG LWORD (64-bit) units is enabled |
| 2 | P_LWS | 0 | R/W | WPF Output Data Swapping in long word Units<br>The effect of this bit setting is the same as data swapping in the RPF; refer to Table 60.14.<br>0: Data swapping in long word (32-bit) units is disabled<br>1: Data swapping in long word (32-bit) units is enabled |
| 1 | P_WDS | 0 | R/W | WPF Output Data Swapping in Word Units<br>The effect of this bit setting is the same as data swapping in the RPF; refer to Table 60.14.<br>0: Data swapping in word (16-bit) units is disabled<br>1: Data swapping in word (16-bit) units is enabled |
| 0 | P_BTS | 0 | R/W | WPF Output Data Swapping in Byte Units<br>The effect of this bit setting is the same as data swapping in the RPF; refer to Table 60.14.<br>0: Data swapping in byte (8-bit) units is disabled<br>1: Data swapping in byte (8-bit) units is enabled |

Table 60.14 shows the data order before and after swapping according to the long long word, long word, word, and byte swapping settings.

When data order in memory for each format is the same as in Table 60.20 for RGB format Figure 60.12 for YCbCr format, set 4'b1111 to {*_LLS, *_LWS, *_WDS, *_BTS}. If data order is not the same as the definition, change data order within 16byte unit by these bits as shown in Table 60.14.

### 60.2.7.6 WPF0 Rounding Control Registers (VI6_WPF0_RNDCTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | CBRM | – | – | ABRM [1:0] | | ATHRESH [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | CLMD [1:0] | | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29,<br>27, 26,<br>15,14,<br>11 to 0 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 | CBRM | 0 | R/W | Bit Count Reduction Method Selection for Data Storage in Packed RGB<br>This bit specifies the method for reducing the number of bits when data is stored in the bit fields indicated as R, G, and B in Table 60.20 and the target bit fields are not 8 bits.<br>0: Bit count conversion: The lower-order bits are truncated<br>1: Bit count conversion: Rounding (rounding off) |
| 25, 24 | ABRM<br>[1:0] | All 0 | R/W | Bit Count Reduction Method Selection for Data Storage in PAD<br>These bits specify the method for reducing the number of bits when the data selected through VI6_WPF0_OUTFMT.PXA is stored in the bit fields indicated as PAD or P in Table 60.20 and the target bit field is four bits or one bit.<br>A value of b'10 can be specified only when the packed RGB format specified through VI6_WPF0_OUTFMT.WRFMT includes a 1-bit P field. In this case, when the data selected through VI6_WPF0_OUTFMT.PXA is greater than the ATHRESH value, 1 is stored in the P field; when the selected data is not greater than the ATHRESH value, 0 is stored.<br>b'00: Bit count conversion: The lower-order bits are truncated<br>b'01: Bit count conversion: Rounding (rounding off)<br>b'10: Bit count conversion: Comparison with the threshold value (this setting is allowed only when the storage field is one bit)<br>b'11: Setting prohibited |
| 23 to 16 | ATHRESH<br>[7:0] | All 0 | R/W | Threshold for Conversion to 1-Bit α Data<br>These bits specify the threshold value used for conversion from 8-bit α data to one bit when the ABRM bits are set to b'10. When the 8-bit α value before bit count reduction is equal to or smaller than the ATHRESH value, 0 is stored as the reduced 1-bit α data. In other cases, 1 is stored as the 1-bit α data.<br>A value from 0 to 255 can be specified. |
| 13, 12 | CLMD<br>[1:0] | All 0 | R/W | Color Data Clipping<br>These bits specify the method for clipping the YCbCr color data output from the WPF. When RGB color data is output from the WPF, specify 0 in these bits.<br>b'00: Output value is not clipped (0-255)<br>b'01: Output value is clipped:<br>YCbCr mode 1 (16-235 (Y), 16-240 (Cb/Cr))<br>b'10: Output value is clipped: YCbCr mode 2 (Y/Cb/Cr = 1-254)<br>b'11: Setting prohibited |

### 60.2.7.7 WPF0 Destination Y Plane Memory Stride Registers (VI6_WPF0_DSTM_STRIDE_Y)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PICT_STRD_Y [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 0 | PICT_STRD_Y [15:0] | 0 | R/W | Memory Stride of Destination Picture Y/RGB Plane<br>These bits specify in 1-byte units the memory stride of the destination picture in the external memory to be written to by the WPF0 as shown in Figure 60.22.<br>A value from H'0000 to H'FFFF can be specified. |

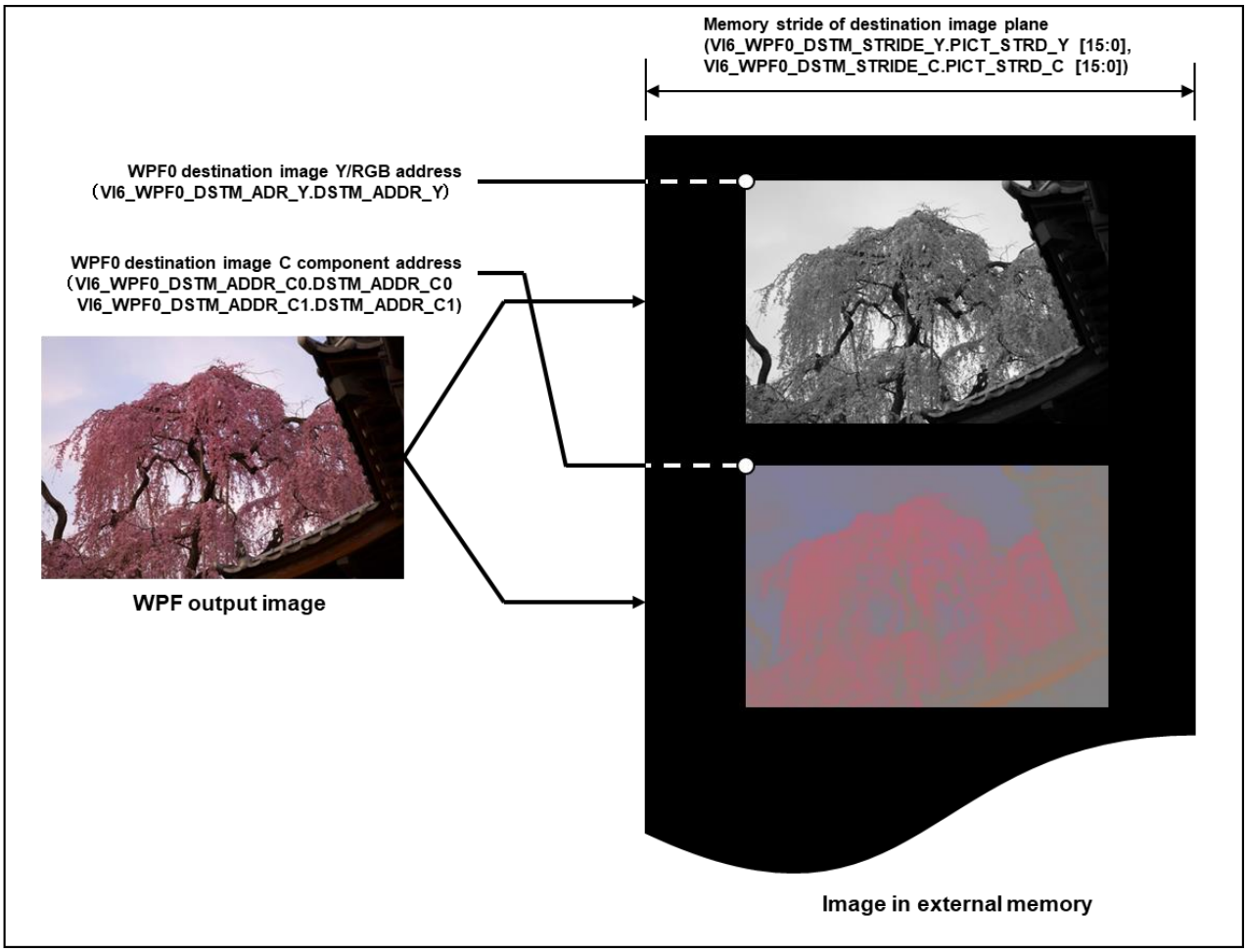



**Figure 60.22 Writing Image Data to Destination Area in WPF0**

### 60.2.7.8 WPF0 Destination C Plane Memory Stride Registers (VI6_WPF0_DSTM_STRIDE_C)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PICT_STRD_C [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 15 to 0 | PICT_STRD_C [15:0] | 0 | R/W | Memory Stride of Destination Picture C Plane<br>These bits specify in 1-byte units the memory stride for the C plane of the destination picture in the external memory to be written to by the WPF0 as shown in Figure 60.22. When the WPF0 outputs images in an RGB format, this setting is not used. When the WPF0 outputs images in YCbCr planar format, this setting is applied to both the Cb and Cr planes.<br>A value from 0x0000 to 0xFFFF can be specified. |

### 60.2.7.9 WPF0 Destination Y/RGB Address Registers (VI6_WPF0_DSTM_ADDR_Y)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTM_ADDR_Y [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTM_ADDR_Y [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSTM_ ADDR_Y [31:0] | 0 | R/W | Destination Image Y/RGB Plane Storing Address<br>These bits specify in 1-byte units the address for storing the destination-image Y plane or packed RGB plane to be written to by the WPF0 in the method described later.<br>A value from H'0000 0000 to H'FFFF FFFF can be specified. |

### 60.2.7.10 WPF0 Destination Chroma Address Registers 0 (VI6_WPF0_DSTM_ADDR_C0)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTM_ADDR_C0 [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTM_ADDR_C0 [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial | R/W | Description |
|---|---|---|---|---|

| | | Value | | |
|---|---|---|---|---|
| 31 to 0 | DSTM_ ADDR_C0 [31:0] | 0 | R/W | Destination Image C Plane Storing Address 0<br>These bits specify in 1-byte units the address for storing the destination-image C plane to be written to by the WPF0. Refer to the description of VI6_WPF0_DSTM_ADDR_Y for settings.<br>Here, the C plane indicates the combined CbCr plane when a semi-planar format is selected from the packed YCbCr formats shown in Table 60.21 or the Cb plane when a planar format is selected. When an interleaved format is selected or the output is in an RGB format, this setting is not used.<br>A value from H'0000 0000 to H'FFFF FFFF can be specified. |

### 60.2.7.11 WPF0 Destination Chroma Address Registers 1 (VI6_WPF0_DSTM_ADDR_C1)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTM_ADDR_C1 [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTM_ADDR_C1 [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSTM_ ADDR_C1 [31:0] | 0 | R/W | Destination Image C Plane Storing Address 1<br>These bits specify in 1-byte units the address for storing the Cr plane when the WPF0 outputs images to the external memory in YCbCr planar format shown in Table 60.21. Refer to the description of VI6_WPF0_DSTM_ADDR_Y for settings.<br>This setting is not used when the WPF outputs in YCbCr format that is not a planar format or in an RGB format.<br>A value from H'0000 0000 to H'FFFF FFFF can be specified. |

### 60.2.7.12 WPF0 LIF Write Back Control Registers (VI6_WPF0_WRBCK_CTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | – | | | | | | | | – | – | – | – | – | – | – | WBMD |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | WBMD | 0 | R/W | Display Data Write Back Control<br>This bit is used for selecting the write back mode when the value of VI6_LIF0_CTRL.LIF_EN bit is set to 1.<br>Relation between register setting and destination of output image is shown in Table 60.22.<br>0 : Write Back Disabled<br>1 : Write Back Enabled |

**Table 60.22 Register settings and destination of output image**

| Register setting | | Destination | |
|---|---|---|---|
| VI6_LIF0_CTRL .LIF_EN | VI6_WPF0_WRBCK_CTRL. WBMD | Output image to display | Output image to memory |
| 0 | 0 | | ✓ |
| 0 | 1 | Forbidden | |
| 1 | 0 | ✓ | |
| 1 | 1 | ✓ | ✓ |

### 60.2.8 DPR Control Registers

#### 60.2.8.1 Concept of DPR Settings

Figure 60.23 shows node value of all sub modules implemented in all VSPs.

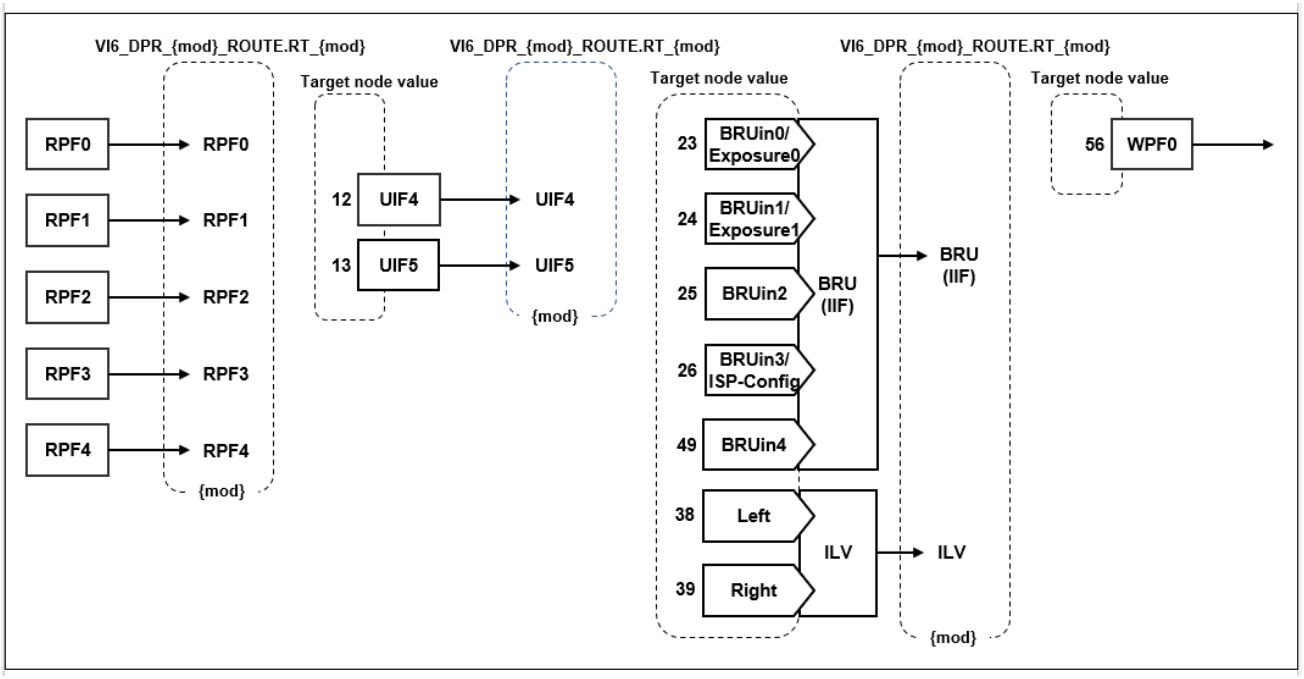



Figure 60.23 Node Register Names and Target Node Values on Data Path Router

In the VSP2 internal data path, the order of processes can be specified as desired. The module for performing each process has a unique node value.

For DPR settings, all of the following restrictions should be observed. If any of them is violated, correct operation will not be guaranteed.

1. Specify 63 for the output node values of all RPFs and processing modules that are not used in the DPR. Here, make sure that no module is connected to a module for which 63 is specified as the node value. Note: VI6_DPR_RPFn_ROUTE (n = 4) corresponding to RPFn is not implemented in VSP which doesn't have RPFn (n = 4). Note: VI6_DPR_{mod}_ROUTE corresponding to {mod} is not implemented in VSP which doesn't have {mod}. See Figure 60.1 to Figure 60.2 whether each VSP has {mod} or not.
2. When specifying a value other than 63 for an output node value in the DPR, make sure that valid inputs (RPF0 to RPF4 or virtual RPF) and a target WPF are determined.
3. Only one module can be connected to each module; specifying the same target node value for two or more modules is prohibited.
4. Desired modules can be connected between each RPF and BRU input port, but all RPFs specified as the sources for a BRU input port should have the same target WPF. Same manner must be applied for ILV.
5. Make appropriate routing or RPF register settings so that the color space formats (RGB/YCbCr) for all BRU input ports are the same. Same manner must be applied for ILV.
6. Do not connect the output of any module as the input to the same module (in the BRU case, any input port) even when there is another module between the output and input (creating a loop is prohibited).

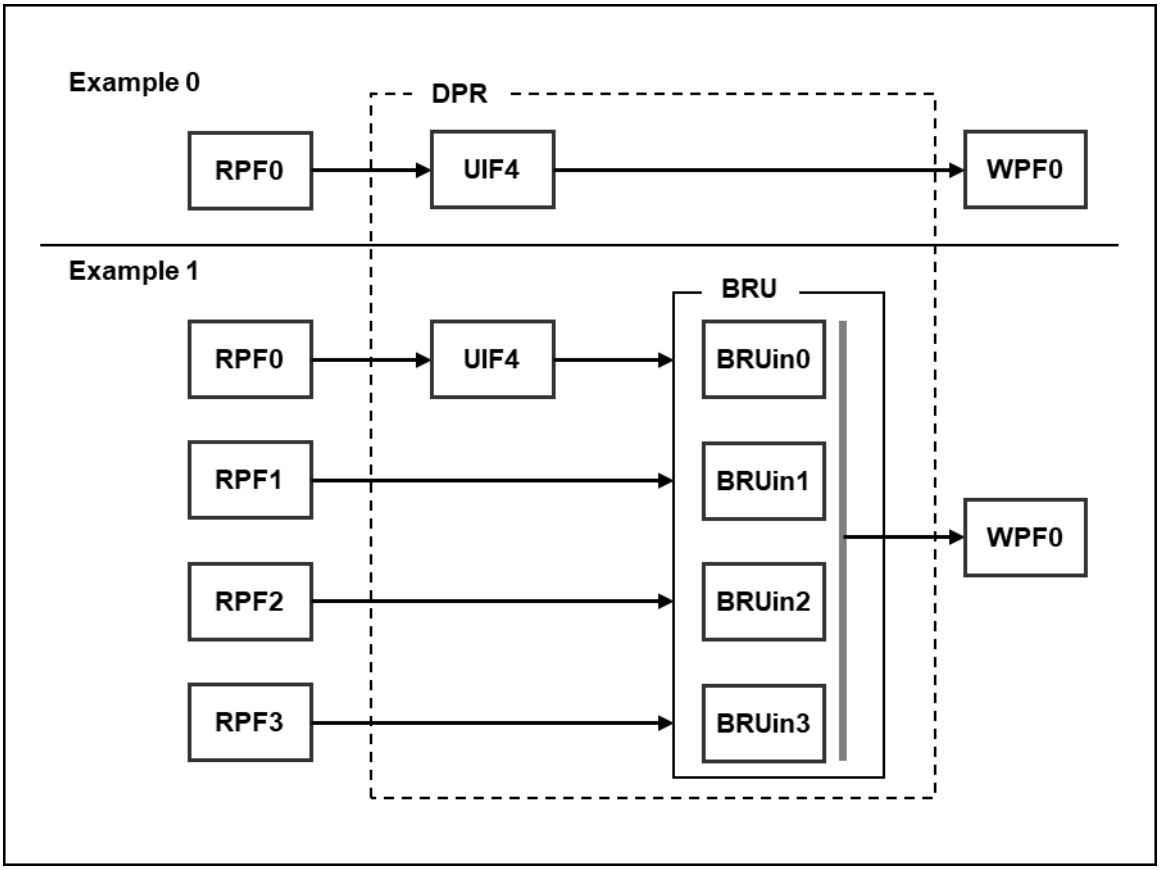



**Figure 60.24 Examples of Internal Data Path Routing**

Figure 60.24 shows examples of internal data path routing. The VI6_ DPR_*_ROUTE settings for these examples are shown in Table 60.23. The bit fields for the modules that are not used in the examples should be set to 63.

**Table 60.23 VI6_DPR_*_ROUTE Register Settings in Connection Examples Shown in Figure 60.25**

| | Register Name | Setting |
|---|---|---|
| Example 0 | VI6_DPR_RPF0_ROUTE.RT_RPF0 | 12 (To UIF4) |
| | VI6_DPR_RPF1_ROUTE.RT_RPF1 | 63 (UNUSED) |
| | VI6_DPR_RPF2_ROUTE.RT_RPF2 | 63 (UNUSED) |
| | VI6_DPR_RPF3_ROUTE.RT_RPF3 | 63 (UNUSED) |
| | VI6_DPR_RPF4_ROUTE.RT_RPF4 | 63 (UNUSED) |
| | VI6_DPR_UIF4_ROUTE.RT | 56 (To WPF0) |
| | VI6_DPR_ILV _ROUTE.RT | 63 (UNUSED) |
| | VI6_DPR_BRU_ROUTE.RT | 63 (UNUSED) |
| Example 1 | VI6_DPR_RPF0_ROUTE.RT_RPF0 | 12 (To UIF4) |
| | VI6_DPR_RPF1_ROUTE.RT_RPF1 | 24 (To BRUin1) |
| | VI6_DPR_RPF2_ROUTE.RT_RPF2 | 25 (To BRUin2) |
| | VI6_DPR_RPF3_ROUTE.RT_RPF3 | 26 (To BRUin3) |
| | VI6_DPR_RPF4_ROUTE.RT_RPF4 | 63 (UNUSED) |
| | VI6_DPR_UIF4_ROUTE.RT | 23 (To BRUin0) |

| | | |
|---|---|---|
| | VI6_DPR_BRU_ROUTE.RT | 56 (To WPF0) |

Although both examples shown in Figure 60.24 include operation in image processing modules, connect the RPF to the WPF directly when only image format conversion or packed format conversion is required.

### 60.2.8.2 RPFn Routing Register (VI6_DPR_RPFn_ROUTE : n=0, 1, 2, 3, 4)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | RT_RPFn [5:0] | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | RT_RPFn [5:0] | All 0 | R/W | RPFn Target Node Value<br>These bits specify the target node value for RPFn. When using RPFn, refer to Figure 60.23 for settings.<br>When RPFn is not started through the VI6_WPF0_SRCRPF setting, specify 63. |

### 60.2.8.3 WPF0 Timing Control Register (VI6_DPR_WPF0_FPORCH)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | FP_WPF0 [5:0] | | | | | | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| R/W : | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14,<br>7 to 0 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 8 | FP_WPF0<br>[5:0] | All 0 | R/W | WPF0 Internal Operation Timing Setting<br>Specify 5. |

#### 60.2.8.4 {mod} Routing Register (VI6_DPR_{mod}_ROUTE : {mod}=ILV,UIFn (n=4, 5) ,BRU)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | IIFSEL | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | FP [5:0] | | | | | | – | – | RT [5:0] | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29, 27 to 14, 7, 6 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 | IIFSEL | 0 | R/W | This bit is valid only for VI6_DPR_BRU_ROUTE.<br>0: BRU is selected<br>1: IIF is selected<br>Set 0 to this bit for VSPD to select BRU. Set 1 to this bit for VSPX to select IIF.<br>BRU and IIF shares data path route. When IIF is selected (VSPX), the VSP outputs data to the connected streaming IP. When BRU is selected (VSPD), the VSP outputs data to external memory or DU display according to setting of VI6_LIF0_CTRL. |
| 13 to 8 | FP [5:0] | All 0 | R/W | {mod} Internal Operation Timing Setting<br>Specify 0. |
| 5 to 0 | RT [5:0] | All 0 | R/W | {mod}Target Node Value<br>These bits specify the target node value for the {mod}. When using the {mod}, refer to Figure 60.23 for settings. When not using the {mod}, specify 63. |

### 60.2.9 BRU Control Registers

#### 60.2.9.1 BRU Input Control Register (VI6_BRU_INCTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | NRM | ODE4 | DITH 4 [2:0] | | | – | – | – | D4ON | D3ON | D2ON | D1ON | D0ON |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ODE3 | DITH3 [2:0] | | | ODE2 | DITH2 [2:0] | | | ODE1 | DITH1 [2:0] | | | ODE0 | DITH0 [2:0] | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29,<br>23 to 21 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 | NRM | 0 | R/W | Color Data Normalization<br>Enables or disables division by the α value of the color data in BRU blending operation. This is used when converting the RGB color data format to which the α value is multiplied (pre-multiplied color) into the RGB color data format to which the α value is not multiplied (non pre-multiplied color). Do not use this for the YCbCr format.<br>0: Divider (DIV unit in Figure 60.25) does not divide the color value by α<br>1: Divider (DIV unit in Figure 60.25)) divides the color value by α |
| 27 | ODE4 | 0 | R/W | Ordered Dither (mode A) of CH4 Input to BRU Enable/Disable<br>0: Ordered dither (mode A) of BRUin4 is disabled.<br>1: Ordered dither (mode A) of BRUin4 is enabled.<br>Ordered dither is available only for 18bpp. So, when ODE4 bit is equal to 1, set DITH4 = 1.<br>When ODE4 bit is equal to 0, BRUin4 dither method is specified by D4ON in the register<br>Ordered dither (mode A) is recommended rather than ordered dither (mode B) in case of 18bpp. |
| 26 to 24 | DITH4[2:0] | 0 | R/W | Dithering of CH4 Input to BRU<br>These bits specify how to perform dithering of the CH4 input to the BRU. The setting method is the same as that for the DITH3 bits. Read the description of the DITH3 bits with BRUin4 and D4ON replacing BRUin4 and D4ON, respectively. |
| 20 | D4ON | 0 | R/W | Ordered dither (mode B) Enable of BRU Input 4<br>Enables or disables dithering (color reduction) of BRU input 4 (BRUin4 in Figure 60.25)).<br>0: Dithering (mode B) of BRUin4 is disabled<br>1: Dithering (mode B) of BRUin4 is enabled<br>When ODE4 in this bit is set to 1, set 0 to this bit. |
| 19 | D3ON | 0 | R/W | Ordered dither (mode B) Enable of BRU Input 3<br>Enables or disables dithering (color reduction) of BRU input 3 (BRUin3 in Figure 60.25)).<br>0: Dithering (mode B) of BRUin3 is disabled<br>1: Dithering (mode B) of BRUin3 is enabled<br>When ODE3 in this bit is set to 1, set 0 to this bit. |
| 18 | D2ON | 0 | R/W | Ordered dither (mode B) Enable of BRU Input 2 |

| | | | | |
|---|---|---|---|---|
| | | | | Enables or disables dithering (color reduction) of BRU input 2 (BRUin2 in Figure 60.25).<br>0: Dithering (mode B) of BRUin2 is disabled<br>1: Dithering (mode B) of BRUin2 is enabled<br>When ODE2 in this bit is set to 1, set 0 to this bit. |
| 17 | D1ON | 0 | R/W | Ordered dither (mode B) Enable of BRU Input 1<br>Enables or disables dithering (color reduction) of BRU input 1 (BRUin1 in Figure 60.25).<br>0: Dithering (mode B) of BRUin1 is disabled<br>1: Dithering (mode B) of BRUin1 is enabled<br>When ODE1 in this bit is set to 1, set 0 to this bit. |
| 16 | D0ON | 0 | R/W | Ordered dither (mode B) Enable of BRU Input 0<br>Enables or disables dithering (color reduction) of BRU input 0 (BRUin0 in Figure 60.25).<br>0: Dithering (mode B) of BRUin0 is disabled<br>1: Dithering (mode B) of BRUin0 is enabled<br>When ODE0 in this bit is set to 1, set 0 to this bit. |
| 15 | ODE3 | 0 | R/W | Ordered Dither (mode A) of CH3 Input to BRU Enable/Disable<br>0: Ordered dither (mode A) of BRUin3 is disabled.<br>1: Ordered dither (mode A) of BRUin3 is enabled.<br>Ordered dither is available only for 18bpp. So, when ODE3 bit is equal to 1, set DITH3 = 1.<br>When ODE3 bit is equal to 0, BRUin3 dither method is specified by D3ON in the register<br>Ordered dither (mode A) is recommended rather than ordered dither (mode B) in case of 18bpp. |
| 14 to 12 | DITH3 [2:0] | All 0 | R/W | Dithering of CH3 Input to BRU<br>These bits specify the number of colors for pixels after dithering (color reduction) when dithering (color reduction) for pixel information is enabled through the D3ON bit. When dithering (color reduction) for pixel information is disabled, specify 0 in these bits.<br>0: Dithering of BRUin3 input image is disabled<br>1: Dithering of BRUin3 input image at 18 bpp (RGB666: 260000 colors)<br>2: Dithering of BRUin3 input image at 16 bpp (RGB565: 65535 colors)<br>3: Dithering of BRUin3 input image at 15 bpp (RGB555: 32768 colors)<br>4: Dithering of BRUin3 input image at 12 bpp (RGB444: 4096 colors)<br>5: Dithering of BRUin3 input image at 8 bpp (RGB332: 256 colors)<br>6, 7: Setting prohibited |
| 11 | ODE2 | 0 | R/W | Ordered Dither (mode A) of CH2 Input to BRU Enable/Disable<br>0: Ordered dither (mode A) of BRUin2 is disabled.<br>1: Ordered dither (mode A) of BRUin2 is enabled.<br>Ordered dither is available only for 18bpp. So, when ODE2 bit is equal to 1, set DITH2 = 1.<br>When ODE2 bit is equal to 0, BRUin2 dither method is specified by D2ON in the register |

| | | | | |
|---|---|---|---|---|
| | | | | Ordered dither (mode A) is recommended rather than ordered dither (mode B) in case of 18bpp. |
| 10 to 8 | DITH2 [2:0] | All 0 | R/W | Dithering of CH2 Input to BRU<br>These bits specify how to perform dithering of the CH2 input to the BRU. The setting method is the same as that for the DITH3 bits. Read the description of the DITH3 bits with BRUin2 and D2ON replacing BRUin3 and D3ON, respectively. |
| 7 | ODE1 | 0 | R/W | Ordered Dither (mode A) of CH1 Input to BRU Enable/Disable<br>0: Ordered dither (mode A) of BRUin1 is disabled.<br>1: Ordered dither (mode A) of BRUin1 is enabled.<br>Ordered dither is available only for 18bpp. So, when ODE1 bit is equal to 1, set DITH1 = 1.<br>When ODE1 bit is equal to 0, BRUin1 dither method is specified by D1ON in the register<br>Ordered dither (mode A) is recommended rather than ordered dither (mode B) in case of 18bpp. |
| 6 to 4 | DITH1 [2:0] | All 0 | R/W | Dithering of CH1 Input to BRU<br>These bits specify how to perform dithering of the CH1 input to the BRU. The setting method is the same as that for the DITH3 bits. Read the description of the DITH3 bits with BRUin1 and D1ON replacing BRUin3 and D3ON, respectively. |
| 3 | ODE0 | 0 | R/W | Ordered Dither (mode A) of CH0 Input to BRU Enable/Disable<br>0: Ordered dither (mode A) of BRUin0 is disabled.<br>1: Ordered dither (mode A) of BRUin0 is enabled.<br>Ordered dither is available only for 18bpp. So, when ODE0 bit is equal to 1, set DITH0 = 1.<br>When ODE0 bit is equal to 0, BRUin0 dither method is specified by D0ON in the register<br>Ordered dither (mode A) is recommended rather than ordered dither (mode B) in case of 18bpp. |
| 2 to 0 | DITH0 [2:0] | All 0 | R/W | Dithering of CH0 Input to BRU<br>These bits specify how to perform dithering of the CH0 input to the BRU. The setting method is the same as that for the DITH3 bits. Read the description of the DITH3 bits with BRUin0 and D0ON replacing BRUin3 and D3ON, respectively. |

Figure 60.25 shows the configuration of the BRU. For the BRU inputs, there are five inputs from the DPR and one internal input as a virtual RPF. BRUin0 to BRUin4 are input ports that have the target node values shown in Figure 60.23, and they can be connected to any module on the DPR. The same color space (YCbCr or RGB) has to be used for the five inputs from the DPR to the BRU.

The virtual RPF inside the BRU is an input unit not connected to the DPR. It is called the "virtual RPF" because it outputs images internally created by the BRU. Starting of the virtual RPF is controlled by VI6_WPF0_SRCRPF.VIR_ACT, and the single-color data created at the virtual RPF can be used for blending or raster operation (ROP) with data from the other input units BRUin0 to BRUin4. The color space for the single color to be set for the virtual RPF needs to match the color space of the five inputs from the DPR to the BRU. For this setting method, see section 60.2.9.4.

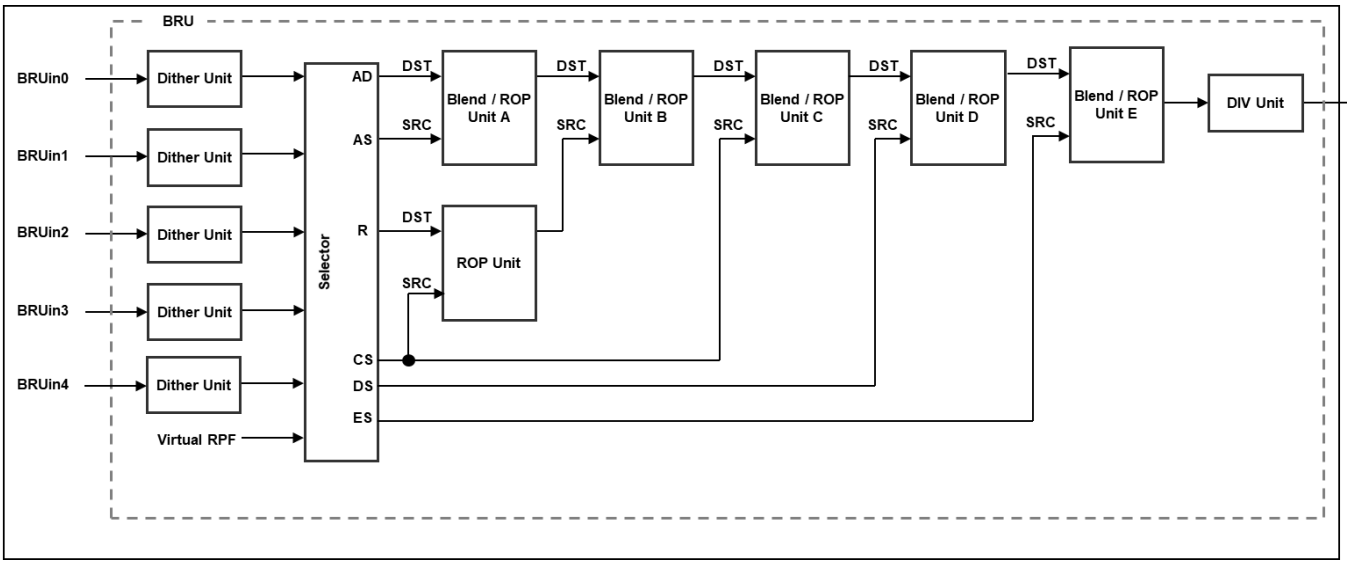



**Figure 60.25 BRU Configuration**

The selector in Figure 60.25 is used to select the SRC and DST inputs to blending/ROP units A to E and the ROP unit from BRUin0 to BRUin4 which are inputs from the DPR and the virtual RPF. The SRC and DST input sources for blending/ROP units A to E and the ROP unit are either uniquely determined based on the configuration shown in Figure 60.25 (Table 60.24) or selected as desired by registers. The input sources that can be arbitrarily selected by registers are AD, AS, R, CS, DS and ES, which correspond to the registers shown in Table 60.25.

**Table 60.24 SRC and DST with Unique Input Sources**

| Input | Input Source | Register Bits |
|---|---|---|
| Blending/ROP unit B - DST | Output from blending/ROP unit A | – |
| Blending/ROP unit B - SRC | Output from ROP unit | – |
| Blending/ROP unit C - DST | Output from blending/ROP unit B | – |
| Blending/ROP unit D - DST | Output from blending/ROP unit C | – |
| Blending/ROP unit E - DST | Output from blending/ROP unit D | – |
| ROP unit - SRC | CS which is a selector output | VI6_BRUC_CTRL.SRCSEL |

**Table 60.25 Correspondence between Selector Output Destinations and Register Bits**

| Selector Output | Output Destination | Register Bits |
|---|---|---|
| AD | Blending/ROP unit A - DST | VI6_BRUA_CTRL.DSTSEL |
| AS | Blending/ROP unit A - SRC | VI6_BRUA_CTRL.SRCSEL |
| R | ROP unit - DST | VI6_BRU_ROP.DSTSEL |
| CS | Blending/ROP unit C - SRC | VI6_BRUC_CTRL.SRCSEL |
| DS | Blending/ROP unit D - SRC | VI6_BRUD_CTRL.SRCSEL |
| ES | Blending/ROP unit E - SRC | VI6_BRUE_CTRL.SRCSEL |

### 60.2.9.2 Size Register of BRU Input Virtual RPF (VI6_BRU_VIRRPF_SIZE)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | VIR_HSIZE [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | VIR_VSIZE [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29,<br>15 to 13 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 16 | VIR_HSIZE [12:0] | All 0 | R/W | Virtual RPF Horizontal Size<br>These bits set the horizontal size of an image from the virtual RPF shown in Figure 60.25. A value from 1 to 8190 can be specified. |
| 12 to 0 | VIR_VSIZE [12:0] | All 0 | R/W | Virtual RPF Vertical Size<br>These bits set the vertical size of an image from the virtual RPF shown in Figure 60.25. A value from 1 to 8190 can be specified. |

The virtual RPF has only a function to output a fixed α value and a fixed pixel value. The virtual RPF can internally create a single-color image without accessing external memory via the MAU. Same as images from the other BRU input ports, a sublayer can be blended on an image created in this manner with the image used as the background (master layer). In turn, when using the image as a sublayer, it can be drawn on the master layer as a window.

Note that the virtual RPF is fixed to input port 5 of the BRU, and the data path route cannot be changed by the DPR.

### 60.2.9.3 Display Location Register of BRU Input Virtual RPF (VI6_BRU_VIRRPF_LOC)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | HCOORD [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | VCOORD [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29,<br>15 to 13 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 16 | HCOORD [12:0] | All 0 | R/W | Horizontal Coordinate of Virtual RPF Location on Master Layer<br>These bits specify the horizontal coordinate of where to locate the left-edge pixel of the virtual RPF's layer, with the left-edge pixel of the master layer set at coordinate 0. This setting should be made in pixel units. A value from 0 to 8189 can be specified.<br>When the virtual RPF is specified as the master layer by VI6_WPF0_SRCRPF.VIR_ACT, set these bits to 0. |
| 12 to 0 | VCOORD [12:0] | All 0 | R/W | Vertical Coordinate of Virtual RPF Location on Master Layer<br>These bits specify the vertical coordinate of where to locate the top-edge pixel of the virtual RPF's layer, with the top-edge pixel of the master layer set at coordinate 0. This setting should be made in pixel units. A value from 0 to 8189 can be specified.<br>When the virtual RPF is specified as the master layer by VI6_WPF0_SRCRPF.VIR_ACT, set these bits to 0. |

### 60.2.9.4 Color Information Register of BRU Input Virtual RPF (VI6_BRU_VIRRPF_COL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COL_A [7:0] | | | | | | | | COL_RCR [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COL_GY [7:0] | | | | | | | | COL_BCB [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | COL_A [7:0] | All 0 | R/W | Fixed α of Virtual RPF<br>These bits set the fixed α value when the virtual RPF creates a virtual plane. A value from 0 to 255 can be specified. |
| 23 to 16 | COL_RCR [7:0] | All 0 | R/W | Fixed R/Cr of Virtual RPF<br>These bits set the fixed R/Cr value when the virtual RPF creates a virtual plane. A value from 0 to 255 can be specified. |
| 15 to 8 | COL_GY [7:0] | All 0 | R/W | Fixed G/Y of Virtual RPF<br>These bits set the fixed G/Y value when the virtual RPF creates a virtual plane. A value from 0 to 255 can be specified. |
| 7 to 0 | COL_BCB [7:0] | All 0 | R/W | Fixed B/Cb of Virtual RPF<br>These bits set the fixed B/Cb value when the virtual RPF creates a virtual plane. A value from 0 to 255 can be specified. |

The transparency information and color information of the single color that is created by the virtual RPF are set in the bits of this register. As described earlier, the color information is set for the YCbCr or RGB color space. The color space to be set in this register depends on the register settings of the environment and other modules to which the BRU is connected by the DPR. Two cases can be considered. Since the α value (COL_A) is transparency information and irrelevant to the concept of color space, the same setting is made for either the YCbCr or RGB color space.

(Case 1: When an input other than the virtual RPF is used)

When the source RPF is connected to any one of the BRU input ports (BRUin0 to BRUin4) other than the virtual RPF and valid data is being supplied, the same color space data as the color space for the BRU inputs should be set in this register as the color space for the virtual RPF's color information. This is based on the restriction of "all BRU inputs must have the same color space", as described in section 60.2.8.1 or 60.2.9.1.

(Case 2: When only the virtual RPF is used)

When only the virtual RPF is used as the source RPF of WPF0, RPFn is not connected to the BRU, as shown in Figure 60.26. Thus, there is no color space for another input that the color space for the virtual RPF has to follow, as in case 1. This means that the color space of the data output by the BRU is determined by the WPF setting.

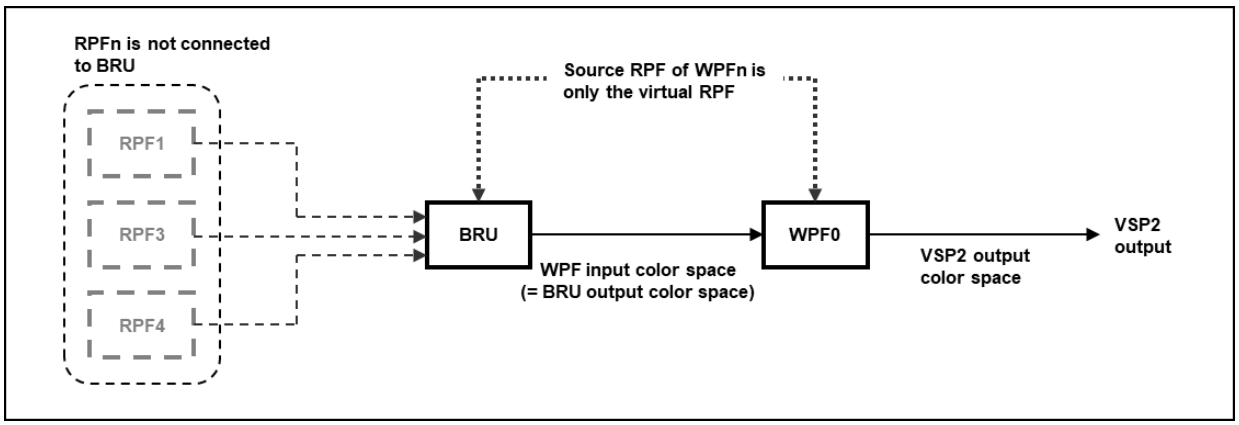



**Figure 60.26 Relationship between DPR Connection and Color Space When Only Virtual RPF is Used**

Figure 60.26, the output color space of the VSP2 (= output color space of WPF0) is determined by VI6_WPF0_OUTFMT.WRFMT. When bit 6 in VI6_WPF0_OUTFMT.WRFMT (WRFMT [6]) is 0, the color space is RGB, while when it is 1, the color space is YCbCr. Next, the WPF input color space (= BRU output color space) is determined by the relationship between the WPF output color space and VI6_WPF0_OUTFMT.CSC. When VI6_WPF0_OUTFMT.CSC is 0, the WPF output color space and WPF input color space (= BRU output color space) are the same. When VI6_WPF0_OUTFMT.CSC is 1, the WPF output color space and WPF input color space (= BRU output color space) are the opposite. This relationship is summarized in Table 60.26.

The color space for the virtual RPF's color information should be set in this register according to the "BRU output color space" shown in Table 60.26.

**Table 60.26 Relationship between WPF Output Color Space and BRU Output Color Space**

| VI6_WPF0_OUTFMT Register Bit Settings | | BRU Output Color Space (= WPF Input Color Space) |
|---|---|---|
| Bit 6 in WRFMT | CSC | |
| 0 (WPF output is RGB) | 0 (YCbCr → RGB conversion is disabled) | RGB |
| 0 (WPF output is RGB) | 1 (YCbCr → RGB conversion is enabled) | YCbCr |
| 1 (WPF output is YCbCr) | 0 (RGB → YCbCr conversion is disabled) | YCbCr |
| 1 (WPF output is YCbCr) | 1 (RGB → YCbCr conversion is enabled) | RGB |

### 60.2.9.5 BRU Control Registers (VI6_BRUm_CTRL: M=A, B, C, D, E)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBC | – | – | – | – | – | – | – | – | DSTSEL [2:0] | | | – | SRCSEL [2:0] | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | CROP [3:0] | | | | AROP [3:0] | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RBC | 0 | R/W | Operation Type of Blending/ROP Unit m (m = A, B, C, D, E)<br>Specifies the operation type for blending/ROP unit m (m = A, B, C, D, E) shown in Figure 60.25.<br>0: ROP (raster operation)<br>1: Blending operation |
| 30 to 23,<br>19,<br>15 to 8 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 22 to 20 | DSTSEL [2:0] | All 0 | R/W | Input Selection for DST Side of Blending/ROP Unit A<br>These bits select the input for the DST side of blending/ROP unit A shown in Figure 60.25. These bits specify the connection between the BRU input port and the DST separately from the setting of connections between other modules and the BRU input port through the DPR.<br>b'000: BRU input 0 (BRUin0) is input to DST<br>b'001: BRU input 1 (BRUin1) is input to DST<br>b'010: BRU input 2 (BRUin2) is input to DST<br>b'011: BRU input 3 (BRUin3) is input to DST<br>b'100: Virtual RPF is input to DST<br>b'101: BRU input 4 (BRUin4) is input to DST<br>b'110 to 111: Setting prohibited<br>[Note]<br>The DSTSEL bits for blending/ROP unit m (m = B, C, D, E) are reserved according to Table 60.24. The write value should always be 0. |
| 18 to 16 | SRCSEL [2:0] | All 0 | R/W | Input Selection for SRC Side of Blending/ROP Unit m (m = A, C, D, E)<br>These bits select the input for the SRC side of blending/ROP unit m (m = A, C, D, E) shown in Figure 60.25. These bits specify the connection between the BRU input port and the SRC separately from the setting of connections between other modules and the BRU input port through the DPR.<br>b'000: BRU input 0 (BRUin0) is input to SRC<br>b'001: BRU input 1 (BRUin1) is input to SRC<br>b'010: BRU input 2 (BRUin2) is input to SRC<br>b'011: BRU input 3 (BRUin3) is input to SRC<br>b'100: Virtual RPF is input to SRC<br>b'101: BRU input 4 (BRUin4) is input to SRC<br>b'110 to 111: Setting prohibited<br>[Note]<br>The SRCSEL bits for blending/ROP unit B are reserved according to Table 60.24. The write value should always be 0. |
| 7 to 4 | CROP [3:0] | All 0 | R/W | Color Data ROP Operator<br>These bits select the ROP operator of the color data in blending/ROP unit m (m = A, B, C, D, E). Select the opcode for ROP operation from Table 60.27. |
| 3 to 0 | AROP [3:0] | All 0 | R/W | α Data ROP Operator<br>These bits select the ROP operator of the α data in blending/ROP unit m (m = A, B, C, D, E). Select the opcode for ROP operation from Table 60.27. |

**Table 60.27 ROP Operator of Blending/ROP Unit m (m = A, B, C, D)**

| Opcode | Operator |
|---|---|
| b'0000 | NOP(D) |
| b'0001 | AND(S & D) |
| b'0010 | AND_REVERSE(S & ~D) |
| b'0011 | COPY(S) |
| b'0100 | AND_INVERTED(~S & D) |

| | |
|---|---|
| b'0101 | CLEAR(0) |
| b'0110 | XOR(S ^ D) |
| b'0111 | OR(S \| D) |
| b'1000 | NOR(~(S \| D)) |
| b'1001 | EQUIV(~(S ^ D)) |
| b'1010 | INVERT(~D) |
| b'1011 | OR_REVERSE(S \| ~D) |
| b'1100 | COPY_INVERTED(~S) |
| b'1101 | OR_INVERTED(~S \| D) |
| b'1110 | NAND(~(S & D)) |
| b'1111 | SET(all1) |

### 60.2.9.6 BRU Blend Control Registers (VI6_BRUm_BLD : M=A, B, C, D, E)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CBES | CCMDX [2:0] | | | – | CCMDY [2:0] | | | ABES | ACMDX [2:0] | | | – | ACMDY [2:0] | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COEFX [7:0] | | | | | | | | COEFY [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CBES | 0 | R/W | Blending Expression Selection<br>Selects the blending expression of the color data in the BRU (VI6_BRUm_CTRL.RBC = 1). Blending coefficients are specified by the CCMDX and CCMDY bits.<br>0: CCMDX * (DST color data) + CCMDY * (SRC color data)<br>1: CCMDX * (DST color data) – CCMDY * (SRC color data) |
| 30 to 28 | CCMDX [2:0] | All 0 | R/W | Blending Coefficient X Selection<br>These bits specify coefficient X used in the blending expression determined by the CBES bit.<br>b'000: DST α data is used as blending coefficient X<br>b'001: 255 – (DST α data) is used as blending coefficient X<br>b'010: SRC α data is used as blending coefficient X<br>b'011: 255 – (SRC α data) is used as blending coefficient X<br>b'100: Fixed α value 0 (COEFX setting) |
| 27, 19 | – | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 26 to 24 | CCMDY [2:0] | All 0 | R/W | Blending Coefficient Y Selection<br>These bits specify coefficient Y used in the blending expression |

| | | | | |
|---|---|---|---|---|
| | | | | determined by the CBES bit.<br>b'000: DST α data is used as blending coefficient Y<br>b'001: 255 – (DST α data) is used as blending coefficient Y<br>b'010: SRC α data is used as blending coefficient Y<br>b'011: 255 – (SRC α data) is used as blending coefficient Y<br>b'100: Fixed α value 1 (COEFY setting) |
| 23 | ABES | 0 | R/W | Blending α Creation Expression<br>Specifies the expression for creating α data after blending by blending/ROP unit m (m = A, B, C, D, E). α creation coefficients are specified by the ACMDX and ACMDY bits.<br>0: ACMDX * (DST α data) + ACMDY * (SRC α data)<br>1: ACMDX * (DST α data) – ACMDY * (SRC α data) |
| 22 to 20 | ACMDX [2:0] | All 0 | R/W | α Creation Coefficient X<br>These bits specify α creation coefficient X used in the α creation expression determined by the ABES bit.<br>b'000: (α creation coefficient X) = (DST α data)<br>b'001: (α creation coefficient X) = 255 – (DST α data)<br>b'010: (α creation coefficient X) = (SRC α data)<br>b'011: (α creation coefficient X) = 255 – (SRC α data)<br>b'100: (α creation coefficient X) = Fixed α value 0 (COEFX setting)<br>b'101 to b'111: Setting prohibited |
| 18 to 16 | ACMDY [2:0] | All 0 | R/W | α Creation Coefficient Y<br>These bits specify α creation coefficient Y used in the α creation expression determined by the ABES bit.<br>b'000: (α creation coefficient Y) = (DST α data)<br>b'001: (α creation coefficient Y) = 255 – (DST α data)<br>b'010: (α creation coefficient Y) = (SRC α data)<br>b'011: (α creation coefficient Y) = 255 – (SRC α data)<br>b'100: (α creation coefficient Y) = Fixed α value 1 (COEFY setting)<br>b'101 to b'111: Setting prohibited |
| 15 to 8 | COEFX [7:0] | All 0 | R/W | Fixed α Value 0<br>These bits specify fixed α value 0 used when the CCMDX or ACMDX bits are set to b'100. A value from 0x00 to 0xFF can be specified. |
| 7 to 0 | COEFY [7:0] | All 0 | R/W | Fixed α Value 1<br>These bits specify fixed α value 1 used when the CCMDY or ACMDY bits are set to b'100. A value from 0x00 to 0xFF can be specified. |

### 60.2.9.7 BRU Raster Operation Control Register (VI6_BRU_ROP)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | DSTSEL [2:0] | | | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | CROP [3:0] | | | | AROP [3:0] | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23,<br>19 to 8 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 22 to 20 | DSTSEL [2:0] | All 0 | R/W | Input Selection for DST Side of ROP Unit<br>These bits select the input for the DST side of the ROP unit. For the SRC side, the same data as the input for the SRC side of blending/ROP unit C is input.<br>b'000: BRU input 0 (BRUin0) is input to DST<br>b'001: BRU input 1 (BRUin1) is input to DST<br>b'010: BRU input 2 (BRUin2) is input to DST<br>b'011: BRU input 3 (BRUin3) is input to DST<br>b'100: Virtual RPF is input to DST<br>b'101: BRU input 4 (BRUin4) is input to DST<br>b'110 to 111: Setting prohibited |
| 7 to 4 | CROP [3:0] | All 0 | R/W | Color Data ROP Operator<br>These bits select the ROP operator of the color data in the ROP unit. Select the opcode for ROP operation from Table 60.27. |
| 3 to 0 | AROP [3:0] | All 0 | R/W | α Data ROP Operator<br>These bits select the ROP operator of the α data in the ROP unit. Select the opcode for ROP operation from Table 60.27. |

### 60.2.10 ILV Control Register

#### 60.2.10.1 ILV Control Register (VI6_ILV_CTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | ILV_MODE [1:0] | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | ILV_EN |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18,<br>15 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17 to 16 | ILV_MODE<br>[1:0] | 0 | R/W | Format Setting for 3D-Format Conversion<br>These bits specify the format for 3D-format conversion.<br>0: Side by side (output is enlarged twice in horizontal direction)<br>1: Line by line (output is enlarged twice in vertical direction)<br>2: Pixel by pixel (output is enlarged twice in horizontal direction)<br>3: Doubler (output is enlarged twice in horizontal and vertical directions) |
| 0 | ILV_EN | 0 | R/W | 3D-Format Conversion Enable/Disable<br>Enables or disables 3D-format conversion by the ILV. Before setting this bit to 1, make appropriate connections with the DPR so that valid image data is sent to two input lines of the ILV. When this bit is set to 0, the input to the ILVL passes through the ILV and the output size becomes the same as the input size.<br>0: 3D-format conversion is disabled.<br>1: 3D-format conversion is enabled. |

Interleave method of ILVL and ILVR is shown in Figure 60.27. "ILVL" or "L" in the figure is output image from module that target node setting is 38. "ILVR" or "R" in the figure is output image from module that target node setting is 39. See section 60.2.8.1 about target node.

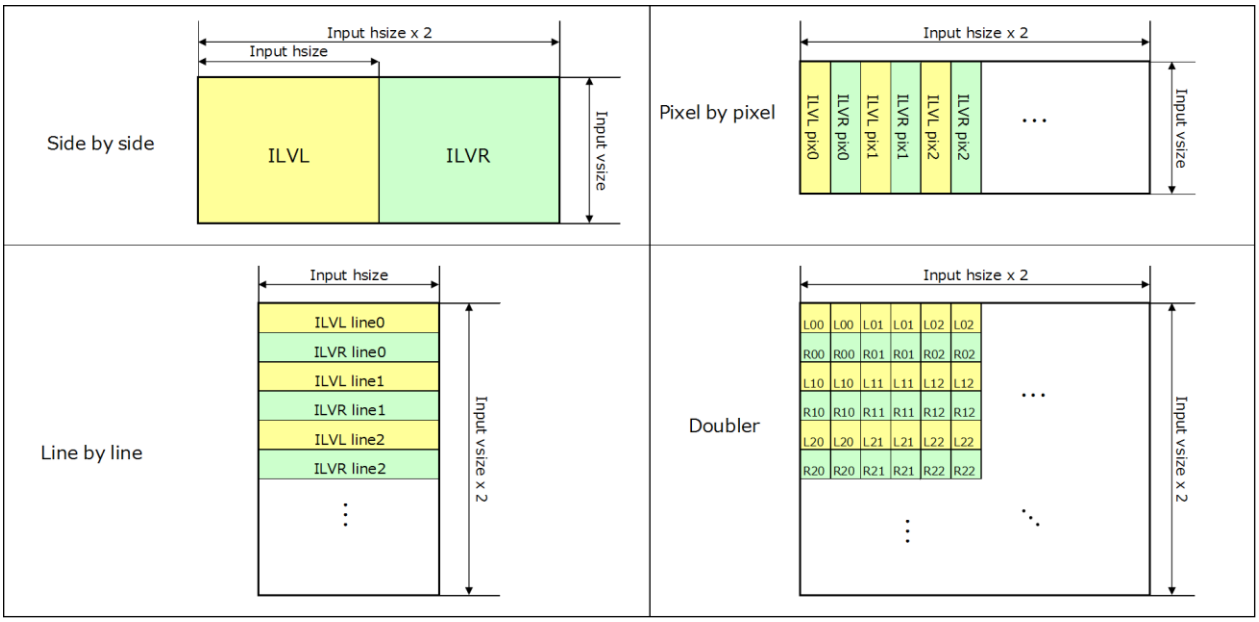



**Figure 60.27 Interleave method of ILVL and ILVR**

### 60.2.11 LIF Control Registers

#### 60.2.11.1 LIF0 Control Register (VI6_LIF0_CTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | OBTH [11:0] | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | PADL | – | – | – | CFMT | – | – | REQSEL | LIF_EN |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28,<br>15 to 9, 7 to 5,<br>3 to 2 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 16 | OBTH [11:0] | 0 | R/W | Buffer Threshold for Start Ready Notification to Display Module<br>Always specify 3000, when LIF_EN is set to 1. |
| 8 | PADL | 0 | R/W | Enable/Disable of padding with dummy lines to output-image to the DU.<br>Enables or disables of padding with dummy lines to output-image to the DU.<br>0: Padding with dummy lines to output image to DU is disabled.<br>1: Padding with dummy lines to output image to DU is enabled. |
| 4 | CFMT | 0 | R/W | Chroma Format<br>This bit selects the output format from the LIF module to the display module. When RGB format is used, this bit shall be set to 0.<br>0: YCbCr444 or RGB Format<br>1: YCbCr422 Format<br>[Note] The DU cannot receive YCbCr422 format. Therefore, when CFMT is 1, YCbCr444 data which information contents is equal to YCbCr422 is output to DU as shown in Figure 60.28. Set 1 to DEF10Rm.YCDF{0,1} (DU register) for DU to perform simple down sample for Cb/Cr. |
| 1 | REQSEL | 0 | R/W | External Display Module Selection<br>0: This value is setting prohibited when 1 is set to LIF_EN<br>1: DU is selected as the destination external display module. |
| 0 | LIF_EN | 0 | R/W | Enable/Disable of Data Output to External Display Module<br>Enables or disables data output from the LIF to the external display module (DU).<br>0: Data output to the external display module is disabled. |

| | | | | |
|---|---|---|---|---|
| | | | | 1: Data output to the external display module is enabled. |

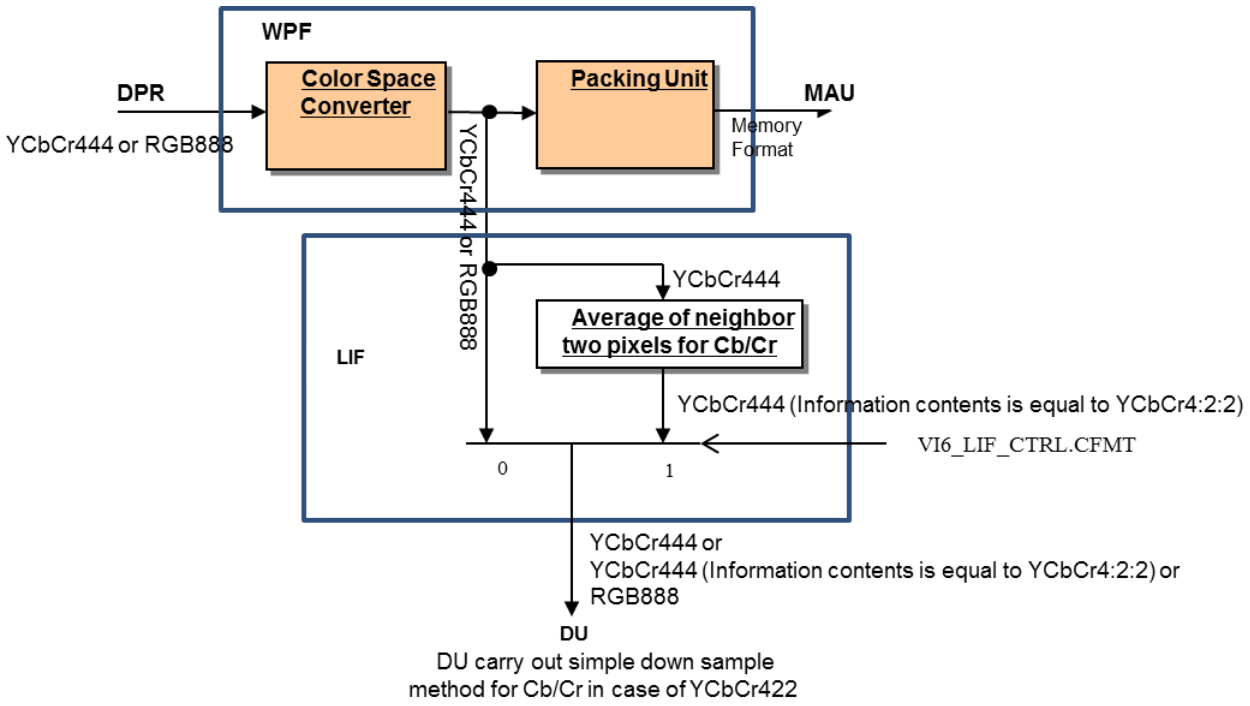



**Figure 60.28 Data flow between LIF and DU**

### 60.2.11.2 LIF0 Clock Stop Buffer Control Register (VI6_LIF0_CSBTH)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | HBTH [11:0] | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | LBTH [11:0] | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28,<br>15 to 12 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 27 to 16 | HBTH [11:0] | 0 | R/W | Buffer Threshold for Clock Stop in Dynamic Clock Control<br>Set HBTH[11:0] = 0 (fixed value) |
| 11 to 0 | LBTH [11:0] | 0 | R/W | Buffer Threshold for Clock Start in Dynamic Clock Control<br>Set LBTH[11:0] = 0 (fixed value) |

### 60.2.11.3 LIF0 Padding Line Cycle Register (VI6_LIF0_PADLN_CYC)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | PADLN_CYC [5:0] | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | PADLN_CYC [5:0] | 0 | R/W | Cycle of padding pattern.<br>These bits specify the minimum number of cycles of the padding pattern of valid lines and dummy lines. For example, if PADLN_CYC [5:0] is 4, lower 4 bits of PADLN_PATTERN [3:0] indicates padding pattern. A value from 1 to 32 can be specified. |

### 60.2.11.4 LIF0 Padding Line Pattern Register (VI6_LIF0_PADLN_PT)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PADLN_PATTERN [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PADLN_PATTERN [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PADLN_PATTERN [31:0] | 0 | R/W | Pattern of padding with dummy lines<br>These bits specify the padding pattern of valid lines and dummy lines. In each bit, B'0 indicates an active line, B'1 indicates a dummy line. It starts from the lower bit (Bit 0).<br>First line should be valid line (Always set 0 to PADLN_PATTERN [0]). |

### 60.2.11.5 LIF0 Padding Line Value Register (VI6_LIF0_PADLN_VAL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PADA [7:0] | | | | | | | | PADR [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PADG [7:0] | | | | | | | | PADB [7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | PADA [7:0] | 0 | R/W | Padding data of EDC code for dummy line. |
| 23 to 16 | PADR [7:0] | 0 | R/W | Padding data of R/Cr for dummy line. |
| 15 to 8 | PADG [7:0] | 0 | R/W | Padding data of G/Y for dummy line. |
| 7 to 0 | PADB [7:0] | 0 | R/W | Padding data of B/Cb for dummy line. |

Note: Set H'24FFFFFF to the register, when output format is YUV422. Set H'2EFFFFFF to the register in other case.

### 60.2.11.6 LIF0 Padding Line Size Register (VI6_LIF0_PADLN_SIZE)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | – | – | – | PADLN_VSIZE [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 to 0 | PADLN_VSIZE [12:0] | 0 | R/W | Vertical Size of LIF output in case of enabling Padding line<br>When VI6_LIF0_CTRL.PADL is 1, dummy lines are included in the output data from LIF, so set the LIF output VSIZE including dummy lines to these bits. If the number of output lines does not reach the value of PADLN_VSIZE [12:0] even if all valid lines are output according to padding pattern, dummy lines are inserted until the number of output lines reaches PADLN_VSIZE [12:0].<br>- Output vsize of LIF is same with input vsize to LIF when VI6_LIF0_CTRL.PADL is 0<br>- Output vsize of LIF is PADLN_VSIZE [12:0] when VI6_LIF0_CTRL.PADL is 1 |

### 60.2.12 VSP Internal Registers

#### 60.2.12.1 IP Version Register (VI6_IP_VERSION)

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | IP Version [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | IP Version [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0/1 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0/1 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | IP Version [31:0] | – | R | IP version register to indicate module kind and IP version. |

| | | | | |
|---|---|---|---|---|
| | | | | R-Car V4M VSPD : H'0101 1C08<br>R-Car V4M VSPX : H'0101 1D08 |

### 60.2.13 RPFn OSD-CLUT Table

Refer to section 60.3.5.1

### 60.2.14 IIF Control Registers (VI6_IIF_CTRL)

#### 60.2.14.1 IIF Control Registers (VI6_IIF_CTRL)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | CTRL [4:0] | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 to 0 | CTRL | 0 | W | A single write access to this register is required before activating VSPX in order to transfer ISP config data.<br>Set 0x13 before starting VSPX to this register. Read value is not guaranteed. These bits are only for VSPX. |

## 60.3 Operation

### 60.3.1 Registers to Set Fixed Value

Set fixed value to following registers in any case.

[1] Always set H'0000 0808 to VI6_CLK_DCSWT

[2] Always set D'256 to VI6_DL_CTRL.AR_WAIT [7:0] in case of using display list.

[3] Always set D'2 to VI6_DL_EXT_CTRL.POLINT [5:0] in case of using extended display list of WPF0.

[4] Always set D'1 to VI6_DL_EXT_CTRL.DLPRI in case of using extended display list of WPF0.

[5] Always set D'0 to VI6_DL_EXT_CTRL.EXPRI in case of using extended display list of WPF0.

[6] Always set H'0000 0500 toVI6_DPR_WPF0_FPORCH in case of using WPF0.

[7] For VSPD, always set D'3000 to VI6_LIF0_CTRL.OBTH[11:0] in case of using LIF0 (VI6_LIF0_CTRL.LIF_EN = 1).

[8] Always set D'1 to VI6_LIF0_CTRL.REQSEL in case of using LIF0 (VI6_LIF0_CTRL.LIF_EN = 1).

[9] Always set D'0 to VI6_LIF0_CSBTH in case of using LIF0 (VI6_LIF0_CTRL.LIF_EN = 1).

### 60.3.2 Concept of VSP2 Operation Starting and Stopping

The VSP2 provides one channel (WPF0) of image processing. The WPF0 is started by setting the corresponding start register. Use the start registers shown in Table 60.28 to start WPF0 module.

After the WPF0 is started and specified processing is completed, the corresponding channel stops operation and notifies the end of processing through an end interrupt. End interrupts are generated through the end interrupt source registers shown in Table 60.28; clearing a source register cancels the corresponding interrupt signal (for details of interrupt processing, refer to section 60.3.4, Interrupt Processing). Each of the operating status registers shown in the table indicates the busy state after the corresponding channel is started through the start register until processing is completed and operation stops.

**Table 60.28 Target Module and Corresponding Registers for Starting and Stopping Operation**

| Target Module | Start Register | End Interrupt Source Register | Operating Status Register |
|---|---|---|---|
| WPF0 | VI6_CMD0.STRCMD | VI6_WPF0_IRQ_STA. FRE | VI6_STATUS.SYS0_ACT |

Figure 60.29 shows relation between start reservation and operating status. When start is reserved while VSP2 is operating, the reservation is accepted after VSP2 status is moved from "operating" into "idle" as shown in Figure 60.29. Therefore, when start reservation is used for VSPX as shown in section 60.3.7, take following procedure to confirm VSP2 isn't operating.

<1> Check VI6_CMD0.STRCMD is zero (Start is not reserved)

<2> Check VI6_STATUS.SYS0_ACT is zero (VSP2 is IDLE status)

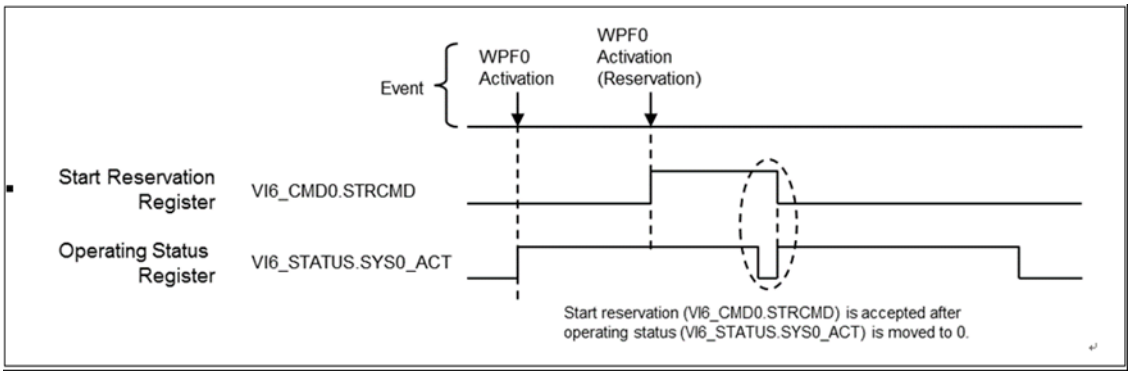



**Figure 60.29 VSP2 Start reservation and Operating status**

### 60.3.3 Display List

#### 60.3.3.1 Functional Description

The VSP2 provides the display list function. As a display list, the VSP2 automatically downloads the register settings except for the control registers (sections 60.2.4 and 60.2.5) from external memory and stores the settings in the VSP2 registers. This function is advantageous in that the interrupt processing or register setting modification processing can be executed without CPU intervention during multiple-frame processing because the register settings used for VSP2 processing are prepared in advance in external memory such as SDRAM.

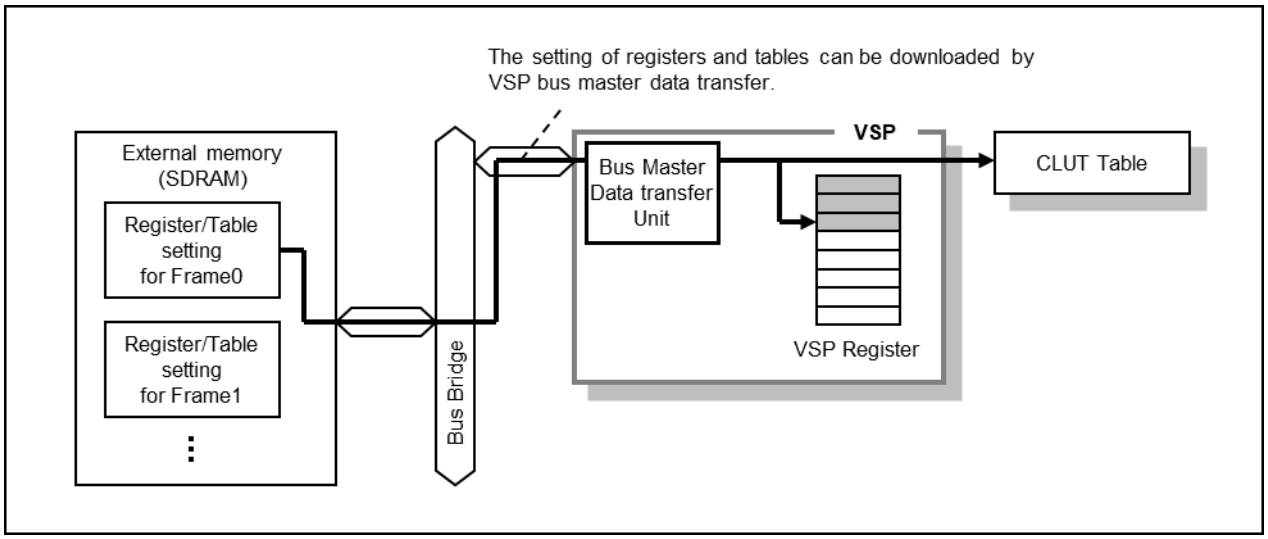



**Figure 60.30 Concept of Display List**

To use display lists, specify the external memory addresses to the display list control registers described in section 60.2.5. The register settings or various types of information should be stored in external memory in the format described in section 60.3.3.2. Figure 60.31 shows the difference between VSP2 operation through normal register settings and through display lists.

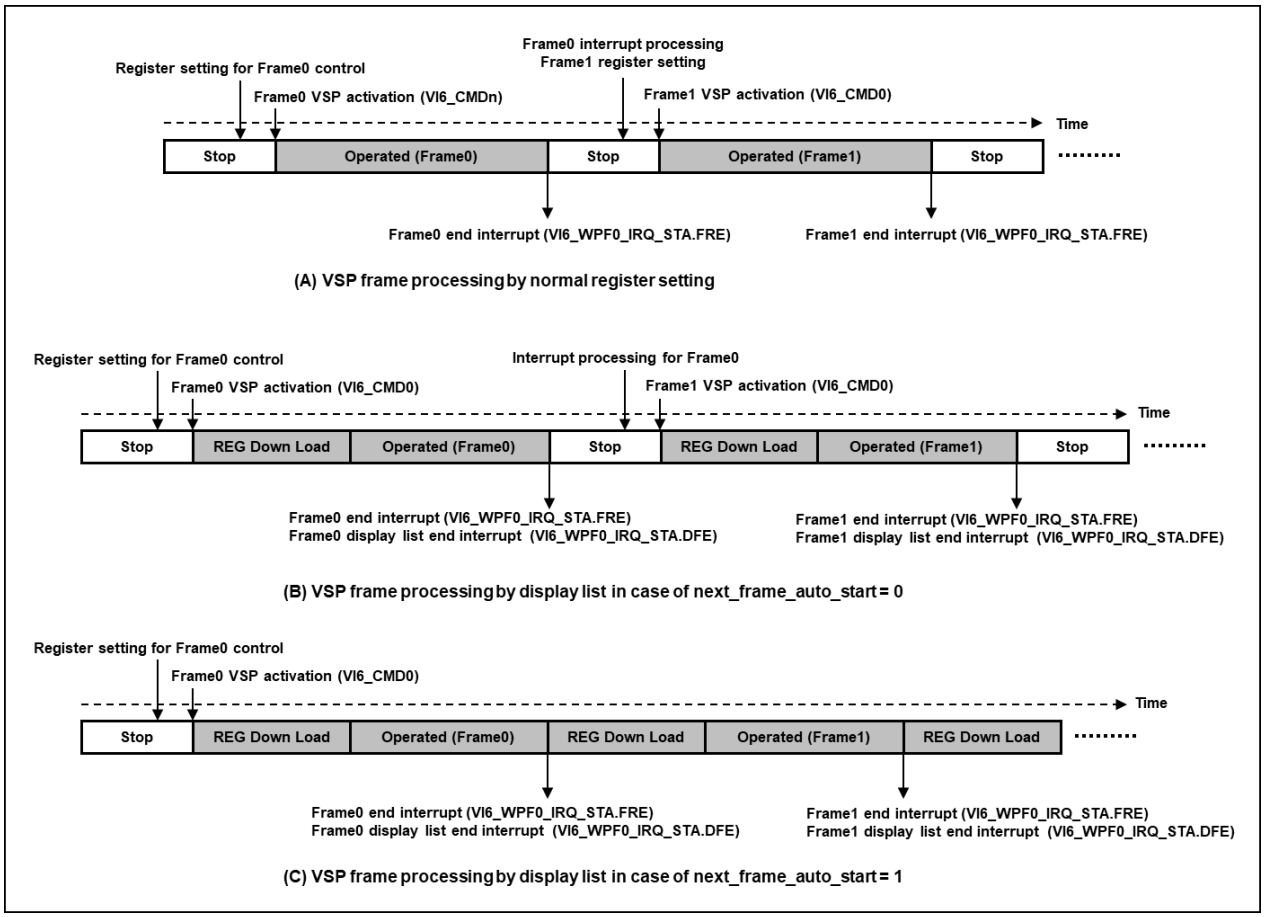



**Figure 60.31 Comparison of VSP2 Operation between Normal Register Settings and Display Lists**

As shown in Figure 60.31 (A), in the VSP2 processing through normal register settings, all registers should be set up before the VSP2 is started for each frame. After the VSP2 processing is completed, the VSP2 outputs a frame end interrupt (VI6_WPF0_IRQ_STA.FRE). This method requires a certain amount of time for register settings or interrupt processing by the CPU between frames. In contrast, when display lists are used, the VSP2 automatically downloads register settings from external memory as shown in Figure 60.31 (B) and (C), which reduces the load on the CPU between frames.

Figure 60.31 (B) shows the display list usage where the VSP2 stops at the end of every frame; only the VSP2 start processing for each frame is done by the CPU. This is suitable for the cases when the CPU controls synchronization of frame processing in frame buffer management or when the amount of register values or table data to be set in the VSP2 is large. In the case shown in Figure 60.31 (C), as soon as the frame processing ends, the VSP2 automatically begins next frame operation and starts downloading new register settings. This is the fastest operation using display lists.

Table 60.29 shows the modes of the display list and the supported functions for each mode. The detail of each mode is described in the following sections.

**Table 60.29 Display list mode and supported functions**

| Mode | Extended Display List | Continuous Frames |
|---|---|---|
| Normal Display List Mode | Supported | Controlled by “next frame auto start” in the header of the display list. |
| Header-less Display List Mode | Not Supported | Controlled by VI6_DL_CTRL.CFM0 bit. |

### 60.3.3.2 Normal Display List Mode

The VSP2 display lists include control information as well as simple register settings in order to control multiple-frame processing in an optimum way for each application. Figure 60.32 shows the display list structure.

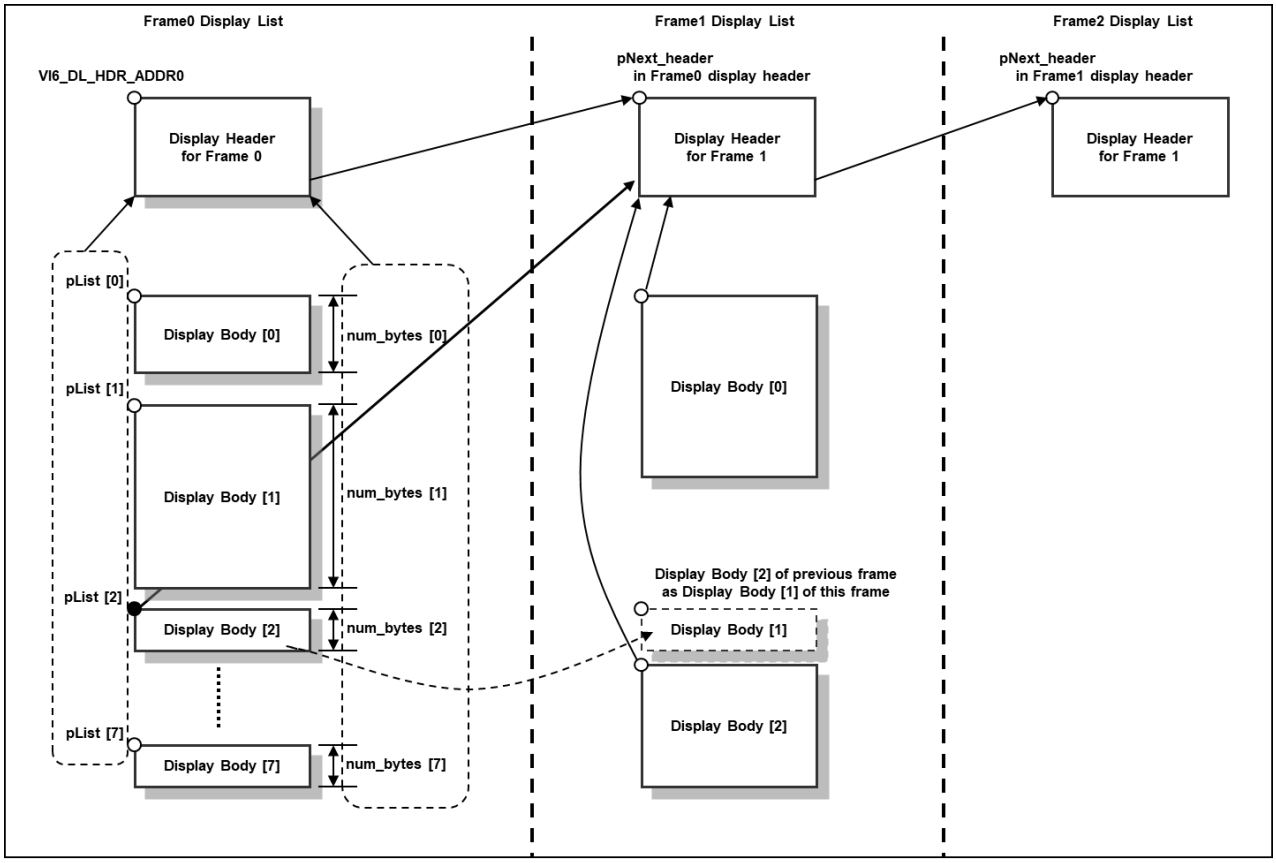



**Figure 60.32 Structure and Concept of VSP2 Display List**

A VSP2 display list consists of two sections; a header section for storing various information and control flags and a body section for storing register and table settings. A combination of these two sections is defined as a display list for a frame. The register and table settings can be divided and stored in up to eight separate bodies allocated in memory. Therefore, the separate bodies storing the register settings for one frame (for example, frame 0) can have non-sequential start addresses; that is, the bodies for one frame can be allocated to areas distant from each other in memory. To gather these bodies and configure the register settings for one frame, a header is used. A display header stores the number of bodies linked with the header and the start address and size of each body.

The VSP2 analyzes the header to gather register and table settings stored in separate memory areas and reconfigures the complete register settings.

The addresses of display headers should be specified in the VSP2 registers described in section 60.2.5. When activated in a mode that uses display lists, the VSP2 downloads display headers from the addresses specified in VI6_DL_HDR_ADDR0, analyzes the numbers of bodies and the address and data size of each body, downloads the bodies, and completes register and table settings. After display list downloading is completed, the VSP2 becomes ready for frame image processing; the VSP2 then starts the actual frame processing.

After processing of a frame ends, the VSP2 proceeds to the next frame processing. Here, there are two modes for starting the next frame processing as shown in . In one mode, the VSP2 stops operation and waits for the next activation by the CPU in the same way as when display lists are not used. In this mode, the address used for downloading the display header of the next frame is kept by the internal hardware. Therefore, if valid address information is not stored in the display header for the previous frame, a correct value should be specified in VI6_DL_HDR_ADDR0 while the VSP2 is stopped. When the VSP2 is started after a correct value is specified, the VSP2 starts next frame processing with the same procedure for the previous frame. In contrast to this mode, which stops the VSP2 after the end of one-frame processing, there is another mode for automatically starting next frame processing. In automatic start mode, the VSP2 downloads the

next display header as soon as the previous frame processing ends. After downloading ends, the VSP2 starts image processing. The information regarding mode selection, that is, whether to automatically start next frame processing, should be stored in the display header downloaded for the previous frame.

In the automatic start mode, the VSP2 continues processing until the display header for a frame specifies that the next frame should not be started automatically. To stop processing during automatic execution, use a software reset (VI6_SRESET).

To strictly define the display list format described above, the following shows the grammatical structure of a display list using pseudo-code. First, to simplify the description of the display list format in the following pages, Table 60.30 defines a function. Function zero bits (num bits) generates a string of one-bit 0s for the number of bits specified by the parameter for the function. By using this function, Table 60.31 defines the header section format of a display list and Table 60.32 defines the body section format and Table 60.33 defines the extended display list body section format.

Data order of Display List Header Section, Display List Body Section and Extended Display List Body Section are assumed as the data is stored in SDRAM by big endian. Table 60.34 shows an example of data order. If data order is not same as the definition, data order within 8byte unit can be adjusted by setting VI6_DL_SWAP.

**Table 60.30 Definition of a Function for Simple Description**

| Syntax | Bit Count |
|---|---|
| zero_bits (num_bits) | |
| { | |
| for (i=0; i<num_bits; i++) { | |
| **zero_bit** | 1 |
| } | |
| } | |

| Bit String | Contents |
|---|---|
| zero_bit | zero_bit indicates a 1-bit integer having a value of 0. |

**Table 60.31 Format of Display List Header Section**

| Syntax | Bit Count |
|---|---|
| display_header () /* Fixed length */ | |
| { | |
| zero_bits (29) | |
| **num_list_minus1** | 3 |
| for (i=0; i<8; i++) | |
| zero_bits (15) | |
| **num_bytes [i]** | 17 |
| **pList [i]** | 32 |
| } | |
| **pNext_header** | 32 |
| zero_bits (23) | |
| **wait_wup** | **1** |
| zero_bits (3) | |
| **ignore_upd_dl** | **1** |
| zero_bits (2) | |
| **current_frame_int_enable** | 1 |

| | |
|---|---|
| **next_frame_auto_start** /* 76 bytes from the beginning of this header*/ | 1 |
| if (VI6_DL_EXT_CTRL.EXT) { | |
| zero_bits (32) /* padding zero 4 bytes for alignment */ | |
| zero_bits (6) | |
| **pre_ext_dl_exec** | 1 |
| **post_ext_dl_exec** | 1 |
| zero_bits (8) | |
| **pre_ext_dl_num_cmd** | 16 |
| **pre_ext_dl_pList** | 32 |
| zero_bits (16) | |
| **post_ext_dl_num_cmd** | 16 |
| **post_ext_dl_pList** /* 96 bytes from the beginning of this header*/ | 32 |
| } | |
| } | |

| Bit String | Contents |
|---|---|
| num_list_minus1 | Specifies the value obtained by subtracting 1 from the total number of display list bodies linked with the display header. For example, when this bit field is set to 0, this display list uses one body. |
| num_bytes [i] | Specifies the number of bytes in the i-th display list body (indicated by index i). Be sure to specify a multiple of eight bytes. For the bodies that are not defined in num_list_minus1 (for example, i = 5 to 7 when num_list_minus1 is set to 4), specify 0. |
| pList [i] | Specifies the start address of the i-th display list body (indicated by index i). Be sure to specify an address aligned with an 8-byte boundary (the lower-order three bits are 0). For the bodies that are not defined in num_list_minus1 (for example, i = 5 to 7 when num_list_minus1 is set to 4), specify 0. |
| pNext_header | Specifies the address of the display list header for the next frame. After display list downloading ends, the VSP2 keeps its value in the internal memory and uses it in the next display list header downloading.<br>Be sure to specify an address aligned with an 8-byte boundary (the lower-order three bits are 0). |
| wait_wup | When this bit is 1, VSP2 starts reading image data from external memory after WUP (Wake Up) signal from any channel of VIN asserts. This bit is available for WPF0. |
| ignore_upd_dl | If this bit is set to 1, new display list pointed by VI6_DL_HDR_ADDR0 is not downloaded in VSP2 H/W for the next frame, even if S/W sets VI6_DL_HDR_ADDR0 during the frame. New display list is downloaded in VSP2 H/W at next frame of the processing which this bit is 0. This bit is available for WPF0. |
| current_frame_int_enable | This is a flag that indicates whether to set the display list end interrupt source (VI6_WPF0_IRQ_STA.DFE) to 1 when the current frame processing ends. If this flag is set to 0, the display list end interrupt source (VI6_WPF0_IRQ_STA.DFE) is not set to 1 when one-frame processing by this display header ends. In this state, even if the display list end interrupt is enabled (VI6_WPF0_IRQ_ENB.DFEE is set to 1), no interrupt will be generated.<br>If this flag is set to 1, the display list end interrupt source (VI6_WPF0_IRQ_STA.DFE) is set to 1 when one-frame processing by this display header ends. In this state, if the display list end interrupt is enabled (VI6_WPF0_IRQ_ENB.DFEE is set to 1), the VSP2 generates an interrupt. |
| Next_frame_auto_start | Enables or disables automatic start of next frame processing when one-frame processing by this display header ends. If this bit is set to 1, the VSP2 starts next frame processing as soon as one-frame processing by this display header ends, and starts downloading the next frame display header from the |

| | |
|---|---|
| | pNext_header address specified in this display header. If this bit is set to 0, the VSP2 stops operation when one-frame processing by this display header ends. In this case, start the VSP2 through VI6_CMD0 to process the next frame. |
| pre_ext_dl_exec | Enables execution of the extended display list for frame preprocessing when VI6_DL_EXT_CTRL.EXT is 1. If this bit is set to 1, the VSP2 executes the extended display list for frame preprocessing. The VSP2 does not execute it if this bit is set to 0. This bit setting is ignored when VI6_DL_EXT_CTRL.EXT is 0. |
| post_ext_dl_exec | Set 0 to this bit. |
| pre_ext_dl_num_cmd | Specifies the number of commands in the extended display list body section for frame preprocessing when VI6_DL_EXT_CTRL.EXT is 1. The number of commands that can be specified is 1, and a command is 16 bytes.<br>When pre_ext_dl_exec is set to 0, the extended display list for frame preprocessing is not executed; specify 0 in this bit. This bit setting is ignored when VI6_DL_EXT_CTRL.EXT is 0. |
| pre_ext_dl_pList | Specifies the start address of the area where the extended display list body section for frame preprocessing is stored when VI6_DL_EXT_CTRL.EXT is 1. Be sure to specify an address aligned with a 16-byte boundary (lower-order four bits are 0). When pre_ext_dl_exec is set to 0, the extended display list for frame preprocessing is not executed; specify 0 in this bit. This bit setting is ignored when VI6_DL_EXT_CTRL.EXT is 0. |
| post_ext_dl_num_cmd | Set 0 to this bit. |
| post_ext_dl_pList | Set 0 to this bit. |

**Table 60.32 Format of Display List Body Section**

| Syntax | Bit Count |
|---|---|
| display_list (num_bytes) /* Variable length (num_bytes) */ | |
| for (i=0; i<num_bytes; i+=8) { | |
| **set_address** | 32 |
| **set_data** | 32 |
| } | |
| } | |

| Bit String | Contents |
|---|---|
| set_address | Specifies the address where the value specified by set_data is to be stored. Specify a register address.<br>Note :<br>Register space of VSP is 32Kbyte (0x0000 - 0x7FFC). So, set 0 to upper 17bit of set_address in Display List Body Section. |
| set_data | Specifies the value to be stored in the address specified by set_address. Specify a value to be set in a register. |

**Table 60.33 Format of Extended Display List Body Section**

| Syntax | Bit width |
|---|---|
| ext_dl_display_list (num_llw) /* Variable length (pre/post_ext_dl_num_bytes) */ | |
| for (i=0; i<num_llw; i+=2) { | |
| **ext_dl_cmd** | 64 |
| **ext_dl_data** | 64 |
| } | |
| } | |

See section 60.3.6.3 for setting extended display list body section.

**Table 60.34 Data order of display list body**

| Address | Syntax |
|---|---|
| pList | set_address [0] (bit31-24) |
| pList + 1 | set_address [0] (bit23-16) |
| pList + 2 | set_address [0] (bit15-8) |
| pList + 3 | set_address [0] (bit7-0) |
| pList + 4 | set_data [0] (bit31-24) |
| pList + 5 | set_data [0] (bit23-16) |
| pList + 6 | set_data [0] (bit15-8) |
| pList + 7 | set_data [0] (bit7-0) |
| pList + 8 | set_address [1] (bit31-24) |
| pList + 9 | set_address [1] (bit23-16) |
| pList + 10 | set_address [1] (bit15-8) |
| pList + 11 | set_address [1] (bit7-0) |
| pList + 12 | set_data [1] (bit31-24) |
| pList + 13 | set_data [1] (bit23-16) |
| pList + 14 | set_data [1] (bit15-8) |
| pList + 15 | set_data [1] (bit7-0) |
| ~~~ | ~~~ |

#### 60.3.3.3 Header-less Display List Mode

The header-less display list does not have the display header listed in Table 60.31, and it has the simplest structure that has only single body. The extended display list function is not available in case of the header-less display list mode. Set the value 1 to VI6_DL_CTRL.NH0 (see the section 60.2.5.1) to use the header-less display list. The start address downloaded by the header-less display list should be set to VI6_DL_HDR_ADDR0. And the size of the display body which is originally defined in the display header should be set to VI6_DL_BODY_SIZE0. The header-less display list is available only in WPF0.

#### 60.3.3.4 Restrictions on Display List Usage

Access to the general control registers and display list control registers through a display list is prohibited. When using display lists, be sure to observe the following restrictions on register access by the CPU.

1. Do not execute write access to the same register (same address) from the CPU and through a display list at the same time. If such a conflict occurs, correct operation of the VSP2 is not guaranteed.
2. Do not execute write access to the same OSD-CLUTlookup table from the CPU and through a display list at the same time. If such a conflict occurs, correct operation of the VSP2 is not guaranteed. Here, the same lookup table means the address space having the same space name shown in Table 60.35.
3. When read access by the CPU and write access through a display list to the same register (same address) occur at the same time, the read value returned to the CPU is not guaranteed.
4. When read access by the CPU and write access through a display list to the same OSD-CLUT lookup table occur at the same time, the read value returned to the CPU is not guaranteed. Here, the same lookup table means the address space having the same space name shown in Table 60.35.
5. For other restrictions on values and timing of register setting through display lists, refer to the restrictions on normal register settings described in section 60.2.2.
6. Manipulation and setting of the registers described in sections section 60.2.4 and 60.2.5 through a display list is prohibited.

### 60.3.4 Interrupt Processing

When the VSP2 module generates an internal source that should be notified, an interrupt signal is output. As internal sources are generated in each WPF independently, the VSP2 has the following registers to control interrupts in each WPF.

WPF0 Interrupt Enable Registers (VI6_WPF0_IRQ_ENB)

WPF0 Interrupt Status Registers (VI6_WPF0_IRQ_STA)

**(Note1)**

To use interrupts, enable them through WPF interrupt enable registers. If an interrupt source register has already been set to 1 for some reason, an unintended interrupt will occur as soon as the corresponding interrupt enable register is set as enabled. To avoid this, before enabling interrupts through WPF interrupt enable registers, be sure to clear WPF interrupt sources to be enabled to 0. Be careful about this procedure when setting up registers before starting the VSP2.

### 60.3.5 Lookup Table Settings

#### 60.3.5.1 OSD-CLUT

For a single entry to the OSD-CLUT space (Table 60.35) of the VSP2, the OSD-CLUT data is set by a write access in the format shown in Figure 60.33.

The entry address of each space is (start address of the space) + (entry number counting from the base point 0 × 4). For example, the address of entry 7 to the OSD-CLUT2 space is (offset address) + 0x4800 + 7 × 4 = (offset address) + 0x481C.

Table 60.35 shows the spaces for which entries can be made. Note that if the module that references that space is operating, write accesses to the relevant space are prohibited. For example, while RPF2 is operating, accesses to the entire space of VI6_CLUT2_TBL are prohibited. When a read access is made to the relevant space during operation, undefined values will be read out. VI6_STATUS shows operation status.

**Table 60.35 OSD-CLUT Space Addresses**

| Space Name | Space Addresses | Entry Count | Module that References the Space in the Left Column |
|---|---|---|---|
| VI6_CLUT2_TBL | (offset address) + 0x 4800 to (offset address) + 0x 4BFC | 256 | RPF2 |

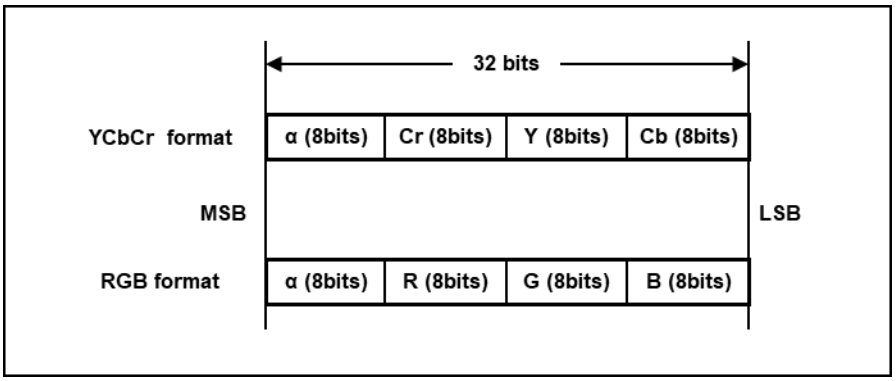



**Figure 60.33 VI6_CLUT_TBL Formats**

### 60.3.6 Linked with DU

#### 60.3.6.1 Operation Flow of VSPD and DU

This section shows operation flow of VSPD by using Normal Display List Mode. For details of Normal Display List Mode, refer to section 60.3.3.2.

Figure 60.34 to Figure 60.36 shows a procedure (register setting flow) to display image data to display panel. Setting not only VSPD registers, but also FCPV/DU registers are needed. So, see FCP User's Manual / DU User's Manual to know the contents of FCPV/DU registers mentioned in Figure 60.34 and Figure 60.36.

Figure 60.34 shows setting flow to start VSPD linked with DU. Figure 60.35 shows setting flow to update display image. Figure 60.36 shows setting flow to stop VSPD linked with DU.

Using display list with enabling next_frame_auto_start shown in section 60.3.3 is required to take the flow from Figure 60.34 to Figure 60.36.

Note: If display sync operation ([1-8] step in Figure 60.34) is set before activating the VSPD, background color specified in DU register (BPORn or DOORn) is displayed until image data read from external memory is ready to be transferred to DU. In this case, image data from external memory is displayed late by one vsync or more.

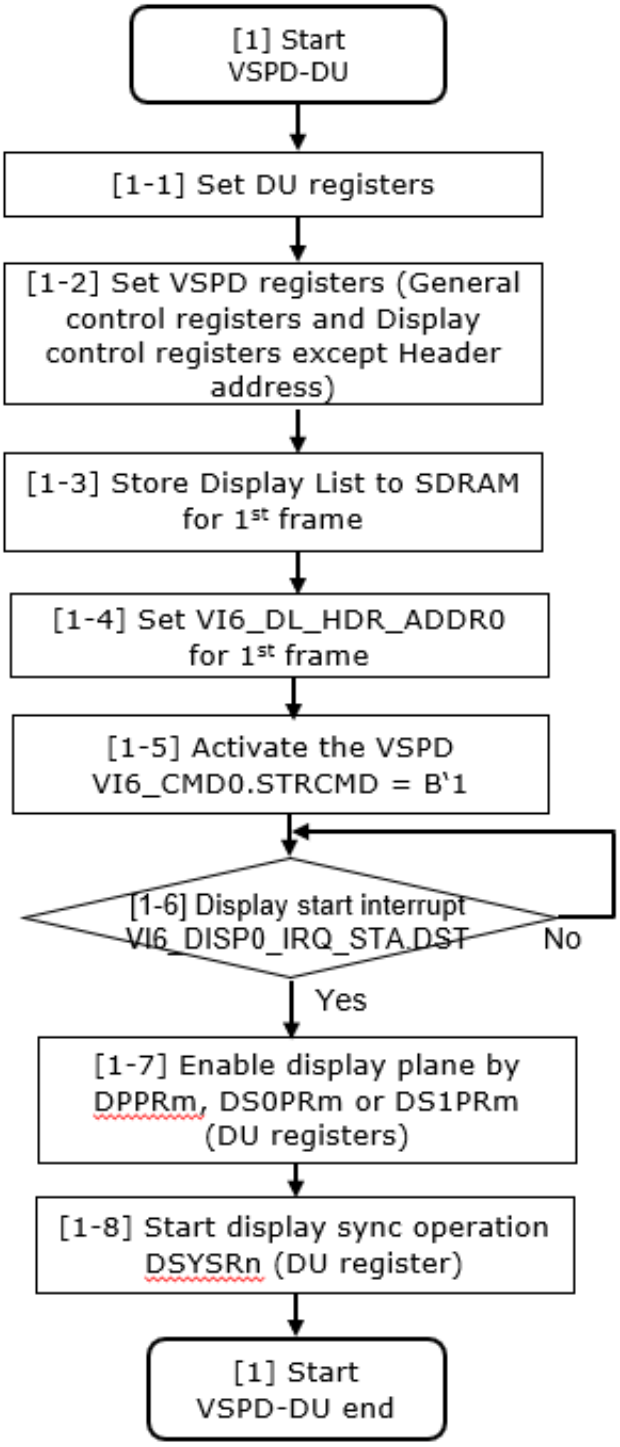



**Figure 60.34 Setting flow to start VSPD linked with DU**

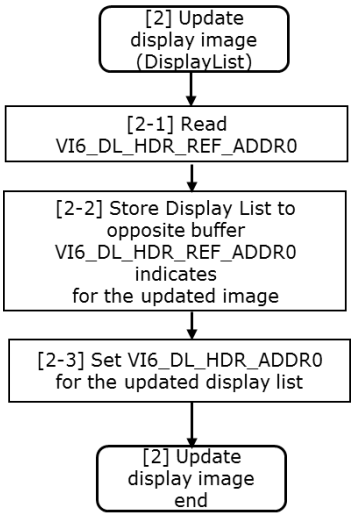



**Figure 60.35 Setting flow to update display image**

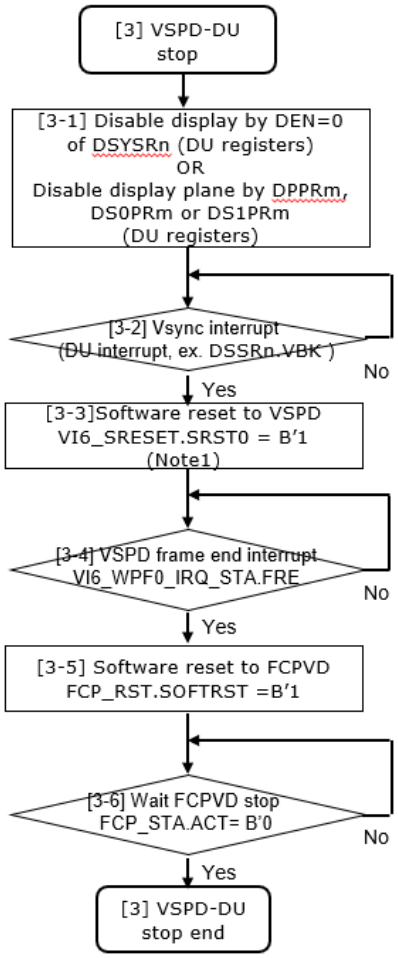



**Figure 60.36 Setting flow to stop VSPD linked with DU**

Note1: As shown in WPF0 Interrupt Status Registers (VI6_WPF0_IRQ_STA), FRE interrupt can occur by 3 factors. Therefore, clear the interrupt before applying software reset not to mistake FRE interrupt of finishing one frame for the FRE interrupt of completing software reset.

It is possible to update display list at any time (arbitrary timing) as shown in Figure 60.35. Don't overwrite display list in external memory pointed by VI6_DL_HDR_REF_ADDR. It is because VSP2 is reading the display list pointed by VI6_DL_HDR_REF_ADDR or will be reading the display list soon.

Detail explanation is shown below.

As shown in section 60.2.5.6, when VSP2 is reading display list from external memory (Period [A] in Figure 60.37), VI6_DL_HDR_REF_ADDR0 indicates header address of the display list referred by VSP2. When VSP2 is not reading display list from external memory (Period [B] in Figure 60.37), VI6_DL_HDR_REF_ADDR0 indicates the value of VI6_DL_HDR_ADDR0, and there is possibility that the display list is being read soon by VSP2 at next frame start timing. Therefore, keep display list in external memory pointed by VI6_DL_HDR_REF_ADDR. Detail software sequence to update display list and VSP2-H/W behavior are shown below.

[1] Store latest display list in external memory area different from the area pointed by VI6_DL_HDR_REF_ADDR, because the area pointed by VI6_DL_HDR_REF_ADDR is being read or is being read soon by VSP2.

[2] Set header address of new display list into VI6_DL_HDR_ADDR0.

[3] VSP2 reads display list from the external memory area pointed by VI6_DL_HDR_ADDR0 at frame start timing.

Refer also to section 60.2.5.3 and section 60.2.5.6 .

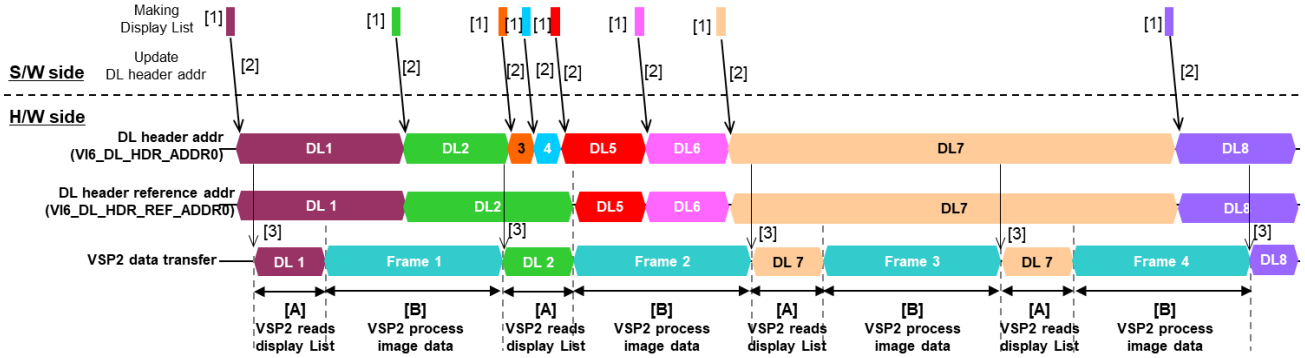
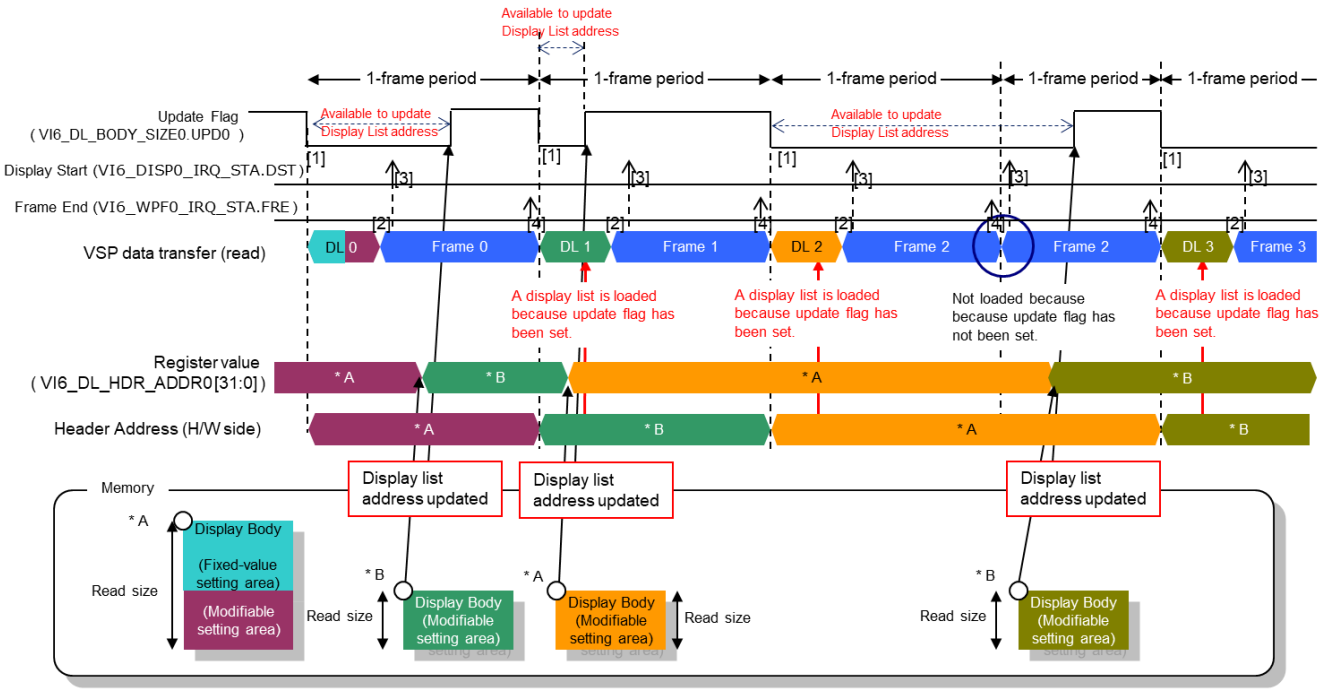



**Figure 60.37 Updating display list at arbitrary timing**

### 60.3.6.2 Controlling Two Register Planes Using Display Lists

This section shows operation flow of VSPD by using Header-less Display List Mode. Start procedure and stop procedure are same with Normal Display List Mode in section 60.3.6.1 except for the point that VI6_DL_BODY_SIZE0 should be set before step [1-5] in start procedure shown in Figure 60.34. The procedure to update display list is different with Normal Display List Mode, and is described later. Figure 60.38 shows the control of two register planes using header-less display lists (see section 60.3.3.3) and its timing. In the description hereafter, the use of header-less display lists is always assumed and they are simply called display lists.




Display list plane A: "Fixed-value settings" and "modifiable settings" are stored as a set in contiguous area A.
Display list plane B: Only the "modifiable settings" are stored in area B, which is separate from display list A.

To switch settings for every frame, modify two display list planes alternately. After every modification, update the display list address register (VI6_DL_HDR_ADDR0[31:0]) and writing 1 to update flag (VI6_DL_BODY_SIZE0.UPD0) in the VSP.
Update flag (VI6_DL_BODY_SIZE0.UPD0) is negated automatically after Header Address is loaded into H/W side.

**Figure 60.38 Controlling Two Register Planes Using a Display List**

The VSP2 downloads a display list immediately after activation. In the start frame, download the display list that contains all necessary settings. From the next frame on, only the necessary register or table values should be specified in a display list.

When the update flag of VI6_DL_BODY_SIZE0.UPD0 is set to 1, VSP2 downloads a new display list at the start of the next frame. When the update flag of VI6_DL_BODY_SIZE0.UPD0 is set to 0, the register settings acquired from the display list previously downloaded are retained and used for the next operation without downloading a new display list.

Order of each VSP2-H/W Events mentioned as [1], [2], [3] and [4] in Figure 60.38 are shown below.

[1] VI6_DL_BODY_SIZE0.BS0 and VI6_DL_HDR_ADDR0 are downloaded in H/W side. And Update Flag (VI6_DL_BODY_SIZE0.UPD0) is negated automatically.

[2] Reading DisplayList from external memory into VSP2-H/W is finished through master access.

[3] Display start interrupt status (VI6_DISP0_IRQ_STA.DST) asserts.

[4] Frame end interrupt status (VI6_WPF0_IRQ_STA.FRE) asserts.

Don’t overwrite the display list in external memory for a period from "[4] VI6_WPF0_IRQ_STA.FRE" to next "[3] VI6_DISP0_IRQ_STA.DST", because VSP is reading the display list for the period.

#### 60.3.6.3 Auto Display Function (AUTO-DISP)

VSPD have AUTO-DISP function. This function allows VSPD to display latest image in triple buffer output from VIN without S/W control of triple buffer. And VSPD selects the field which should be read from TOP or BOTTOM automatically without S/W control. Outline of this function is shown in Figure 60.39. VSPD replaces source address registers with source address information stored in pre extended display list according to latest buffer information from VIN and field information from DU.

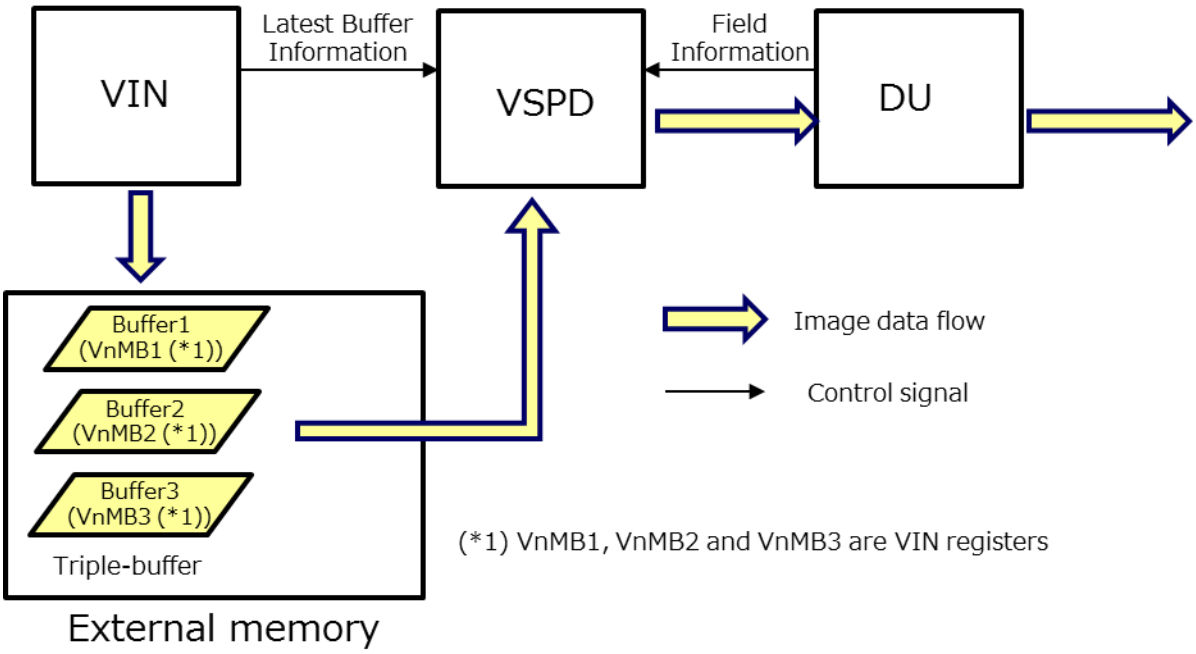



Figure 60.39 Outline of AUTO-DISP

Method to realize this function is shown below.

[1] Use Display List (Normal Display List Mode)

Register setting about display list is shown below.

VI6_DL_CTRL.DLEn = 1

VI6_DL_CTRL.NH0 = 0

VI6_DL_CTRL.CFM0 = 0

VI6_DL_CTRL.RLM0 = 0

VI6_DL_HDR_ADDR0 = <Set start address of Display list header>

[2] Use extended display list.

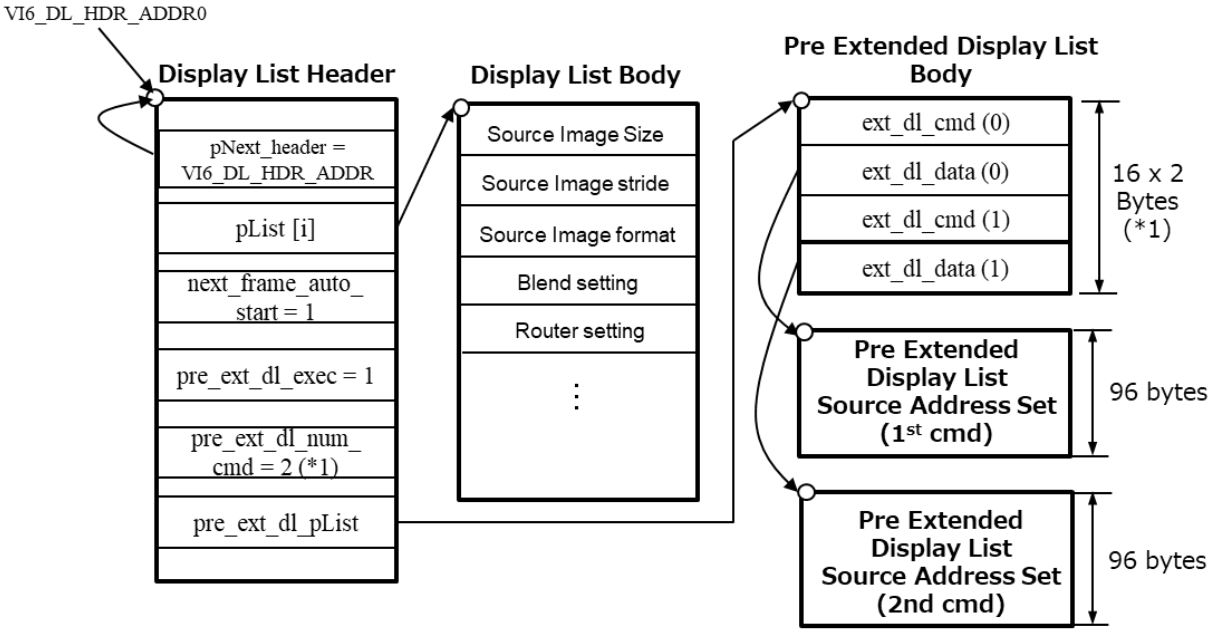
Register setting about extended display list is shown below.

VI6_DL_EXT_CTRL = H'0000 0221

[3] Contents of display list

Structure of display list to realize AUTO-DISP function is shown in blow figure. Contents of each display list is shown below.

- Display List Header: Table 60.36
- Display List Body: Prepare it in the same way with the case which AUTO-DISP is not used.
- Pre Extended Display List Body: Table 60.37
- Pre Extended Display List Source Address Set: Table 60.38




(*1) This example : Blending two images output from two channels of VIN, and displaying it.

**Figure 60.40 Structure of DisplayList of AUTO-DISP**

**Table 60.36 Display List Header for AUTO-DISP**

| Syntax | Value |
|---|---|
| num_list_minus1 | Set these bits in the same way with the case which AUTO-DISP is not used. |
| num_bytes [i] | Set these bits in the same way with the case which AUTO-DISP is not used. |
| pList [i] | Set these bits in the same way with the case which AUTO-DISP is not used. |
| pNext_header | Set same address of VI6_DL_HDR_ADDR0 for progressive display.<br>See Figure 60.42 and Figure 60.43 for interlace display.<br>Be sure to specify an address aligned with a 8-byte boundary (lower-order three bits are 0). |
| current_frame_int_enable | 0 |
| next_frame_auto_start | 1 |
| pre_ext_dl_exec | 1 |

| | |
|---|---|
| post_ext_dl_exec | 0 |
| pre_ext_dl_num_cmd | Set the number of connected VIN channels <1 to 4> |
| pre_ext_dl_pList | Set start address of extended display list body<br>Be sure to specify an address aligned with a 16-byte boundary (lower-order four bits are 0). |
| post_ext_dl_num_cmd | 0 |
| post_ext_dl_pList | 0 |

**Table 60.37 Pre Extended Display List Body for AUTO-DISP**

| Bit String | bits | Symbol | value |
|---|---|---|---|
| ext_dl_cmd | 63 to 40 | - | 0 |
| | 39 to 32 | Opcode | 2 (AUTO_DISP is selected) |
| | 31 to 14 | - | 0 |
| | 13 to 11 | VIN ch no | Set VIN channel number <0 ~ 3><br>0: for VIN0<br>1: for VIN1<br>2: for VIN8<br>3: for VIN9<br>4~7: Reserved |
| | 10 to 8 | RPF ch no | Set RPF channel number <0 ~ 4> |
| | 7 to 3 | - | 0 |
| | 2 | 1st field | 0: TOP field first (DSMRn.ODEV=1)<br>1: BOT field first (DSMRn.ODEV=0) |
| | 1 | Correction | Set 0 for progressive display.<br>See Figure 60.43 and Figure 60.44 for interlace display. |
| | 0 | Int | 0: Progressive display<br>1: Interlace display |
| ext_dl_data | 63 to 32 | pAddres_set | Set start address of Source Address Set<br>Be sure to specify an address aligned with a 16-byte boundary (lower-order four bits are 0). |
| | 31 to 0 | Reserved | 0 |

**Table 60.38 Pre Extended Display List Source Address Set for AUTO-DISP**

| Address | Contents |
|---|---|
| ext_dl_data [63:32] + 0 | Buffer1 TOP field address of Y/RGB |

| | |
|---|---|
| ext_dl_data [63:32] + 4 | Buffer1 BOTTOM field address of Y/RGB |
| ext_dl_data [63:32] + 8 | Buffer1 TOP field address of C0 |
| ext_dl_data [63:32] + 12 | Buffer1 BOTTOM field address of C0 |
| ext_dl_data [63:32] + 16 | 0 |
| ext_dl_data [63:32] + 20 | 0 |
| ext_dl_data [63:32] + 24 | 0 |
| ext_dl_data [63:32] + 28 | 0 |
| ext_dl_data [63:32] + 32 | buffer2 TOP field address of Y/RGB |
| ext_dl_data [63:32] + 36 | buffer2 BOTTOM field address of Y/RGB |
| ext_dl_data [63:32] + 40 | buffer2 TOP field address of C0 |
| ext_dl_data [63:32] + 44 | buffer2 BOTTOM field address of C0 |
| ext_dl_data [63:32] + 48 | 0 |
| ext_dl_data [63:32] + 52 | 0 |
| ext_dl_data [63:32] + 56 | 0 |
| ext_dl_data [63:32] + 60 | 0 |
| ext_dl_data [63:32] + 64 | buffer3 TOP field address of Y/RGB |
| ext_dl_data [63:32] + 68 | buffer3 BOTTOM field address of Y/RGB |
| ext_dl_data [63:32] + 72 | buffer3 TOP field address of C0 |
| ext_dl_data [63:32] + 76 | buffer3 BOTTOM field address of C0 |
| ext_dl_data [63:32] + 80 | 0 |
| ext_dl_data [63:32] + 84 | 0 |
| ext_dl_data [63:32] + 88 | 0 |
| ext_dl_data [63:32] + 92 | 0 |

How VSPD replaces source address registers is described as below. VSP reads extended display list body specified in Table 60.37 and source address set specified in Table 60.38 during the term of “Ext-DL” specified in Figure 60.41. And VSPD replaces source address registers by step 1 to step 7 as below.

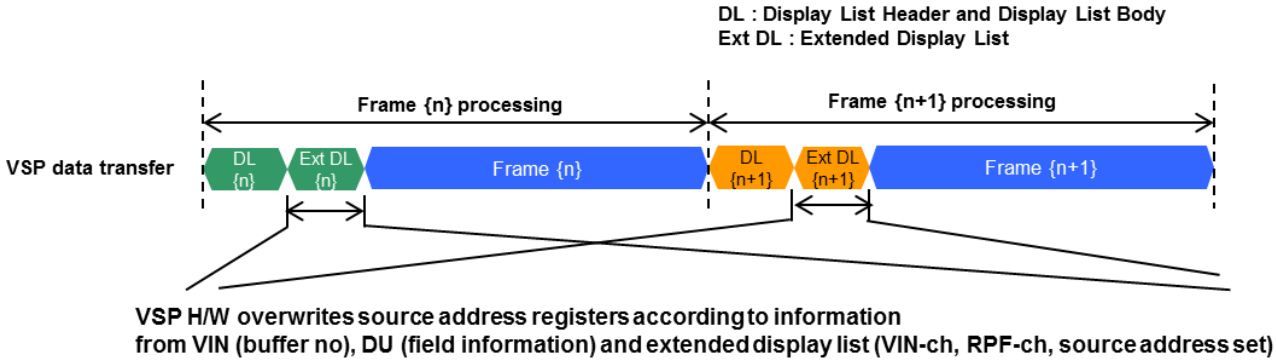



**Figure 60.41 Timing to read extended display list**

1. VSP reads extended display list body from external memory
2. VSP gets latest buffer information from VINs.

Note: information is updated at only 1st field in case of interlace.

3. VSP gets field information from DU.
4. VSP reads address information of the latest buffer of the VIN ext_dl_cmd [13:11] indicates from source address set specified in Table 60.38.
5. VSP selects source address from TOP (*1) address or BOTTOM (*1) address in source address set according to ext_dl_cmd [2:0].

   Note: Set start address of progressive image in both TOP address bits and BOTTOM address bits in extended display list source address set in case of progressive display.
6. VSP overwrites source address registers for the RPF specified in ext_dl_cmd [10:8] at source address read from external memory.

   Note: C1 address and Alpha address registers for the RPF are overwritten by 0 value for the RPF specified in ext_dl_cmd [10:8].
7. VSP repeats step 4 to step 6 for number of times of pre_ext_dl_num_cmd.

(*1) In this document, TOP field is used as field image including 0th, 2nd, 4th … line. BOTTOM field is used as field image including 1st, 3rd, 5th … line.

[4] Progressive and Interlace

There are restrictions about combination of interlace and progressive for input/output as shown in Table 60.39. When VIN captures interlace image, DU should display image in interlace method. When VIN captures progressive image, DU should display image in progressive method.

[4-1] Particular restriction for VIN registers

- Set full interlace mode in VIN register (VnMC.IM[1:0]=3) in case of interlace display.

- No particular restriction is required in case of progressive display.

[4-2] Particular restriction for DU registers

- No particular restriction is required

[4-3] Particular setting and status of VSP

- Set <2 x source stride> to source stride register (VI6_RPFn_SRCM_PSTRIDE) in case of interlace display.

- VI6_STATUS.FLDST0 in section 60.2.4.11 indicates field information of last displayed frame.

[4-4] The number of Display List Header and Pre Extended Display List body

One display list header and one pre extended display list body is needed for progressive display. All frames can be controlled by one display list. Set same value of VI6_DL_HDR_ADDR0 to pNext_haeder.

Field signal from DU is not stable in first two fields. Therefore, to display first two fields by appropriate field (TOP or BOTTOM), set bit1 in ext_dl_cmd as shown in Figure 60.42 for TOP field first and Figure 60.43 for BOTTOM field first. Three display list headers and three pre extended display list bodies are needed to adjust first two fields for interlace display.

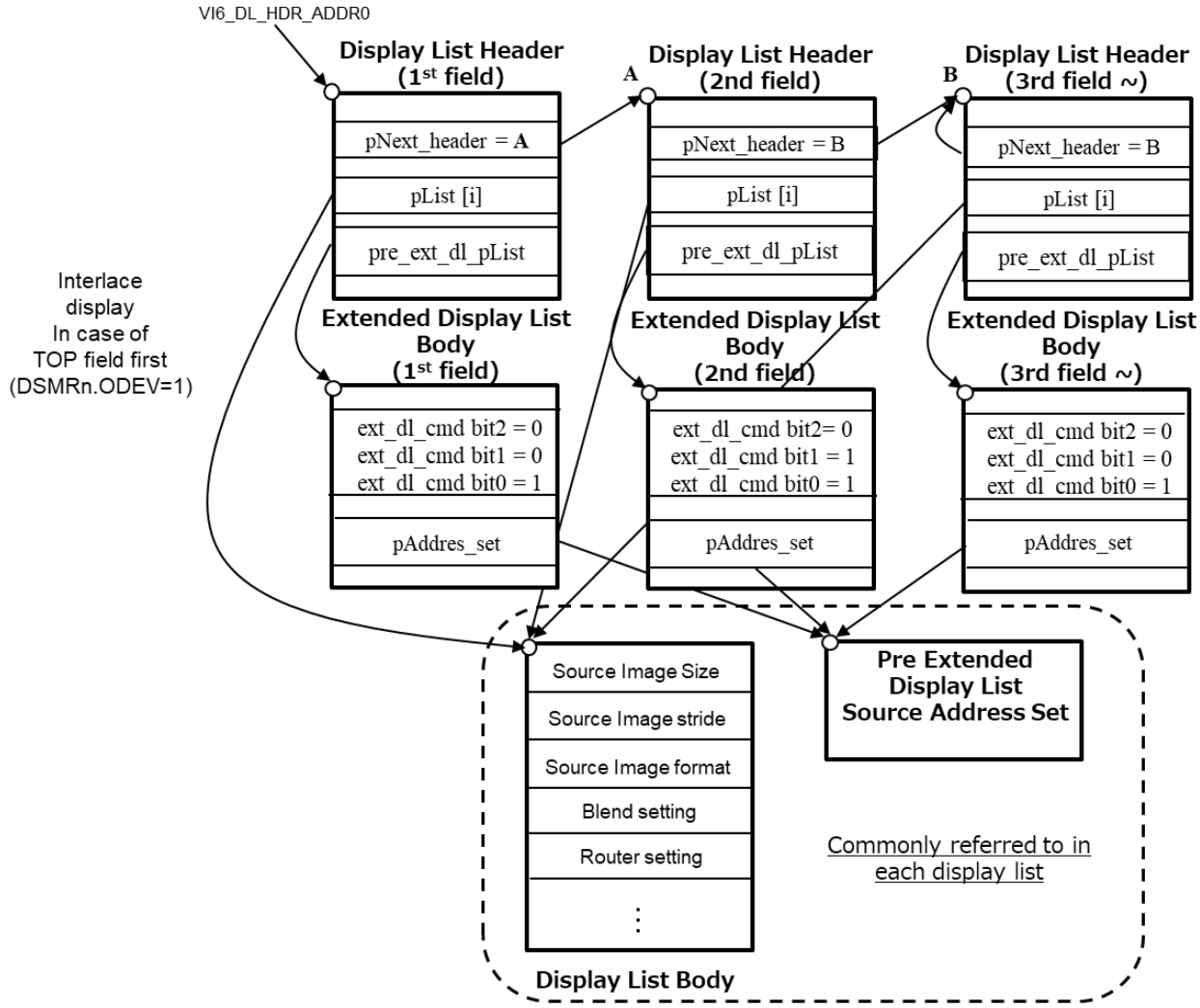



**Figure 60.42 Correcting first two fields in case of TOP field first**

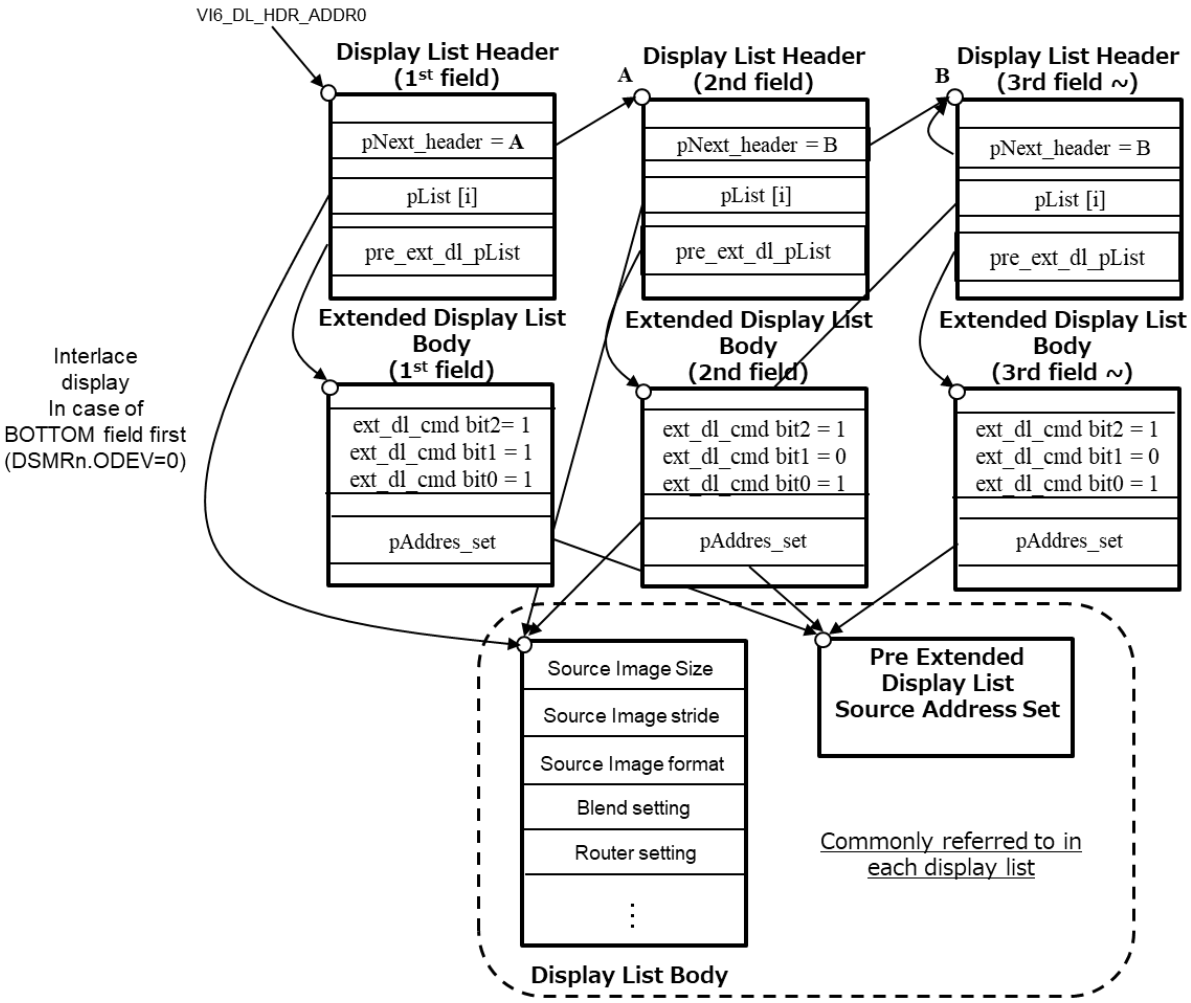



**Figure 60.43 Correcting first two fields in case of BOTTOM field first**

[5] Startup image before first image is output from VIN

VSP reads image data in buffer3 address specified in extended display list source address set until VIN completes writing first frame's image into buffer1. Therefore, prepare startup image in buffer3. This image is overwritten after VIN is activated.

[6] Procedure of activation

Activate VIN with continuous frame capture mode (VnFC.CC=1). After that, activate VSPD and DU by procedure of Figure 60.34. Activating VIN and VSPD should be after initializing both VIN and VSPD. The operation is not guaranteed if VIN is initialized during VSPD operation.The reverse is also true.

[7] Restrictions

There are several restrictions as shown below.

[7-1] Restriction about combination of interlace and progressive for input/output

There are restriction about combination of interlace and progressive for input/output as shown in below table.

**Table 60.39** Restriction about combination of interlace and progressive for input/output

| Video IN | Display | Support |
|---|---|---|
| Progressive | Progressive | √ |
| Progressive | Interlace | |
| Interlace | Progressive | |
| Interlace | Interlace | √ |

[7-2] Restriction about frame rate of input and output

Frame rate of VIN should be equal to or smaller than frame rate of display.

[7-3] Restriction about selecting VIN channel.

There is following restriction about selecting VIN channel.

VSPD can read VIN0, VIN1, VIN8, VIN9 output image when this function is used.

There is no restriction about selecting VIN channel.

### 60.3.7 Start Reservation for VSPX

This section shows sequence of software when start reservation is used. The way of start reservation is for VSPX.

Software can reserve starting VSPX processing of one frame is completed. Start reservation can remove interrupt response time and software overhead by reserving execution of VSPX before completion of previous frame. Use frame end interrupt (VI6_WPF0_IRQ_ENB.FRE) for VSPX.

(a) below shows software sequence of start reservation.

(b) below shows software sequence to manage the number of interrupt generation.

(a) Software sequence of start reservation

[1] ~ [9] shown in  shows S/W flow to set registers for start reservation of VSPX

[1] Wait until 0 is read from VI6_CMD0.STRCMD to confirm that starting WPF0 is not reserved.

[2] Wait until 0 is read from VI6_STATUS.SYS0_ACT to confirm WPF0 is IDLE.

[3] Set registers with two planes for frame 0 (1st frame) except DLH (Display List Header) address. Vi6_DL_HDR_ADDR0 has two planes.

[4] Set header address of display list for frame 0 to VI6_DL_HDR_ADDR0. Then VI6_CMD0.UPDHDR is asserted automatically by VSP2 to indicate downloading registers with two planes is reserved.

[5] Activate WPF0 for frame 0 (1st frame) by writing H'0000 0001 to VI6_CMD0.

If next frame's execution of VSPX is needed, go to step [6].

[6] Wait until start reservation is accepted and registers with two planes are downloaded into H/W side. If 0 is read from VI6_CMD0, it means that starting WPF0 is accepted and register with two planes are downloaded into internal registers H/W side refers.

[7] Set registers with two planes for next frame (fr{m}) except DLH (Display List Header) address.

[8] Set header address of display list for next frame (fr{m}) to VI6_DL_HDR_ADDR0. Then VI6_CMD0.UPDHDR is asserted automatically by VSP2 to indicate downloading registers with two planes is reserved.

[9] Reserve activating WPF0 for next frame (fr{m}) by writing H'0000 0001 to VI6_CMD0.

If next frame's execution of VSPX is needed, go to step [6].

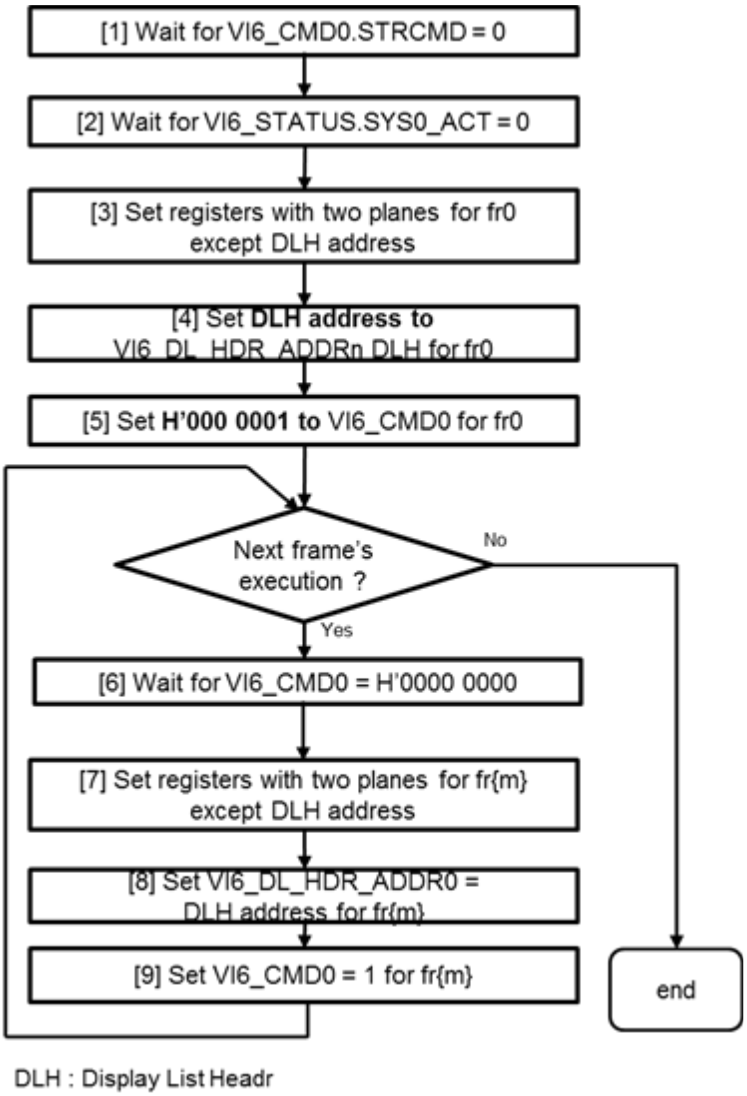



**Figure 60.44 Flow chart of setting registers by S/W in case of VSPX start reservation**

(b) Software sequence to manage the number of interrupt generation.

The way of using interrupt is shown below.

[1] Set 1 to the VI6_WPF0_IRQ_ENB.FREE to use frame end interrupt (VI6_WPF0_IRQ_STA.FRE)

[2] Write 0 to VI6_WPF0_IRQ_STA.FRE to clear status after interrupt processing as shown in Figure 60.45.

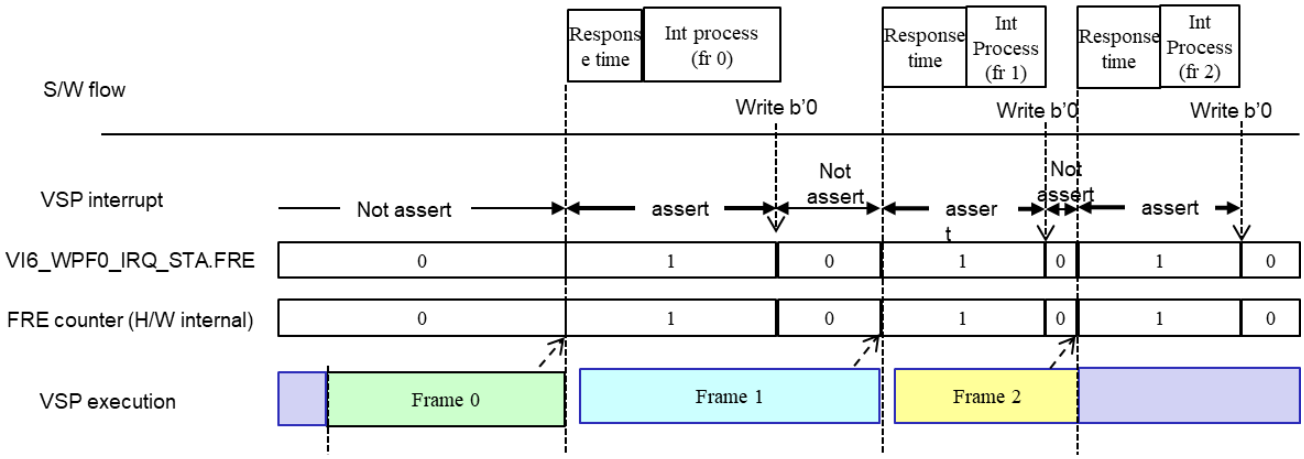



**Figure 60.45 Timing of interrupt processing by S/W in case of VSPX start reservation (Case1)**

Be careful following points as shown in Figure 60.46.

[1] Set 0 to VI6_WPF0_IRQ_STA.FRE for the number of interrupt generation to clear VI6_WPF0_IRQ_STA.FRE.

[2] The VSPX H/W can count at most two of interrupt generation. If three times of interrupts occur before clearing the interrupt factor, interrupt counter overflows. Therefore, wait to reserve starting VSPX of frame {n+2} until interrupt factor of the frame {n} is cleared.

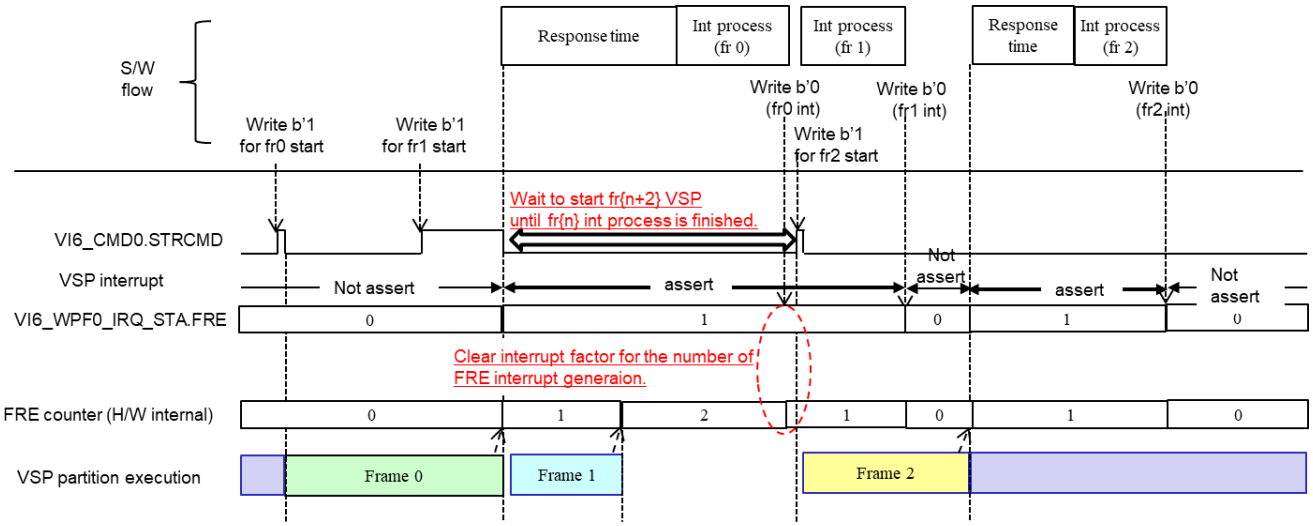



**Figure 60.46 Timing of interrupt processing by S/W in case of VSPX start reservation (Case2)**

### 60.3.8 VSPX Setting and Operation

#### 60.3.8.1 VSPX function and restriction

VSPX inputs RAW image and ISP-Config data from external memory and sends to ISP. In other word, VSPX functions as a bus interface for data input of ISP. There are following restrictions.

- RAW image
  - Up to 2 RAW image data can be read to support multi exposures.
  - RPF0, RPF1 can be used for reading RAW image. Use RPF0 in case of 1 RAW image, and use RPF0 and RPF1 in case of 2 RAW images.
  - RAW8/RAW10/RAW12/RAW14/RAW16/RAW20/RAW24 can be read for RPF0
  - RAW8/RAW10/RAW12 canbe read for RPF1
  - All RPFs must have the same image size.
- ISP Config data
  - It can be transferred in the same frame as RAW image.
  - RPF0 can be used for reading ISP config data.
  - VSPX reads the number of bytes corresponding to the horizontal size x vertical size as the ISP config data.
  - Set 19 (4 byte/pix) to VI6_RPF0_INFMT.RDFMT. VSPX transfers the number of bytes equal to VI6_RPF0_SRC_BSIZE.BHSIZE x VI6_RPF0_SRC_BSIZE.BVSIZE x 4 to the ISP as ISP-Config data.
  - Set 19 to VI6_IIF_CTRL.
  - When transferring ISP-Config data, the horizontal size (VI6_RPF0_SRC_BSIZE.BHSIZE, VI6_RPFn_SRC_ESIZE.EHSIZE) should be set in 2-pixel units. If the required ISP Config data is less than the size calculated by VI6_RPF0_SRC_BSIZE.BHSIZE x VI6_RPF0_SRC_BSIZE.BVSIZE x 4, store dummy data in the latter half. For details of format of ISP Config data, refer to the ISP manual.
  - ISP Config data and RAW data cannot be transferred to ISP in the same frame of VSPX processing. Frame of VSPX for RAW data transfer should be executed after ISP-Config data transfer frame. By setting 1 to next_frame_auto_start in Display List Header for ISP-config frame, it is possible for S/W to active VSPX only once per two frames.

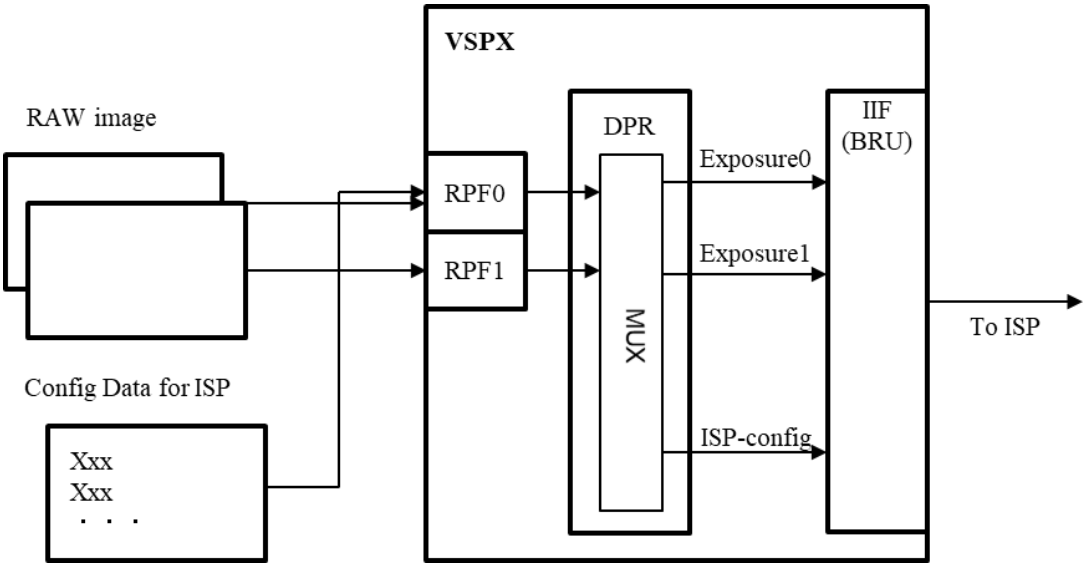



**Figure 60.47 VSPX data flow**

In the usecase where the output image from VIN to the memory is input to VSPX, VSPX can start a frame processing after waiting for the wake-up assertion from VIN.

- Features of VIN wake-up
  - Wake-up counter per each VIN channel (Total 16 channels VIN 0 to VIN15)
  - Enabling/disabling wake-up assertion of VIN is in the VIN register
  - When wait_wup in display list is 1, VSPX waits for the wake-up from the VIN channel enabled by corresponding bit in VI6_DL_VIN_WUPEN0, and start the frame operation.
    - Ex. If wait_wup is 1 and VI6_DL_VIN_WUPEN0.WE3 is 1, VSPX waits for assertion of VIN3 wake-up, and start the frame operation after that.
  - The maximum outstanding number of wake-up is 15 (4-bit wake-up counter)
    - Wake-up counter increments by assertion of wake-up from each VIN.
    - Wake-up counter decrements when the processing of the frame in which wait_wup in display list is 1 and the corresponding bit of VI6_DL_VIN_WUPEN0 is set to 1 is started.
  - Error detection
    - Wake-up error (VI6_WUP0_IRQ_STA.VWERn) occurs when wake-up counter (VI6_DL_VIN_WUPCNT0.WUPCNTn / VI6_DL_VIN_WUPCNT1.WUPCNTn) is 15 and wake-up from VINn asserts. Wake-up cannot be recognized correctly when error occurs, so user needs to control VSPX so that this error does not occur.
    - Ignore any errors other than the VIN channel that output image that VSPX inputs. Interrupt can be masked by the interrupt enable register.

### 60.3.8.2 VSPX Related Registers

This section shows registers that can be set in VSPX. For VSPX, the registers shown in Table 60.40 can be set. All registers except Table 60.40 are reserved area for VSPX.

**Table 60.40 Registers VSPX supports**

| Classification | Register Name | Restriction |
|---|---|---|
| General Control Registers | VI6_CMD0 | |
| | VI6_CLK_DCSWT | |
| | VI6_SRESET | |
| | VI6_STATUS | |
| | VI6_WPF0_IRQ_ENB | UNDE bit cannot be used. |
| | VI6_WPF0_IRQ_STA | UND bit cannot be used. |
| | VI6_WUP0_IRQ_ENB | |
| | VI6_WUP0_IRQ_STA | |
| | VI6_WPF0_WDT | |
| Display List Control Registers | VI6_DL_CTRL | |
| | VI6_DL_HDR_ADDR0 | |
| | VI6_DL_SWAP0 | |
| | VI6_DL_BODY_SIZE0 | |
| | VI6_DL_HDR_REF_ADDR0 | |
| | VI6_DL_VIN_WUPCNT0 | |
| | VI6_DL_VIN_WUPCNT1 | |
| | VI6_DL_WUPCNT0 | |
| DPR Control Registers (*1) | VI6_DPR_RPF0_ROUTE | Set D'23 to RT [5:0] (BRUin0/Exposure0) when RPF0 reads raw-image.<br>Set D'26 to RT [5:0]: (BRUin3/ISP-Config) when RPF0 reads ISP-config. |
| | VI6_DPR_RPF1_ROUTE | Set D'24 to RT [5:0] (BRUIN1/Exposure1) when RPF1 read raw-image.. |
| | VI6_DPR_WPF0_FPORCH | Set D'5 to FP_WPF0 [5:0] |
| | VI6_DPR_BRU_ROUTE | Set D'56 to RT [5:0] (WPF0).<br>Set D'1 to IIFSEL. |
| RPF Control Registers (RPF0, 1) | VI6_RPFn_SRC_BSIZE (n=0, 1) | Set same size for all RPFs. |
| | VI6_RPFn_SRC_ESIZE (n=0, 1) | It should be same setting with VI6_RPFn_SRC_BSIZE |
| | VI6_RPFn_INFMT (n=0, 1) | Set 0 (RAW 1-byte/pix), 6 (RAW 2-byte/pix) or 19 (RAW 4-byte/pix) to RDFMT of RPF0 for RAW data.<br>Set 0 (RAW 1-byte/pix) or 6 (RAW 2-byte/pix) to RDFMT of RPF1. |

| | | |
|---|---|---|
| | | Set 19 to RDFMT of RPF0 for ISP config data.<br>All bits except RDFMT are reserved bits. |
| | VI6_RPFn_DSWAP (n=0, 1) | All bits except P_LLS, P_LWS, P_WDS and P_BTS are reserved bits. |
| | VI6_RPFn_ALPH_SEL (n=0, 1) | Set H'0000 0000 in the register. |
| | VI6_RPFn_SRCM_PSTRIDE (n=0, 1) | |
| | VI6_RPFn_SRCM_ADDR_Y( n=0, 1) | |
| IIF Control Registers | VI6_IIF_CTRL | Set 0x13 in the register before starting VSPX |
| WPF Control Registers (WPF0) | VI6_WPF0_SRCRPF | All bits except RPF0_ACT [1:0] and RPF1_ACT [1:0] are reserved bits. |

(*1) IIF is a function of BRU, and works by setting 1 to VI6_DPR_BRU_ROUTE.IIFSEL. Therefore, the BRU must be connected to the DPR to use IIF.

#### 60.3.8.3 Operation in case of using wake-up of VIN

The figure below shows the connection of the VIN wake-up signals. Wake-up signals are shared by multiple VSPX and other modules like IMR. In wake-up control, the operation method in the case of Multi-Exposure and the case of Multi-Sensor is shown below. There is also a use-cause that wake-up asserts at a specific vertical position on the image (vertical segmentation). The operation method at the vertical segmentation is described in section 60.3.8.3.

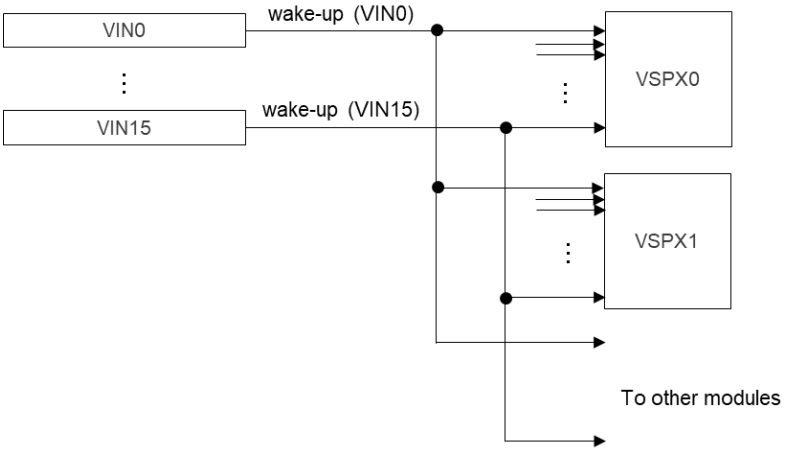



**Figure 60.48 wake-up connection**

The startup method for processing the images output from two sensors via VIN in a time-sharing method using one VSPX is explained. As an example, one sensor (sensor0) is a multi-exposure of VIN0, VIN1 and VIN2, the other sensor (sensor1) is the case of single-exposure of VIN7.

- Settings before operation (sensor0)

  1. Set wake-up related registers of VIN0, VIN1, and VIN2. It is also possible to set wake-up related registers of only the VIN that memory output is completed lastly. Refer to VIN User's Manual related to VIN registers.

  2. Clear error status of VWER and wake-up counters about VIN0, VIN1 and VIN2

     1.1 VI6_WUP0_IRQ_STA = 32'H 0000 FFF8
     (Clear VWER status of VIN0, VIN1 and VIN2, and keep other VIN wake-up status)
     R. VI6_DL_VIN_WUPCNT0 = 32'H FFFF F000
     (Clear wake-up counter of VIN0, VIN1 and VIN2, and keep other VIN wake-up counter)

  3. Enable interrupt status of VWER of VIN0, VIN1 and VIN2. Because wake-up signals are common between each VSPX and other modules like IMR, wake-up counter of other VIN may increment by the VIN's timing. Therefore, disable interrupt enable of VWER for other VIN channels unrelated to the VSPX.

     3.1 Set 1 to VWER0E, VWER1E and VWER2E in VI6_WUP0_IRQ_ENB.

- Settings for 1-frame processing (sensor0)

  1. Enable waiting wake-up of VIN0, VIN1 and VIN2. Then VSPX waits for wake-up of VIN0, VIN1 and VIN2 before starting one frame processing.

     1.1 VI6_DL_VIN_VIN_WUPEN0 = 32'H 0000 0007

  2. Set 1 to wait_wup in display list header to wait wake-up

- Settings before operation (sensor1)

1. Set wake-up related registers of VIN7. Refer to VIN User's Manual related to VIN registers.

2. Clear error status of VWER and wake-up counters about VIN7

   1.2 VI6_WUP0_IRQ_STA = 32'H 0000 FF7F
   (Clear VWER status of Vin7, and keep other VIN wake-up status)
   s. VI6_DL_VIN_WUPCNT0 = 32'H 0FFF FFFF
   (Clear wake-up counter of VIN7, and keep other VIN wake-up counter)

3. Enable interrupt status of VWER of VIN7. Because wake-up signals are common between each VSPX and other modules like IMR, wake-up counter of other VIN may increment by the VIN's timing. Therefore, disable interrupt enable of VWER for other VIN channels unrelated to the VSPX.

   3.1 Set 1 to VWER7E in VI6_WUP0_IRQ_ENB.

- Settings for 1-frame processing (sensor1)

  1. Enable waiting wake-up of VIN7. Then VSPX waits for wake-up of VIN0, VIN1 and VIN2 before starting one frame processing.

     1.1 VI6_DL_VIN_VIN_WUPEN0 = 32'H 0000 00080

  2. Set 1 to wait_wup in display list header to wait wake-up

- Initialization

  If VSPX is initialized, the wake-up information of all channels (VIN0 to 15) is initialized.

#### 60.3.8.4 Operation of segment processing.

**(1) Segment Processing (Vertical)**

VIN and VSPX have a function of wake up notification. VIN can notify VSPX of irburstg irburs the specified number of lines by wake up signal without S/W intervention.VSPX waits for the notification from VIN and starts to read the number of lines after the wake up notification. VSPX can read image data output from VIN by shorter latency than one frame by wake up notification as shown in Figure 60.49.

When "Wake up" function is used, it is necessary to process input image data by dividing the image in the vertical direction. The way of dividing in the vertical direction depends on the setting of "Wake Up Line" in VIN register. Hereafter, one area after dividing one frame's image in the vertical directions is called segment. When processing per segment units, display list with header described in section 60.3.3 has to be used.

An example of dividing one frame image into segments is shown in Figure 60.49. Divide one frame in the vertical direction by wake up line set in VIN register like Figure 60.49. Overlapping pixels in adjacent segments may be needed as shown in Figure 60.49. Regarding the number of pixels of overlapping pixels, refer to ISP User's Manual.

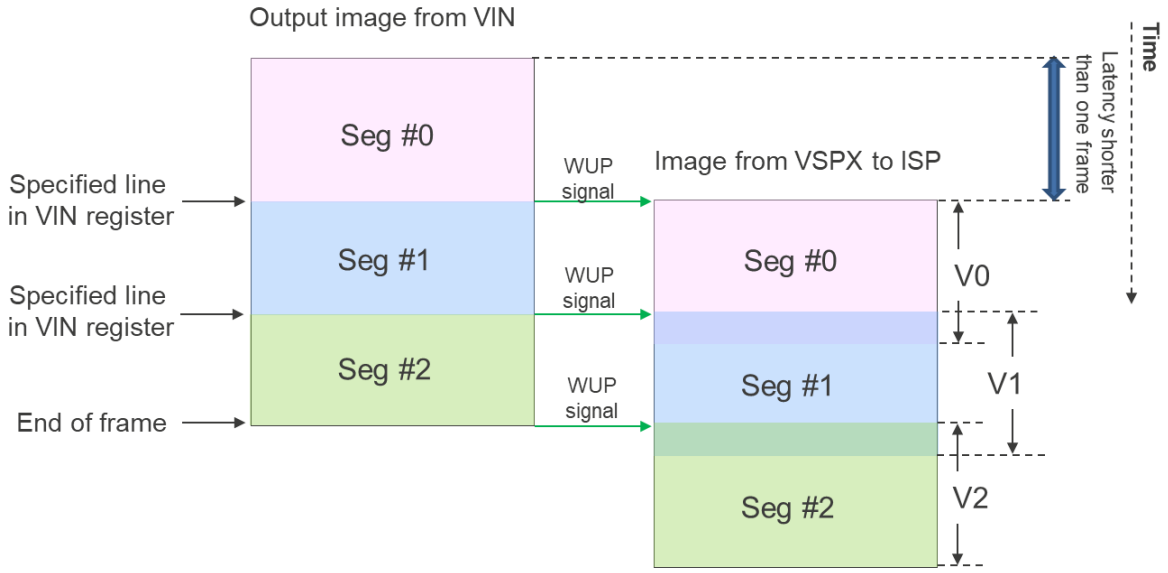



**Figure 60.49** Wake up notification and dividing one frame of VSPX input image into segments (Vertical)

The setting method of display list of each vertical segment is shown in Figure 60.50. Vertical segment processing needs the following features as shown in Figure 60.51.

[1] S/W only activate VSPX once per frame

[2] VSPX waits for the wake up notification from VIN before starting data processing

[3] Frame end interrupt asserts only once in one frame

For realizing the features, set registers related to display list as below.

(1) Store all segments' display list in external memory.

(2) In order to get better performance, don't set registers which are not updated per segment in display body from 2nd segment to last segment.

(3) Set 1 to next_frame_auto_start in display list header except last segment. Then S/W has to activate VSPX only once for 1st segment.

(4) Set 1 to current_frame_int_enable in display list header for only last segment. Set H'0000 0002 to VI6_WPF0_IRQ_ENB to enable Display List Frame End (DFE) interrupt and to disable Frame End (FRE) interrupt. Then DFE interrupt occurs after all segments' processing is finished by VSPX-H/W.

(5) Set 1 to wait_wup in display list header. Segment with wait_wup=1 waits for processing to start until wake up signal from VIN asserts.

(6) Set below to VI6_DL_CTRL

- VI6_DL_CTRL.DLE0 = 1 (Use display list)
- VI6_DL_CTRL.NH0 = 0 (Use Header Mode)
- VI6_DL_CTRL.RLM0 = 1 (Display List Header Address <VI6_DL_HDR_ADDR0> is not downloaded into H/W side at segment boundary)

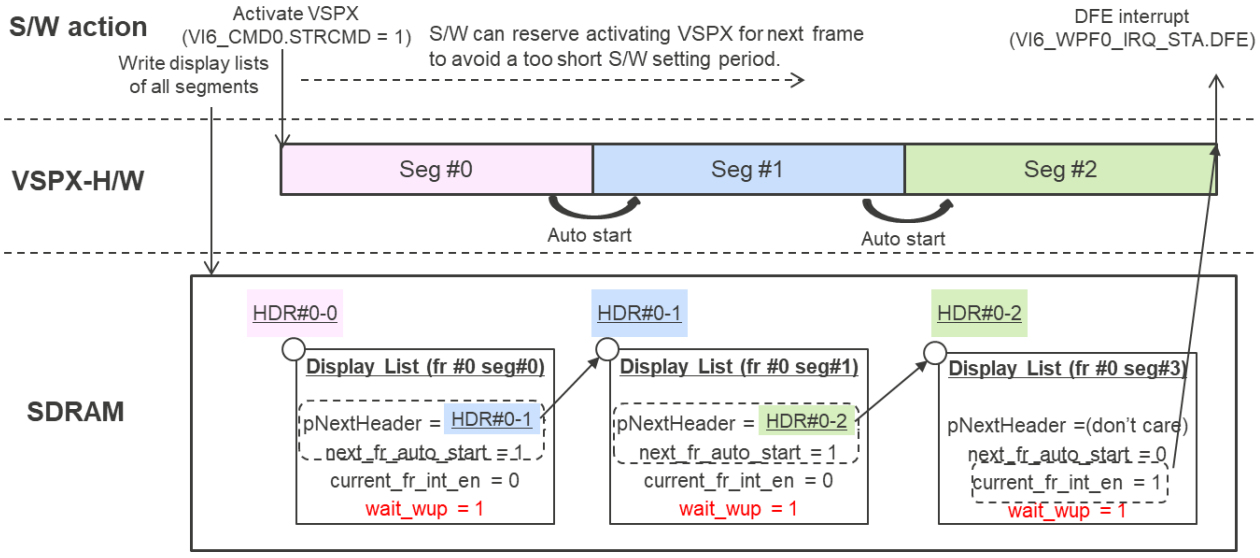



**Figure 60.50 Method of Display List for segment (Vertical)**

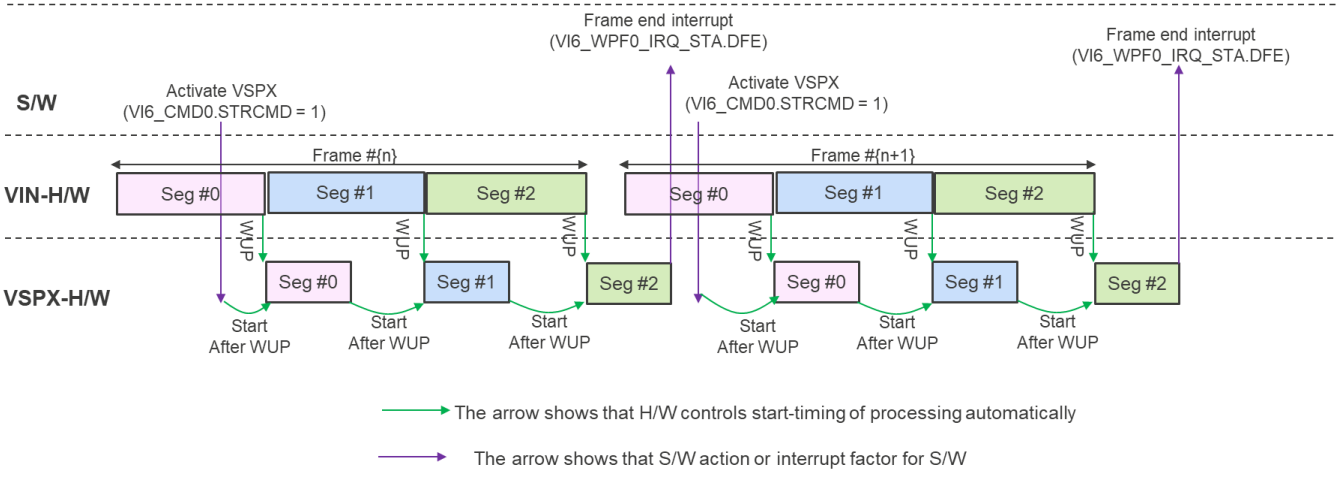



**Figure 60.51** Processing flow of segment (Vertical)

### (2) Operation flow

Operation Flow in case of segment processing is shown below.

[1] Start procedure

1. Set 0 to relevant bit in VI6_WUP0_IRQ_STA and VI6_DL_VIN_WUPCNT[0-1]. (*1)
2. Activate ISP. Then the ISP waits for VSPX to output first pixel to ISP.
3. Stor all display lists of all segments of one frame in external memory.
4. Read VI6_CMD0.STRCMD. If VI6_CMD0.STRCMD is 0, S/W can activate VSPX by VI6_CMD0.STRCMD.
5. Activate VSPX.
6. Activate VIN by continuous frame capture mode. (*1)

7. Waits for VIN to assert frame end interrupt. (*1) (*2)
8. Repeat step 3 to 5 per one frame.

[2] Stop procedure

1. Stop ISP after receiving VSPX frame end interruption corresponding to last activation of VSPX. Then, the ISP stop the operation.
2. Stop VIN. (*1)

(*1) Take step 1, 6 and 7 of start procedure and step 2 of stop procedure only when wake up notification is used.

(*2) Since step 6 depends on the use case and S/W structure, this procedure can be deleted depending on the use case and S/W structure.

#### 60.3.8.5 Restrcition in Case of VSPX Operation

Software reset (VI6_SRESET.SRST0) cannot be applied to ensure data transfer to ISP.

## 60.4 Usage Notes

### 60.4.1 Assignment in Memory Space

Make sure that VSP2 register space shall be mapped to Non-Cache region.

### 60.4.2 Limitations on Software Reset and Module Standby

Refer to Usage Notes of FCP User's Manual

### 60.4.3 Input Image Size

Table 60.41 is a list of input size specifications.

**Table 60.41 List of Input Size Specifications**

| Module | Min. Input Size | Max. Input Size | Restrictions on Setting Unit |
|---|---|---|---|
| RPF | 1 (horizontal)<br>x 1 (vertical) pixel | 8190 (horizontal)<br>x 8190 (vertical) pixels | YcbCr422: 2-pixel units horizontally and 1-pixel units vertically.<br>YcbCr420: 2-pixel units both horizontally and vertically.<br>Other formats: 1-pixel units both horizontally and vertically.<br>Notes: 1. When the 1-bpp alpha plane[*1] is input, the size can always be specified in 8-pixel units both horizontally and vertically regardless of the input format.<br>T. These restrictions including note 1 are applied to the following.<br>VI6_RPFn_SRC_BSIZE<br>VI6_RPFn_SRC_ESIZE |
| UIFn | - | - | Refer to DISCOM section. |
| ILV | 1 (horizontal)<br>x 1 (vertical) pixel | Side by Side Mode<br>4095 (horizontal)<br>x 8190 (vertical) pixels<br>Line by Line Mode<br>8190 (horizontal)<br>x 4095 (vertical) pixels<br>Pixel by Pixel Mode<br>4095 (horizontal)<br>x 8190 (vertical) pixels<br>Doubler Mode<br>4095 (horizontal)<br>x 4095 (vertical) pixels | 1-pixel units both horizontally and vertically. |
| IIF | 128 (horizontal) x<br>32 (vertical) pixels | 5120 (horizontal) x<br>4096(vertical) pixels | 1-pixel units both horizontally and vertically. |
| BRU | 1 (horizontal)<br>x 1 (vertical) pixel | 8190 (horizontal)<br>x 8190 (vertical) pixels | 1-pixel units both horizontally and vertically. |
| LIF | 128 (horizontal)<br>× 96 (vertical) | 8190 (horizontal)<br>x 8190 (vertical) pixels | VI6_LIFn_CTRL.CFMT is 1: 2-pixel units horizontally and 1-pixel units vertically.<br>Other formats: 1-pixel units both horizontally and vertically. |
| SMG | 128 (horizontal)<br>× 96 (vertical) pixel | 8190 (horizontal)<br>× 8190 (vertical) pixels | VI6_LIFn_CTRL.CFMT is 1: 2-pixel units horizontally and 1-pixel units vertically.<br>Other formats: 1-pixel units both horizontally and vertically. |
| WPF | 1 (horizontal)<br>x 1 (vertical) pixel | 8190 (horizontal)<br>x 8190 (vertical) pixels | YcbCr422: 2-pixel units horizontally and 1-pixel units vertically.<br>YcbCr420: 2-pixel units both horizontally and vertically.<br>Other formats: 1-pixel units both horizontally and vertically. |

| | | | |
|---|---|---|---|
| | | | Note 1: This restriction on the WPF only applies to the WPF output size. The WPF input size should be specified in 1-pixel units. |

The most important restriction shown in Table 60.41 is the setting unit. Make appropriate register settings so that the size of the image input to each module does comply with setting units and does not exceed the limits shown in Table 60.41.

### 60.4.4 Output Image Size

The size of the output from each WPF is determined by the results of processing in the modules connected with the DPR. As shown in Figure 60.1 to Figure 60.2, the data input to the VSP2 is sent to the WPF output modules through the RPF and the modules connected with the DPR. When there is no processing that changes the image size through this data path, the WPF output size is the same as the RPF input size. Table 60.42 is a list of the processing that changes image size.

**Table 60.42 Image Processing that changes image size**

| Module | Function*1 | Related Register | Size of Output from Module |
|---|---|---|---|
| WPF | Input size clipping | VI6_WPF0_HSZCLIP<br>VI6_WPF0_VSZCLIP | When this function is disabled, the input size and the output size are the same. When this function is enabled, the output is in the following size.<br>Horizontal output size:<br>VI6_WPF0_HSZCLIP.HCL_SIZE setting<br>Vertical output size:<br>VI6_WPF0_VSZCLIP.VCL_SIZE setting |
| ILV | 3D format conversion | VI6_ILV_CTRL | Side by side mode or Pixel by pixel mode<br>Horizontal : Input size× 2, Vertical : Input size<br>Line by line mode<br>Horizontal : Input size, Vertical : Input size x 2<br>Doubler mode<br>Horizontal : Input size x 2, Vertical : Input size x 2 |
| LIF | Padding line | VI6_LIF0_CTRL | When padding with dummy lines is disabled, output size is same with Input size.<br>When padding with dummy lines is enabled:<br>Horizontal output size: input size<br>Vertical output size: VI6_LIF0_PADLN_SIZE setting |

The input image size can be changed only with the modules and functions shown in Table 60.42. With the other modules and functions, the input size and output size are the same. Accordingly, after module connections with the DPR are determined, the VSP2 output size can also be determined through the following steps.

1. The image size (VI6_RPFn_SRC_ESIZE) read from the external memory and sent to the DPR by the RPFn is the initial value.
2. When the module connected with the DPR is not a module (function) shown in Table 60.42, the output size from the module is the same as the input size; the image size does not need to be updated.
3. When a module connected with the DPR is a module (function) shown in Table 60.42, the output image size should be updated to the size shown in the table, which should be used as the input image size for the module connected behind it.
4. When the final image size at the WPF0 is determined, this size is the VSP2 output size

Make appropriate register settings in each module so that the VSP2 output image size determined as shown in the figure does not violate the restrictions shown in Table 60.42.

### 60.4.5 Configuration of VSP2

Table 60.43 show the configuration of each type of VSP2 unit on this LSI.

All the registers and functions listed in [Module] category of Table 60.43 should be treated as un-implemented functions in each VSP2 unit. Do not set any registers related to the un-implemented function without any statement.

For VSPX, refer to 60.3.8.1

**Table 60.43 VSPD configuration**

| Category | Configuration | Restricted contents |
|---|---|---|
| Function | RPF/ OSD-CLUT | OSD-CLUT is available only in RPF2. OSD-CLUT is not available in RPF0, RPF1, RPF3 and RPF4.<br><br>Related registers.<br><br>VI6_RPFn_INFMT.RDFMT, VI6_CLUT0_TBL*, VI6_CLUT1_TBL*,VI6_CLUT3_TBL*, VI6_CLUT4_TBL* |

### 60.4.6 Usage of DISCOM

When DISCOM is used in VSPD, setting registers in DISCOM is needed. See Subsection of this document.

### 60.4.7 Performance of VSP2

VSPX and VSPD performance is shown in this section. VSP performance shown in below is based on evaluation under the condition that only one IP is operated.

#### 60.4.7.1 Performance of VSPD

In this section, the maximum dot clock frequency is the one VSPD can support regardless of display type. Therefore, it may be higher than the maximum dot clock frequency specified by an particular display type. Please refer to DU User's Manual and Electrical Characteristics to grasp the maximum dot clock frequency.

250 Mpix/s process rate (support up to 300MHz in case of DU display) per 1 set of VSPD.

#### 60.4.7.2 Performance of VSPX

720 Mpix/s process rate per 1 set of VSPX.

To meet the performance, it is required to apply QoS fix-level 4 or QoS fix-level 5 to VSPX. The execution of the VSPX is up to at most 520 frames per second.

### 60.4.8 Legend of Terms

This section describes the configurations of VSP2 on this LSI. Descriptions in this section show the full-function specifications of the VSP2 IP. Some of the functions described in this section are not available on VSP2 with restrictions written here.

#### (1) Terminology

This section explains the terminology for VSP2.

Virtual RPF

There is an image compositing sub module named as BRU which can execute image blending or raster operation. The BRU has 5 input ports for compositing multiple images. There must be a RPF module as the start of data processing. Each RPF can be a source of data input to BRU. In addition to these RPF modules, the BRU has a function which generates monochrome color image internally. This functionality is useful because the data access to an external memory is not required. This function is called as virtual RPF.

Refer to section 60.2.7.1, 60.2.9.2, 60.2.9.3 and 60.2.9.4 for virtual RPF.

Target WPF

The VSP2 has five RPFs (RPF 0 to 4) and four WPFs (WPF0 to 1) and execute image processing. One data path consists of more than one RPF, one WPF and other image functional modules if necessary. The WPF which is reached from specific RPF(s) in the data path is called as target WPF for the RPF(s). The number of target WPFs is always one for the data path.

Source RPF

The data source in the data path for the specific WPF is called as source RPF for that WPF. The number of source RPFs is not always one, may be more than one for image blend operation or raster operation.

Layer

In case of image composite operation such as image blending or raster operation for multiple images, there is hierarchical relationship for each image. One layer is assigned to one RPF or one virtual RPF indicating one image in hierarchical relationship.

Master Layer

The master layer is a base layer for image compositing of multiple images, and the size of the master layer equals to that of BRU output. When the image compositing is not applied, there is one layer. This layer is also called as the master layer.

Sub Layer

The Layers except the master layer are called as sub layer.

Alpha

The VSP2 can execute both color data and its transparent data. The alpha is 8-bit data and expresses the transparency of pixel. The value 0xFF represents opaque pixel, and the value 0x00 represents transparent pixel.

Plane

One layer consists of color components such as R/G/B or Y/Cb/Cr, and alpha data representing its transparency of the pixel. The frame which is one of color components and alpha data is called as a plane.

Picture Plane

The picture plane represents color data in one layer.

Alpha Plane

The alpha plane represents transparent data in one layer.

Blending

The image blending means the image composite operation with alpha data. The composite ratio of image blending depends on the relationship of each layer and alpha values.

Raster Operation (ROP)

The bitwise operation such as AND or OR is called as raster operation (ROP).

### (2) Definition of Operators and Functions

The following operators, notations are defined for explaining the functions of VSP2.

$\langle x \rangle$

Discard decimal places of value x

rdown (*x*)

Discard decimal places of value *x* : Ex, rdown (3.81) = 3

clip0 (*x*)

Clip to 0 value if x is less than 0. [ $x = (x < 0)? 0 : x$ ]

clip3 (*min*, *max*, *x*)

The value x shall be clipped so that x shall be $min < x < max$. [ $x = (x < min)? Min : ((x > max)? Max : x)$ ]

### (3) Unit

The unit is defined here for explaining VSP2's functions.

[bpp]: bits per pixel

VSP2 can handle a number of image formats. Each image format has different number of data bits. The unit [bpp] shows a number of data bits for one pixel. For example, RGB 8bpp means that its format is RGB and the pixel consists of 8 data bits.

[bpc] : bits per component

VSP2 can handle a number of image formats and bit-depth. The unit [bpc] shows a number of data bits for one component.

[bypp] : bytes per pixel

The unit [bypp] shows a number of data byte for one pixel.

[bypc] : bytes per component

VSP2 can handle 2 types of data size of YUV's component: 1 byte per component (1bypc) when it's 8-bit, and 2 bytes per component (2bypc) when it's more than 8-bit (10/12-bits).

## 60.5 Safety Mechanism

### 60.5.1 Watch Dog Timer, Frame Counter and SRAM EDC Error Detection of FCPVX.

#### 60.5.1.1 Ovierview

The VSPX and VSPD has watchdog timer to detect hang-up. VSPX has FCPVX SRAM EDC error status and error injection.

#### 60.5.1.2 Hardware Description

**(1) Features**

**(a) Watch Dog Timer of VSPD**

The VSPD has watchdog timer to detect hang-up. Feature is shown below.

- 32 bits watchdog timer (Up to about 8 second)
- Interrupt assertion occurs when timer reaches Time-out cycles set in register
- The interruption is notified to ECM
- Time-out cycle can be set per 2^16 cycles (About 0.12 ms step)
- Min/Max/Step of Time-out cycles is shown below.

  Min : About 0.12 ms (2^16 cycles at 533MHz)

  Max : About 8 sec (2^32 – 1 cycles at 533MHz)

  Step : About 0.12 ms (2^16 cycles at 533MHz)

If the VSP hangs-up due to some H/W fault, initialize the VSP2 by module reset in section .

**(b) Watch Dog Timer of VSPX**

The VSPX has watchdog timer to detect hang-up. Feature is shown below.

- 32 bits watchdog timer (Up to about 8 second)
- Interrupt assertion occurs when timer reaches Time-out cycles set in register
- The interruption is notified to ECM
- Time-out cycle can be set per 2^16 cycles (About 0.12 ms step)
- Min/Max/Step of Time-out cycles is shown below.

  Min : About 0.08 ms (2^16 cycles at 800MHz)

  Max : About 5.3 sec (2^32 – 1 cycles at 800MHz)

  Step : About 0.08 ms (2^16 cycles at 800MHz)

If the VSP hangs-up due to some H/W fault, initialize the VSP2 by module reset in section 60.2.4.10.

**(c) EDC Error Status of FCPV RRSP0BUF SRAM**

VSPX has a SRAM EDC status of FCPV.

**(d) Error Injection of VSPX ECM Notification**

VSPX notifies ECM of WatchdogTimer Error and FCPVX-SRAM-EDC error.VSPX has Error Injection as a function that S/W can set error notification to ECM forcibly.

**(2) Register Configurations of General Control Registers**

**Table 60.44 Register Map**

| Register Type | Register Name | Address Offset |
|---|---|---|
| CTL registers | VI6_WPF0_WDT | H'00C4 |
| | VI6_WPF0_FRCNT | H'00D4 |
| | VI6_ECM0_IRQ_ENB | H'00DC |
| | VI6_ECM0_IRQ_STA | H'00E0 |
| | VI6_CTL_ERRINJ | H'00E8 |

**(3) Register Descriptions of General Control Registers**

**(a) WPF0 Watch Dog Timer Registers (VI6_WPF0_WDT)**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | | – | – | – | – | – | – | – | – | – | – | – | WDTEN |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIMER_CYC [15:0] | | | | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| | | | | |
|---|---|---|---|---|
| 16 | WDTEN | 0 | R/W | Enable/Disable hung-up timer for WPF0<br>When this bit is set to 1 and one-frame processing is not finished within the cycles specified in TIMER_CYC [15:0], VI6_ECM0_IRQ_STA.W0ALM is set to 1.<br>0: Disable timer<br>1: Enable timer |
| 15 to 0 | TIMER_CYC [15:0] | 0 | R/W | Hung-Up Timer setting for WPF0<br>These bits specify hung-up timer setting in cycle unit. The number of cycles of watch dog timer can be specified by (TIMER_CYC + 1) x 2 ^ 16 cycles. Example is below.<br>VSPD (533.33 MHz): If D'49999 is set in TIMER_CYC [15:0], timer setting is 6.144 (sec). This value can be obtained by following expression.<br>((49999+1) x 2^16) / (533.33*1000*1000) = 6.144 |

**(b) WPF0 Frame Count Register (VI6_WPF0_FRCNT)**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | FRCNT [11:0] | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | FRCNT [11:0] | 0 | R/W | WPF0 Frame Counter<br>These bits are used for counting the number of frames executed by WPF0. The count is automatically incremented at the interrupt timing of the frame end (VI6_WPF0_IRQ_STA.FRE). Set desired value in these bits before activating VSP as initial frame count number. |

**(c) ECM0 Interrupt Enable Registers (VI6_ECM0_IRQ_ENB)**

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | W0ALME | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | RRSP0E |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

Each bit controls the interrupt enable of the corresponding interrupt source.

0 : Interrupt Disabled

1 : Interrupt Enabled

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25, 23 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 24 | W0ALME | 0 | R/W | Interrupt Enable for WPF0 Watch dog timer alarm. |
| 0 | RRSP0E | 0 | R/W | Interrupt Enable for EDC error of FCPV RRSP0BUF SRAM |

Each bit in VI6_ECM0_IRQ_STA is set to 1 when the corresponding interrupt source is generated. VI6_ECM0_IRQ_ENB specifies whether to output an interrupt signal for the generated source. When an interrupt is disabled in this register, no interrupt signal is generated even when the corresponding bit in VI6_ECM0_IRQ_STA is set to 1. When an interrupt is enabled in this register, an interrupt signal is output when the corresponding bit in VI6_ECM0_IRQ_STA is set to 1. This interrupt signal is notified to ECM.

### (d) ECM0 Interrupt Status Registers (VI6_ECM0_IRQ_STA)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | W0ALM | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | RRSP0 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

The read value from each bit is the status of the interrupt source, and the write access to each bit controls the interrupt status.

[Read Access] Interrupt Status

0 : No interrupt

1 : Interrupt activated

[Write Access] Interrupt Clear

0 : The interrupt status is cleared to 0

1 : Hold the interrupt status value

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25, 23 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 24 | W0ALM | 0 | R/W | Interrupt Status and Clear for WPF0 Watch dog timer alarm.<br>This interrupt source bit is set to 1 when VI6_WPF0_WDT.WDTEN is 1 and VSP2 does not complete one-frame processing within the period specified in VI6_WPF0_WDT register. Refer to VI6_WPF0_WDT register for more detail. |
| 0 | RRSP0 | 0 | R/W | Interrupt Status and Clear for EDC error of FCPV RRSP0BUF SRAM<br>This interrupt source bit is set to 1 when EDC Error of RRSP0BUF occurs. |

VI6_ECM0_IRQ_STA indicates the state of the interrupt sources generated in the VSP2. Whether to output a VSP2 interrupt when an interrupt source is generated and the corresponding bit is set to 1 is determined by the corresponding bit setting in VI6_ECM0_IRQ_ENB. While an interrupt is disabled in VI6_ECM0_IRQ_ENB, the VSP2 does not output an interrupt signal even when an interrupt source is generated, but the source flag in this register is set to 1.

Note that the interrupt source bits in this register cannot be cleared by write access using a display list.

**(e) ECM0 Error Injection Registers (VI6_CTL_ERRINJ)**

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | ERRINJ0 |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | ERRINJ0 | 0 | R/W | Forced assertion of error notification to ECM<br>0: Disable error injection to error notification to ECM<br>1: Enable error injection to error notification to ECM<br>Error injection can assert an error notification signal to the ECM without activating the status in VI6_ECM0_IRQ_STA. |

### 60.5.2 Display Output Compare Unit (DISCOM)

This module is treated as safety mechanism in case of the products supporting functional safety. For details, refer to the functional safety documents. However, this module can be also treated as non-safety-related function.

The DISCOM calculates the CRC of input image plane and checks whether the CRC is equal to an expectation value or not.

The VSPD, which is one of video signal processing module and specified for image blending of output image, has the DISCOM. Usual use case of the DISCOM is calculation CRC of selected input plane show as following figure.

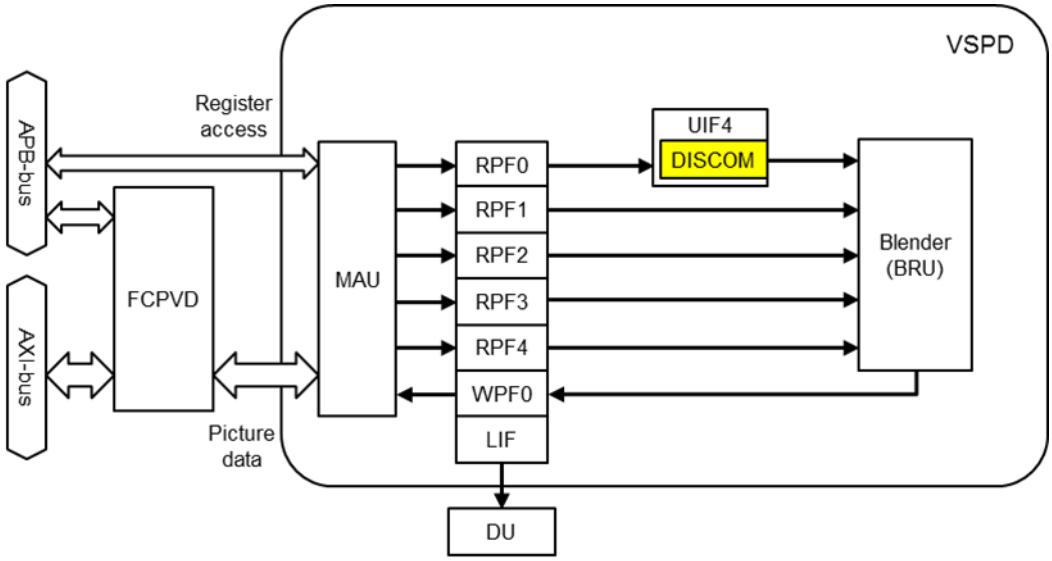



**Figure 60.52 Example use case of the DISCOM**

#### 60.5.2.1 Overview

Display Output Compare Unit (DISCOM) is used to check the input or blending images of VSPD in VSP2. DISCOM can detect the fault in the VSP2 itself and FCP (Figure 60.53 ) by comparing the CRC of arbitrary rectangular area of the image with the expected value. For detail, refer to Display Output Compare Unit (DISCOM) section in VSP2 User's Manual.

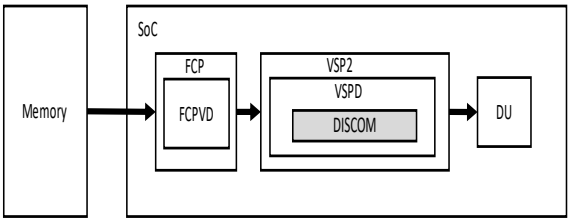



**Figure 60.53 Target of Fault Detection by DISCOM (Output Signal Path)**

DISCOM can also detect the fault in the Camera sensor on input signal path (Figure 60.54 ) by comparing the CRC of arbitrary rectangular area of the image with the previous one. It means DISCOM can detect input image frozen when the CRC value is the same as the previous one.

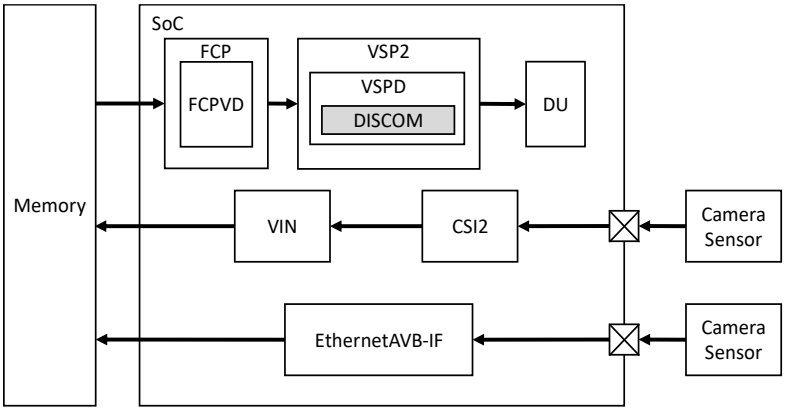



**Figure 60.54 Target of Fault Detection by DISCOM (Input Signal Path) or Frozen Image Detection**

The DISCOM calculates the CRC of input image plane or blending image and checks whether the CRC is equal to an expectation value or not. It can also check the CRC of input image plane is the same value as the CRC of previous picture. These checking are exclusively.
The VSPD, which is one of video signal processing module and specified for image blending of output image, has the DISCOM. Usual use case of the DISCOM is calculation CRC of selected input plane show as following figure. The DISCOM also can calculate the CRC after blending image.
Figure 60.55 shows a block diagram of DISCOM. The DISCOM.CMPI is an interrupt request for detection of unmatching the expectation CRC value. And the DISCOM.CMPI2 is an interrupt request for detection of image frozen.
When detection of mismatching the expectation CRC value is used for blending image, all the following conditions is required as shown in Figure 60.56 .

- Target image of CMPI detection is blended in the foreground by complete opaque
- Target image of CMPI detection is not moving (still image)

And then set size registers and offset registers of target image.

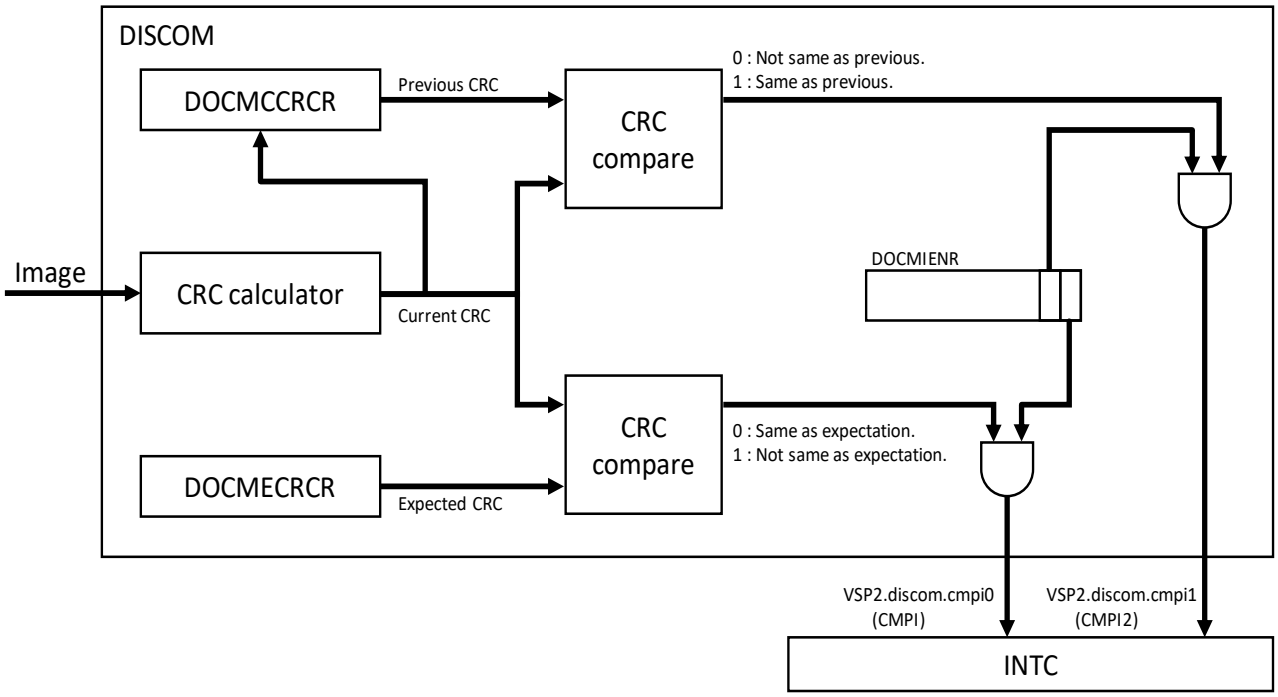



Figure 60.55 Block Diagram of DISCOM

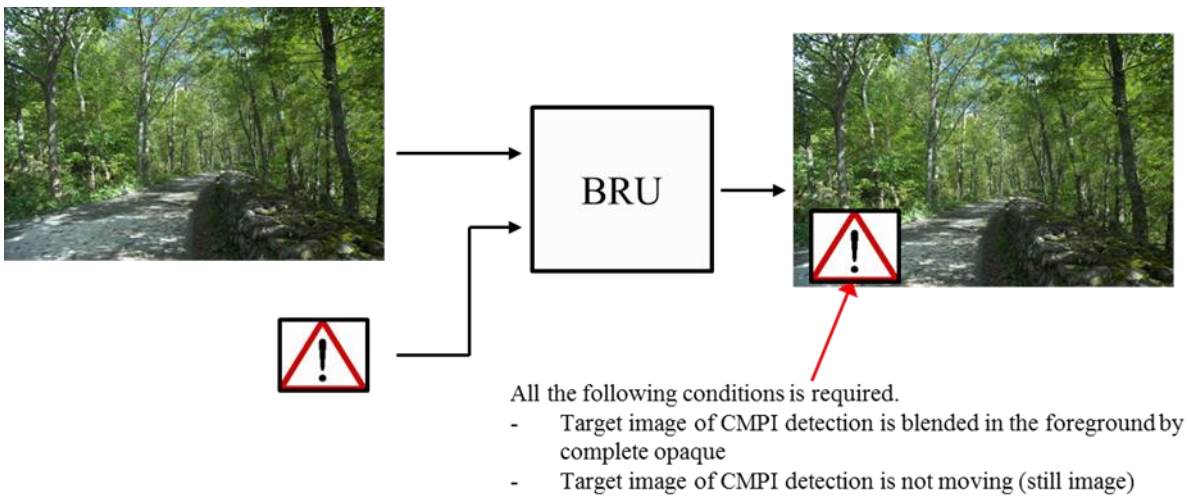



**Figure 60.56 Detecting mismatching with expectation CRC value for blending image by DISCOM.CMPI**

#### 60.5.2.2 Features

The DISCOM has the CRC checking feature shown in following table.

**Table 60.45 Main Function of the DISCOM**

| Item | Description |
|---|---|
| Functions | • Calculation the CRC of input image<br>• Comparing the CRC with an expectation value<br>• Comparing the CRC with previous one<br>• CRC calculation area is configurable. |
| Picture size supported | Minimum:<br>Same as VSPD<br>Maximum:<br>Same as VSPD |
| Color format | Supported color format:<br>• ARGB8888<br>• RGB888<br>• YcbCr444 |

#### 60.5.2.3 Block Diagram

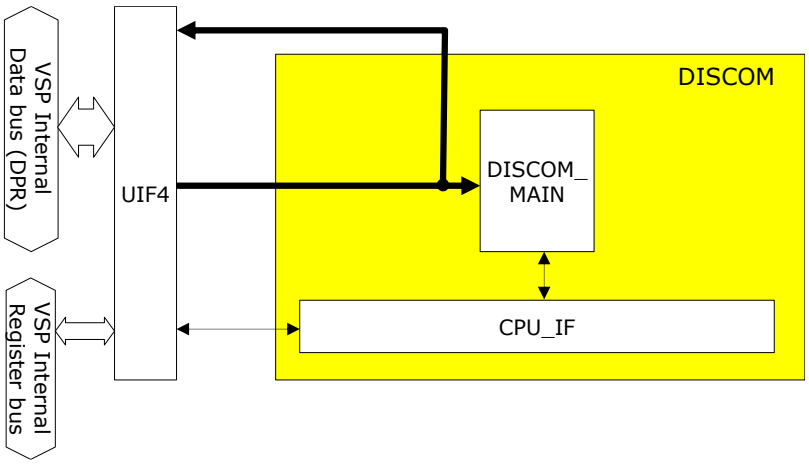



**Figure 60.57 Block Diagram of DISCOM**

#### 60.5.2.4 Register Configuration

The DISCOM's register can be accessed via VSPD's user logic interface (UIF) register address space 4 and space 5 for VSPD. All parameters should be set in the same manner as the other VSP's register, for example, setting by the display list.

**Table 60.46 Register Map**

| Register Type | Register Name | Address Offset |
|---|---|---|
| DISCOM (UIF4) control | VI6_UIF4_DICOM_DOCMCR | 0x1C00 |
| | VI6_UIF4_DICOM_DOCMSTR | 0x1C04 |
| | VI6_UIF4_DICOM_DOCMCLSTR | 0x1C08 |
| | VI6_UIF4_DICOM_DOCMIENR | 0x1C0C |
| | VI6_UIF4_DICOM_DOCMMDR | 0x1C10 |
| | VI6_UIF4_DICOM_DOCMPMR | 0x1C14 |
| | VI6_UIF4_DICOM_DOCMECRCR | 0x1C18 |
| | VI6_UIF4_DICOM_DOCMCCRCR | 0x1C1C |
| | VI6_UIF4_DICOM_DOCMSPXR | 0x1C20 |
| | VI6_UIF4_DICOM_DOCMSPYR | 0x1C24 |
| | VI6_UIF4_DICOM_DOCMSZXR | 0x1C28 |
| | VI6_UIF4_DICOM_DOCMSZYR | 0x1C2C |
| | VI6_UIF4_DICOM_DOCMCRCIR | 0x1C30 |
| | Setting forbidden for other UIF4 register space | 0x1C00 to 0x1CFC except for above offset |
| DISCOM (UIF5) control | VI6_UIF5_DICOM_DOCMCR | 0x1D00 |
| | VI6_UIF5_DICOM_DOCMSTR | 0x1D04 |
| | VI6_UIF5_DICOM_DOCMCLSTR | 0x1D08 |
| | VI6_UIF5_DICOM_DOCMIENR | 0x1D0C |
| | VI6_UIF5_DICOM_DOCMMDR | 0x1D10 |
| | VI6_UIF5_DICOM_DOCMPMR | 0x1D14 |
| | VI6_UIF5_DICOM_DOCMECRCR | 0x1D18 |
| | VI6_UIF5_DICOM_DOCMCCRCR | 0x1D1C |
| | VI6_UIF5_DICOM_DOCMSPXR | 0x1D20 |
| | VI6_UIF5_DICOM_DOCMSPYR | 0x1D24 |
| | VI6_UIF5_DICOM_DOCMSZXR | 0x1D28 |
| | VI6_UIF5_DICOM_DOCMSZYR | 0x1D2C |
| | VI6_UIF5_DICOM_DOCMCRCIR | 0x1D30 |
| | Setting forbidden for other UIF5 register space | 0x1D00 to 0x1DFC except for above offset |

#### 60.5.2.5 Connected Module

Table 60.47 shows modules connected to the DISCOM.

**Table 60.47 Connected Module**

| Module | Connected Module | Description |
|---|---|---|
| DISCOM (UIF4) | VSPD | VSP for image blending and output |
| DISCOM (UIF5) | VSPD | VSP for image blending and output |

### 60.5.2.6 Register Description

#### (1) VI6_UIFn_DICOM_DOCMCR (n=4,5)

DOCMCR controls the DISCOM operation.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | CMPRU |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | CMPR |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | CMPRU | 0 | R | This bit indicates an internal reflected bit of CMPR.<br>The register except for DOCMCR.CMPR, DOCMCLSTR, and DOCMIENR should be updated only when CMPRU is 0. For detail, refer section 60.5.2.7(6) Register Update Timing. |
| 0 | CMPR | 0 | R/W | DISCOM execution enable.<br>This bit is reflected to DISCOM behavior at the timing of frame staring. For detail, refer section (6) Register Update Timing.<br>0 : Stop DISCOM<br>1 : Execute DISCOM |
| 31 to 17, 15 to1 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

#### (2) VI6_UIFn_DICOM_DOCMSTR (n=4,5)

DOCMSTR indicates the compare result of the CRC code. It is reflected at the end of frame.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | CMPPRE | CMPST |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | CMPPRE | 0 | R | Compare result between the CRC and the previous one.<br>0: Not same as previous CRC code<br>1: The CRC has the same code of (VI6_UIFn_DICOM_ DOCMMDR.INTHRH+1) consecutive times. |
| 0 | CMPST | 0 | R | Compare result between the CRC and its expectation<br>0 : Same as expectation<br>1 : Not same as expectation |
| 31 to 2 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (3) VI6_UIFn_DICOM_DOCMCLSTR (n=4,5)

DOCMCLSTR clear the DOCMSTR. When each bit of DOCMCLSTR is set to 1, the corresponding bit of DOCMSTR is cleared. Note that the clear bit takes the fixed time. Therefore after setting DOCMCLSTR, it is necessary to confirm that DOCMSTR has been cleared.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | CMPPRECLCNT | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | CMPCLPRE | CMPCLST |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | CMPPRECLCNT | 0 | R/W | Clear for frozen counter bit<br>When this bit is set to 1, internal frozen counter is cleared.<br>This bit is always read as 0. |
| 1 | CMPCLPRE | 0 | R/W | Clear for CMPPRE status bit.<br>When this bit is set to 1, DOCMSTR.CMPPRE is cleared.<br>This bit is always read as 0. |
| 0 | CMPCLST | 0 | R/W | Clear for CMPST status bit<br>When this bit is set to 1, DOCMSTR.CMPST is cleared.<br>This bit is always read as 0. |

| | | | | |
|---|---|---|---|---|
| 31 to 18, 16 to 2 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(4) VI6_UIFn_DICOM_DOCMIENR(n=4,5)**

DOCMIENR specifies the enable of the interruption.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | EEP_ERR_FWD |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | CMPPREIEN | CMPIEN |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | EEP_ERR_FWD | 0 | R/W | Error forwarding to ECM<br>This bit determines whether DISCOM forwards interrupts to ECM as error signals or to INTC as interrupt signals.<br>0: Not forward to ECM (Forward interrupts to INTC)<br>1: Forward to ECM. |
| 1 | CMPPREIEN | 0 | R/W | Interruption enable for CMPI2.<br>This bit specifies the interruption enable if DOCMSTR. CMPPRE becomes 1.<br>0: Disable CMPI2 interruption.<br>1: Enable CMPI2 interruption. |
| 0 | CMPIEN | 0 | R/W | Interruption enable for CMPI<br>This bit specifies the interruption enable if DOCMSTR. CMPST becomes 1.<br>0 : Disable CMPI interruption<br>1 : Enable CMPI interruption |
| 31 to 17, 15 to 2 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(5) VI6_UIFn_DICOM_DOCMMDR (n=4,5)**

DOCMMDR specifies the compare mode of the DISCOM.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | INTHRH[7:0] | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R/W : | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 23 to 16 | INTHRH[7:0] | 0 | R/W | This parameter specified a threshold value for an interruption. If the cause of interruption is continuously occurred more than (INTHRH) frames, the interruption request is generated.<br>Note that the value H'FF is setting prohibited.<br>When the frame rate for output is greater than that for input, the same input frame may be compared. Accordingly, these bits are used to set a limit on the number of matches. |
| 31 to 24, 15 to 0 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (6) VI6_UIFn_DICOM_DOCMPMR (n=4,5)

DOCMPMR specifies the pixel data format.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit : | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPDFF[1:0] | | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit : | CMPDFA[7:0] | | | | | | | | CMPDAUF | — | — | — | SEL[3:0] | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 to 17 | CMPDFF[1:0] | 0 | R/W | This parameter should be set to 0.<br>00 : ARGB8888, RGB888, or YcbCr444<br>Others : Setting prohibited |
| 15 to 8 | CMPDFA[7:0] | 0 | R/W | This parameter specifies the default alpha value.<br>It is used when CMPDAUF is 1. |
| 7 | CMPDAUF | 0 | R/W | This flag specifies the alpha value which is used for CRC calculation.<br>0 : Use alpha value with pixel value<br>1 : Use default alpha value (CMPDFA) |

| | | | | |
|---|---|---|---|---|
| 3 to 0 | SEL[3:0] | 0 | R/W | This parameter should be set to 9. |
| 31 to 19, 16, 6 to 4 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (7) VI6_UIFn_DICOM_DOCMECRCR (n=4,5)

DOCMECRCR specifies the expectation value of the CRC code.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPECRC[31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPECRC[15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMPECRC[31:0] | 0 | R/W | This parameter specifies the expectation of the CRC. |

### (8) VI6_UIFn_DICOM_DOCMCCRCR (n=4,5)

DOCMCCRCR indicates the CRC value of current image. It is reflected at the end of frame.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPCCRC[31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPCCRC[15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMPCCRC[31:0] | 0 | R | This parameter indicates the CRC value of current image. |

### (9) VI6_UIFn_DICOM_DOCMSPXR (n=4,5)

DOCMSPXR specifies horizontal offset the CRC calculation area in the image.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | CMPSPX[12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 0 | CMPSPX [12:0] | 0 | R/W | This parameter specifies horizontal offset of the CRC calculation area. It should be less or equal to horizontal size of input image.<br>Horizontal offset is value of CMPSPX for other products. |
| 31 to 13 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

Valid bits for each product is shown below.

### (10) VI6_UIFn_DICOM_DOCMSPYR (n=4,5)

DOCMSPYR specifies vertical offset the CRC calculation area in the image.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | CMPSPY[12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 0 | CMPSPY[12:0 ] | 0 | R/W | This parameter specifies vertical offset of the CRC calculation area. It should be less or equal to vertical size of input image. |
| 31 to 13 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

Valid bits for each product is shown below.

### (11) VI6_UIFn_DICOM_DOCMSZXR (n=4,5)

DOCMSZXR specifies horizontal size of the CRC calculation area.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | CMPSZX[12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 0 | CMPSZX[12:0] | 0 | R/W | This parameter specifies horizontal size of the CRC calculation area. It should be less or equal to (horizontal size of input image – CMPSPX).<br>Horizontal size is value of CMPSZX for other products. |
| 31 to 13 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

Valid bits for each product is shown below.

### (12) VI6_UIFn_DICOM_DOCMSZYR (n=4,5)

DOCMSZYR specifies vertical size of the CRC calculation area.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | CMPSZY[12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 12 to 0 | CMPSZY [12:0] | 0 | R/W | This parameter specifies vertical size of the CRC calculation area. It should be less or equal to (vertical size of input image – CMPSPY). |
| 31 to 13 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

Valid bits for each product is shown below.

**(13) VI6_UIF4_DICOM_DOCMCRCIR (n=4,5)**

DOCMCRCIR specifies the initial value of the CRC.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CRCINI[31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CRCINI[15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CRCINI[31:0] | H'FFFF_FFFF | R/W | This parameter specifies the initial value of the CRC. |

### 60.5.2.7 Operation

**(1) Overview of Process**

The DISCOM calculates the CRC of arbitrary rectangular area of the input image, and compares with the expectation value.

1. If the CRC code is same as its expectation value, the display output image is as expected. This case the input image is such as graphics data, and the DISCOM is used for checking it is displayed correctly or not.

The DISCOM has the following features.

- The input image of the DISCOM can be selected from five input images of VSPD by using data path routing function of the VSPD.
- Any CRC computation rectangular area can be specified.
- Supported color formats are ARGB8888, RGB888, and YcbCr.
- The interruption (CMPI) is occurred if the CRC value doesn't match with the expectation value.

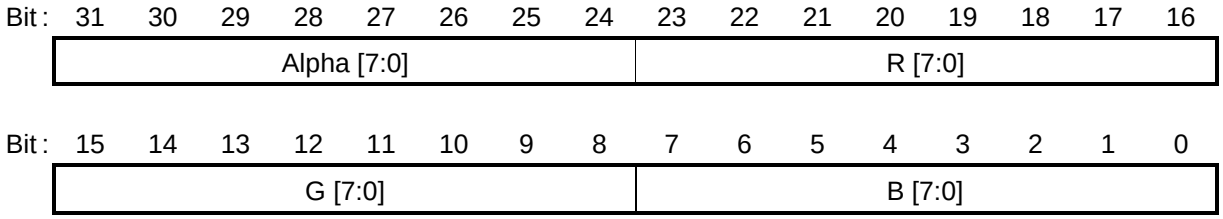
### (2) CRC Calculation

The DISCOM uses following equation to calculate the 32-bit CRC value. Each bit of the final CRC result is logically inverted.

$X^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$

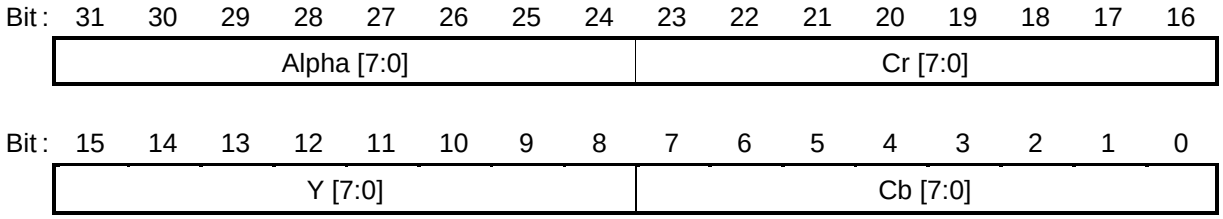
The CRC calculation is operated in pixels and the pixel order of input image is a raster scan order. The pixel data is 32-bit data and processed from LSB to MSB. The format of pixel data is shown as follows:

#### (a) ARGB8888/RGB888




If DOCMPMR.CMPDAUF is set to 0, the alpha value is from the data path router of the VSPD. Otherwise the alpha value is set as DOCMPMR.CMPDFA.

#### (b) YcbCr444




If DOCMPMR.CMPDAUF is set to 0, the alpha value is from the data path router of the VSPD. Otherwise the alpha value is set as DOCMPMR.CMPDFA.

### (3) Image Format on the External Memory

The read pixel formatter of the VSPD converts the various pixel format shown in Table 60.10 and Table 60.12 to the internal pixel format of the DISCOM (shown in previous section). For the detail, refer section 60.2.6.

### (4) Setting Rectangle Region

This section describes how to set the rectangular area to the CRC calculation.

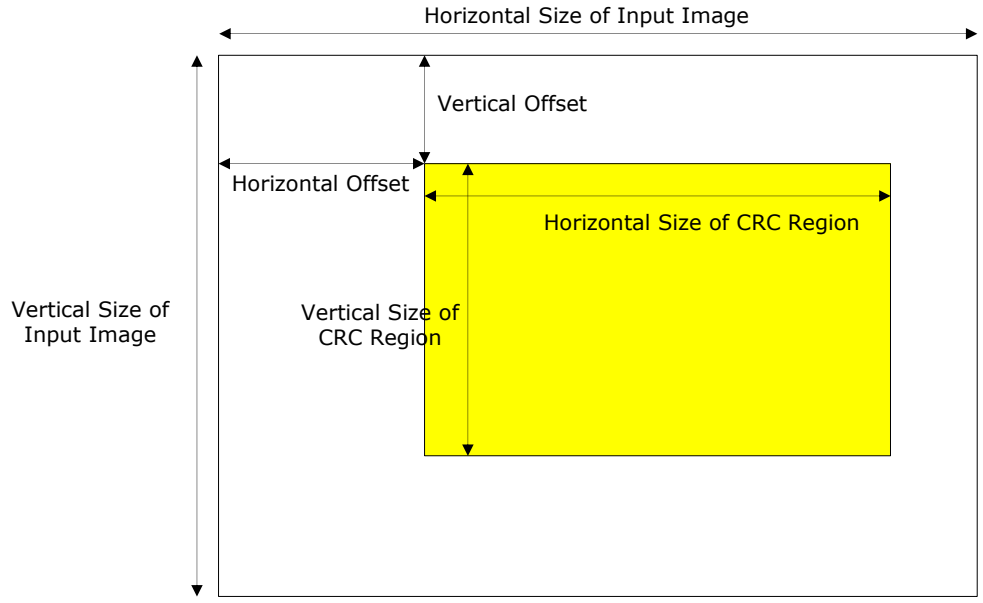



**Figure 60.58 Example Case of the CRC Computation Area**

**Table 60.48 Register Setting for the CRC Computation Area**

| Word in Figure | Register of DISCOM | Description |
|---|---|---|
| Horizontal Size of Input Image | | This parameter is horizontal size of input image for the DISCOM. It is set by VSPD's register. |
| Vertical size of Input Image | | This parameter is vertical size of input image for the DISCOM. It is set by VSPD's register. |
| Horizontal Offset | DOCMSPXR | This parameter specifies horizontal offset position of the CRC calculation area from left border of the input image. The offset value is the pixel unit. |
| Vertical Offset | DOCMSPYR | This parameter specifies vertical offset position of the CRC calculation area from top border of the input image. The offset value is the line unit. |
| Horizontal Size of CRC Region | DOCMSZXR | This parameter specifies horizontal size of the CRC calculation area. It should be less or equal to (Horizontal size of input image – Horizontal offset). The size is the pixel unit. |
| Vertical Size of CRC Region | DOCMSZYR | This parameter specifies vertical size of the CRC calculation area. It should be less or equal to (Vertical size of input image – Vertical offset). The size is the line unit. |

### (5) Timing Chart of CRC Computation and Comparing

Following figure shows the timing chart example of CRC computation and comparing.

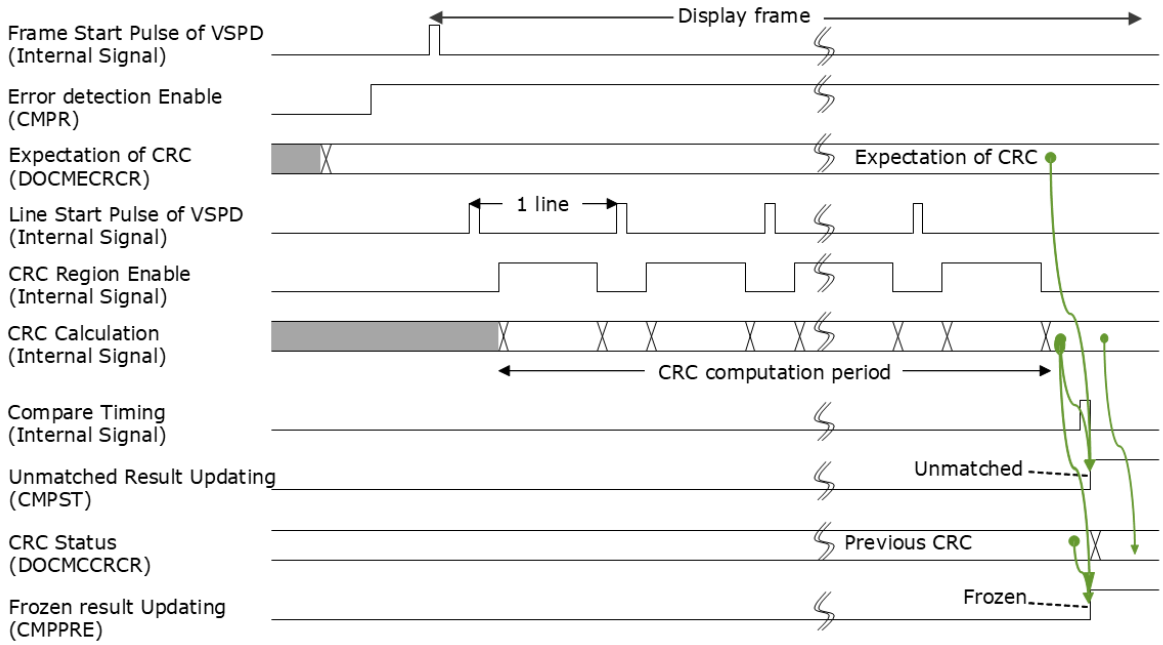



**Figure 60.59 Example Timing of CRC Computation and Comparing**

1. The DISCOM starts operation from the following frame after setting DOCMCR.CMPR to 1. For register update timing, refer section (6).
2. The CRC code is calculated in the specified rectangular area.
3. The result of the CRC code is compared with the expectation (DOCMECRCR) and the previous value (CMPCCRC).
4. If the CRC code doesn't match with the expectation value, DOCMSTR.CMPST bit is set to 1.
5. If the CRC code matches with the previous value, DOCMSTR.CMPPRE bit is set to 1.

   Note: When VSPD software reset is issued as shown in Figure 60.36, CRC calculation may be halfway finished. CRC calculation is not same with the expectation, and false error occurs as CMPST interrupt in the case. Therefore, disable VI6_UIFn_DICOM_DOCMIENR.CMPIEN before VSPD software reset is issued to restrain false error.

### (6) Register Update Timing

#### (a) Update Timing of Register Setting

All registers except for DOCMCR.CMPR are reflected in the internal register immediately. Therefore, they should be set during DOCMCR.CMPRU is 0.

The DOCMCR.CMPR is reflected in the internal register at the beginning of the frame processing. It can be updated at any time.

#### (b) Update Timing of Status Register

The DOCMSTR and DOCMCCRCR are updated at the end of the frame processing. The DOCMCR.CMPRU is updated at the beginning of the frame processing.

The following figure shows an example timing of register updating.

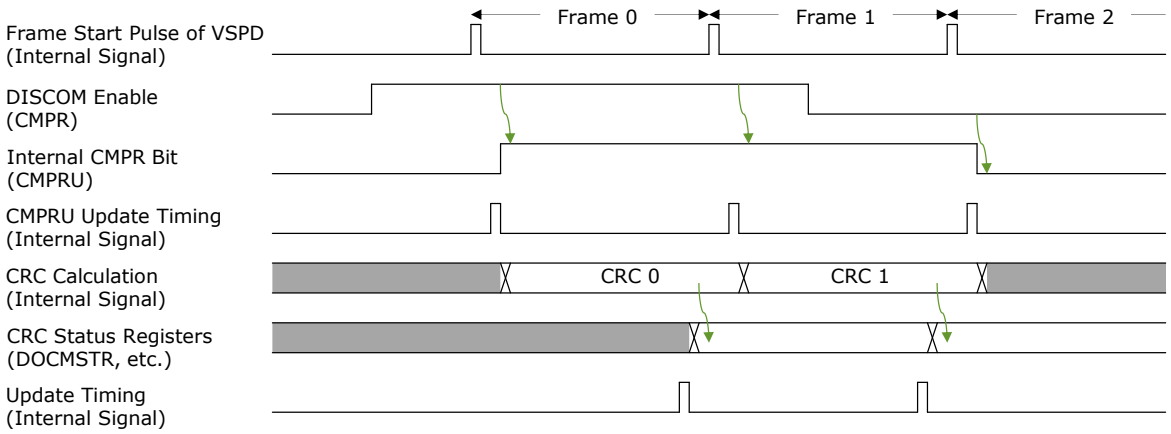



**Figure 60.60 Example Timing of Register Updating**

**(7) Interruption**

The DISCOM has two interrupt request, CMPI and CMPI2. The CMPI request is occurred when both DOCMIENR.CMPIEN and DOCMSTR.CMPST are set to 1. CMPI2 is functional safety use only. Refer to functional safety documents for the detail.

#### 60.5.2.8 Usage Notes

Set 9 to VI6_UIFn_DICOM_DOCMPMR.SEL[3:0] (Section 60.5.2.6(6)) before starting VSP2 in case of using UIFn(DISCOM). Otherwise, VSP may not work correctly.

### 60.5.3 Safety Mechanism Generator (SMG)

#### 60.5.3.1 Ovierview

SMG (Safety Mechanism Generator) is implemented in VSPD to support below safety mechanism of FUSA:

- (1) To send EDC code for each pixel from LIF0 to Display (Pixel protection)
- (2) To send CRC code for each picture from LSI to LSI transmission (End-to-End protection)
- (3) To detect unmatched picture data from External Memory to WPF (Unmatched error notification)
- (4) To detect frozen picture data from External Memory to WPF (Frozen error notification)

To realize safety mechanism (1) SMG calculates EDC code for each pixel and sends EDC code with corresponding pixel data to Display for evaluating (Pixel protection).

To realize safety mechanism (2) SMG calculates CRC code for each frame and sends it with corresponding frame to the other end terminal for evaluating (End-to-End protection).

To realize safety mechanism (3), (4), SMG compare current frame CRC with an expectation or with previous frame CRC result respectively. Result of detection is reflected in registers and notify to ECM for error handling of FUSA

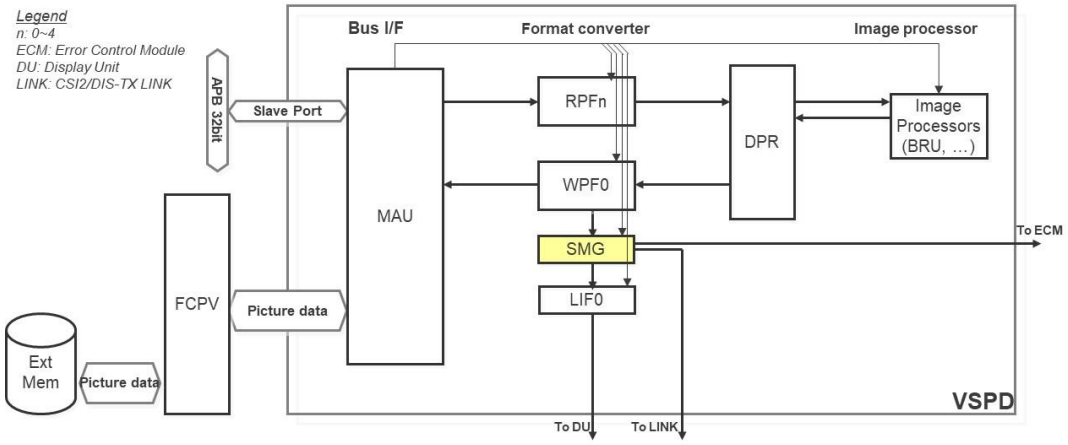



**Figure 60.61** Sample use case of SMG

### 60.5.3.2 Hardware Description

**(1) Features**

Table 60.49 Main functions of SMG

| Item | Description |
|---|---|
| Functions | • Calculating the EDC code for each output pixels to DU<br>• Calculating the CRC for each output frame to TX-LINK<br>• Comparing the CRC with an expectation value<br>• Comparing the CRC with previous one<br>• CRC calculation area is configurable. |
| Image size | Minimum:<br>Same as VSPD<br>Maximum:<br>Same as VSPD |
| Color format | Supported color format for EDC calculation:<br>• RGB888<br>• YcbCr422<br>Supported color format for CRC calculation:<br>• RGB888 4bytes (MSB byte is set by register)<br>• RGB888<br>• YcbCr422 |

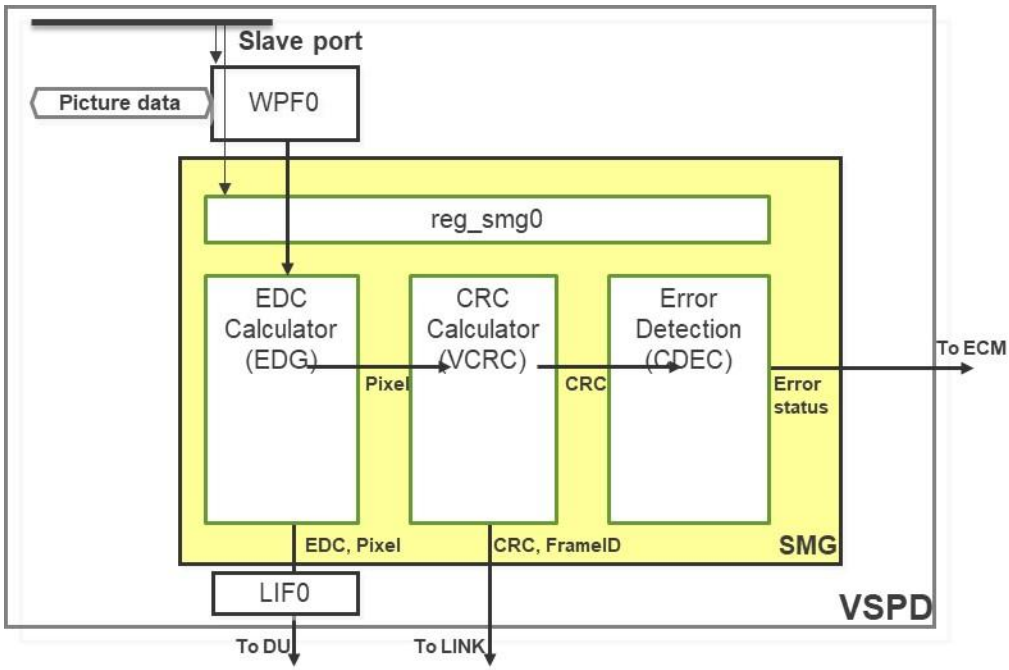



Figure 60.62 Block Diagram of SMG

SMG contains three main blocks:

✓ EDC calculator supports calculating EDC code for each output pixels to DU.

- ✓ CRC calculator supports calculating CRC code for both E2E protection and Error detection.
- ✓ Error detection supports either unmatched error or Frozen error notification.

### (2) Register Configuration

SMG registers can be accessed via either CPU interface or Display List interface of VSP. SMG registers are allocated as offset in below table.

Table 60.50 Register Configuration

| Register Type | Register Name | Address Offset |
|---|---|---|
| SMG control | VI6_SMG_MCR | 0x1400 |
| | VI6_SMG_MSTR | 0x1404 |
| | VI6_SMG_MCLSTR | 0x1408 |
| | VI6_SMG_MIENR | 0x140C |
| | VI6_SMG_MMDR | 0x1410 |
| | VI6_SMG_MPMR | 0x1414 |
| | VI6_SMG_MECRCR | 0x1418 |
| | VI6_SMG_MCCRCR | 0x141C |
| | VI6_SMG_MSPXR | 0x1420 |
| | VI6_SMG_MSPYR | 0x1424 |
| | VI6_SMG_MSZXR | 0x1428 |
| | VI6_SMG_MSZYR | 0x142C |
| | VI6_SMG_MCRCIR | 0x1430 |
| | VI6_SMG_IODCFG | 0x1434 |
| | VI6_SMG_PEQU | 0x1438 |
| | VI6_SMG_PSZM1 | 0x143C |
| | VI6_SMG_FXOR | 0x1440 |
| | VI6_SMG_FRID | 0x1444 |
| | VI6_SMG_ERRINJ | 0x1448 |
| | VI6_SMG_ECRC | 0x1450 |

### (3) Register Description

#### (a) VI6_SMG_MCR

MCR controls Enable/Stop CRC calculation for error detection feature of SMG

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPRU |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPR |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | CMPRU | 0 | R | CRC calculation status flag (for error detection)<br>This bit indicates an internal reflected bit of CMPR.<br>The register except for MCR.CMPR, MCLSTR, and MIENR should be updated only when CMPRU is 0.0: There is no CRC calculation (for error detection)<br>1: CRC calculation for error detection is on-going |
| 0 | CMPR | 0 | R/W | CRC execution enable for error detection.<br>This bit is reflected to SMG behavior at the timing of frame staring.<br>0: Stop CRC calculation for error detection<br>1: Start CRC calculation for error detection |
| 31 to 17,<br>15 to1 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(b) VI6_SMG_MSTR**

MSTR indicates the compare result of the CRC code. It is reflected at the end of display frame.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPPRE | CMPST |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | CMPPRE | 0 | R | Compare result between the CRC and the previous one.<br>0: Not same as previous CRC code<br>1: The CRC has the same code of (VI6_SMG_MMDR.INTHRH1+1) consecutive times. |
| 0 | CMPST | 0 | R | Compare result between the CRC and its expectation<br>0: Same as expectation<br>1: The CRC is not same as the expectation (VI6_SMG_MMDR.INTHRH+1) consecutive times. |
| 31 to 2 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(c) VI6_SMG_MCLSTR**

MCLSTR clear the MSTR. When each bit of MCLSTR is set to 1, the corresponding bit of MSTR is cleared. Note that the clear bit takes the fixed time. Therefore, after setting MCLSTR, it is necessary to confirm that MSTR has been cleared.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPPRECLCNT | — |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPCLPRE | CMPCLST |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | CMPCLPRECLCNT | 0 | R/W | Clear for frozen counter bit<br>When this bit is set to 1, internal frozen counter is cleared.<br>This bit is always read as 0. |
| 1 | CMPCLPRE | 0 | R/W | Clear for CMPPRE status bit<br>When this bit is set to 1, MSTR.CMPPRE is cleared.<br>This bit is always read as 0. |
| 0 | CMPCLST | 0 | R/W | Clear for CMPST status bit<br>When this bit is set to 1, MSTR.CMPST is cleared.<br>This bit is always read as 0. |
| 31 to 18,<br>16 to 2 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(d) VI6_SMG_MIENR**

MIENR specifies the enable of the error notification to ECM.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | UIF5_CMPPREIEN | UIF5_CMPIEN | — | — | UIF4_CMPPREIEN | UIF4_CMPIEN | — | — | CMPPREIEN | CMPIEN |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R/W : R R R R R R R/W R/W R R R/W R/W R R R/W R/W

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | UIF5_CMPPREIEN | 0 | R/W | Enable frozen error notification for DISCOM (UIF5) to ECM<br>This bit specifies the error notification enable if VI6_UIF5_DICOM_DOCMSTR.CMPPRE becomes 1 and VI6_UIF5_DICOM_DOCMIENR.CMPPREIEN = 1<br>0 : Disable frozen error notification<br>1 : Enable frozen error notification |
| 8 | UIF5_CMPIEN | 0 | R/W | Enable unmatched error notification for DISCOM (UIF5) to ECM<br>This bit specifies the error notification enable if VI6_UIF5_DICOM_DOCMSTR.CMPST becomes 1 and VI6_UIF5_DICOM_DOCMIENR.CMPIEN = 1<br>0 : Disable unmatched error notification<br>1 : Enable unmatched error notification |
| 5 | UIF4_CMPPREIEN | 0 | R/W | Enable frozen error notification for DISCOM (UIF4) to ECM<br>This bit specifies the error notification enable if VI6_UIF4_DICOM_DOCMSTR.CMPPRE becomes 1 and VI6_UIF4_DICOM_DOCMIENR.CMPPREIEN = 1<br>0 : Disable frozen error notification<br>1 : Enable frozen error notification |
| 4 | UIF4_CMPIEN | 0 | R/W | Enable unmatched error notification for DISCOM (UIF4) to ECM<br>This bit specifies the error notification enable if VI6_UIF4_DICOM_DOCMSTR.CMPST becomes 1 and VI6_UIF4_DICOM_DOCMIENR.CMPIEN = 1<br>0 : Disable unmatched error notification<br>1 : Enable unmatched error notification |
| 1 | CMPPREIEN | 0 | R/W | Enable frozen error notification to ECM<br>This bit specifies the error notification enable if MSTR. CMPPRE becomes 1.<br>0 : Disable frozen error notification<br>1 : Enable frozen error notification |
| 0 | CMPIEN | 0 | R/W | Enable unmatched error notification to ECM<br>This bit specifies the error notification enable if MSTR. CMPST becomes 1.<br>0 : Disable unmatched error notification<br>1 : Enable unmatched error notification |
| 31 to 10,<br>7 to 6,<br>3 to 2 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(e) VI6_SMG_MMDR**

MMDR specifies the threshold of frozen or unmatched error notification. SMG uses two different error thresholds: INTHRH1 for frozen error notification and INTHRH for unmatched error notification.

Bit : 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

| | INTHRH1[7:0] | | | | | | | | INTHRH [7:0] | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | INTHRH1[7:0] | 0 | R/W | Threshold for frozen error notification<br>This parameter specifies a threshold value for an error notification. If the cause of error notification is continuously occurred more than (INTHRH1) frames, the error request is generated.<br>Note that the value H'FF is setting prohibited. |
| 23 to 16 | INTHRH [7:0] | 0 | R/W | Threshold for unmatched error notification<br>This parameter specified a threshold value for an error notification. If the cause of error is continuously occurred more than (INTHRH) frames, the error request is generated.<br>Note that the value H'FF is setting prohibited. |
| 15 to 0 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(f) VI6_SMG_MPMR**

MPMR specifies the pixel data format

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | - | — | — | — | — | — | — | | CMPBT |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPDFA[7:0] | | | | | | | | CMPDAUF | — | — | — | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 16 | CMPBT | 0 | R/W | Invalid bits truncation<br>0: No truncation (For ARGB8888, RGB888, or YcbCr444, YcbCr422)<br>1: Apply truncation (For RGB565) |
| 15 to 8 | CMPDFA[7:0] | 0 | R/W | Default alpha value in case of RGB888 (4 bytes format)<br>This parameter specifies the default alpha value.<br>It is used when CMPDAUF is 1. |
| 7 | CMPDAUF | 0 | R/W | This flag specifies the alpha value, which is used for CRC calculation in case of RGB888 (4 bytes format).<br>0: Use alpha value with pixel value<br>1: Use default alpha value (CMPDFA) |
| 31 to 17,<br>6 to 0 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(g) VI6_SMG_MECRCR**

MECRCR specifies the expectation value of the CRC code for unmatched error detection.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPECRC [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPECRC [15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMPECRC [31:0] | 0 | R/W | Expected CRC for unmatched error detection<br>This parameter specifies the expectation of the CRC. |

**(h) VI6_SMG_MCCRCR**

MCCRCR indicates the CRC value of current display image. It is reflected at the end of whole frame (V-valid)

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPCCRC [31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPCCRC [15:0] | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMPCCRC [31:0] | 0 | R | CRC of current display frame.<br>This parameter indicates the CRC value of current image. These bits are valid when SMG enable error detection features |

### (i) VI6_SMG_MSPXR

MSPXR specifies horizontal offset the CRC calculation area in the image.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | CMPSPX [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 0 | CMPSPX [12:0] | 0 | R/W | Horizontal offset of the CRC calculation area.<br>This parameter specifies horizontal offset of the CRC calculation area. It should be less or equal to horizontal size of input image (up to 8190) and below condition:<br>VI6_SMG_MSPXR.CMPSPX[12:0] + VI6_SMG_MSZXR.CMPSZX[12:0] <= “WPF output horizontal size to SMG” |
| 31 to 13 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (j) VI6_SMG_MSPYR

MSPYR specifies vertical offset the CRC calculation area in the image.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | CMPSPY [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 0 | CMPSPY [12:0] | 0 | R/W | Vertical offset of the CRC calculation area<br>This parameter specifies vertical offset of the CRC calculation area. It should be less or equal to vertical size of input image (up to 8190) and below condition:<br>VI6_SMG_MSPYR.CMPSPY[12:0] + VI6_SMG_MSZYR.CMPSZY[12:0] <= “WPF output vertical size to SMG” |
| 31 to 13 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (k) VI6_SMG_MSZXR

MSZXR specifies horizontal size of the CRC calculation area.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | CMPSZX [12:0] | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 0 | CMPSZX [12:0] | 0 | R/W | Horizontal size of the CRC calculation area.<br>This parameter specifies horizontal size of the CRC calculation area. It should be less or equal to (horizontal size of input image – CMPSPX) (up to 8190) and below condition:<br>VI6_SMG_MSPXR.CMPSPX[12:0] + VI6_SMG_MSZXR.CMPSZX[12:0] <= “WPF output horizontal size to SMG” |
| 31 to 13 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (l) VI6_SMG_MSZYR

MSZYR specifies vertical size of the CRC calculation area.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | CMPSZY[12:0] | | | | | | | | | | | | |
| Initial Value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 0 | CMPSZY [12:0] | 0 | R/W | Vertical size of the CRC calculation area.<br>This parameter specifies vertical size of the CRC calculation area. It should be less or equal to (vertical size of input image – CMPSPY) (up to 8190) ) and below condition: VI6_SMG_MSPYR.CMPSPY[12:0] + VI6_SMG_MSZYR.CMPSZY[12:0] <= “WPF output vertical size to SMG” |
| 31 to 12 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (m) VI6_SMG_MCRCIR

MCRCIR specifies the initial value of the CRC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CRCINI[31:16] | | | | | | | | | | | | | | | |
| Initial Value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CRCINI[15:0] | | | | | | | | | | | | | | | |
| Initial Value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 31 to 0 | CRCINI[31:0] | H'FFFF_FFFF | R/W | CRC Initial Remainder<br>These bits specify the initial value of reminder. If size of polynomial is smaller than 32-bit, set to LSB side. |

**(n) VI6_SMG_IODCFG**

IODCFG specifies input/output data configuration for CRC calculation. The CRC calculation result is used for both E2E protection feature and Error detection feature.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | REFLECT_REMAINDER | — | — | — | REFLECT_INPUT | — | BYTEORDER_INPUT [2:0] | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R/W | R | R | R | R/W | R | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | | | | | | | — | — | — | — | — | — | DATA_SIZE_M1 [1:0] | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 24 | REFLECT_REMAINDER | 1 | R/W | Remainder Output Reflection<br>This bit specifies the reflection flag of remainder output. When this bit is set to 1, the bit order of remainder output data is reversed.<br>0: No reflection<br>1: Apply reflection |
| 20 | REFLECT_INPUT | 1 | R/W | Input Data Reflection<br>This bit specifies the reflection flag of input byte data. After applying byte order modification by BYTEORDER_INPUT [2:0] bits, the bit order of each byte of input data is reversed before CRC operation when this bit is set to 1.<br>0: No reflection<br>1: Apply reflection |
| 18 to 16 | BYTEORDER_INPUT [2:0] | 0 | R/W | Input Data Byte Order<br>This bit specifies the byte swap setting for input data.<br>H'0: The bit order from MSB is bit 31 to 0<br>H'1: The bit order from MSB is bit 23 to 16, 31 to 24, 7 to 0, 15 to 8<br>H'2: The bit order from MSB is bit 15 to 0, 31 to 16<br>H'3: The bit order from MSB is bit 7to 0, 15 to 8, 23 to 16, 31 to 24<br>H'4: The bit order from MSB is bit 23 to 0, 31 to 24<br>H'5: The bit order from MSB is bit 15 to 8, 23 to16, 31 to 24, 7 to 0<br>H'6: Setting prohibited |

| | | | | |
|---|---|---|---|---|
| | | | | H'7: Setting prohibited |
| 1 to 0 | DATA_SIZE_M1 [1:0] | 0 | R/W | Input Data Byte length<br>These bits specify the value of byte length of input data.<br>00: 1-byte data (bit 31 to 24) are valid<br>01: 2-byte data (bit 31 to 16) are valid<br>10: 3-byte data (bit 31 to 8) are valid<br>11: 4-byte data (bit 31 to 0) are valid<br>Note: 1. This bit specifies the byte length of modified data by BYTEORDER_INPUT [2:0] and REFLECT_INPUT bits. |
| 31 to 25,<br>23 to 21,<br>19,<br>15 to 2 | - | All 0 | R/W | Reserved<br>These bits are always read as 0. The write value should always be 0. |

**(o) VI6_SMG_PEQU**

PEQU indicates coefficient of polynomial equation. If polynomial is smaller than 32-bit, set it to LSB side.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VCRCPOLYEQU[31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VCRCPOLYEQU[15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | VCRCPOLYEQU[31:0] | H'04C11DB7 | R/W | CRC Polynomial Equation Coefficient<br>**These bits specify the** coefficient of CRC polynomial equation. If polynomial is smaller than 32-bit, set to LSB side. |

**(p) VI6_SMG_PSZM1**

PSZM1 indicates bit length of polynomial – 1.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | — | — | — | — | — | — | — | — | — | — | — | VCRCPOLY_SIZE_M1[4:0] | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 to 0 | VCRCPOLY_ SIZE_M1[4:0] | H'1F | R/W | Polynomial Equation Coefficient Data Bit Length<br>These bits specify the valid bit of coefficient which is specified by the VnCRPEC, VnCRIR, VnCRXRC.<br>H'00: 1-bit data (bit 0) is valid<br>H'01: 2-bit data (bit 1 and 0) are valid<br>H'02: 3-bit data (bit 2 to 0) are valid<br>H'03: 4-bit data (bit 3 to 0) are valid<br>~<br>H'0F: 16-bit data (bit 15 to 0) are valid<br>H'10: 17-bit data (bit 16 to 0) are valid<br>~<br>H'1E: 31-bit data (bit 30 to 0) are valid<br>**H'1F: 32-bit data (bit 31 to 0) are valid** |
| 31 to 5 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (q) VI6_SMG_FXOR

FXOR indicates final XOR equation's coefficient to remainder. If equation is smaller than 32-bit, set it to LSB side. If no XOR operation is required, set 0 to FXOR.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FINAL_XOR[31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FINAL_XOR[15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W : | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 31 to 0 | FINAL_XOR [31:0] | H'FFFF_FFFF | R/W | CRC Final XOR Coefficient<br>These bits specify the XOR operation coefficient to remainder. If size of polynomial is smaller than 32-bit, set to LSB side. If no XOR operation is applied, set these bits to H'0000 0000. |

### (r) VI6_SMG_FRID

FRID indicates previous frame-ID for End-to-End (E2E) protection feature. This register is used as backup purpose only and accessed by host CPU.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VALID | UNIQUEID[2:0] | | | FRAMENUM[11:0] | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | VALID | 0 | R | CRC valid flag of previous frame<br>This bit indicates previous frame ecrc_valid flag from SMG to LINK. When VALID is 0 , UNIQUEID and FRAMENUM is invalid due to previous frame is error frame (under-run) |
| 30 to 28 | UNIQUEID[2:0] | 0 | R/W | Frame Unique ID for each VSP<br>This parameter indicates a unique-ID for each VSP channels. |
| 27 to 16 | FRAMENUM[11:0] | 0 | R | Frame number of previous frame<br>This parameter indicates frame number of previous frame in continuous order. |
| 15 to 0 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (s) VI6_SMG_ERRINJ

ERRINJ indicates error is forced injecting to each corresponding error notification to ECM.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ERR_INJ[1:0] | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 to 0 | ERR_INJ[1:0] | 0 | R/W | Force error injection to ECM.<br>This bit indicates error is forced injecting to each corresponding error notification to ECM. Bit [0] is for unmatched error notification; Bit [1] is for frozen error notification. For each bits, below setting is can be applied<br>0: No error is injected to corresponding error notification<br>1: Error is injected to corresponding error notification |
| 31 to 2 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### (t) VI6_SMG_ECRC

ECRC indicates CRC of previous display frame for End-to-End (E2E) protection feature. This register is used as backup purpose only and accessed by host CPU.

| Bit : | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECRCDATA[31:16] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit : | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECRCDATA[15:0] | | | | | | | | | | | | | | | |
| Initial Value : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W : | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ECRCDATA[31:0] | 0 | R | CRC of previous display frame |

### (4) EDC Calculation

EDC calculation is no cycle delay processing. EDC code is generated in the same cycle of input pixel. EDC calculation is carried out as below steps:

SMG receives pixel output from WPF0 in format YCBCR444 or RGB. After EDC calculation, SMG send data to Display through LIF0. In case pixel format send to Display Unit is YCBCR422, set 1 to VI6_LIF0_CTRL.CFMT

Calculate EDC code for each pixel data (24bits) depending on format YCBCR422 or RGB

Output EDC is 6bits and corresponding 24bits pixel data in below figure.

**Figure 60.63 EDC calculation for each pixel format**

**(5)** CRC calculation for ECRC (End to End CRC)

VSPD calculates CRC for one frame of output image data and outputs it to DSI/CSI2-TX as 32-bits Frame CRC.

VSPD also outputs following values to DSI/CSI2-TX for Frame CRC packet.

- The value set in VI6_SMG_FRID.UNIQUEID[2:0] as Frame ID.
- The value in VI6_WPF0_FRCNT.FRCNT[31:0] as Frame number value.

Write desirable initial value to VI6_WPF0_FRCNT.FRCNT[31:0]. VI6_WPF0_FRCNT.FRCNT[31:0] is automatically incremented by plus 1 at end of frame.

The operator can calculate CRC of any polynomial by VI6_SMG_PEQU.VCRCPOLYEQU [31:0], VI6_SMG_PSZM1VCRCPOLY_SIZE_M1 [4:0], VI6_SMG_IODCFG.REFLECT_INPUT, VI6_SMG_IODCFG.REFLECT_REMAINDER, VI6_SMG_MCRCIR.CRCINI [31:0] and VI6_SMG_FXOR.FINAL_XOR [31:0].

Following table shows setting examples.

**Table 60.51** Setting Examples of CRC

| CRC Type | Register | Setting Value |
|---|---|---|
| CRC-32 IEEE 802.3 | VI6_SMG_PEQU.VCRCPOLYEQU[31:0] | H'04C1 1DB7 |
| | VI6_SMG_PSZM1VCRCPOLY_SIZE_M1[4:0] | 31 |
| | VI6_SMG_IODCFG.REFLECT_INPUT | 1 |
| | VI6_SMG_IODCFG.REFLECT_REMAINDER | 1 |
| | VI6_SMG_MCRCIR.CRCINI | H'FFFF FFFF |
| | VI6_SMG_FXOR.FINAL_XOR | H'FFFF FFFF |
| CRC-32 AUTOSAR 4.2.2 | VI6_SMG_PEQU.VCRCPOLYEQU[31:0] | H'F4AC FB13 |
| | VI6_SMG_PSZM1VCRCPOLY_SIZE_M1[4:0] | 31 |
| | VI6_SMG_IODCFG.REFLECT_INPUT | 1 |

| | | |
|---|---|---|
| | VI6_SMG_IODCFG.REFLECT_REMAINDER | 1 |
| | VI6_SMG_MCRCIR.CRCINI | H'FFFF FFFF |
| | VI6_SMG_FXOR.FINAL_XOR | H'FFFF FFFF |
| CRC-16 CCITT-FALSE | VI6_SMG_PEQU.VCRCPOLYEQU[31:0] | H'0000 1021 |
| | VI6_SMG_PSZM1VCRCPOLY_SIZE_M1[4:0] | 15 |
| | VI6_SMG_IODCFG.REFLECT_INPUT | 0 |
| | VI6_SMG_IODCFG.REFLECT_REMAINDER | 0 |
| | VI6_SMG_MCRCIR.CRCINI | H'0000 FFFF |
| | VI6_SMG_FXOR.FINAL_XOR | H'0000 0000 |
| CRC-8 SAE J1850 | VI6_SMG_PEQU.VCRCPOLYEQU[31:0] | H'0000 001D |
| | VI6_SMG_PSZM1VCRCPOLY_SIZE_M1[4:0] | 7 |
| | VI6_SMG_IODCFG.REFLECT_INPUT | 0 |
| | VI6_SMG_IODCFG.REFLECT_REMAINDER | 0 |
| | VI6_SMG_MCRCIR.CRCINI | H'0000 00FF |
| | VI6_SMG_FXOR.FINAL_XOR | H'0000 00FF |
| CRC-21 0x102899 | VI6_SMG_PEQU.VCRCPOLYEQU[31:0] | H'0010 2899 |
| | VI6_SMG_PSZM1VCRCPOLY_SIZE_M1[4:0] | 20 |
| | VI6_SMG_IODCFG.REFLECT_INPUT | 1 |
| | VI6_SMG_IODCFG.REFLECT_REMAINDER | 1 |
| | VI6_SMG_MCRCIR.CRCINI | H'001F FFFF |
| | VI6_SMG_FXOR.FINAL_XOR | H'001F FFFF |

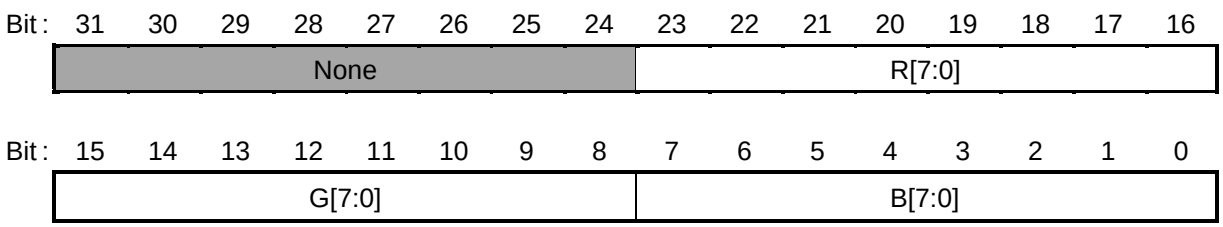
Description below shows the pixel formats and corresponding settings for CRC calculation:

**(a) RGB888 (3bytes)**

This format is used for CRC calculation 3 bytes per pixel.

- ✓ Set 2 to VI6_SMG_IODCFG.DATA_SIZE_M1
- ✓ Set 4 to VI6_SMG_IODCFG.BYTEORDER_INPUT (In case of VI6_SMG_IODCFG.REFLECT_INPUT= 0)

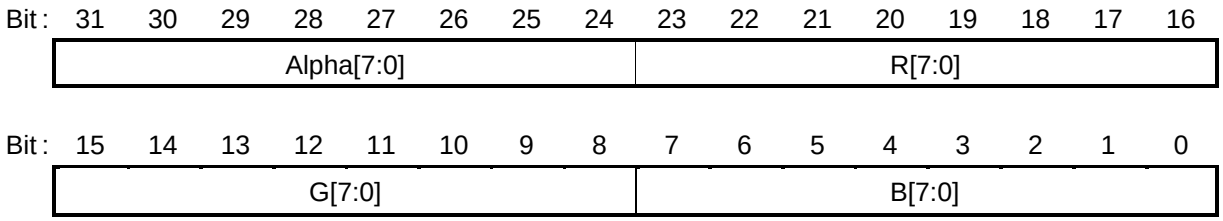
  Set 3 to VI6_SMG_IODCFG.BYTEORDER_INPUT (In case of VI6_SMG_IODCFG.REFLECT_INPUT= 1)
- ✓ Set 0 to VI6_LIF0_CTRL.CFMT



**(b) ARGB8888 (4bytes) /YcrCr444 (4bytes)**

This format is used for CRC calculation 4 bytes per pixel.

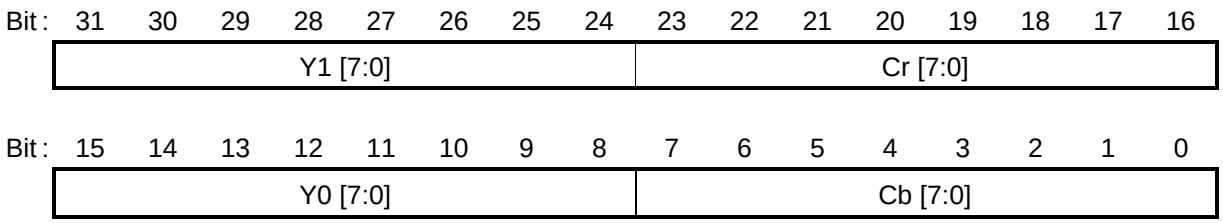
- ✓ Set 1 to VI6_SMG_MPMR.CMPDAUF
- ✓ Set alpha value to VI6_SMG_MPMR.CMPDFA
- ✓ Set 3 to VI6_SMG_IODCFG.DATA_SIZE_M1
- ✓ Set 0 to VI6_SMG_IODCFG.BYTEORDER_INPUT (In case of VI6_SMG_IODCFG.REFLECT_INPUT= 0)Set 3 to VI6_SMG_IODCFG.BYTEORDER_INPUT (In case of VI6_SMG_IODCFG.REFLECT_INPUT= 1)
- ✓ Set 0 to VI6_LIF0_CTRL.CFMT




**(c) YcbCr422**

For YcbCr422 format, SMG converts input pixel format YcbCr444 – 8bit to YcbCr422 8bit before CRC calculation. SMG allocates two neighbor pixels as below format for CRC calculation:

- ✓ Set 3 to VI6_SMG_IODCFG.DATA_SIZE_M1
- ✓ Set 0 to VI6_SMG_IODCFG.BYTEORDER_INPUT (In case of VI6_SMG_IODCFG.REFLECT_INPUT= 0)Set 3 to VI6_SMG_IODCFG.BYTEORDER_INPUT (In case of VI6_SMG_IODCFG.REFLECT_INPUT= 1)
- ✓ Set 1 to VI6_LIF0_CTRL.CFMT




**(6) Setting Rectangle Region**

Below figure describes how to set the rectangular area to the CRC calculation.

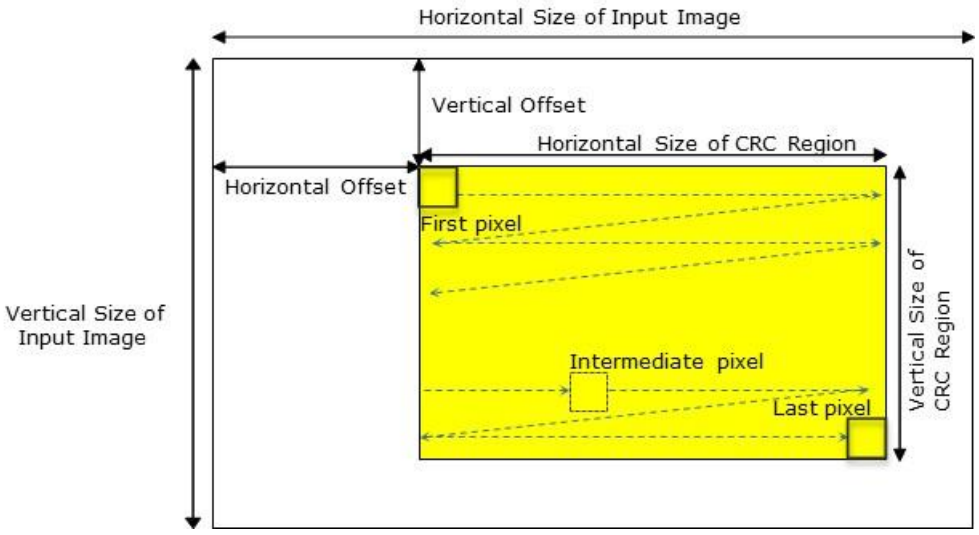



**Figure 60.64** Example Case of the CRC computation area

**Table 60.52** Register setting for the CRC Computation Area

| Word in Figure | Register of SMG | Description |
|---|---|---|
| Horizontal Size of Input Image | | This parameter is horizontal size of input image for the SMG. It is set by VSPD's register. |
| Vertical size of Input Image | | This parameter is vertical size of input image for the SMG. It is set by VSPD's register. |
| Horizontal Offset | MSPXR | This parameter specifies horizontal offset position of the CRC calculation area from left border of the input image. The offset value is the pixel unit. |
| Vertical Offset | MSPYR | This parameter specifies vertical offset position of the CRC calculation area from top border of the input image. The offset value is the line unit. |
| Horizontal Size of CRC Region | MSZXR | This parameter specifies horizontal size of the CRC calculation area. It should be less or equal to (Horizontal size of input image – Horizontal offset). The size is the pixel unit. |
| Vertical Size of CRC Region | MSZYR | This parameter specifies vertical size of the CRC calculation area. It should be less or equal to (Vertical size of input image – Vertical offset). The size is the line unit. |

### (7) Timing Chart of CRC Computation for unmatched/frozen error detection

Following figure shows the timing chart example of CRC computation and comparing for unmatched/frozen error detection.

- ✓ SMG starts error detection operation from the following frame after setting MCR.CMPR to 1.
- ✓ The CRC code is calculated in the specified rectangular area.
- ✓ The result of the CRC code is compared with the expectation (MECRCR) and the previous value (CMPCCRC).
- ✓ If the CRC code doesn't match with the expectation value, MSTR.CMPST bit is set to 1.
- ✓ If the CRC code matches with the previous value, MSTR.CMPPRE bit is set to 1.

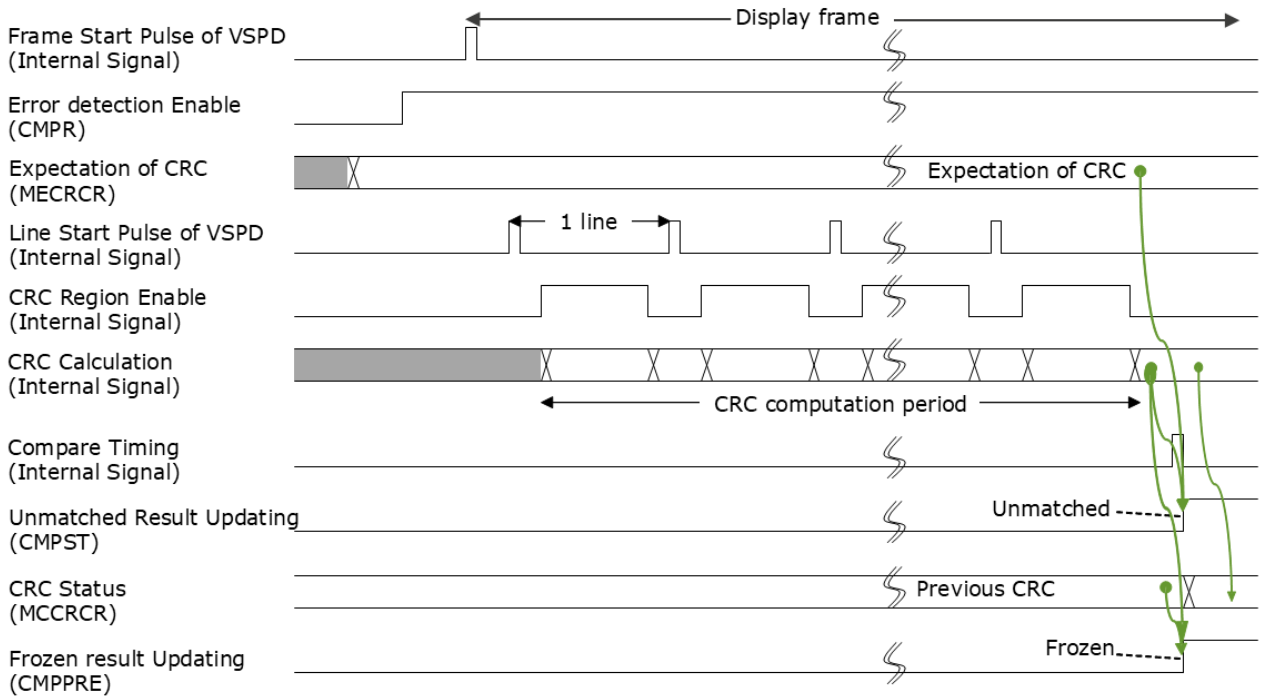



**Figure 60.65 Timing of CRC Computation and Comparing for Error Detection**

**(8) Register Update Timing**

**(a) Update Timing of Register Setting in Error Detection feature**

All registers except for VI6_SMG_MCR.CMPR are reflected in the internal register immediately. Therefore they should be set during VI6_SMG_MCR.CMPRU is 0.

The VI6_SMG_MCR.CMPR is reflected in the internal register at the beginning of the frame processing. It can be updated at any time.

**(b) Update Timing of Status Register in Error Detection feature**

The VI6_SMG_MSTR and VI6_SMG_MCCRCR are updated at the end of the frame processing. The VI6_SMG_MCR. CMPRU is updated at the beginning of the frame processing.

The following figure shows an example timing of register updating.

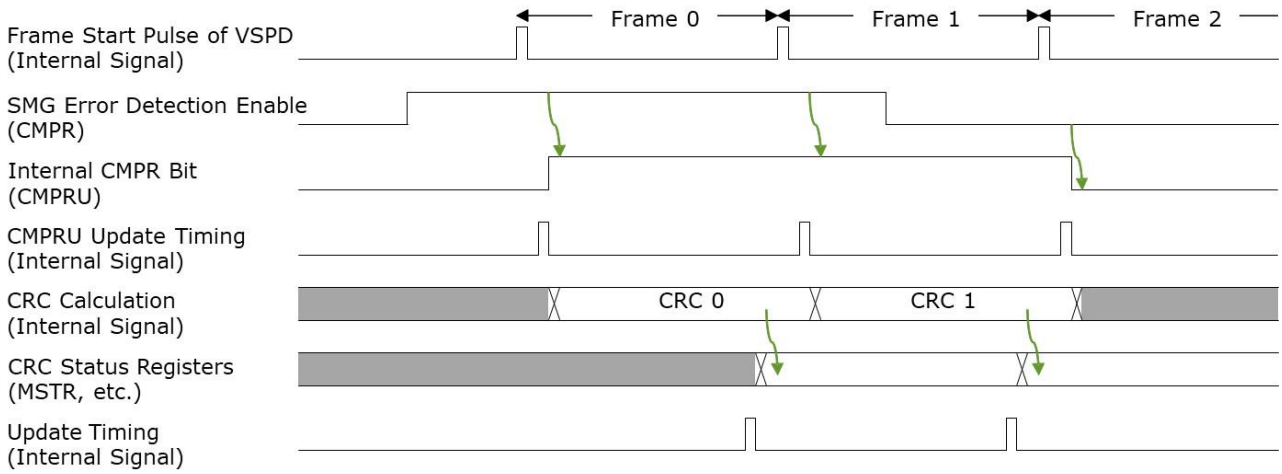



Figure 60.66 Example Timing of Register Updating

# 62. Frame Compression Processor (FCP)

## 62.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The FCP is a companion module of a video processing module in R-Car 4th generation. It provides data compression and decompression, data caching, and converting of AXI transaction in order to reduce the memory bandwidth.

## 62.2 Reserved

## 62.3 FCP for VSP (FCPV)

The FCPV conducts the bus accesses with 256bit AXI bus for the Renesas video signal processor (VSP). This module is under control of software for the VSP with the control function of the FCPV executed on the host CPU.

There are two configurations of the FCPV. Those are FCPVX and FCPVD.

**Table 62.1 Configuration Type of the FCPV**

| Type | Description |
|---|---|
| FCPVX | For VSPX |
| FCPVD | For VSPD |

### 62.3.1 Features

The FCPVD and FCPVX have the following features.

- Support out-of-order for the whole outstanding transactions

**Table 62.2 Main Function of the FCPV**

| Item | Description |
|---|---|
| Picture size supported | Minimum: 1pixels x 1lines<br>Maximum: 8190pixels x 8190 lines<br>The picture sizes must follow the restriction specified in the section “60. Video Signal Processor (VSP2)” (sections “Usage Notes”, “Input Image Size” and “Output Image Size”). |
| Color format supported | Support all formats specified in the section “60. Video Signal Processor (VSP2)” (sections “Register Descriptions”, “RPFn Input Format Registers (VI6_RPFn_INFMT)” and “WPF0 Output Format Registers (VI6_WPF0_OUTFMT)”). |
| AXI transactions | Read access:<br>Access at 256 bytes boundary<br>Support out-of-order for the whole outstanding transactions<br>Support data interleaving<br>Write access:<br>Access at 256 bytes boundary<br>Support out-of-order for the whole outstanding transactions |

### 62.3.2 Block Diagram

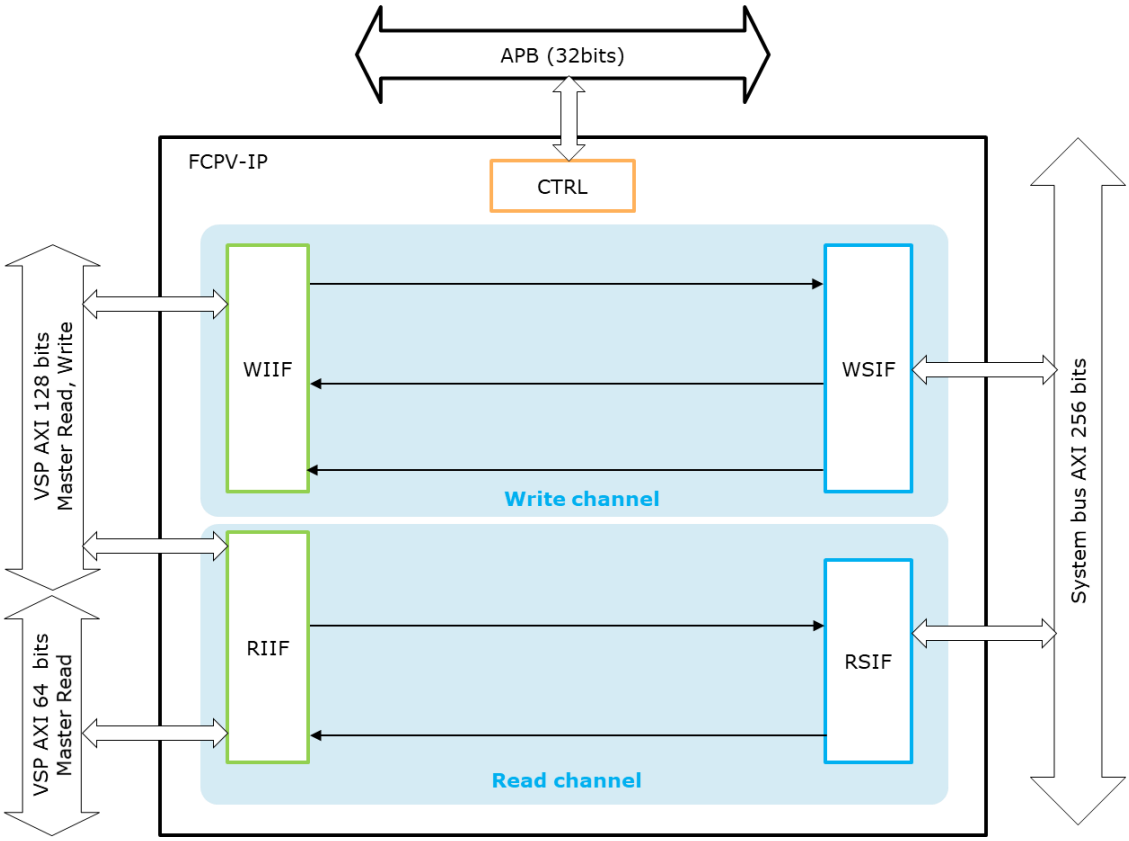



**Figure 62.1 Overall Configuration of the FCPV**

**Table 62.3 Sub-module Description**

| Abbr. | Description | Abbr. | Description |
|---|---|---|---|
| WIIF | VSP write channel interface | CTRL | FCPV controller |
| WSIF | System AXI write channel interface | RIIF | VSP read channel interface |
| - | - | RSIF | System AXI read channel interface |

### 62.3.3 IP Set and Bus Access Functions

FCPV does not work alone but in companion with other IPs (**VSPD/VSPX**) in an IP set. For each IP set the bus access function conducted by FCPV is different.

**Table 62.4 Bus access function for each IP set**

| IP set | Bus access |
|---|---|
| VSPD + FCPVD | Following functions are conducted by FCPV<br>- Read linear addressing image data<br>- Read display list data<br>- Write image data |

| | |
|---|---|
| VSPX + FCPVX | Following functions are conducted by FCPV<br>- Read linear addressing image data<br>- Read display list data |

### 62.3.4 Register Configuration

The FCPV contains the registers which can directly be read and written to by the host CPU. The host can access any long word (32 bits) in the registers. Therefore, read-modify-write is needed to change partial bits in 32bits register.

The grayed portions represent reserved areas for debugging or future functional expansion. Do not access the reserved areas. Never access the registers when the FCPV is in the module standby mode (with the operation clock placed in hold status).

The following are the base addresses of the FCPV channels.

- FCPVD0: H'FEA1 0000
- FCPVX0: H’FEDB 0000

**Table 62.5 List of FCPV Register**

| Name of Register | Abbreviation | R/W | Offset from Base Address | Initial Value | Access size |
|---|---|---|---|---|---|
| FCPV version control register | FCP_VCR | R | H’0000 | VCR_INIT (*1) | 32 |
| | | | H’0004-H’000C | | |
| FCP reset register | FCP_RST | R/W | H’0010 | H’0000 0000 | 32 |
| | | | H’0014 | | |
| FCP status register | FCP_STA | R | H’0018 | H’0000 0000 | 32 |
| | | | H’001C-H’01FC | | |

(*1) VCR_INIT: depend on product generation, refer to section 62.3.6.1 for detail

### 62.3.5 Connected Module

Following table shows modules connected to the FCPV.

**Table 62.6 Connected Module**

| Module | Connected Module | Description |
|---|---|---|
| FCPVX | VSPX | VSP for ISP data transferring |
| FCPVD | VSPD | VSP for blending and display output |

### 62.3.6 Register Description

#### 62.3.6.1 FCP_VCR: FCPV version Control Register

FCP_VCR is the version control register of the FCPV. The value read is fixed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CATEGORY | | | | | | | | REVISION | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | CATEGORY | H'01 | R | FCP use case category.<br>1: R-Car |
| 7 to 0 | REVISION | H'0A | R | Version of LSI product<br>H'0A: FCP revision is R-Car V4H<br>H'0B: FCP revision is R-Car V4M |

#### 62.3.6.2 FCP_RST: FCP Reset Register

FCP_RST: FCP Reset Register

FCP_RST controls the FCPV reset operation. The reset operation of the FCPV highly depends on the reset sequence of the connected VSP. For detailed information, refer to the section 62.3.7.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | MODRST | — | — | — | SOFTRST |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | MODRST | 0 | R/W | FCPV MODule ReSeT<br>This reset stops the FCPV module immediately regardless of the bus transaction state. To operate the FCPV after this reset, reconfigure all FCPV registers.<br>After the active period of MODRST is finished, it is switched back to 0 automatically.<br>0: NOP<br>1: Forcibly terminate the FCPV operation.<br>NOTE: Do not set both MODRST and SOFTRST bit to 1. |
| 3 to 1 | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | SOFTRST | 0 | R/W | FCPV SOFT ReSeT<br>This reset stops the FCPV module after finishing active bus transaction. To operate the FCPV after this reset, reconfigure all FCPV registers.<br>After the active period of SOFTRST is finished, it is switched back to 0 automatically.<br>0: NOP<br>1: Terminates the FCPV operation.<br>NOTE: Do not set both MODRST and SOFTRST bit to 1. |

### 62.3.6.3 FCP_STA: FCP Status Register

FCP_STA indicates the FCPV status.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ACT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0 or 1. Do not care about this value. The write value should always be 0. |
| 0 | ACT | 0 | R | FCPV Active Flag<br>0: Indicates the FCPV is not active.<br>1: Indicates the FCPV is active. |

### 62.3.7 Operation

FCPV is not invoked directly by register setting. It works if there are bus accesses from the connected IP VSP. This section explains how to control operation for FCPV connected with VSP.

#### 62.3.7.1 Operation Control Setting IP set VSPX+FCPVX

(a) FCPVX initialization

FCPVX must be initialized with following settings. This initialization must be conducted before starting image process. After initialization, those settings are kept no change until power-off or hardware reset, therefore, they don't need to be set for every image process.

(b) To start image process of VSPX + FCPVX

FCPVX registers do not need to be set when starting an image process. Therefore, refer section "60. Video Signal Processor (VSP2)" for the guidance of starting an image process.

(c) To stop image process of VSPX+FCPVX immediately

Do following steps to stop immediately an image process of VSPX + FCPVX

1. Set value 1 to VSPX register VI6_WPF0_IRQ_ENB.FREE so that frame end interruption can be issued.
2. Set value 1 to VSPX register VI6_SRESET.SRST0. VSPX will invoke termination process immediately.
3. Wait frame end interruption from VSPX or until the register bit VI6_WPF0_IRQ_STA.FRE is set to 1. After it occurs, VSPX has finished its processing.
4. Set value 1 to FCPVX register FCP_RST.SOFTRST. FCPVX will invoke termination process immediately.
5. Wait until ACT bit of the FCP_STA register and become 0. After it occurs, FCPVX has finished its processing.

With this procedure, VSPX can stop its process quickly, but the output frame of the last frame is corrupted.

#### 62.3.7.2 Operation Control Setting for IP set VSPD+FCPVD

(a) FCPVD initialization

FCPVD must be initialized with following settings. This initialization must be conducted before starting image process. After initialization, those settings are kept no change until power-off or hardware reset, therefore, they don't need to be set for every image process.

(b) To start image process of VSPD+FCPVD

FCPVD registers do not need to be set when starting an image process. Therefore, refer the section "60.Video Signal Processor (VSP2)" for the guidance of starting an image process.

(c) To stop image process of VSPD+FCPVD immediately

Do following steps to stop immediately an image process

1. Stop the DU first if it is connected with VSPD (VI6_LIF0_CTRL.LIF_EN = 1). Refer to the section "64. Display Unit (DU)" for the way of stopping the DU.
2. Set value 1 to VSPD register VI6_WPF0 _IRQ_ENB.FREE so that frame end interruption can be issued.
3. Set value 1 to VSPD register VI6_SRESET.SRST0. VSPD will invoke termination process immediately.
4. Wait frame end interruption from VSPD or until the register bit VI6_WPF0_IRQ_STA.FRE is set to 1. After it occurs, VSPD has finished its processing.
5. Set value 1 to FCPVD register FCP_RST.SOFTRST. FCPVD will invoke termination process immediately. (*1)
6. Wait until ACT bit of the FCP_STA register become 0. After it occurs, FCPVD has finished its processing. (*1)

NOTE:

With this procedure, VSP can stop its process quickly, but the output frame of the last frame is corrupted.

#### 62.3.7.3 Reset Operation

FCPV reset can be conducted by software reset and power on reset. The software reset which is controlled by software, is used to stop the FCPV operation. The software reset operation is not executed immediately; it is executed on completion of the bus transaction. The other is power on reset which initialize the whole FCPV logic immediately.

**(1) Software Reset**

1) Activate software reset sequence

   Set FCP_RST.SOFTRST to the value 1.

2) Confirm software reset sequence finished:

   FCP_STA.ACT becomes 0. All remain transactions with the bus system finished.

The procedure to stop operation of FCPV and connected IP is descripted at section 62.3.7.1.

**(2) Power-on Reset, Hardware reset**

The power-on reset is controlled by an externally input signal called hardware reset. This reset initializes the whole internal logic of FCPV.

### 62.3.8 Usage Notes

#### 62.3.8.1 Manipulating FCPV+VSP with CPG module stop, software reset

When using Module stop and Software Reset for VSP and FCPV, follow the notes 1) to 2) to prevent FCPV, VSP from the unstable status which may cause hang or wrong operation.

Note 1) Stop sequence of VSP and FCPV.

Take stop process of VSP and FCPV guide in 62.3.7 before applying Software Reset and enabling Module Stop specified in section 9.

Note 2) Start sequence of VSP and FCPV

Take start process of VSP and FCPV guide in section “60. Video Signal Processor (VSP2)” after applying Software Reset and disabling Module Stop

### 62.3.9 Safety mechanism

#### 62.3.9.1 Safety mechanism of ISP M2M path (include VSPX/FCPVX)

The Runtime region include the whole path from FCPVX to ISP output. However, because the Runtime test does not cover for MBIST the SRAM of FCPVX supports EDC for the safety of SRAM. FCPVX supports EDC for response buffer. The EDC error is sent to VSP to issue error notification (interruption) to ECM.

# 63. Data Compression Processor (FCPR)

## 63.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The FCPR is a data compression and decompression module. It provides lossless data compression and decompression to reduce the memory bandwidth.

The FCPR consists of three modules as FCPRC, FCPRS and FCPRM.

The following Figure 63.1 shows an overview of each module.

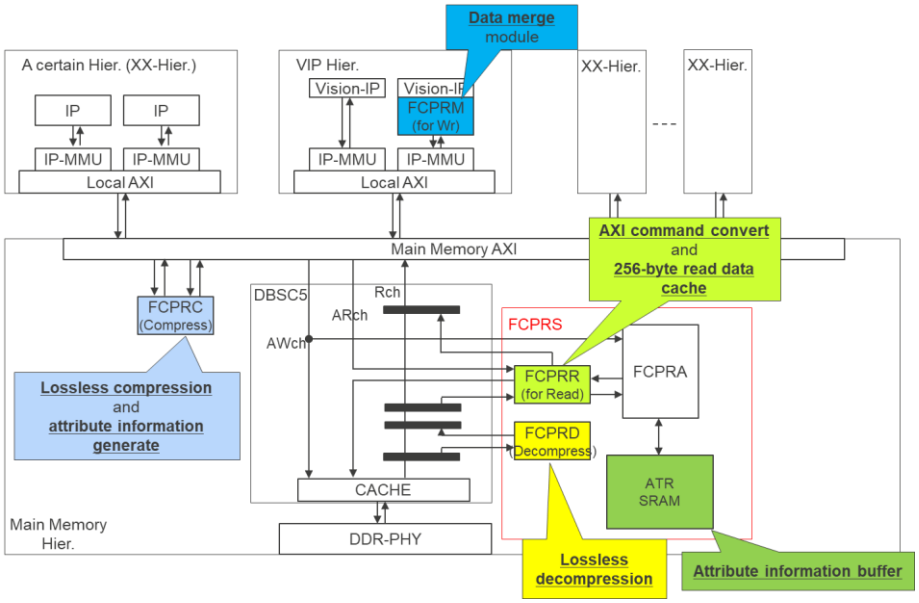



**Figure 63.1 LSI System Block Diagram**

### (1) FCPRC

The FCPRC is connected to AXMM for data compression.

The FCPRC is a lossless compression module to reduce the memory bandwidth. In compression, it also generates attribute information (ATR).

### (2) FCPRS

The FCPRS is connected to DBSC for data decompression.

The FCPRS is a lossless decompression module. This module consists of FCPRR, FCPRA and FCPRD.

### (3) FCPRM

The FCPRM is a module that merges write data of less than 256 bytes.

### (4) FCPRR

The FCPRR is an AXI command converter for lossless decompression. The FCPRR cooperates with FCPRD to decompress data.

### (5) FCPRA

The FCPRA manages the SRAM buffer of ATR.

### (6) FCPRD

The FCPRD is a decompression core module.

### 63.1.1 Features

The feature of the FCPR is lossless compression.

Lossless compression ratio is typically 60% (compressed/original). It may not be compressed since lossless compression relies on the statistical nature of the original data.

### 63.1.2 Register Configuration

The FCPRC and FCPRM contain the registers which the host CPU can read and write directly. The host can access any long word (32-bit) in the registers. All FCPR registers are writable and readable by only 32-bit unit. To write partial bits in 32-bit, read-modify-write is needed.

The grayed portions represent reserved areas for debugging or future functional expansion. Do not access the reserved areas. In the module standby mode (with the operation clock placed in hold status), FCPR ignores write data. Read data is unstable and meaningless.

Do not access registers of FCPR during CPG soft reset or module stop processing.

Table 63.1 is the base addresses of the FCPR.

Table 63.2, Table 63.3, Table 63.4 and Table 63.5 show list of FCPR register.

**Table 63.1 FCPR base address**

| Bus Domain | Master IP | FCPR | FCPR base address |
|---|---|---|---|
| VIP | DOF | FCPRM_DOF | H'E7AB 1000 |
| MM | - | FCPRC | H'E67A 3000 |
| | AXMM | FCPR | H'E678 5700 |
| | DBSC5 | FCPRS | H'E679 0000 |

**Table 63.2 List of FCPRC Register**

| Name of Register | Abbreviation | R/W | Offset from Base Address | Initial Value | Access size |
|---|---|---|---|---|---|
| FCPRC identification information register | FCPRC_VID | R | H'000C | H'0000 144F | 32 |
| FCPRC reset register | FCPRC_RST | R/W | H'0010 | H'0000 0000 | 32 |
| FCPRC clock stop control register | FCPRC_CLK_STOP | R/W | H'001C | H'0000 0000 | 32 |
| FCPRC lossless compression status 0 register | FCPRC_CMP_STA0 | R/W | H'0040 | H'0000 0000 | 32 |
| FCPRC lossless compression status 1 register | FCPRC_CMP_STA1 | R/W | H'0044 | H'0000 0000 | 32 |
| FCPRC error detection status register | FCPRC_EDC_STS | R/W | H'0074 | H'0000 0000 | 32 |
| FCPRC bus error detection status register | FCPRC_BUSEDC_STS | R/W | H'0080 | H'0000 0000 | 32 |
| FCPRC bus error detection source0 register | FCPRC_BUSEDC_SRC0 | R/W | H'0084 | H'0000 0000 | 32 |
| FCPRC bus error detection control register | FCPRC_BUSEDC_CTL | R/W | H'008C | H'0000 0000 | 32 |

**Table 63.3 List of FCPRM Register**

| Name of Register | Abbreviation | R/W | Offset from Base Address | Initial Value | Access size |
|---|---|---|---|---|---|
| FCPRM identification information register | FCPRM_VID | R | H'000C | H'0000 144F | 32 |
| FCPRM reset register | FCPRM_RST | R/W | H'0010 | H'0000 0000 | 32 |
| FCPRM buffer clear register | FCPRM_BRST | R/W | H'0020 | H'0000 0000 | 32 |
| FCPRM buffer flush register | FCPRM_BFLS | R/W | H'0024 | H'0000 0000 | 32 |
| FCPRM data merge control register | FCPRM_MRG_CTRL | R/W | H'0030 | H'0000 0000 | 32 |
| FCPRM error detection status register | FCPRM_EDC_STS | R/W | H'0074 | H'0000 0000 | 32 |
| FCPRM merge enabled area start address 0 register | FCPRM_MRG_STADR0 | R/W | H'0084 | H'0000 0000 | 32 |
| FCPRM merge enabled area end address 0 register | FCPRM_MRG_EDADR0 | R/W | H'0088 | H'0000 0000 | 32 |
| FCPRM merge enabled area start address 1 register | FCPRM_MRG_STADR1 | R/W | H'0094 | H'0000 0000 | 32 |
| FCPRM merge enabled area end address 1 register | FCPRM_MRG_EDADR1 | R/W | H'0098 | H'0000 0000 | 32 |
| FCPRM merge enabled area start address 2 register | FCPRM_MRG_STADR2 | R/W | H'00A4 | H'0000 0000 | 32 |
| FCPRM merge enabled area end address 2 register | FCPRM_MRG_EDADR2 | R/W | H'00A8 | H'0000 0000 | 32 |

| Name of Register | Abbreviation | R/W | Offset from Base Address | Initial Value | Access size |
|---|---|---|---|---|---|
| FCPRM merge enabled area start address 3 register | FCPRM_MRG_STADR3 | R/W | H'00B4 | H'0000 0000 | 32 |
| FCPRM merge enabled area end address 3 register | FCPRM_MRG_EDADR3 | R/W | H'00B8 | H'0000 0000 | 32 |
| FCPRM ID0 setting register | FCPRM_ID0 | R/W | H'00C0 | H'0000 0000 | 32 |
| FCPRM ID1 setting register | FCPRM_ID1 | R/W | H'00C4 | H'0000 0000 | 32 |
| FCPRM ID2 setting register | FCPRM_ID2 | R/W | H'00C8 | H'0000 0000 | 32 |
| FCPRM ID3 setting register | FCPRM_ID3 | R/W | H'00CC | H'0000 0000 | 32 |
| FCPRM ID4 setting register | FCPRM_ID4 | R/W | H'00D0 | H'0000 0000 | 32 |
| FCPRM ID5 setting register | FCPRM_ID5 | R/W | H'00D4 | H'0000 0000 | 32 |
| FCPRM ID6 setting register | FCPRM_ID6 | R/W | H'00D8 | H'0000 0000 | 32 |
| FCPRM ID7 setting register | FCPRM_ID7 | R/W | H'00DC | H'0000 0000 | 32 |
| FCPRM ID8 setting register | FCPRM_ID8 | R/W | H'00E0 | H'0000 0000 | 32 |
| FCPRM ID9 setting register | FCPRM_ID9 | R/W | H'00E4 | H'0000 0000 | 32 |

**Table 63.4 List of FCPR Register in AXMM**

| Name of Register | Abbreviation | R/W | Offset from Base Address | Initial Value | Access size |
|---|---|---|---|---|---|
| FCPR lossless compression control register | FCPR_CMP_CTRL | R/W | H'0030 | H'0000 0000 | 32 |
| FCPR 4G-byte space index register | FCPR_CMP_SPACE | R/W | H'0080 | H'0000 0000 | 32 |
| FCPR compression enabled area start address register | FCPR_CMP_STADR | R/W | H'0084 | H'0000 0000 | 32 |
| FCPR compression enabled area end address register | FCPR_CMP_EDADR | R/W | H'0088 | H'0000 0000 | 32 |

**Table 63.5 List of FCPRS Register in DBSC5**

| Name of Register | Abbreviation | R/W | Offset from Base Address | Initial Value | Access size |
|---|---|---|---|---|---|
| FCPRS control register | FCPRS_CTRL | R/W | H'0110 | H'0000 0000 | 32 |
| FCPRS clock stop control register | FCPRS_CLK_STOP | R/W | H'0600 | H'0000 0000 | 32 |
| FCPRS cache line 0 control register | FCPRS_CL0_CTRL | R/W | H'0604 | H'0000 0000 | 32 |
| FCPRS cache line 1 control register | FCPRS_CL1_CTRL | R/W | H'0608 | H'0000 0000 | 32 |
| FCPRS cache line 2 control register | FCPRS_CL2_CTRL | R/W | H'060C | H'0000 0000 | 32 |
| FCPRS cache line 3 control register | FCPRS_CL3_CTRL | R/W | H'0610 | H'0000 0000 | 32 |
| FCPRS cache line 4 control register | FCPRS_CL4_CTRL | R/W | H'0614 | H'0000 0000 | 32 |

| Name of Register | Abbreviation | R/W | Offset from Base Address | Initial Value | Access size |
|---|---|---|---|---|---|
| FCPRS cache line 5 control register | FCPRS_CL5_CTRL | R/W | H'0618 | H'0000 0000 | 32 |
| FCPRS cache line 6 control register | FCPRS_CL6_CTRL | R/W | H'061C | H'0000 0000 | 32 |
| FCPRS cache line 7 control register | FCPRS_CL7_CTRL | R/W | H'0620 | H'0000 0000 | 32 |
| FCPRS error detection control register | FCPRS_EDC_CTL | R/W | H'7700 | H'0000 0000 | 32 |
| FCPRS error detection status register | FCPRS_EDC_STA | R | H'7704 | H'0000 0000 | 32 |
| FCPRS error detection status clear register | FCPRS_EDC_STA_CLR | R/W | H'7708 | H'0000 0000 | 32 |
| FCPRS lossless decompression status 0 register | FCPRS_ATR_STA0 | R | H'770C | H'0000 0000 | 32 |
| FCPRS lossless decompression status 0 clear register | FCPRS_ATR_STA0_CLR | R/W | H'7710 | H'0000 0000 | 32 |
| FCPRS lossless decompression status 1 register | FCPRS_ATR_STA1 | R | H'7714 | H'0000 0000 | 32 |
| FCPRS lossless decompression status 1 clear register | FCPRS_ATR_STA1_CLR | R/W | H'7718 | H'0000 0000 | 32 |

### 63.1.3 Connected Module

Table 63.6, Table 63.7 and Table 63.8 show modules connected to the FCPRC, FCPRS and FCPRM.

**Table 63.6 FCPRC Connected Module**

| Module | Connected Module | Description |
|---|---|---|
| FCPRC | CPG | Clock controller |
| | ECM | Error control module |
| | Hire ECM | BUS hierarchy error control module |
| | AXI-BUS | Bus |

**Table 63.7 FCPRS Connected Module**

| Module | Connected Module | Description |
|---|---|---|
| FCPRS | DBSC5 | External Bus Controller for SDRAM |
| | CPG | Clock controller |
| | ECM | Error control module |

**Table 63.8 FCPRM Connected Module**

| Module | Connected Module | Description |
|---|---|---|
| FCPRM | DOF | Optical Flow |
| | CPG | Clock controller |
| | ECM | Error control module |
| | AXI-BUS | Bus |

## 63.2 Register Description

### 63.2.1 FCPRC Register Description

#### 63.2.1.1 FCPRC Identification Information Register (FCPRC_VID)

FCPRC_VID is identification information register of the FCPRC. The read value is fixed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VCID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 15 to 0 | VCID | H’144F | R | Identification number (FCP) |

### 63.2.1.2 FCPRC Reset Register (FCPRC_RST)

FCPRC_RST controls the FCPRC reset operation. For detailed information, refer to the section 63.3.9.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | FCP RSTF | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 4 | FCPRSTF | 0 | R/W | FCPRC module forced reset.<br>This reset stops the FCPRC module immediately. To operate the FCPRC after this reset, reconfigure all FCPRC registers.<br>After the active period of FCPRSTF is finished, it is switched back to 0 automatically.<br>0: NOP<br>1: Stop the FCPRC operation. |
| 3 to 0 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |

### 63.2.1.3 FCPRC Clock Stop Control Register (FCPRC_CLK_STOP)

FCPRC_CLK_STOP controls the clock stop operation for the FCPRC lossless compression core (COMP).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CC STP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 0 | CCSTP | 0 | R/W | COMP Clock Supply Setting.<br>Control enabling / disabling of the clock stop operation.<br>0: Clock supply begins at when the outstanding counter is not 0. When the outstanding counter is 0, the clock supply to it stops. (Initial value)<br>1: Always supplies the operation clock. |

#### 63.2.1.4 FCPRC Lossless Compression Status 0 Register (FCPRC_CMP_STA0)

FCPRC_CMP_STA0 indicates the FCPRC lossless compression status.

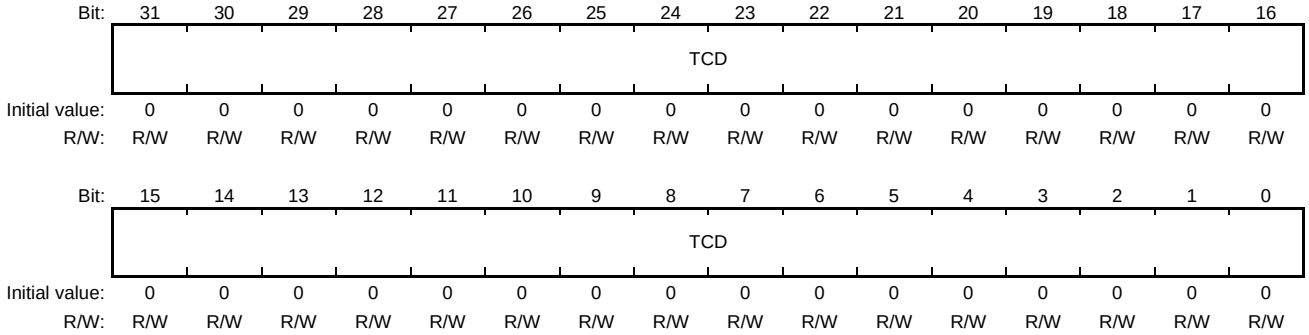
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TWD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TWD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TWD | 0 | R/W | Total amount of write data to compression area in 64-byte unit.<br>Data size before compression. To clear it, write 0 to this register. Note that writing not 0 doesn't change it. |

### 63.2.1.5 FCPRC Lossless Compression Status 1 Register (FCPRC_CMP_STA1)

FCPRC_CMP_STA1 indicates the FCPRC lossless compression status.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TCD | 0 | R/W | Total amount of compressed data in 64-byte unit.<br>Data size after compression. To clear it, write 0 to this register. Note that writing not 0 doesn't change it. |

### 63.2.1.6 FCPRC Error Detection Status Register (FCPRC_EDC_STS)

FCPRC_EDC_STS indicates the FCPRC error detection status.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CRC ERR | RAM ERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 1 | CRCERR | 0 | R/W | CRC error status<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it.<br>For details, refer 63.5.6. |
| 0 | RAMERR | 0 | R/W | RAM checksum error status<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it.<br>For details, refer 63.5.1. |

Note: error injection function

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | – | – | W | If this register is set to 0x5361_6c74, FCPRC issue error notification to ECM module.<br>Note that DCLSERR and RAMERR don't change its value. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | – | – | W | If this register is set to 0x5361_6c75, CTRL issue error injection pulse to DCLS comparator module.<br>Note that DCLSERR and RAMERR don't change its value. |

### 63.2.1.7 FCPRC Bus Error Detection Status Register (FCPRC_BUSEDC_STS)

FCPRC_BUSEDC_STS indicates the Bus EDC error detection status. For overview of bus safety mechanisms, refer section 18.5.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | WBST ERR | BTID ERR | — | — | — | — | — | BEDC ERR | WEDC ERR | AWED CERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 9 | WBSTERR | 0 | R/W | AXI W channel BstID error status<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it. |
| 8 | BTIDERR | 0 | R/W | AXI B channel TID error status<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it. |
| 7 to 3 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 2 | BEDCERR | 0 | R/W | AXI B channel EDC error status<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it. |
| 1 | WEDCERR | 0 | R/W | AXI W channel EDC error status<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it. |
| 0 | AWEDCERR | 0 | R/W | AXI AW channel EDC error status<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it. |

### 63.2.1.8 FCPRC Bus Error Detection Source0 Register (FCPRC_BUSEDC_SRC0)

FCPRC_BUSEDC_SRC0 indicates the SrcID of error transaction. For overview of bus safety mechanisms, refer section 18.5.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | BSRCID | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WSRCID | | | | | | | | AWSRCID | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 23 to 16 | BSRCID | 0 | R/W | SrcID of error transaction at B channel<br>To clear it, write 0 to this register. Note that writing not 0 doesn't change it |
| 15 to 8 | WSRCID | 0 | R/W | SrcID of error transaction at W channel<br>To clear it, write 0 to this register. Note that writing not 0 doesn't change it. |
| 7 to 0 | AWSRCID | 0 | R/W | SrcID of error transaction at AW channel<br>To clear it, write 0 to this register. Note that writing not 0 doesn't change it. |

### 63.2.1.9 FCPRC Bus Error Detection Control Register (FCPRC_BUSEDC_CTL)

FCPRC_BUSEDC_CTL controls the error injection mode for the Bus EDC module. For overview of bus safety mechanisms, refer section 18.5.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | BERRINJ | WERRINJ | AWERRINJ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 2 | BERRINJ | 0 | R/W | Error Injection mode enable at B channel<br>Writing 1 to this bit assert error injection to BUS EDC module. When 0 is written, the error injection is deasserted. |
| 1 | WERRINJ | 0 | R/W | Error Injection mode enable at W channel<br>Writing 1 to this bit assert error injection to BUS EDC module. When 0 is written, the error injection is deasserted. |
| 0 | AWERRINJ | 0 | R/W | Error Injection mode enable at AW channel<br>Writing 1 to this bit assert error injection to BUS EDC module. When 0 is written, the error injection is deasserted. |

### 63.2.2 FCPRM Register Description

#### 63.2.2.1 FCPRM Identification Information Register (FCPRM_VID)

FCPRM_VID is identification information register of the FCPRM. The read value is fixed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VCID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 15 to 0 | VCID | H’144F | R | Identification number (FCP) |

#### 63.2.2.2 FCPRM Reset Register (FCPRM_RST)

FCPRM_RST controls the FCPRM reset operation. For detailed information, refer to the section 63.3.9.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | FCP RSTF | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 4 | FCPRSTF | 0 | R/W | FCPRM module forced reset.<br>This reset stops the FCPRM module immediately. To operate the FCPRM after this reset, reconfigure all FCPRM registers.<br>After the active period of FCPRSTF is finished, it is switched back to 0 automatically.<br>0: NOP<br>1: Stop the FCPRM operation. |
| 3 to 0 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |

### 63.2.2.3 FCPRM Buffer Clear Register (FCPRM_BRST)

FCPRM_BRST controls reset operation of the buffer of write data. For detailed information, refer to the section 63.3.6.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | BRST ID9 | BRST ID8 | BRST ID7 | BRST ID6 | BRST ID5 | BRST ID4 | BRST ID3 | BRST ID2 | BRST ID1 | BRST ID0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 9 | BRSTID9 | 0 | R/W | Reset the buffer of write data of ID9.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 8 | BRSTID8 | 0 | R/W | Reset the buffer of write data of ID8.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 7 | BRSTID7 | 0 | R/W | Reset the buffer of write data of ID7.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 6 | BRSTID6 | 0 | R/W | Reset the buffer of write data of ID6.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 5 | BRSTID5 | 0 | R/W | Reset the buffer of write data of ID5.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 4 | BRSTID4 | 0 | R/W | Reset the buffer of write data of ID4.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 3 | BRSTID3 | 0 | R/W | Reset the buffer of write data of ID3.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 2 | BRSTID2 | 0 | R/W | Reset the buffer of write data of ID2.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 1 | BRSTID1 | 0 | R/W | Reset the buffer of write data of ID1.<br>0: NOP<br>1: reset<br>The read value is always 0. |
| 0 | BRSTID0 | 0 | R/W | Reset the buffer of write data of ID0.<br>0: NOP<br>1: reset<br>The read value is always 0. |

#### 63.2.2.4 FCPRM Buffer Flush Register (FCPRM_BFLS)

FCPRM_BFLS controls flush operation for the buffer of write data. For detailed information, refer to the section 63.3.6.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | BFLS ID9 | BFLS ID8 | BFLS ID7 | BFLS ID6 | BFLS ID5 | BFLS ID4 | BFLS ID3 | BFLS ID2 | BFLS ID1 | BFLS ID0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 9 | BFLSID9 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID9 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |
| 8 | BFLSID8 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID8 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |
| 7 | BFLSID7 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID7 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |
| 6 | BFLSID6 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID6 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |
| 5 | BFLSID5 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID5 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |
| 4 | BFLSID4 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID4 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |
| 3 | BFLSID3 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID3 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | BFLSID2 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID2 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |
| 1 | BFLSID1 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID1 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |
| 0 | BFLSID0 | 0 | R/W | Flush the buffer of write data.<br>The read value indicates FCPRM stays in flush process of ID0 (flush of merging buffer). When flush process is finished, this bit becomes 0 automatically.<br>0: not flush<br>1: flush |

### 63.2.2.5 FCPRM Data Merge Control Register (FCPRM_MRG_CTRL)

FCPRM_MRG_CTRL controls the FCPRM data merge operation.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MRGEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 0 | MRGEN | 0 | R/W | Data mearge enabled.<br>0: disabled<br>1: enabled |

### 63.2.2.6 FCPRM Error Detection Status Register (FCPRM_EDC_STS)

FCPRM_EDC_STS indicates the FCPRM error detection status.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RAM ERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 0 | RAMERR | 0 | R/W | RAM checksum error status<br>0: no error<br>1: error occurred<br>To clear it, write 0 to this bit. Note that writing 1 doesn't change it.<br>For details, refer 63.5.3. |

Note: error injection function

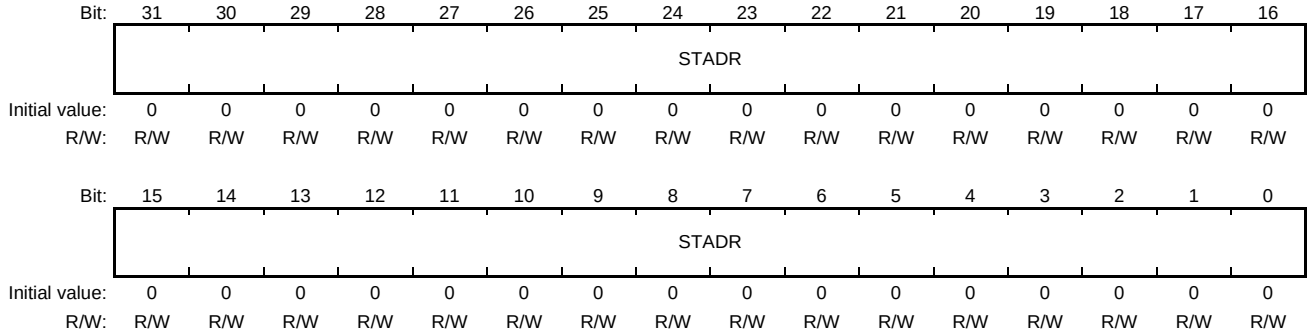
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | – | – | W | By writing 0x5361_6c74, FCPRM issues error notification to ECM module as error injection function.<br>The field bit [31:1] in this register returns always 0 regardless written value. Written value is used for just error injection function.<br>Note that field bit [0] doesn't change its value. |

### 63.2.2.7 FCPRM Merge Enabled Area Start Address Register (FCPRM_MRG_STADRn (n = 0 to 3))

FCPRM_MRG_STADR*n* specifies the start address of merge enabled area *n* (*n* = 0 to 3).




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | STADR | 0 | R/W | Start address of merge enabled area *n* (*n* = 0 to 3)<br>This register is available when data merge is enabled.<br>It shall be 256-byte aligned address. |

### 63.2.2.8 FCPRM Merge Enabled Area End Address Register (FCPRM_MRG_EDADRn (n = 0 to 3))

FCPRM_MRG_EDADR*n* specifies the end address of merge enabled area *n* (*n* = 0 to 3).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EDADR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EDADR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | EDADR | 0 | R/W | End address of merge enabled area *n* (*n* = 0 to 3)<br>This register is available when data merge is enabled.<br>It shall be (256-byte aligned address + 255). |

### 63.2.2.9 FCPRM ID Setting Register (FCPRM_IDn (n = 0 to 9))

FCPRM_IDn specifies the FCPRM IDn setting (n = 0 to 9). These registers shall be set as shown in Table 63.9.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ID VALID | — | — | — | IDPAT | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | IDMASK | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

**Table 63.9 Setting FCPRM_ID*n* register**

| Bus Domain | Connected Module | Setting of FCPRM_ID*n* (*n* = 0 to 9) |
|---|---|---|
| VIP | DOF | FCPRM_ID0: H'8000 0007 |
| | | FCPRM_ID1: H'8001 0007 |
| | | FCPRM_ID2: H'8002 0007 |
| | | FCPRM_ID3: H'8003 0007 |
| | | FCPRM_ID4: H'8004 0007 |
| | | Others: H'0000 0000 |

### 63.2.3 FCPR Register Description in AXMM

#### 63.2.3.1 FCPR Lossless Compression Control Register (FCPR_CMP_CTRL)

FCPR_CMP_CTRL controls the FCPR lossless compression operation.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CMP EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R/W | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 0 | CMPEN | 0 | R/W | Lossless compression enabled.<br>0: disabled<br>1: enabled |

### 63.2.3.2 FCPR 4G-byte Space Index Register (FCPR_CMP_SPACE)

FCPR_CMP_SPACE specifies the index of 4G-byte space for FCPR_CMP_STADR/FCPR_CMP_EDADR.

When compression is enabled, all FCPR_CMP_SPACE/FCPR_CMP_STADR/FCPR_CMP_EDADR registers shall be set for compression area.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SPACE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | – | All 0 | R/W | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 7 to 0 | SPACE | 0 | R/W | Index of 4G-byte space for FCPR_CMP_STADR/FCPR_CMP_EDADR.<br>It is higher 8-bit of 40-bit address. Use only the lower 6 bits.<br>Ignore the high-order bits 39 and 38 of the address.<br>This register is available when compression is enabled. |

#### 63.2.3.3 FCPR Compression Enabled Area Start Address Register (FCPR_CMP_STADR)

FCPR_CMP_STADR specifies the start address of compression enabled area.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STADR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STADR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | STADR | 0 | R/W | Start address of compression enabled area<br>This register is available when compression is enabled.<br>It shall be 1G-byte aligned address. Lower 30 bits shall be 0. |

#### 63.2.3.4 FCPR Compression Enabled Area End Address Register (FCPR_CMP_EDADR)

FCPR_CMP_EDADR specifies the end address of compression enabled area.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EDADR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EDADR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | EDADR | 0 | R/W | End address of compression enabled area<br>This register is available when compression is enabled.<br>It shall be (256-byte aligned address + 255). |

### 63.2.4 FCPRS Register Description in DBSC5

#### 63.2.4.1 FCPRS Control Register (FCPRS_CTRL)

FCPRS_CTRL controls the FCPRS operation.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CCDS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 0 | CCDS | 0 | R/W | Cache disabled.<br>0: enabled<br>1: disabled |

### 63.2.4.2 FCPRS Clock Stop Control Register (FCPRS_CLK_STOP)

FCPRS_CLK_STOP controls the clock stop operation for the FCPRS lossless decompression core (DCMP).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DCSTP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 0 | DCSTP | 0 | R/W | DCMP Clock Supply Setting.<br>Control enabling / disabling of the clock stop operation.<br>0: Stop the supply of the operating clock. (Initial value)<br>1: Always supplies the operation clock. |

### 63.2.4.3 FCPRS Cache Line [n] Control Register (n=0-7) (FCPRS_CL[n]_CTRL)

FCPRS_CL[n]_CTRL specifies the cache line n ID setting (n = 0 to 7). FCPRS has an 8-entry simple cache for read data. FCPRS can assign cache lines to dedicated IP by ARID and SrcIDs in ARUSER. An ID that concatenates ARID and SrcID is called Cache_line_ID set by this register.

Transactions matching Cache_line_ID of certain cache lines are restricted to use those only. Transactions not matching any Cache_line_ID use cache lines which have no registration (IDVALID bit is B'0) only. If no cache line is available for a transaction (e.g., in case all cache lines support registration and none is matching), the transaction is transparently passed through. If transactions match multiple Cache_line_ID, a round-robin decision for the used cache line is made.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IDVALID | IDPAT | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | IDMASK | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | IDVALID | 0 | R/W | Valid of Cache_line_ID<br>0: Cache_line_ID n is invalid. (No registration)<br>1: Cache_line_ID n is valid. (Registration) |
| 30 to 16 | IDPAT | H'0 | R/W | Cache_line_ID pattern bit.<br>These bits are the bit concatenation of ARID*1 and SrcID*2.<br>{ARID[6:0], SrcID[7:0]} |
| 15 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 14 to 0 | IDMASK | H'0 | R/W | Cache_line_ID mask bit.<br>Setting these bits masks the comparison bit when checking for a transaction ID and IDPAT match. Bits set to 1 are valid. Bits set to 0 are masked (ignored). |

*1: The ARIDs that can be set are shown in Table 63.10.

*2: The SrcIDs that can be set are shown in Table 63.11.

**Table 63.10 ARID setting**

| IP | DMA Channel | ARID |
|---|---|---|
| DOF | 5 | H'0 |
| | 6 | H'1 |
| | 7 | H'2 |
| | 8 | H'3 |
| | 9 | H'4 |
| | 10 | H'5 |
| | 11 | H'6 |
| | 12 | H'7 |
| | 13 | H'8 |
| SMD-PS | 3 | H'0 |
| | 4 | H'1 |
| | 5 | H'2 |
| | 6 | H'3 |
| | 7 | H'4 |
| SMD-POST | 10 | H'0 |
| | 11 | H'1 |
| | 12 | H'2 |

**Table 63.11 SrcID setting**

| IP | SrcID |
|---|---|
| DOF | H'A0 |
| SMD-PS | H'A2 |
| SMD-POST | H'A4 |

#### 63.2.4.4 FCPRS Error Detection control Register (FCPRS_EDC_CTL)

FCPRS_EDC_CTL controls error checking function and injecting dummy error for testing.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DCLS FEN | EDC FEN | — | — | — | — | CRCE INJ | AXIFE INJ | RAME INJ | DCLS EINJ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 9 | DCLSFEN | 0 | R/W | This bit forces the DCLS error notification signal to be asserted.<br>0: Does not assert DCLS error notification signal forcibly.<br>1: Asserts DCLS error notification signal forcibly.<br>For details, refer 63.5.5. |
| 8 | EDCFEN | 0 | R/W | This bit forces the EDC error notification signal to be asserted.<br>0: Does not assert EDC error notification signal forcibly.<br>1: Asserts EDC error notification signal forcibly.<br>For details, refer 63.5.2. |
| 7 to 4 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 3 | CRCEINJ | 0 | R/W | This bit inject error to CRC checker.<br>0: Does not inject error to cheker<br>1: inject error to checker<br>For details, refer 63.5.7 |
| 2 | AXIFEINJ | 0 | R/W | This bit inject error to AXMM I/F EDC checker.<br>0: Does not inject error to cheker<br>1: inject error to checker<br>For details, refer 63.5.2. |
| 1 | RAMEINJ | 0 | R/W | This bit inject error to SRAM EDC checker.<br>0: Does not inject error to cheker<br>1: inject error to checker<br>For details, refer 63.5.2. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | DCLSEINJ | 0 | R/W | This bit inject error to DCLS compareter.<br>0: Does not inject error to compareter<br>1: inject error to compareter<br>For details, refer 63.5.5. |

### 63.2.4.5 FCPRS Error Detection Status Register (FCPRS_EDC_STA)

FCPRS_EDC_STA indicates the FCPRS error detection status.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CRC ERR | AXIF ERR | RAM ERR | DCLS ERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 3 | CRCERR | 0 | R | CRC error status<br>0: no error<br>1: error occurred<br>For details, refer 63.5.7 |
| 2 | AXIFERR | 0 | R | AXMM I/F EDC error status<br>0: no error<br>1: error occurred<br>For details, refer 63.5.2. |
| 1 | RAMERR | 0 | R | SRAM EDC error status<br>0: no error<br>1: error occurred<br>For details, refer 63.5.2. |
| 0 | DCLSERR | 0 | R | DCLS error status<br>0: no error<br>1: error occurred<br>For details, refer 63.5.5. |

### 63.2.4.6 FCPRS Error Detection Status Clear Register (FCPRS_EDC_STA_CLR)

FCPRS_EDC_STA_CLR clear the FCPRS error detection status.

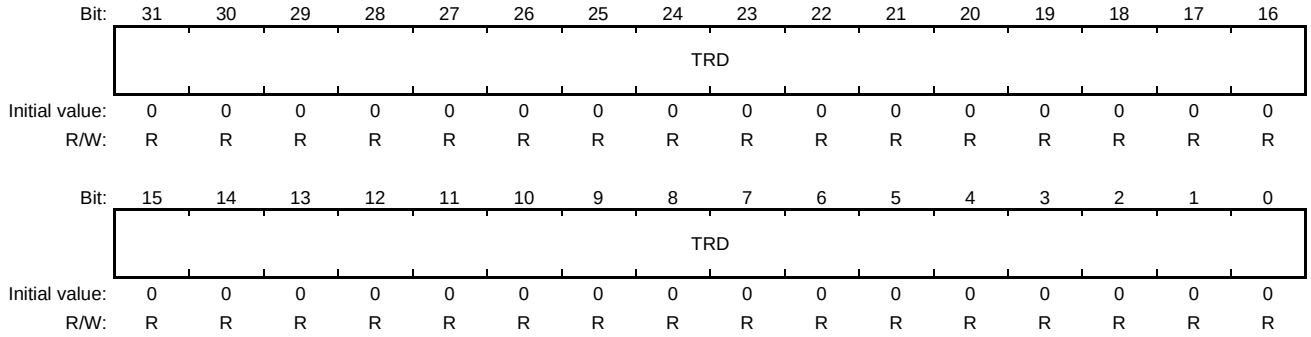
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CRC ECLR | AXIF ECLR | RAM ECLR | DCLS ECLR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 3 | CRCECLR | 0 | R/W | Clear the CRC error status.<br>It is switched back to 0 automatically.<br>0: NOP<br>1: clear the error status<br>For details, refer 63.5.7. |
| 2 | AXIFECLR | 0 | R/W | Clear the AXMM I/F EDC error status.<br>It is switched back to 0 automatically.<br>0: NOP<br>1: clear the error status<br>For details, refer 63.5.2. |
| 1 | RAMECLR | 0 | R/W | Clear the SRAM EDC error status.<br>It is switched back to 0 automatically.<br>0: NOP<br>1: clear the error status<br>For details, refer 63.5.2. |
| 0 | DCLSECLR | 0 | R/W | Clear the DCLS error status.<br>It is switched back to 0 automatically.<br>0: NOP<br>1: clear the error status<br>For details, refer 63.5.5. |

### 63.2.4.7 FCPRS Lossless Decompression Status 0 Register (FCPRS_ATR_STA0)

FCPRS_ATR_STA0 indicates the FCPRS lossless decompression status.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TRD | 0 | R | Total amount of read data from compression area in 64-byte unit.<br>Data size after decompression. |

### 63.2.4.8 FCPRS Lossless Decompression Status 0 Clear Register (FCPRS_ATR_STA0_CLR)

FCPRS_ATR_STA0_CLR clear the FCPRS lossless decompression status.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TRD CLR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 0 | TRDCLR | 0 | R/W | Clear the TRD status.<br>It is switched back to 0 automatically.<br>0: NOP<br>1: clear the status |

### 63.2.4.9 FCPRS Lossless Decompression Status 1 Register (FCPRS_ATR_STA1)

FCPRS_ATR_STA1 indicates the FCPRS lossless decompression status.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDCD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDCD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TDCD | 0 | R | Total amount of compressed data in 64-byte unit.<br>Data size before decompression. |

### 63.2.4.10 FCPRS Lossless Decompression Status 1 Clear Register (FCPRS_ATR_STA1_CLR)

FCPRS_ATR_STA1 clear the FCPRS lossless decompression status.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TDCD CLR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | – | All 0 | R | Reserved<br>These bits are always read as 0. The write value shall always be 0. |
| 0 | TDCDCLR | 0 | R/W | Clear the TDCD status.<br>It is switched back to 0 automatically.<br>0: NOP<br>1: clear the status |

## 63.3 Operation

### 63.3.1 Operation Sequence

The sequence for enabling compression / decompression and activating IP is as follows.

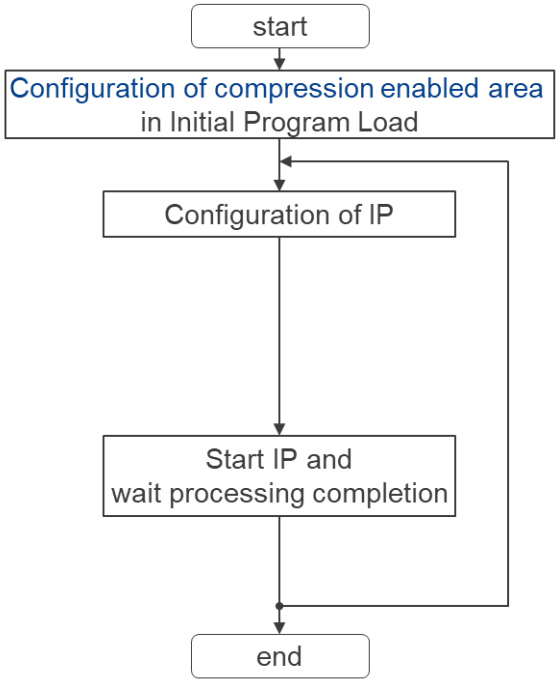



**Figure 63.2 Operation Sequence for Enabling Compression / Decompression**

In case of DOF, in addition to the above need to set the FCPRM for data merge enable. It is possible to change the configuration order between IP and FCPRM. For details of Clear FCPRM buffer and Flush FCPRM buffer, refer to 63.3.6.

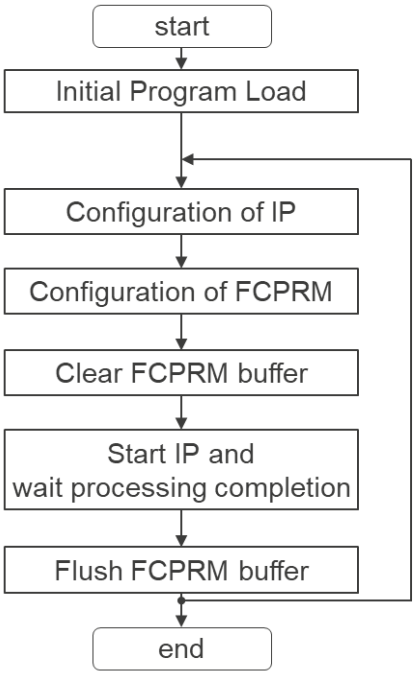



**Figure 63.3 Operation Sequence for Enabling Data merge for DOF**

### 63.3.2 Enabling Compression and Decompression

This section shows how to enable compression or decompression function. In order to enable compression or decompression function of FCPR, set FCPR register in AXMM FCPR_CMP_CTRL to 1. If the compression or decompression function is disabled, no data will be transferred to FCPRC/FCPRS.

### 63.3.3 Setting of Compression Enabled Area

This section shows how to set compression enabled area. Set the start and end address of the area where compressed data is stored in. When IP writes the data to this area, the write data is compressed. When IP reads the data from this area, the compressed data is decompressed. The maximum capacity of the compress enabled area is 1G-byte. The setting registers are following 3 types.

FCPR_CMP_SPACE:

These registers indicate higher 8-bit of 40-bit address setting of FCPR_CMP_STADR/CMP_ EDADR. If IP doesn't support 40-bit address space, it shall be set to 0.

FCPR_CMP_STADR:

Start address of compression enabled area in 1G-byte aligned, Lower 30 bits shall be 0.

FCPR_CMP_EDADR:

End address of compression enabled area. Lower 8-bit shall be 255.

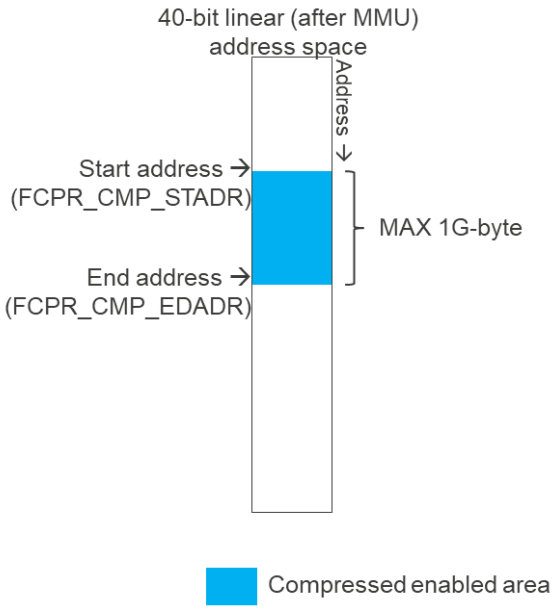



**Figure 63.4 Setting of Compression Enabled Area**

### 63.3.4 Enabling Data Merge in FCPRM

This section shows how to enable data merge function. In order to enable data merge function of FCPRM, set FCPRM_MRG_CTRL to 1. If the data merge function is disabled, FCPRM acts simply as a bus bridge.

### 63.3.5 Setting of Data Merge Enabled Area in FCPRM

This section shows how to set data merge enabled area. Set the start and end address of the area where merged data is stored in. When IP writes the data to this area, the write data is merged. FCPRM can combine contiguous transaction to 256-byte aligned transaction in order to apply compression.

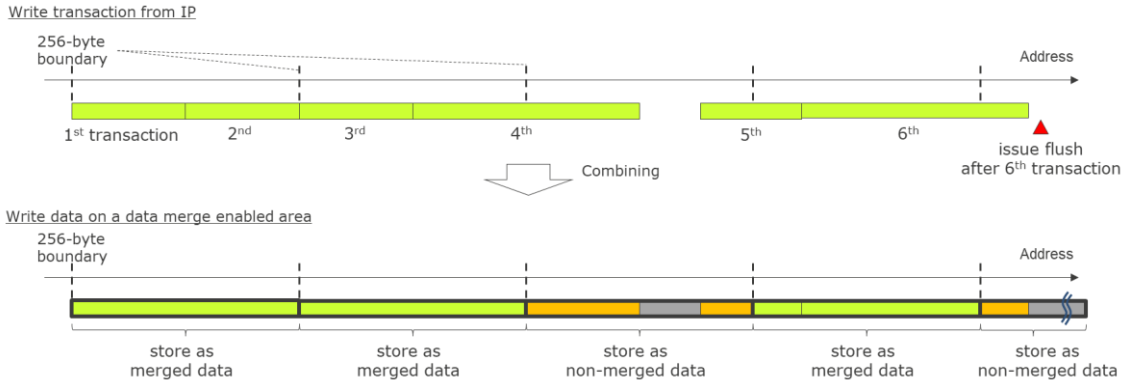



**Figure 63.5 Converting to 256-byte aligned write transaction**

The maximum number of the area is 4 per FCPRM. Set the area by following 2 registers.

FCPRM_MRG_STADR*n* (*n* = 0 to 3):

Start address of merge enabled area. Lower 8-bit shall be 0.

FCPRM_MRG_EDADR*n* (*n* = 0 to 3):

End address of merge enabled area. Lower 8-bit shall be 255.

### 63.3.6 Data Consistency in case of using FCPRM

To keep data consistency, there are two operations before/after IP operation. One is clear operation. The other is flush operation.

In write, both clear and flush operation is necessary. Clear operation before IP operation is required to initialize buffer in FCPRM. FCPRM merges some write requests from IP in compression so that FCPRM may hold these requests after IP operation. Flush operation outputs all requests in FCPRM to external memory.

#### 63.3.6.1 Clear and Flush Write Buffer of FCPRM

All IPs connecting with FCPRM have a function to clear and flush the internal buffer of FCPRM.

**Table 63.12 Clear and Flush Operation**

| IP | Clear/Flush Operation |
|---|---|
| DOF | This IPs have instructions to issue clear and flush. Usually these instructions are listed in the DMA descriptor as below. For detail information, refer to section U25 and IP licensee guide.<br>- Clear buffer<br>- Data transfers for each processing<br>- Flush buffer<br>- Wait completion of flush operation<br>- Assert an end interrupt |

### 63.3.7 Address Management of Compressed Data for using FCPRM

FCPRM of all IPs use an address before Memory Management Unit (MMU) processing (virtual address). So, set virtual address to FCPRM register.

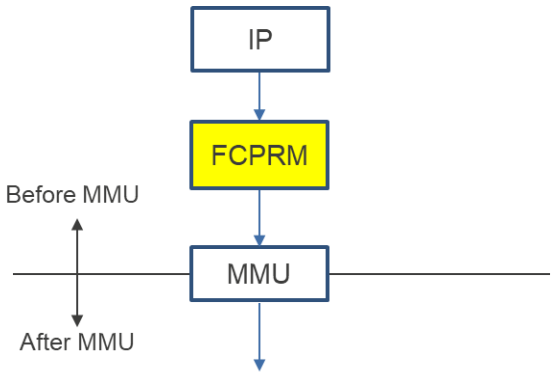



**Figure 63.6 FCPRM use an address before MMU processing**

### 63.3.8 Method of Calculating Compression Ratio

The method of calculating compression ratio (compressed/original) is as follows.

Write:

Compression ratio = (FCPRC_CMP_STA1.TCD / FCPRC_CMP_STA0.TWD) * 100

Read:

Compression ratio = (FCPRS_ATR_STA1.TDCD / FCPRS_ATR_STA0.TRD) * 100

### 63.3.9 Reset Operation

FCPRC and FCPRM has Forced Reset operation controlled by software, which is used to terminate forcibly the whole FCPR immediately upon reset.

The Forced Reset terminates all FCPRC/FCPRM operation immediately even if the bus transaction is being executed. Use this reset in emergency such as the bus sub-system or IP hang.

Set H’0000 0010 to FCPRC_RST/FCPRM_RST register (FCPRSTF bit is set to H’1)

## 63.4 Usage Notes

### 63.4.1 Notes of Compression Enabled Area

- The whole compression enabled area must be located in DRAM space.
- The compression enabled area shall be contiguous region.
- Total capacity of the compress enabled area is up to 1G-byte.
- Virtual addresses pointing to the compression enabled area must be translated to physical addresses assigned the compression enabled area by MMUs.
- If Master IP writes outside the compression enabled area, the write data is not compress.
- If Master IP (inclusive FCPRM) writes smaller bursts than 256 bytes or masks data when writing to the compression enabled area, the data will be written uncompressed. The rest of the 256 byte stays untouched and therefore might be invalid.
- If Master IP reads smaller bursts than 256 bytes when reading to the compression enabled area, the effect of bandwidth compression may be reduced. For the convenience of compressing in 256-byte units, if a read of less than 256 bytes is requested from the Master IP, compressed data with a data length exceeding that may be read from the memory.
- The compression enabled area shall be set to either linear or split mapping mode in AXI-bus, and shall not be mixed (see section 18.3).

### 63.4.2 Exclusive access

In the compression enabled area, exclusive access defined by AXI protocol is prohibited.

### 63.4.3 Note of software reset for the master IP

In case of FCPRM, after resetting the master IP using its internal register, set FCPRM_BFLS register to H’0000 03FF.

### 63.4.4 Limitations on Software Reset, Module Standby and power-shutoff

Before executing following operations for the FCPR, confirm that there is no access to the compression enabled area. If there is access to the compression enabled area, wait completion of access.

- Execute Software Reset by CPG
- Enter Module Standby (stop clock supply) by CPG
- Enter power-shutoff status (stop power supply) by SYSC

## 63.5 Safety Mechanism

### 63.5.1 EDC for FCPRS

#### 63.5.1.1 Overview

EDC protection is applied to internal RAMs in FCPRS.

#### 63.5.1.2 Hardware Description

Figure 63.7 shows the block diagram of EDC protection for FCPRS.

When the EDC error is detected on internal RAMs, the RAMERR bit in the FCPRS_EDC_STA is set to 1.

When the RAMERR bit in the FCPRS_EDC_STA is set to 1, error notification is issued to ECM.

The RAMERR bit is cleared to 0 by FCPRS_EDC_STA_CLR register.

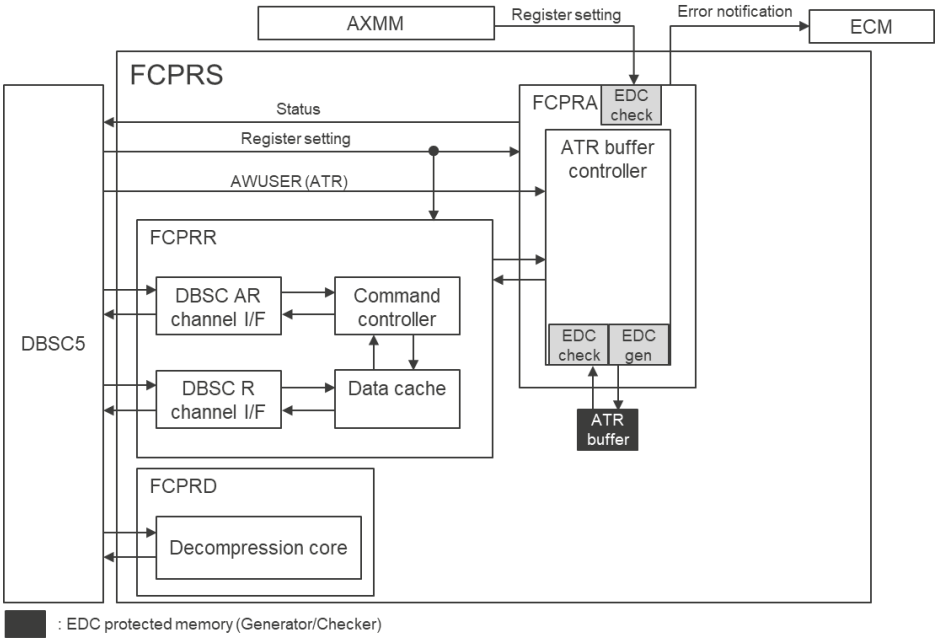



**Figure 63.7 EDC protection for FCPRS**

### 63.5.2 EDC for FCPRM

#### 63.5.2.1 Overview

EDC protection is applied to internal RAMs in FCPRM.

#### 63.5.2.2 Hardware Description

Figure 63.8 shows the block diagram of EDC protection for FCPRM.

When the EDC error is detected on internal RAMs, the RAMERR bit in the FCPRM_EDC_STS is set to 1.

When the RAMERR bit in the FCPRM_EDC_STS is set to 1, error notification is issued to ECM.

The RAMERR bit is cleared to 0 by writing 0.

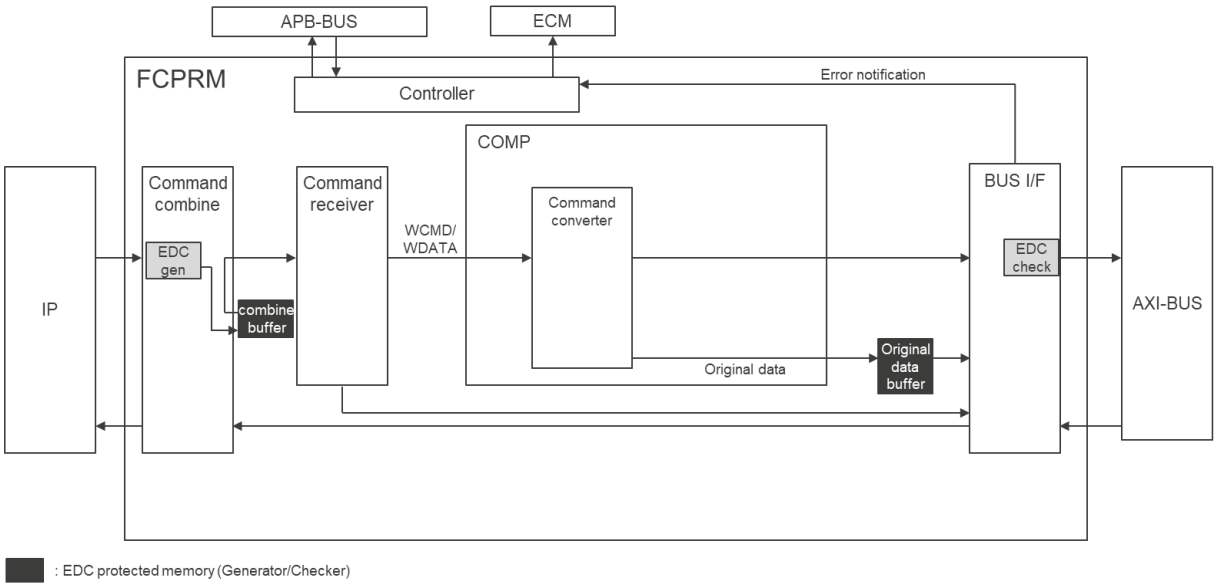



**Figure 63.8 EDC protection for FCPRM**

### 63.5.3 Hardware Redundancy for FCPRS

#### 63.5.3.1 Overview

Hardware redundancy is applied to the control logic in FCPRS in order to detect fault of functional logic.

#### 63.5.3.2 Hardware Description

Figure 63.9 shows the block diagram of FCPRS protection for functional safety.
When an error is detected, the DCLSERR bit in the FCPRS_EDC_STA is set to 1.

When the DCLSERR bit in the FCPRS_EDC_STA is set to 1, error notification is issued to ECM.

The DCLSERR bit are cleared to 0 by FCPRS_EDC_STA_CLR register.

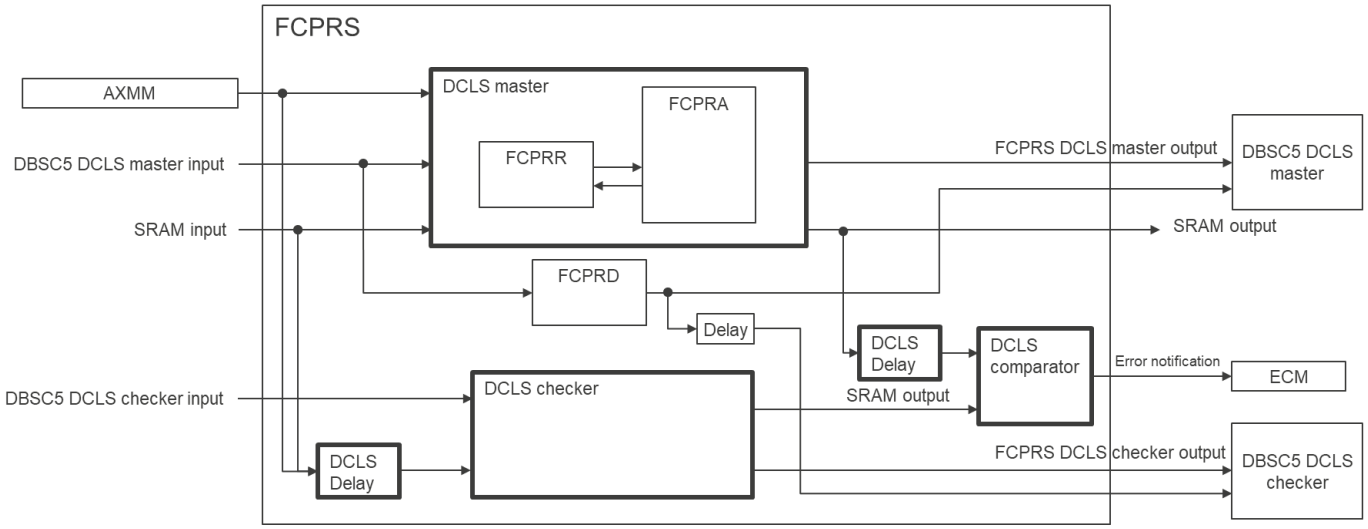



**Figure 63.9 Block diagram of hardware redundancy for FCPRS**

### 63.5.4 CRC for FCPRC

#### 63.5.4.1 Overview

CRC protection is applied to FCPRC and compressed data.

#### 63.5.4.2 Hardware Description

Figure 63.10 shows the block diagram of CRC protection for FCPRC.

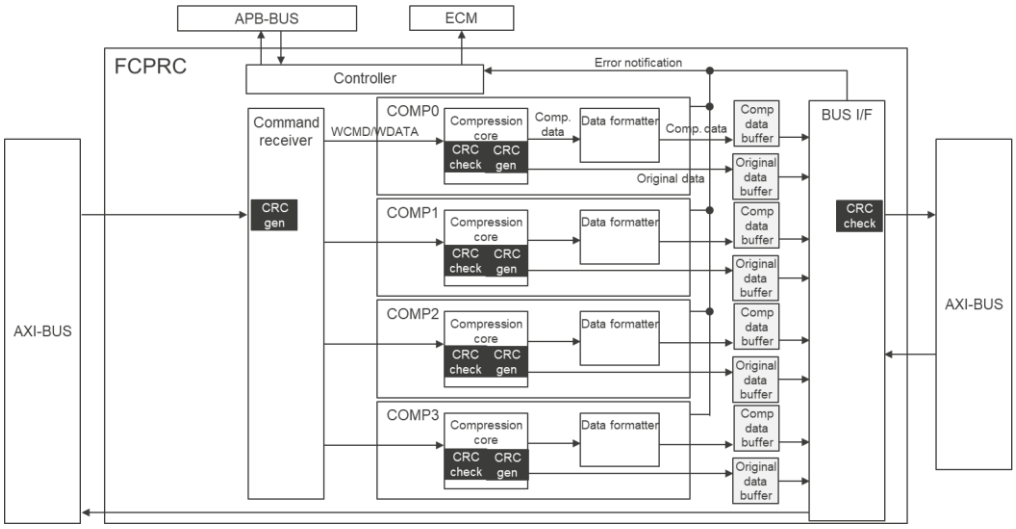



**Figure 63.10 CRC protection for FCPRC**

### 63.5.5 CRC for FCPRD in FCPRS

#### 63.5.5.1 Overview

CRC protection is applied to FCPRD in FCPRS and compressed data.

#### 63.5.5.2 Hardware Description

Figure 63.11 shows the block diagram of CRC protection for FCPRD in FCPRS.

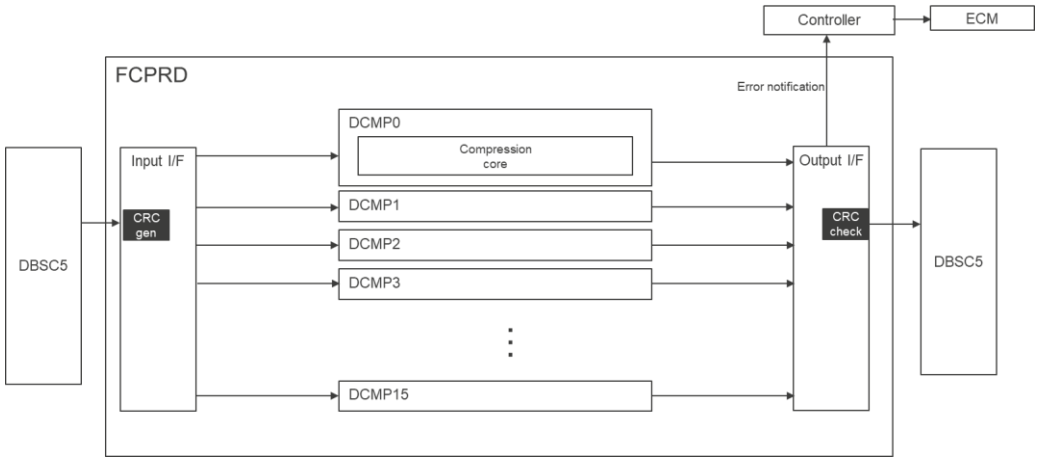



**Figure 63.11 CRC protection for FCPRD in FCPRS**

# 64. Display Unit (DU)

## 64.1 Overview

Display Unit (hereinafter it's called DU) is a module to input an image data from the VSPD and to output the data to the DSI/CSI-2-TX-IF in accordance with optionally settable display timing.

When using display unit, set the desired value in Video Signal Processor for the DU (VSPD) registers.

### 64.1.1 Features

**(1) Plane**

The display surfaces configured to display an image on the screen is called plane in this section. The following parameters for plane can be set through the settings of an internal register.

- Display size
- Display position
- Display data format (32-bit/pixel or YC)
- Plane superimposing

**(2) Synchronization Method**

Internal register settings can be used to select any of three synchronization modes for the display output timing.
Synchronization Method support only master mode (internal sync mode).

**(3) CRT Scan Mode (CRT Scan Method)**

Internal register settings can be used to select from among three scan modes.
CRT Scan Mode support only non-interlaced mode.

**(4) Register Access Control**

The module has internal control registers; these are writable/readable by the APB protocol over the APB. The unit of access is fixed to 32 bits.

**(5) VSPD Connection**

The VSPD output provides RGB or YCbCr444 data format for the DU. The DU is not connected with external memory. For the VSPD connection, refer to Table 64.1.

**(6) DOC Connection**

Connection with the DOC allows the monitoring of images from the DU and thus allows checking of whether the images are being displayed in accord with the expected values. This also allows checking of the rising edge of the VSYNC signal and the completion of processing by the DU. For the DOC connection, refer to Table 64.1

**(7) DSI/CSI-2-TX-IF Connection**

Connection with the DSI/CSI-2-TX-IF allows the output of image data via the DSI/CSI-2-TX-IF. Support data format is RGB or YCbCr422. For the DSI/CSI-2-TX-IF connection, refer to Figure 64.1.

**Table 64.1** Configuration of channel, plane and connection

| Channel | Abbr. | Plane | Connection | | |
|---|---|---|---|---|---|
| Channel0 | DU | Plane | VSPD | DSI/CSI-2-TX-IF | DOC |

### 64.1.2 Block Diagram

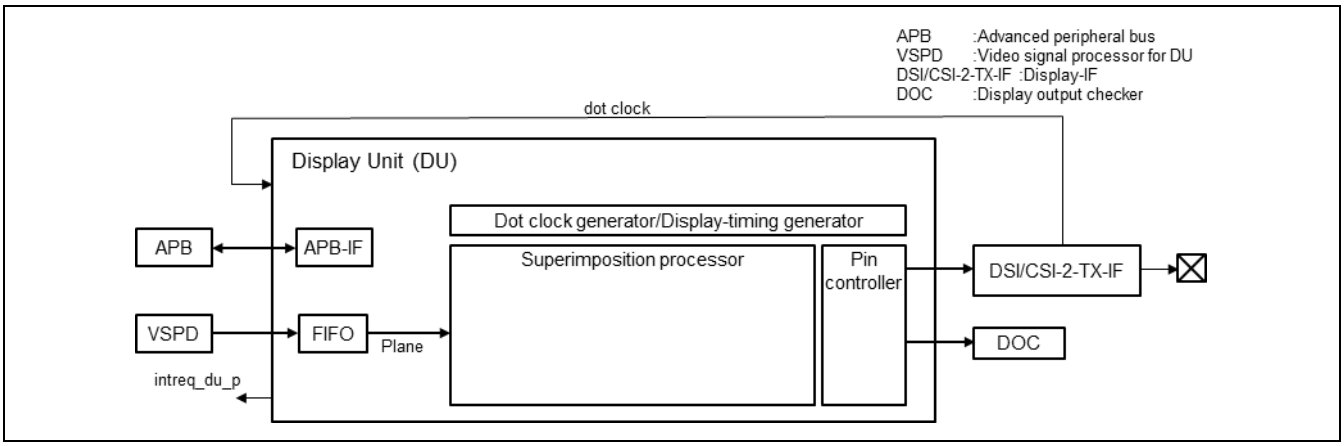



**Figure 64.1 Block Diagram of the DU**

### 64.1.3 Register Configuration

For the DU, there are two register update methods, the external update and the internal update. The internal update has two reflection timings. Refer to Table 64.2 for the update methods and the reflecting timings in the DU registers.

**Table 64.2 Register update methods**

| Update Methods | Reflection timings | Definitions of the Internal Updates in the Bit Description Table | Remarks |
|---|---|---|---|
| External Update | After the end of CPU access | Not available | - |
| Internal Update | When the DRES bit in DSYSR is 1. | DRES | - |
| | When the DRES bit in DSYSR is 1 and at the beginning of each frame, which is the VSYNC falling edge if the VSL bit in DSMR is 0 and the VSYNC rising edge if the VSL bit in DSMR is 1. | Available | Disabled when the IUPD bit in DSYSR is 1. |

#### (1) External Update

An "external update" is an update which reflects the address-mapped register settings made by the CPU after the end of the CPU access. Registers related to display control (for example, DSYSR) and the settings of which are updated through external updates can be overwritten without display flicker by using bit 11 in DSSR indicating the start position of the vertical blanking interval.

#### (2) Internal Update

An "internal update" is an update that reflects the address-mapped register settings with the internal update timing of the DU. Hence in the case of a register with an internal update function, even when the CPU overwrites address-mapped registers related to display operation without being aware of the display timing, display flicker can be prevented.

There are two types of internal update timings. One reflects the register value when the DRES bit is set to 1. The other reflects the register value in the following cases: (a) When the DRES bit is set to 1, and (b) when the VSYNC falling edge occurs if the VSL bit in DSMR is 0 and when the VSYNC rising edge occurs if the VSL bit in DSMR is 1. The type of internal update timings is defined by registers and that appears in the bit description table.

Bits which are internally updated in response to setting of the DRES bit in DSYSR are listed in the column headed "Bit with Internal Update Function" in tables on the following pages.

Updates are performed at the falling edge of VSYNC output when the sync method of DSYSR is master mode (bit 7 = 0, bit 6 = 0).

The address-mapped registers with an internal update function are shown in R-CarV4M_UM_064_DU_r0p81.xlsx. The initial settings for these registers should be made during the interval in which the DRES bit in DSYSR is 1.

**(3) Register Configurations**

Refer to R-CarV4M_UM_064_DU_r0p81.xlsx

### 64.1.4 Connected Module

**Table 64.3 Connected Module**

| Module name | Connected module name | Function of connected module |
|---|---|---|
| | APB | Control to register access |
| | VSPD | Output image data |
| | DOC | Input display data and control signals, after that check display data |
| | DSI/CSI-2-TX-IF | Display-IF |
| | CPG | Output system clocks and dot clock. |
| | Module Standby | Control to stop clocks |
| | Software Reset | Execute software reset |

## 64.2 Register Description

Refer to R-CarV4M_UM_064_DU_r0p81.xlsx

## 64.3 Operation

### 64.3.1 Configuration of Output Screen

The DU executes window displays with one window layer. This window is called a "plane". For plane, display can be turned on and off, the display data format (32-bit/pixel, YC), and other settings can be changed independently.

**Table 64.4 Display Functions of Plane**

| | Display On/Off | Display Data Format | | interleave | Size |
|---|---|---|---|---|---|
| | | YC | 32-bit/pixel | | |
| Plane | ✓ | ✓ | ✓ | No | X, Y (as desired) |
| Back-ground color* | — | — | — | — | — |

Note: The data format for background color is RGB666.

### 64.3.2 Display On/Off

All plane display can be turned on and off using the DEN bit in DSYSR. When the DEN bit is 0, the display data set in DOOR is displayed.

When the value of the DPRS bit in DORCR is 0, DPPR is used to turn the display of plane on and off. When the value of the DPRS bit in DORCR is 1, on the other hand, DSPR is used to turn the display of plane on and off. Under the following display conditions, display data set in BPOR is displayed.

1. When the display of plane is turned off
2. In an area with no plane for display, due to the display size and display position
3. When the pixels in a plane for display are all a transparent color

**Table 64.5 Turning On and Off the Display of Plane(when the DPRS bit in DORCR is 0)**

| Display Plane | Display Plane Priority Register (DPPR) |
|---|---|
| Plane | Plane is selected in one among priority positions 1, and the corresponding enable bit is set to 1 |

**Table 64.6 Turning On and Off the Display of Plane (when the DPRS bit in DORCR is 1)**

| Display Plane | Display Superimpose Priority Register (DSPR) |
|---|---|
| Plane | Plane is selected in one among priority positions 1 (setting to select the plane: B'0001) |

Notes: Even if display on is set using DPPR and DSPR under the following conditions, the setting is handled as display off, and the corresponding plane is not displayed.

- Plane for which the value set in PDPXR is greater than the screen size (horizontal display end (HDE) - horizontal display start (HDS))
- Plane for which the value set in PDPYR is greater than the screen size (vertical display end (VDE) - vertical display start (VDS))
- Plane for which the value set in PDSXR is 0
- Plane for which the value set in PDSYR is 0

### 64.3.3 Plane Parameter

For each plane, a display area start position, memory width, display start position, and display size are set using registers.

The followings are the schematic diagram of start positions and sizes related to plane and the registers used for setting start positions and sizes.

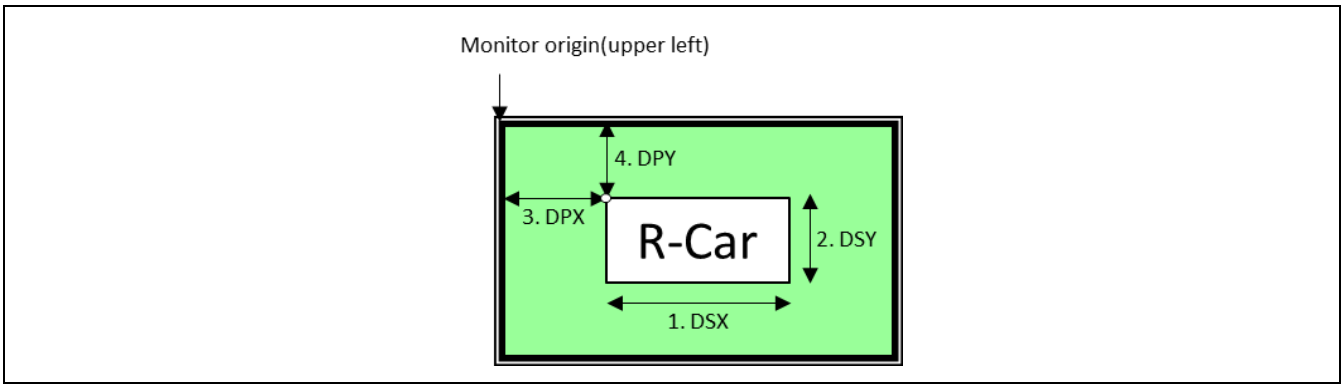



**Figure 64.2 Parameters**

**Table 64.7 Monitor Parameter Setting Registers**

| No. | Name Used in the Figure | Register | Description |
|---|---|---|---|
| 1 | DSX<br>(Display size X) | PDSXR | The X-direction display size of plane is set in dot-clock units. |
| 2 | DSY<br>(Display size Y) | PDSYR | The Y-direction display size of plane is set in raster line units. |
| 3 | DPX<br>(Display position X) | PDPXR | The X-direction distance to the display position is set in dot-clock units, taking the upper-left corner of the monitor as the origin. |
| 4 | DPY<br>(Display position Y) | PDPYR | The Y-direction distance to the display position is set in raster line units, taking the upper-left corner of the monitor as the origin. |

### 64.3.4 Data Format for Input

The following format is used for color data used in display. A data configuration is shown in which data is allocated to the unified memory in little endian.

**(1) 32-bit/pixel: RGB888**

Levels for R, G, and B are represented by 8 bits each (RGB888). "—" Indicates any desired value.

A 32-bit/pixel (RGB888) display should be specified with the PEDF bits in PDDC4R.

**32-bit/pixel data (RGB888 data) format**

| D31 to D0 | 31 — 24 | 23 — 16 | 15 — 8 | 7 — 0 |
|---|---|---|---|---|
| | — | R | G | B |

**(2) 32-bit/pixel: RGB666**

Levels for R, G, and B are represented by 6 bits each (RGB666). "—" Indicates any desired value.

A 32-bit/pixel (RGB666) display should be specified with the PEDF bits in PDDC4R.

**32-bit/pixel data (RGB666 data) format**

| D31 to D0 | 31 — 24 | 23 — 18 | 17 | 16 | 15 — 10 | 9 | 8 | 7 — 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| | — | R | — | — | G | — | — | B | — | — |

**(3) YC: YCbCr444**

Image data has the format YC (YCbCr) = 4:4:4. "—" Indicates any desired value.

A YC(YCbCr444) display should be specified with the PEDF bits in PDDC4R and value other than B’11 is specified in the PDDF bit in PMR .

**YC(YCbCr444) format**

| D31 to D0 | 31 — 24 | 23 — 16 | 15 — 8 | 7 — 0 |
|---|---|---|---|---|
| | — | Y0 | Cb0 | Cr0 |
| | — | Y1 | Cb1 | Cr1 |

### 64.3.5 Data Formats for Output

In the case of digital RGB output from the DU is performed after the display data format has been expanded into the RGB888 format, and then the data are output. The supplementary formats in the case of expansion to RGB888 and output data formats are as indicated in the following table. For YCbCr422 output formats, refer to section 64.3.9 YC Format Display.

**Table 64.8 Output Data Format**

| | | Red | | | | | | | | Green | | | | | | | | Blue | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Expanded data | 32-bit/pixel (RGB888) | R (8 bits) | | | | | | | | G (8 bits) | | | | | | | | B (8 bits) | | | | | | | |
| | 32-bit/pixel (RGB666) | R (6 bits) | | | | | | 0 | 0 | G (6 bits) | | | | | | 0 | 0 | B (6 bits) | | | | | | 0 | 0 |
| After superimposing | | R (8 bits) | | | | | | | | G (8 bits) | | | | | | | | B (8 bits) | | | | | | | |
| Output Data Format | | R (8 bits) | | | | | | | | G (8 bits) | | | | | | | | B (8 bits) | | | | | | | |
| | | Y | | | | | | | | Cb | | | | | | | | Cr | | | | | | | |
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Expanded data | | Y (8 bits) | | | | | | | | Cb (8 bits) | | | | | | | | Cr (8 bits) | | | | | | | |
| After superimposing | | Y (8 bits) | | | | | | | | Cb (8 bits) | | | | | | | | Cr (8 bits) | | | | | | | |
| Output Data Format (YCbCr422) | | Y (8 bits) | | | | | | | | Cb (8 bits) | | | | | | | | Cr (8 bits) | | | | | | | |

### 64.3.6 Superimposing of Plane

Only the specified plane display is supported.

**Table 64.9 Superimposing**

| PSPIM | Superimposing |
|---|---|
| B’000 | Setting prohibited. (Default) |
| B’001 | Setting prohibited. |
| B’010 | Setting prohibited. |
| B’011 | Setting prohibited |
| B’100 | The specified plane is displayed. |
| B’101 | Setting prohibited |
| B’110 | Setting prohibited |
| B’111 | Setting prohibited |

### 64.3.7 Sync Mode

Sync Mode support only master mode (internal sync mode). Selection of master mode is performed using bits 7 and 6 in DSYSR. In master mode (internal sync mode), the position of the falling edge of the vertical sync signal (VSYNC) set by VSPR is detected.The results are then reflected in the bit 11 in DSSR.

**Master Mode (Internal Sync Mode):** By setting the period and pulse width of the horizontal and vertical sync signals (HSYNC, VSYNC) in the display timing generation registers, the corresponding waveforms are output. Also, display data is output in sync with these signals.

### 64.3.8 YC Format Display

In addition to RGB format display, YC format can be displayed. YCbCr444 data from the VSPD convert into YCbCr422 format and display as YC format.

RGB format display and YC format display is selected by setting bits 9 and 8 in DEFR. Bits 9 and 8 setting of B'10 selects 16-bit YC format display (non-multiplexed). A setting of B'00 selects RGB format.

Input data format from the VSPD is selected by DEF10R.

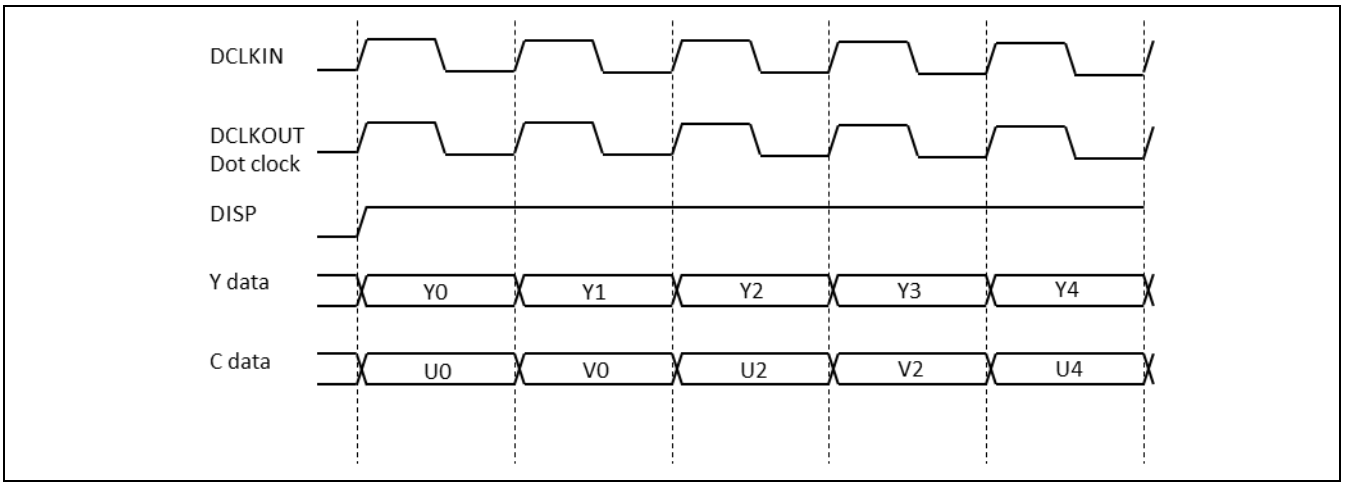



**Figure 64.3 YC Non-Multiplexed**

### 64.3.9 Display Timing Generation

In the DU, display timing is generated for the horizontal direction and vertical direction of the display screen. Display timing is set by using display timing generation registers. (Refer to R-CarV4M_UM_064_DU_r0p81.xlsx)

Figure 64.4 shows the display timing in non-interlaced mode. Here, the display screen is defined in terms of the variables of Table 64.10.

Only the Master Mode (Internal Sync Mode) is supported. Therefore, it has no external sync (EXVSYNC/EXHSYNC).

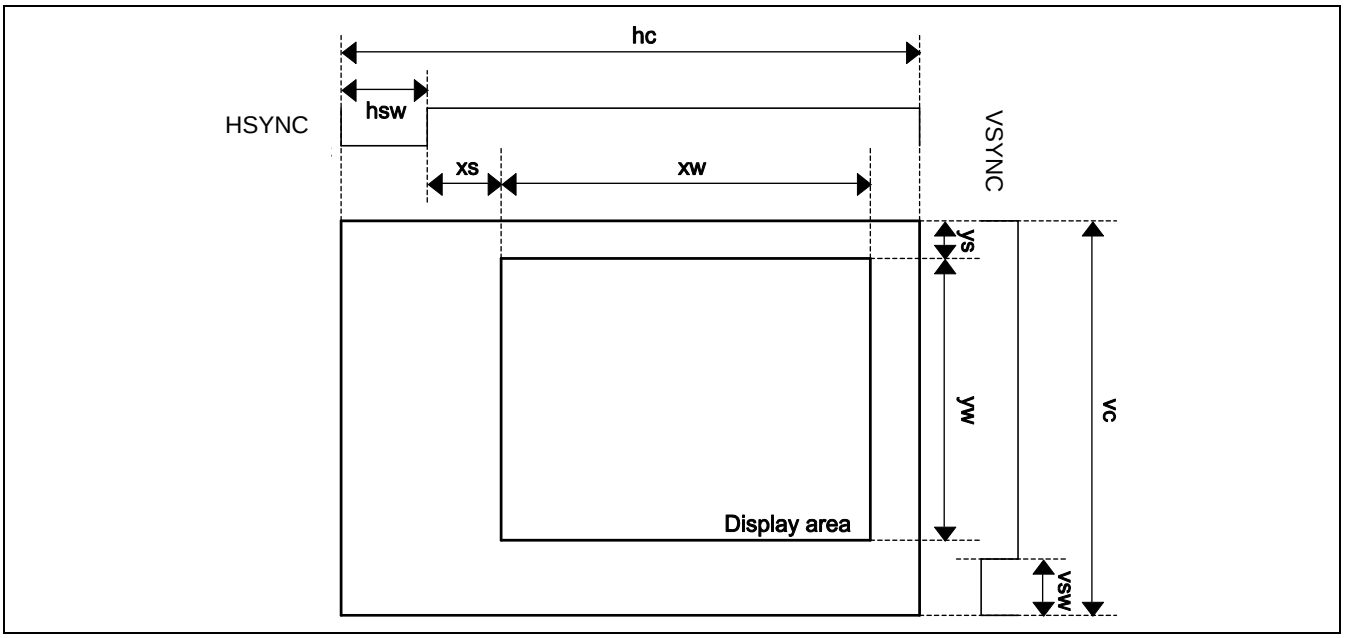



**Figure 64.4 Display Timing Generation for Horizontal Direction and Vertical Direction of Display Screen**

**Table 64.10 Variables Defined in Display Screen**

| Variables | Contents | Units |
|---|---|---|
| hc*1 | Horizontal scan period | Dot clock |
| hsw | Horizontal sync pulse width | Dot clock |
| xs | From rise of HSYNC to display start position in the horizontal direction of the display screen | Dot clock |
| xw | Display width per 1 raster of display screen | Dot clock |
| vc*2 | Vertical scan period | Raster line |
| vsw | Vertical sync pulse width | Raster line |
| ys | From rise of VSYNC to display start position in the vertical direction of the display screen | Raster line |
| yw | Vertical display period of display screen | Raster line |

Notes: 1. Should be set such that hsw + xs + xw < hc
2. Should be set such that vsw + ys + yw < vc

The display timing generation register settings are different depending on the scan method and synchronization method. Hence the value of the display timing generation registers should be set after performing calculations like those indicated in Table 64.11.

**Table 64.11 Correspondence Table of Settings of Display Timing Generation Registers**

| Register Name | Bit Name | Master Mode |
|---|---|---|
| Horizontal display start register (HDSR) | HDS*4 | hsw + xs – 19 |
| Horizontal display end register (HDER) | HDE | hsw + xs – 19 + xw |
| Vertical display start register (VDSR) | VDS | ys – 2 |
| Vertical display end register (VDER) | VDE | ys – 2 + yw |
| Horizontal synch width register (HSWR) | HSW | hsw – 1 |
| Horizontal cycle register (HCR) | HC*3 | hc – 1 |
| Vertical synch point register (VSPR) | VSP | vc – vsw – 1 |
| Vertical cycle register (VCR) | VC | vc – 1 |

Notes: 1. In all scan modes, VDS, VDE, VSP, VC settings are in single-field units.
2. Reserved.
3. HC should be set so as to satisfy HC > HDE.
4. HDS should be set to 1 or greater.

### 64.3.10 Scan Method

The scan method is non-interlaced mode only. The mode is selected using bit 5 and bit 4 in DSYSR.

- Non-interlaced mode
  In this scan method, one frame consists of a single field.
- Example of vertical scan period
  Non-interlaced mode: 1/60 second/field, 1/30 second/field
- Display in non-interlaced method
  In this method, all lines are displayed at once without providing intervals between input video signals.
  This input method is for monitors capable of high-resolution display.

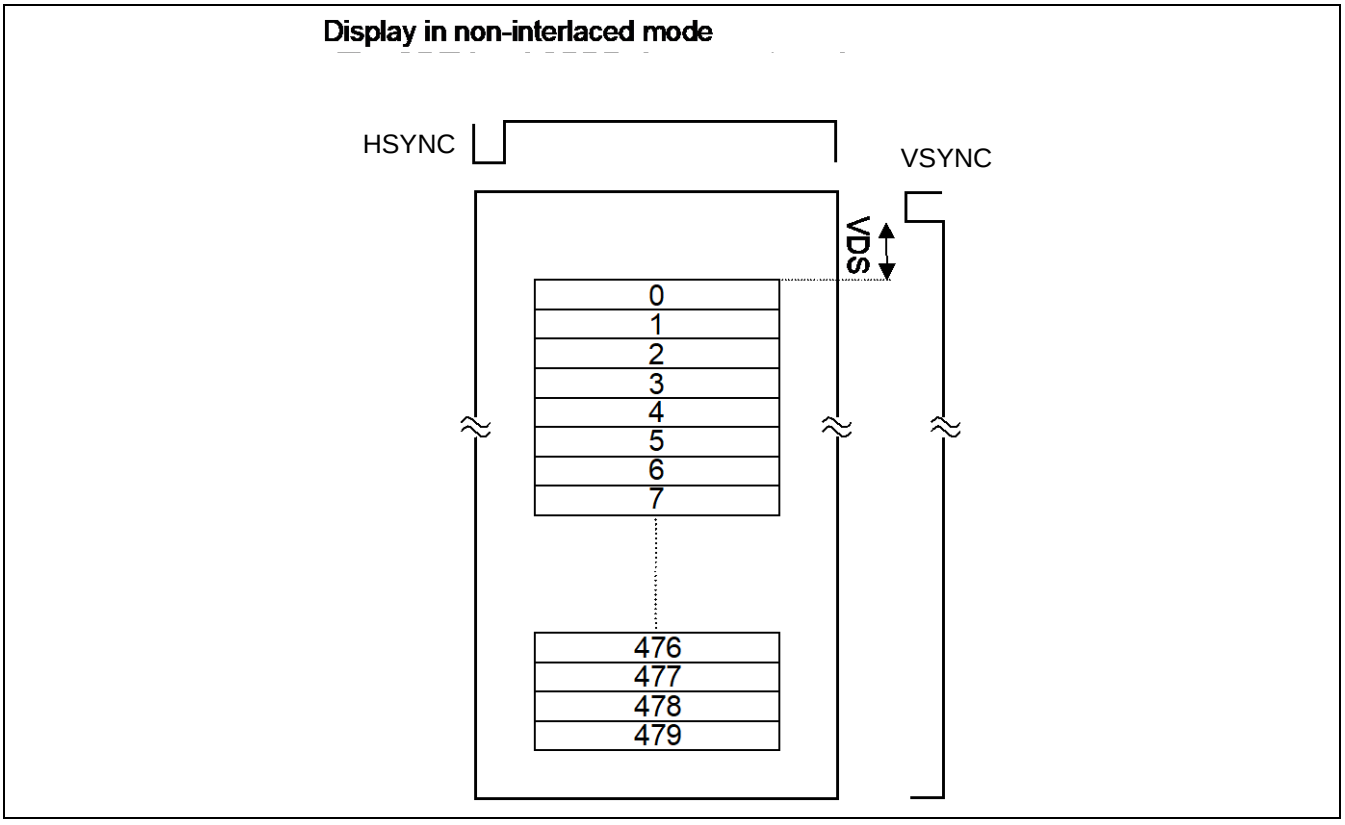



**Figure 64.5 Display in Non-Interlaced Method**

## 64.4 Usage Notes

### 64.4.1 Module Standby Mode

Module Standby mode, in which supply of the clock signal to the DU is stopped, is supported.

Even when the DU enters the standby mode, the register values are retained. Do not access the DU during periods in standby mode.

### 64.4.2 Transition to Module Standby Mode

1. Set both the DEN and DRES bits in DSYSR to 0 to stop access to the VSPD.
2. Test bit 11 in DSSR to confirm the next VBK flag. The setting made in step 1 becomes effective with the timing of VBK and access to the VSPD is stopped. The value of the display data becomes the value set in DOOR
   When the DU is running, test bit 11 of DSSR.
3. Stop the clock.

### 64.4.3 Release from Module Standby Mode and Restarting Display

1. Start the clock.
2. Make settings to turn the display on by setting the DEN and DRES bits in DSYSR to 1 and 0 respectively.

### 64.4.4 Note on Register Settings

At the time of a reset, some registers of the DU have fixed initial values but others do not. The initial values of the latter when the power for the LSI is turned on can be either 0 or 1. From the initial state after power is switched on, be sure to set all of the registers in the DU to the desired values before starting synchronization of the display*[1].

If turning on the display*[2] is attempted while values in registers are unknown, operation may be incorrect due to access by the DU to areas other than those intended (the DU is capable of access to any area).

Notes: 1. Setting (a) or (b) starts synchronization of the display.
   (a) The DRES and DEN bits in DSYSR are B'00.
   (b) The DRES and DEN bits in DSYSR are B'01.
2. Setting (a) or (b) turns the display on.
   (a) When the DPRS bit in DORCR is 0:
   With setting (b) in note 1, setting any of the DPE1 bit in DPPR to 1.
   (b) When the DPRS bit in DORCR is 1:
   With setting (b) in note 1, setting the S0S1 bits in DSPR to B'0001.

### 64.4.5 Transition to Display operation stop by the DRES bit in DSYSR

**(1) Display screen is not disturbed**

1. Execute the procedure of 1. and 2. of Chapter 64.4.2.
2. Set the DRES and DEN bits in DSYSR to B’10.

**(2) Display screen is disturbed**

1. Set the DRES and DEN bits in DSYSR to B’10.
2. Execute software reset to VSPD module.

## 64.5 Safety Mechanism

There is no safety mechanism in the DU.

# 65. Display Output Checker (DOC)

## 65.1 Overview

This module is treated as safety mechanism in case of the products supporting functional safety. However, this module can be also treated as non-safety-related function.

### 65.1.1 Features

DOC supports one channel of DU.

**Table 65.1 DOC's Supported Channels of DU**

| | Connected channel of DU |
|---|---|
| DOC | DU (Supports RGB and YCbCr422) |

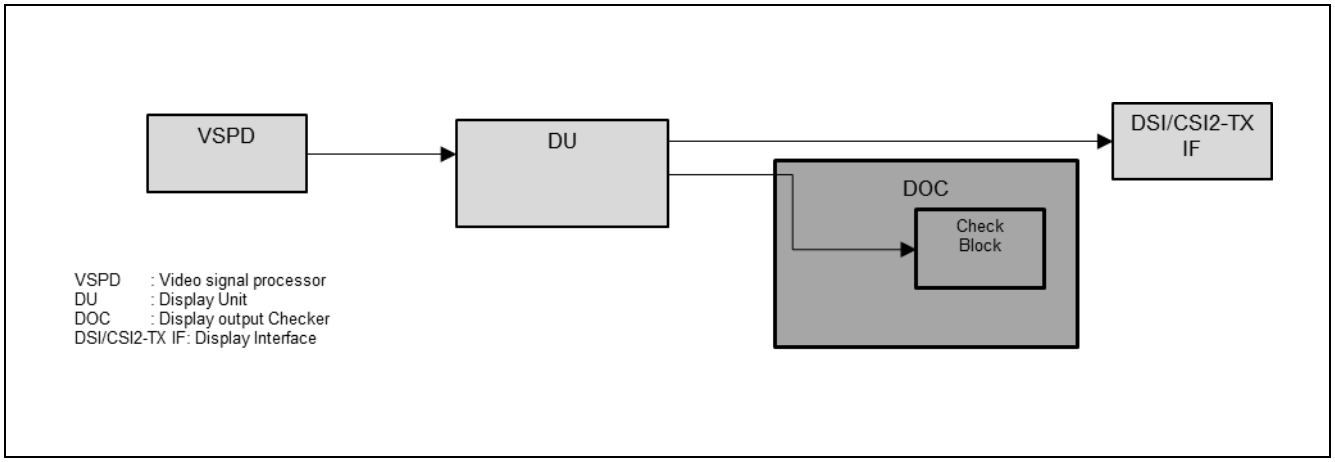



**Figure 65.1 Connected modules with DOC**

Main function is as the following:

1. **Video Output Check**

   The DOC allows to check whether the display content (warning, light, and so on) is correctly output by the Display Unit (DU). If the content is not correct, interrupts are input to the Interrupt Controller (INTC). Refer to Section 65.3.2.1.

   - Up to 4096x2048 pixel in screen size
   - Total of up to 16 monitor areas
   - Up to 65,536 pixels in total over all monitored areas
   - Four programmable reference color ranges for each monitored area
   - Programmable discriminator threshold for fail/pass decision
   - Interrupt generation upon symbol detection mismatch

2. **Active Monitor**

   The function allows to monitor the failure of VOC (Video Output Checker) implemented in DOC. If the VSYNC interspace is different from the register value set by user, interrupts are input to the Interrupt Controller (INTC). Additionally, the interrupts are also asserted in case that the check completion of VOC is not detected as expected (register values). Refer to Section 65.3.2.2.

### 65.1.2 Block Diagram

Display output data from DU is checked by "Video Output Checker (VOC)" which is the main component of the DOC. The format of the display output data is either RGB888 (one of RGB color format) or YCbCr422. VOC can support only

RGB format. Hence, the format of YCbCr422 needs to be converted to RGB format before checking the data by VOC if display output data is YUV format. Refer to Section 65.2.14 to 65.2.15 about the format conversion. Additionally, the DOC has internal control registers which are writable / readable by the APB protocol via the APB Bus. The unit of access is fixed to 32 bits. Refer to Section 65.1.4 about the detail register setting.

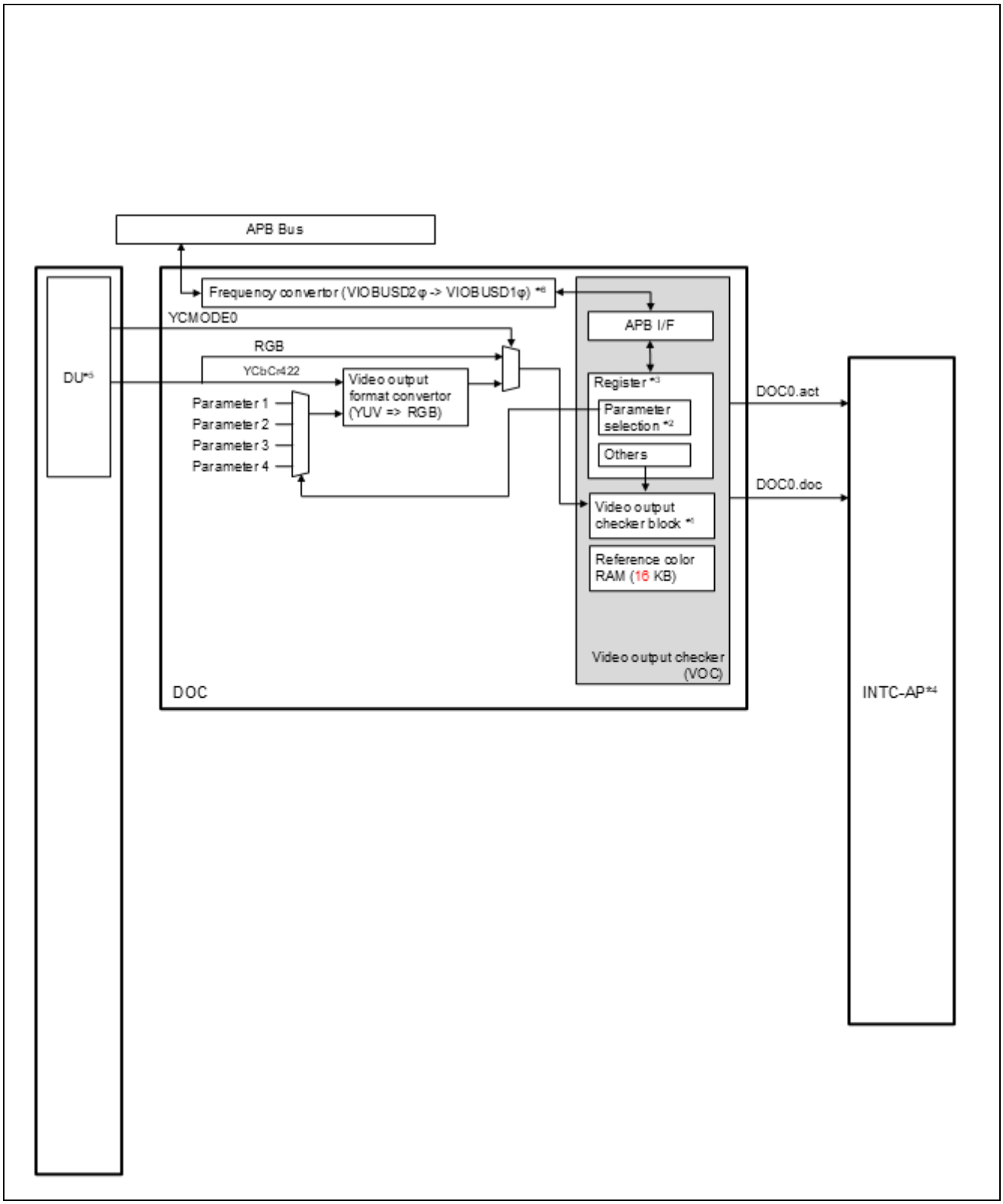



**Figure 65.2 DOC Block Diagram**

Notes:

1. Refer to Figure 65.3 in detail block diagram.
2. Refer to section 65.3.2.1 and 65.3.2.2 in detail.
3. Refer to section 65.3.2 about register setting.
4. Refer to section 15, Interrupt Controller (INTC-AP).
5. Refer to section 64, Display Unit (DU).
6. VIOBUSD2 $\phi$ = 266 MHz, VIOBUSD1 $\phi$ = 533 MHz.

### 65.1.3 External Pins

DOC doesn’t have external pins.

### 65.1.4 Register Configuration

This section contains a description of all registers of the DOC.

#### 65.1.4.1 DOC registers overview

Table 65.2 shows the register configuration.

Table 65.3 shows the register states in each operating mode. Write/Read access to the address not defined below is not allowed. These registers allow only 32bit access.

**Table 65.2 Register Configuration for DOC0** m = 0 to 15

| Register Name | Abbreviation | R/W | Address (DOC0_base = H'FEBA 0000 ) |
|---|---|---|---|
| Status register | DOC0STR | R | <DOC0_base> |
| Control register | DOC0CTL | R/W | <DOC0_base> + H'0004 |
| Video Output and Activity Monitor enable register | DOC0EN | R/W | <DOC0_base> + H'0008 |
| Activity Monitor detection time register for video channel 0 | DOC0TIME0 | R/W | <DOC0_base> + H'0010 |
| Back porch offset of video channel 0 screen | DOC0OFFS0 | R/W | <DOC0_base> + H'0020 |
| Image size of video channel 0 screen | DOC0DISP0 | R/W | <DOC0_base> + H'0028 |
| Video channel 0 selection | DOC0ACT0 | R/W | <DOC0_base> + H'0030 |
| Discriminator register | DOC0DIFF | R | <DOC0_base> + H'0040 |
| Format conversion parameter selection for Video channel 0 | DOC0YCMODE0 | R/W | <DOC0_base> + H'0060 |
| Start point of monitor m registers | DOC0MmCFG0 | R/W | <DOC0_base> + H'0100 + m × H'0020 |
| Range of monitor m registers | DOC0MmCFG1 | R/W | <DOC0_base> + H'0104 + m × H'0020 |
| Start address of monitor m on expected RAM registers | DOC0MmCFG2 | R/W | <DOC0_base> + H'0108 + m × H'0020 |
| Acceptance threshold for sum of pixel of monitor m registers | DOC0MmCFG3 | R/W | <DOC0_base> + H'010C + m × H'0020 |
| CLUT0 of monitor m registers | DOC0MmCFG4 | R/W | <DOC0_base> + H'0110 + m × H'0020 |
| CLUT1 of monitor m registers | DOC0MmCFG5 | R/W | <DOC0_base> + H'0114 + m × H'0020 |
| CLUT2 of monitor m registers | DOC0MmCFG6 | R/W | <DOC0_base> + H'0118 + m × H'0020 |
| CLUT3 of monitor m registers | DOC0MmCFG7 | R/W | <DOC0_base> + H'011C + m × H'0020 |
| Reference color RAM 0 - 4095 register | DOC0EXPD0000 to 4095 | R/W | <DOC0_base> + H'0300 to H'42FC |
| Reference color RAM 4096 - 8191 register | DOC0EXPD4096 to 8191 | R/W | <DOC0_base> + H'4300 to H'82FC |

**Table 65.3 Register State in Each Operation Mode for DOC0** m = 0 to 15

| Register Abbreviation | Reset * | Software Reset | Module Standby |
|---|---|---|---|
| DOC0STR | H'0000 0000 | H'0000 0000 | Retained |
| DOC0CTL | H'0000 0000 | H'0000 0000 | Retained |
| DOC0EN | H'0000 0000 | H'0000 0000 | Retained |
| DOC0TIME0 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0OFFS0 | H'0001 0001 | H'0001 0001 | Retained |
| DOC0DISP0 | H'0001 0001 | H'0001 0001 | Retained |
| DOC0ACT0 | H'0000 0001 | H'0000 0001 | Retained |
| DOC0DIFF | H'0000 0000 | H'0000 0000 | Retained |
| DOC0YCMODE0 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0MmCFG0 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0MmCFG1 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0MmCFG2 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0MmCFG3 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0MmCFG4 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0MmCFG5 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0MmCFG6 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0MmCFG7 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0EXPD0000 - 4095 | H'0000 0000 | H'0000 0000 | Retained |
| DOC0EXPD4096 - 8191 | H'0000 0000 | H'0000 0000 | Retained |

Note: * For details, refer to section 14, RESET (RST).

### 65.1.5 Connected Module

Modules which are connected to DOC are:

- DU
- INTC-AP

## 65.2 Register Description

[Legend]
—: Reserved. The write value should always be 0.
Initial value: Register value after a reset.
R/W: Readable/writable. The written value can be read.
R: Read-only. The write value should always be 0.

### 65.2.1 Status Register (DOC0STR)

This register holds various status and error information.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | VOCAn ACTSTR | VOCAn E18 | VOCAn VOCSTR | VOCAnSELMON[3:0] | | | | VOCAn E17 | VOCAn E16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VOCAnEm | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 24 | VOCAnACTSTR | All 0 | R | Activity Monitor status<br>0: not active<br>1: active |
| 23 | VOCAnE18 | 0 | R | Parameter error<br>Area of Video Output Monitor surpasses the size of video channel image If this error occurs, the related Video Output Monitor error (VOCAnEm) flag is also set. |
| 22 | VOCAnVOCSTR | 0 | R | VOC monitor status<br>0: Video Output Monitor not active in current video channel frame*4<br>1: Video Output Monitor active in current video channel frame |
| 21 to 18 | VOCAnSELMON [3:0] | 0 | R | Monitor number of the current video channel frame<br>VOCAnSELMON[3:0] = m |
| 17 | VOCAnE17 | 0 | R | Activity Monitor of video channel 1 error<br>0: no error occurred<br>1: error occurred*1,*2 |
| 16 | VOCAnE16 | 0 | R | Activity Monitor of video channel 0 error<br>0: no error occurred<br>1: error occurred*1,*2 |
| 15 to 0 | VOCAnEm<br>m = 15 to 0 | 0 | R | Video Output Monitor area m error<br>0: no error occurred<br>1: error occurred*1,*3 |

Note 1. These error flags are cleared via the corresponding clear bits in the control register DOC0CTL.
Note 2. If any Activity Monitor error occurred, the Activity Monitor error interrupt DOC0.act is asserted.
Note 3. If any Video Output Monitor occurred, the Video Output Monitor error interrupt DOC0.doc is asserted.
Note 4. As there is no Video Output Monitor checked during the switching of the video channel, the status of VOCAnVOCSTR will be 0 in this case, even if one (or more) VOC Monitor is enabled (VOCAnENm = 1).

### 65.2.2 Control Register (DOC0CTL)

This register is used to mask interrupts, apply a software reset and to clear the error flags in the status register DOC0STR.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VOCAn MKVOC | VOCAn MKINT | VOCAn SRST | — | — | — | — | — | — | — | — | — | — | VOCAn CL18 | VOCAn CL17 | VOCAn CL16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VOCAnCLm | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Positon | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | VOCAnMKVOC | 0 | R/W | Video Output Monitor error detection mode<br>0: continuous mode: after detection of an area m error the Video Output Monitor continuous monitoring<br>1: stop mode: after detection of the first area m error the Video Output Monitor stops monitoring of other areas |
| 30 | VOCAnMKINT | 0 | R/W | Interrupt masking<br>0: DOC0.doc and DOC0.act interrupts are generated upon error detections<br>1:DOC0.doc and DOC0.act interrupts are not generated upon error detections |
| 29 | VOCAnSRST | 0 | R/W | Software reset<br>Write:<br>0: no function<br>1: apply software reset to DOC module<br>Read:<br>0: software reset released<br>1: software reset active<br>Refer also to the description (1) Software reset procedure after this table. |
| 18 | VOCAnCL18 | 0 | R/W | Parameter error clear<br>0: sets VOCSnCL18 = 0<br>1: clear DOC0STR.VOCAnE18 |
| 17 | VOCAnCL17 | 0 | R/W | Activity Monitor of video channel 1 error flag clear<br>0: sets VOCSnCL17 = 0<br>1: clear DOC0STR.VOCAnE17<br>Clearing the error flag has to be done by a certain procedure. Refer to “Activity Monitors error flag clearing procedure” in Section 65.3.3.2, Activity Monitors error notification for details. |
| 16 | VOCAnCL16 | 0 | R/W | Activity Monitor of video channel 0 error flag clear<br>0: sets VOCSnCL16 = 0<br>1: clear DOC0STR.VOCAnE16<br>Clearing the error flag has to be done by a certain procedure. Refer to “Activity Monitors error flag clearing procedure” in Section 65.3.3.2, Activity Monitors error notification for details. |
| 15 to 0 | VOCAnCLm<br>m = 15 to 0 | 0 | R/W | Video Output Monitor m error flag clear<br>0: sets VOCAnCLm = 0<br>1: clear DOC0STR.VOCAnEm<br>Clearing the error flag has to be done by a certain procedure. Refer to “Video Output Monitor error flag clearing procedure” in Section 65.3.3.1, Video Output Monitor error notification for details. |

Note 1. If both Activity Monitor errors are cleared, the Activity Monitor error interrupt DOC0.act is de-asserted.
Note 2. If all Video Output Monitor errors are cleared, the Video Output Monitor error interrupt DOC0.doc is deasserted.

**(1) Software Reset Procedure**

The following sequence describes the procedure to issue a software reset of the “Video Output Checker” unit.

1. Apply software reset by setting DOC0CTL.VOCAnSRST = 1.
2. Wait until software reset released state reached by polling until DOC0CTL.VOCAnSRST = 0.
3. Wait for occurrence of next Video Data Controller interrupt.

The “Video Output Checker” software reset has been completed. However, this reset is effect only for “Video Output Checker” unit. Hence, the control of reset by CPG is required in case that the whole of DOC is reset.

### 65.2.3 Video Output and Activity Monitor Enable Register (DOC0EN)

This register is enable/disable each Video Output Monitor.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VOCAnEN16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VOCAnENm | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | VOCAnEN16 | 0 | R/W | Enable/disable Activity Monitors<br>0: Activity Monitors disabled<br>1: Activity Monitors enabled |
| 15 to 0 | VOCAnENm<br>m = 15 to 0 | 0 | R/W | Enable/disable Video Output Monitor m<br>0: Video Output Monitor m disabled<br>1: Video Output Monitor m enabled |

Enabled or disabling of the Activity Monitors or of a Video Output Monitor m becomes effective with the next frame of the assigned video channel, i.e. with its next VSYNC.

### 65.2.4 Reserved

### 65.2.5 Video Channel 0 Activity Monitor Detection Time Register (DOC0TIME0)

These registers specify the upper and lower detection time for Activity Monitor for both video channels. The timing values in this register are specified in units of one cycle time of RCLK. RCLK = 32.768 kHz.

The detection time is given or taken 5 cycles of RCLK. Hence, this deviation should be considered for setting the time.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMAX0 [11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMIN0 [11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 to 16 | VOCAnMAX0 [11:0] | All 0 | R/W | Assign upper detection time for Activity Monitor (0~125.1ms)<br><br>**0:** 0.0000ms<br>**1:** 0.0305ms<br>**2:** 0.0610ms<br>**3:** 0.0916ms<br>…<br>**4094:** 124.939ms<br>**4095:** 124.969ms |
| 11 to 0 | VOCAnMIN0 [11:0] | All 0 | R/W | Assign upper detection time for Activity Monitor (0~125.1ms)<br><br>**0:** 0.0000ms<br>**1:** 0.0305ms<br>**2:** 0.0610ms<br>**3:** 0.0916ms<br>…<br>**4094:** 124.939ms<br>**4095:** 124.969ms |

Note : Writing to these registers is only allowed, when the Activity Monitors are disabled, i.e. DOC0EN.VOCAnEN16 = 0.

### 65.2.6 Reserved

### 65.2.7 Video Channel 0 Back Porch Offset Register (DOC0OFFS0)

These registers specify the horizontal and vertical back porch offset of video channel's timing signals. Refer to Section 65.3.2.1, Video Monitor area definition for details about the offset definition.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnHOFFS0 [11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | VOCAnVOFFS0 [10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 to 16 | VOCAnHOFFS0 [11:0] | H'001 | R/W | Horizontal back porch offset of video channel 0 in pixels |
| 10 to 0 | VOCAnVOFFS0 [10:0] | H'001 | R/W | Vertical back porch offset of video channel 0 in lines |

### 65.2.8 Reserved

### 65.2.9 Video Channel 0 Image Size Register (DOC0DISP0)

These registers specify size of the image of the video channels.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnHSIZE0 [11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | VOCAnVSIZE0 [10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 to 16 | VOCAnHSIZE0 [11:0] | H'001 | R/W | Horizontal size of the video channel 0 image in pixels |
| 10 to 0 | VOCAnVSIZE0 [10:0] | H'001 | R/W | Vertical size of the video channel 0 image in pixels |

### 65.2.10 Reserved

### 65.2.11 Video Channel 0 Selection Register (DOC0ACT0)

These registers specify which video channel is available in the device.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VOCAn VOUT0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | VOCAnVOUT0 | B'1 | R/W | Video channel 0 availability selection<br>0: Video channel 0 is not available<br>1: Video channel 0 is available |

Note: Video channel 0 needs to be selected as available (VOCAnVOUT0 = 1) before it can be used for any monitoring purposes.

### 65.2.12 Reserved

### 65.2.13 Video Output Monitor Discriminator Register (DOC0DIFF)

This register shows the calculated discriminator value D for a single Video Output Monitor area. Refer to Section 65.3.2.1, Video Output Monitoring Basic Procedure for details about the discriminator calculation.

This register is to be used only for debugging purposes and in the way shown below (m = 15 to 0):

1. Disable all Video Output Monitor areas m (DOC0EN.VOCAnENm = 0).
2. Enable the single area m to be evaluated (DOC0EN.VOCAnENm = 1).
3. Disable the evaluated single area m (DOC0EN.VOCAnENm = 0).
4. Wait until monitoring becomes inactive, i.e. until DOC0STR.VOCAnVOCSTR = 0.
5. The last calculated discriminator can be read from the VOCAnDIFF register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VOCAnDIFF[17:16] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VOCAnDIFF[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 to 0 | VOCAnDIFF [17:0] | All 0 | R | Last discriminator value |

### 65.2.14 Format Conversion Parameter Selection for Video Channel 0 (DOC0YCMODE0)

One from four type of parameter can be selected for image format conversion (YUV to RGB). As described in Section 65.1.2, the format conversion is required if DU output format is YUV. Refer to the conversion formula in the table below. The conversion parameter is effect during YCMODE0/1 from DU asserting to 1'b1

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | SEL_PARAM [1:0] | | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 to 2 | SEL_PARAM [1:0] | 2'b00 | R/W | **(1) 2'b00 [BT601 Extension type]**<br>R = 1.164*(Y-16) + 1.596*(Cr-128)<br>G = 1.164*(Y-16) – 0.813*(Cr-128) – 0.392*(Cb-128)<br>B = 1.164*(Y-16) + 2.017*(Cb-128)<br>**(2) 2'b01 [BT601 Compressed type]**<br>R = Y + 1.402*(Cr-128)<br>G = Y – 0.344*(Cb-128) – 0.714*(Cr-128)<br>B = Y + 1.722*(Cb-128)<br>**(3) 2'b10 [BT709 Extension type]**<br>R = 1.1644*(Y-16) + 1.7928*(Cr-128)<br>G = 1.1644*(Y-16) – 0.5330*(Cr-128) – 0.2133*(Cb-128)<br>B = 1.1644*(Y-16) + 2.1125*(Cb-128)<br>**(4) 2'b11 [BT709 Compressed type]**<br>R = Y + 1.5397*(Cr-128)<br>G = Y – 0.1832*(Cb-128) – 0.4577*(Cr-128)<br>B = Y + 1.8142*(Cb-128)<br>**Noted**<br>Setting to the register CralCbe not defined aboCre is not allowed. |

### 65.2.15 Reserved

### 65.2.16 Video Output Monitor-m Area Register (DOC0MmCFG0) m = 15 to 0 (monitor number)

These registers specify the screen area to be monitored by the Video Output Monitor m.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmHSTRT [11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | VOCAnMmVSTRT [10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 to 16 | VOCAnMmHSTRT [11:0] | All 0 | R/W | Video Output Monitor m area horizontal start point in pixels. |
| 10 to 0 | VOCAnMmVSTRT [10:0] | All 0 | R/W | Video Output Monitor m area vertical start point in pixels. |

Note : Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.17 Video Output Monitor-m Area Size Register (DOC0MmCFG1) m = 15 to 0 (monitor number)

These registers specify the area size to be monitored by the Video Output Monitor m.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | VOCAnMmHSIZE [7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | VOCAnMmVSIZE [7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 23 to 16 | VOCAnMmHSIZE [7:0] | All 0 | R/W | Video Output Monitor m area horizontal size in pixels |
| 7 to 0 | VOCAnMmVSIZE [7:0] | All 0 | R/W | Video Output Monitor m area vertical size in pixels |

Note: Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.18 Video Output Monitor-m Reference Color RAM Address Register (DOC0MmCFG2) m = 15 to 0 (monitor number)

These registers specify the first address of the 2-bit color index in the reference RAM for Video Output Monitor m reference colors.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | VOCAnMmADDR [12:0] | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 to 0 | VOCAnMmADDR [12:0] | All 0 | R/W | Video Output Monitor m reference RAM start address. |

Note: Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.19 Video Output Monitor-m Acceptance Threshold Register (DOC0MmCFG3) m = 15 to 0 (monitor number)

These registers specify the threshold to be compared against the calculated discriminator.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VOCAnMmTHSH [17:16] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VOCAnMmTHSH [15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 to 0 | VOCAnMmTHSH [17:0] | 0 | R/W | Video Output Monitor m acceptance threshold<br>Setting VOCAnMmTHSH[17:0] = 0 means, the Video Output Monitor always reports an error. |

Note : Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.20 Video Output Monitor-m Reference Color 0 Register (DOC0MmCFG4) m = 15 to 0 (monitor number)

These registers specify the minimum/maximum values of reference color 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmRUP0 [3:0] | | | | VOCAnMmGUP0 [3:0] | | | | VOCAnMmBUP0 [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmRLO0 [3:0] | | | | VOCAnMmGLO0 [3:0] | | | | VOCAnMmBLO0 [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 to 24 | VOCAnMmRUP0 [3:0] | All 0 | R/W | Video Output Monitor m reference color 0: red upper limit |
| 23 to 20 | VOCAnMmGUP0 [3:0] | All 0 | R/W | Video Output Monitor m reference color 0: green upper limit |
| 19 to 16 | VOCAnMmBUP0 [3:0] | All 0 | R/W | Video Output Monitor m reference color 0: blue upper limit |
| 11 to 8 | VOCAnMmRLO0 [3:0] | All 0 | R/W | Video Output Monitor m reference color 0: red lower limit |
| 7 to 4 | VOCAnMmGLO0 [3:0] | All 0 | R/W | Video Output Monitor m reference color 0: green lower limit |
| 3 to 0 | VOCAnMmBLO0 [3:0] | All 0 | R/W | Video Output Monitor m reference color 0: blue lower limit |

Note: Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.21 Video Output Monitor-m Reference Color 1 Register (DOC0MmCFG5) m = 15 to 0 (monitor number)

These registers specify the minimum/maximum values of reference color 1.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmRUP1 [3:0] | | | | VOCAnMmGUP1 [3:0] | | | | VOCAnMmBUP1 [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmRLO1 [3:0] | | | | VOCAnMmGLO1 [3:0] | | | | VOCAnMmBLO1 [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 to 24 | VOCAnMmRUP1 [3:0] | All 0 | R/W | Video Output Monitor m reference color 1: red upper limit |
| 23 to 20 | VOCAnMmGUP1 [3:0] | All 0 | R/W | Video Output Monitor m reference color 1: green upper limit |
| 19 to 16 | VOCAnMmBUP1 [3:0] | All 0 | R/W | Video Output Monitor m reference color 1: blue upper limit |
| 11 to 8 | VOCAnMmRLO1 [3:0] | All 0 | R/W | Video Output Monitor m reference color 1: red lower limit |
| 7 to 4 | VOCAnMmGLO1 [3:0] | All 0 | R/W | Video Output Monitor m reference color 1: green lower limit |
| 3 to 0 | VOCAnMmBLO1 [3:0] | All 0 | R/W | Video Output Monitor m reference color 1: blue lower limit |

Note: Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.22 Video Output Monitor-m Reference Color 2 Register (DOC0MmCFG6) m = 15 to 0 (monitor number)

These registers specify the minimum/maximum values of reference color 2.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmRUP2 [3:0] | | | | VOCAnMmGUP2 [3:0] | | | | VOCAnMmBUP2 [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmRLO2 [3:0] | | | | VOCAnMmGLO2 [3:0] | | | | VOCAnMmBLO2 [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 to 24 | VOCAnMmRUP2 [3:0] | All 0 | R/W | Video Output Monitor m reference color 2: red upper limit |
| 23 to 20 | VOCAnMmGUP2 [3:0] | All 0 | R/W | Video Output Monitor m reference color 2: green upper limit |
| 19 to 16 | VOCAnMmBUP2 [3:0] | All 0 | R/W | Video Output Monitor m reference color 2: blue upper limit |
| 11 to 8 | VOCAnMmRLO2 [3:0] | All 0 | R/W | Video Output Monitor m reference color 2: red lower limit |
| 7 to 4 | VOCAnMmGLO2 [3:0] | All 0 | R/W | Video Output Monitor m reference color 2: green lower limit |
| 3 to 0 | VOCAnMmBLO2 [3:0] | All 0 | R/W | Video Output Monitor m reference color 2: blue lower limit |

Note: Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.23 Video Output Monitor-m Reference Color 3 Register (DOC0MmCFG7) m = 15 to 0 (monitor number)

These registers specify the minimum/maximum values of reference color 3.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmRUP3 [3:0] | | | | VOCAnMmGUP3 [3:0] | | | | VOCAnMmBUP3 [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VOCAnMmRLO3 [3:0] | | | | VOCAnMmGLO3 [3:0] | | | | VOCAnMmBLO3 [3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 to 24 | VOCAnMmRUP3 [3:0] | All 0 | R/W | Video Output Monitor m reference color 3: red upper limit |
| 23 to 20 | VOCAnMmGUP3 [3:0] | All 0 | R/W | Video Output Monitor m reference color 3: green upper limit |
| 19 to 16 | VOCAnMmBUP3 [3:0] | All 0 | R/W | Video Output Monitor m reference color 3: blue upper limit |
| 11 to 8 | VOCAnMmRLO3 [3:0] | All 0 | R/W | Video Output Monitor m reference color 3: red lower limit |
| 7 to 4 | VOCAnMmGLO3 [3:0] | All 0 | R/W | Video Output Monitor m reference color 3: green lower limit |
| 3 to 0 | VOCAnMmBLO3 [3:0] | All 0 | R/W | Video Output Monitor m reference color 3: blue lower limit |

Note: Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.24 Video Output Monitor Reference RAM Register-k [k = 0 to 4095] (DOC0EXPDk).

These registers define the 2-bpp CLUT2 indices of the reference colors. For further details refer to Section 65.3.1.3, Reference color.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | I0 | | I1 | | I2 | | I3 | | I4 | | I5 | | I6 | | I7 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 14 | I0 [1:0] | All 0 | R/W | CLUT2 index of pixel x |
| 13 to 12 | I1 [1:0] | All 0 | R/W | CLUT2 index of pixel x+1 |
| 11 to 10 | I2 [1:0] | All 0 | R/W | CLUT2 index of pixel x+2 |
| 9 to 8 | I3 [1:0] | All 0 | R/W | CLUT2 index of pixel x+3 |
| 7 to 6 | I4 [1:0] | All 0 | R/W | CLUT2 index of pixel x+4 |
| 5 to 4 | I5 [1:0] | All 0 | R/W | CLUT2 index of pixel x+5 |
| 3 to 2 | I6 [1:0] | All 0 | R/W | CLUT2 index of pixel x+6 |
| 1 to 0 | I7 [1:0] | All 0 | R/W | CLUT2 index of pixel x+7 |

Note: Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

### 65.2.25 Video Output Monitor Reference RAM Register-k [k = 4096 to 8191] (DOC0EXPDk).

These registers define the 2-bpp CLUT2 indices of the reference colors. For further details refer to Section 65.3.1.3, Reference color.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | I0 | | I1 | | I2 | | I3 | | I4 | | I5 | | I6 | | I7 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit Position | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 14 | I0 [1:0] | All 0 | R/W | CLUT2 index of pixel x |
| 13 to 12 | I1 [1:0] | All 0 | R/W | CLUT2 index of pixel x+1 |
| 11 to 10 | I2 [1:0] | All 0 | R/W | CLUT2 index of pixel x+2 |
| 9 to 8 | I3 [1:0] | All 0 | R/W | CLUT2 index of pixel x+3 |
| 7 to 6 | I4 [1:0] | All 0 | R/W | CLUT2 index of pixel x+4 |
| 5 to 4 | I5 [1:0] | All 0 | R/W | CLUT2 index of pixel x+5 |
| 3 to 2 | I6 [1:0] | All 0 | R/W | CLUT2 index of pixel x+6 |
| 1 to 0 | I7 [1:0] | All 0 | R/W | CLUT2 index of pixel x+7 |

Note: Changing this register is only allowed, when the respective Video Monitor m is disabled, i.e. DOC0EN.VOCAnENm = 0.

## 65.3 Operation

### 65.3.1 Video-Output-Checker block diagram

The figure below shows the building blocks of the Video-Output-Checker (VOC). Further details about the building blocks are described in the following sub-chapters.

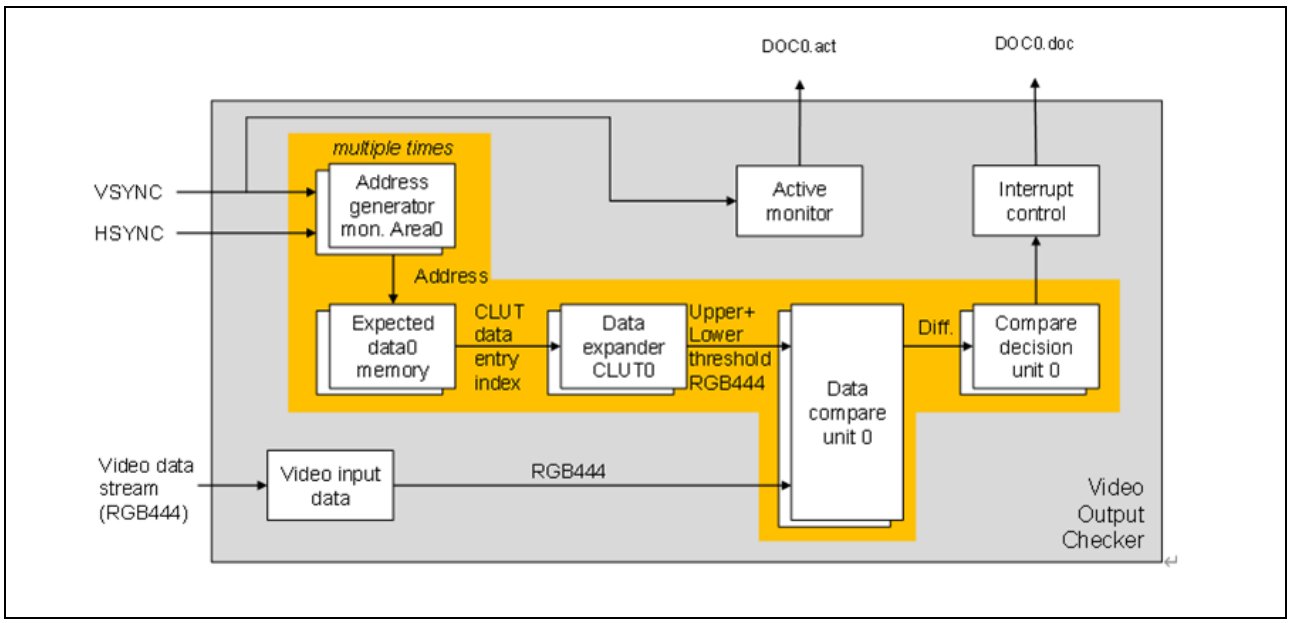



**Figure 65.3 Video-Output-Checker (VOC) diagram**

#### 65.3.1.1 Address Generator

The address generator specifies a rectangular area out of the full video screen to be monitored. The address generator takes care for time multiplexed monitoring of each (enabled) monitoring area. One monitor area is checked each frame. After all active monitored areas are processed, the monitoring processes are repeated.

- This allows that all the monitored data is just a part of the screen.
- Up to 16 monitored areas can be defined.
- Enable / Disable control of monitored areas by register via APB bus interface.
- The locations of monitored areas are register programmable (during monitored area is disabled).

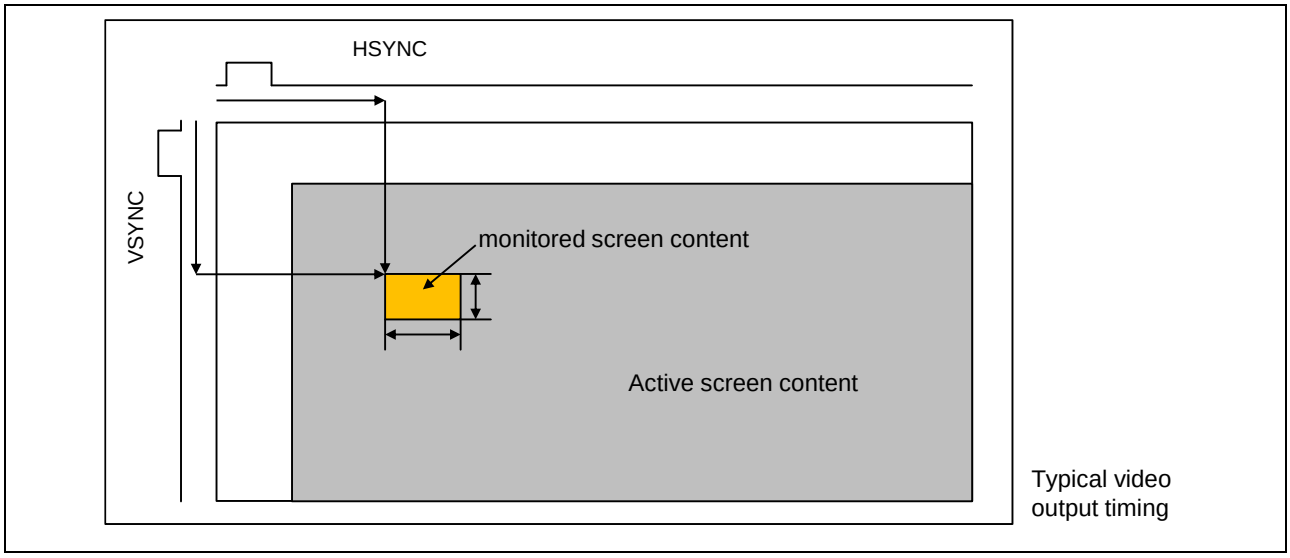



**Figure 65.4 Addressing of monitoring area on the screen**

#### 65.3.1.2 Expected Data Memory

The expected data memory contains the expected data, which are used to compare against the incoming video data. Per incoming pixel of a monitoring area there is one test data for comparison. To save memory the test data is stored as indexed colors in CLUT2 format. This allows 4 different indexed color values for each monitoring area. The 4 values are set by Video Output Monitor reference register (Section 65.2.20 to Section 65.2.23).

#### 65.3.1.3 Data Expander (CLUT)

For each expected data unit, there is one data expander (CLUT). The unit expands the indexed color to a direct mapped RGB444 color for upper and lower threshold limits per color component. In this unit the indexed color is replaced not by a single direct mapped color but by two values per color component.

- RGB444 upper limit

  1. upper limit red (4bit)
  2. upper limit green (4bit)
  3. upper limit blue (4bit)

- RGB444 lower limit

  1. lower limit red (4bit)
  2. lower limit green (4bit)
  3. lower limit blue (4bit)

This threshold values are register programmable. They describe the acceptable color variation of the 4 indexed colors used for the encoding of the test data (telltale image) in the expected data memory before.

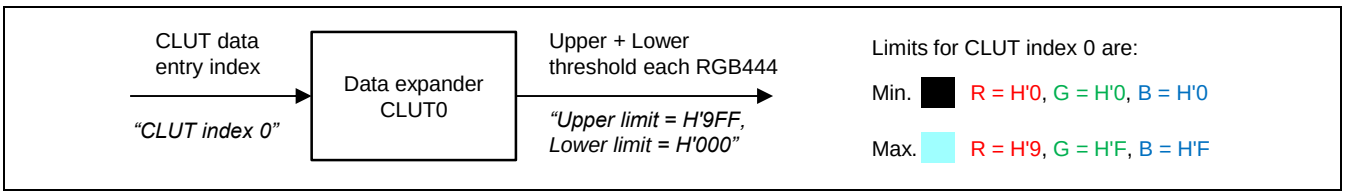



**Figure 65.5 Data expander CLUT block (1 out of 16)**

#### 65.3.1.4 Data Compare Unit

In the data compare unit the incoming video data is compared pixel- and component-wise with the upper and lower thresholds provided by the data expander unit. An incoming pixel is judged as valid as long as for all color components the value is between lower and upper limit of the test data.

If one incoming pixel is valid, the different value is set to be 0. On the other hand, if one incoming pixel is NOT valid, the difference value is calculated as sum over the deviation of each color component. This difference value is provided for each pixel of the monitoring area to the compare decision unit for further processing.

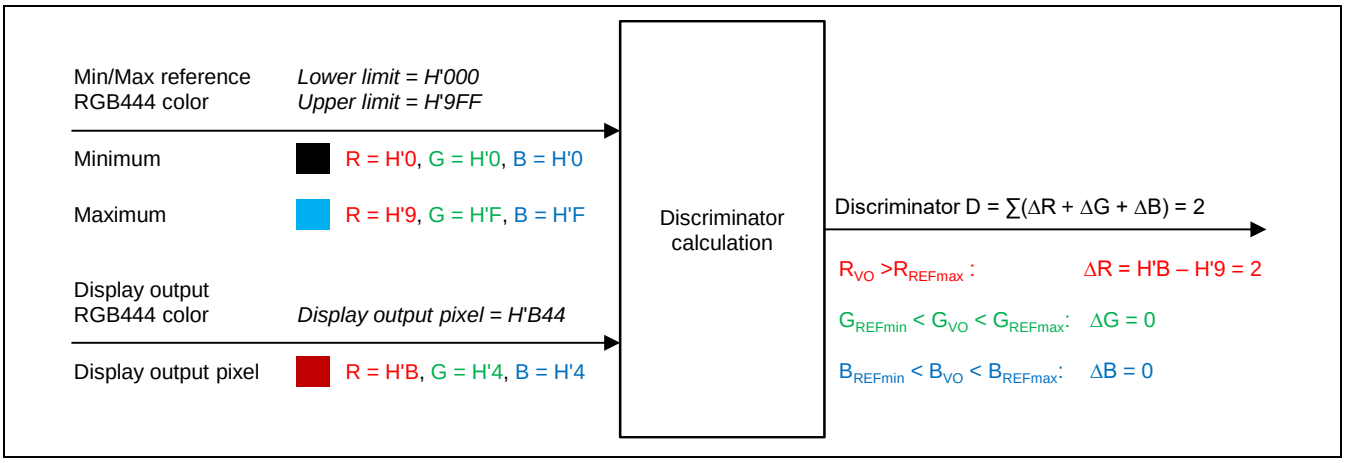



**Figure 65.6 Deviation calculation example**

#### 65.3.1.5 Compare Decision Unit

The compare decision unit sums up all difference values for one monitoring area. After one monitoring area was completed, this unit decides if an expected image was displayed correctly or not. This is done by comparison of the summed up difference values with a register programmed acceptance threshold value.

If the expected image was not correctly displayed, it raises the Compare status interrupt Signaling “fail” condition. Also the index number of the failed monitoring area is stored in a status register. Depending on this status register the CPU can later detect, which of the monitoring areas did not display the expected image.

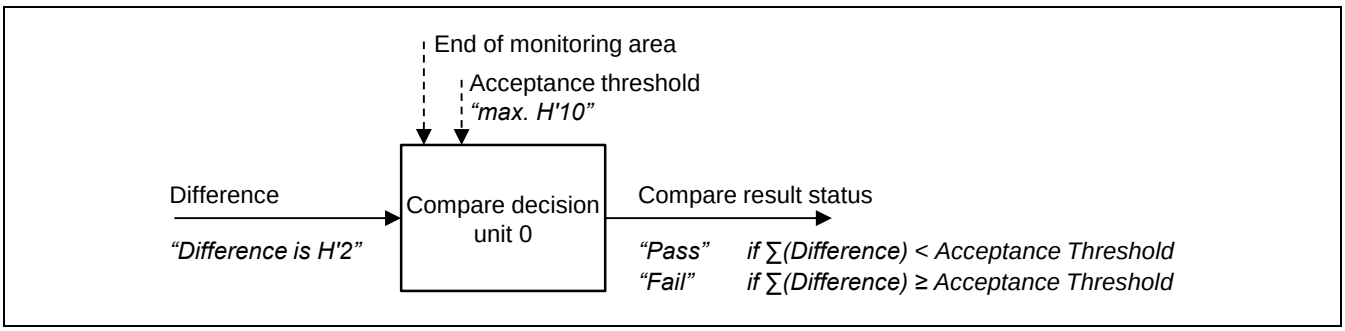



**Figure 65.7 Compare decision unit**

#### 65.3.1.6 Active Monitor

The Activity Monitor controls whether the Video Output Monitor has completed the check of a monitor area within a certain time window. If the Video Output Monitor does not report completion of the monitor area check within this time window, the Activity Monitor notifies an error.

### 65.3.2 Procedure of Parameter Setting

This section describes the procedure of the register setting for using the function of DOC. During register setting, Video Output Monitor is disabled. Each monitor can be enabled and disabled separately.

- DOC0EN.VOCAnENm = 0: Monitor m is disabled (Disable Monitor).
- DOC0EN.VOCAnENm = 1: Monitor m is enabled (Enable Monitor).

Further details about Register Setting in Figure 65.8 are described in sub-chapters.

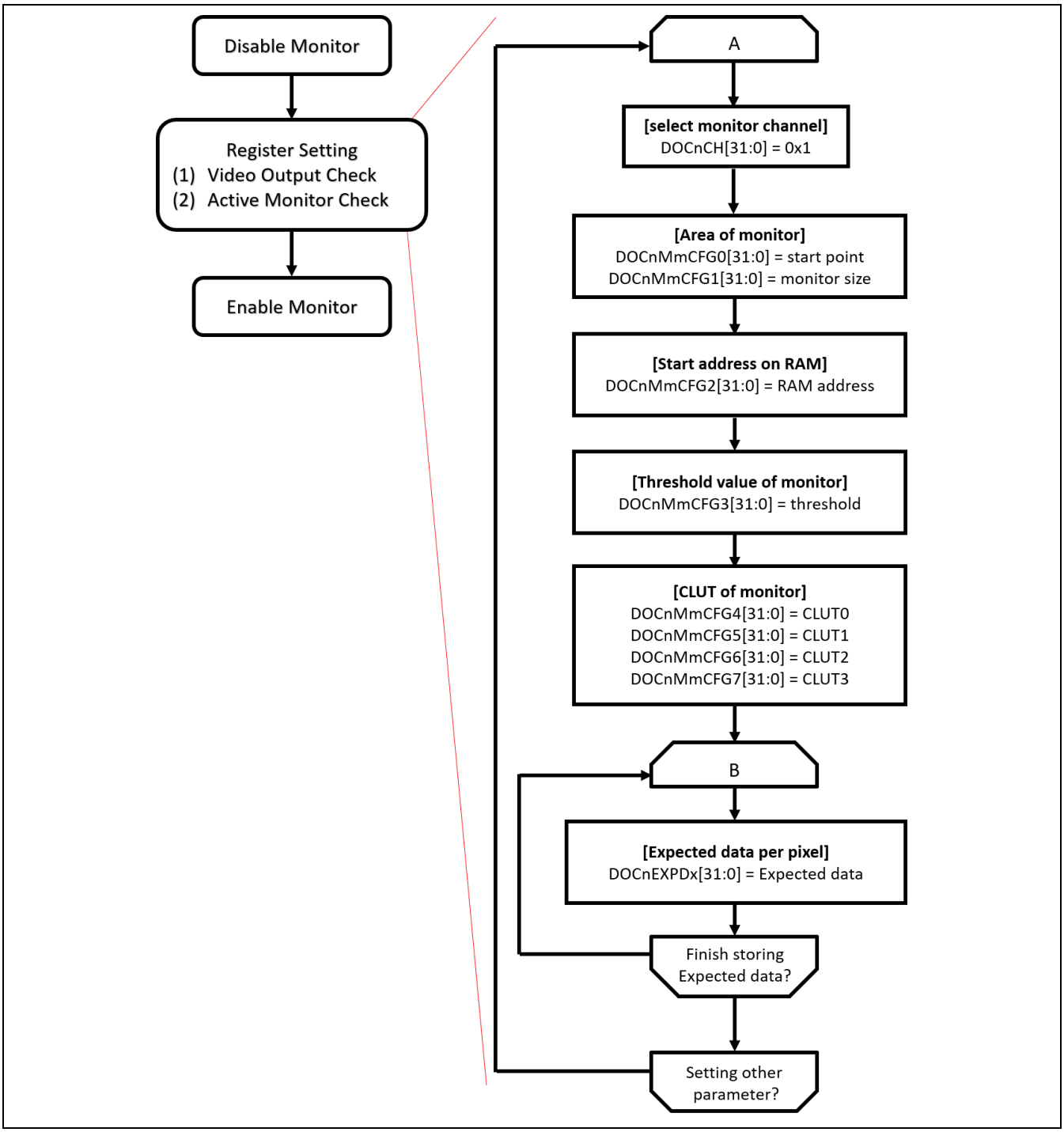



**Figure 65.8 Procedure of the register setting (n = 0, 1)**

#### 65.3.2.1 Register Setting (1) Video Output Check

**(1) Area of Monitor**

The area m to be monitored is defined relative to the HSYNC and VSYNC signals of the video output channel j:

- VOCAnHOFFSj and VOCAnVOFFS set by DOC0OFFSj: (Section 65.2.7 )

  Start value of the active video output screen in horizontal/vertical direction is shown. The back porch offset values specify the number of pixels between the falling edge of the HSYNC or VSYNC pulse and the beginning of the active screen output. Thus the offset values to specify depend on the polarity of the HSYNC and VSYNC signals:

  - If the SYNC signal is high active (refer to Figure 65.9):

    offset value = back porch (time between the end of the SYNC pulse and the active screen output)

  - If the SYNC signal is low active (refer to Figure 65.10):

    offset value = SYNC pulse width + back porch (time between the beginning of the SYNC pulse and the active screen output)

- VOCAnMmHSTRT and VOCAnMmVSTRT set by DOC0MmCFG0: (Section 65.2.16)

  This value shows horizontal/vertical coordination of the first pixel of the monitor area m

- VOCAnMmHSIZE and VOCAnMmVSIZE set by DOC0MmCFG1: (Section 65.2.17)

  This value shows horizontal/vertical size of the monitor area m

All values in horizontal direction (VOCAnHOFFS, VOCAnMmHSTRT, VOCAnMmHSIZE) are defined in pixel units.
All values in vertical direction (VOCAnVOFFS, VOCAnMmVSTRT, VOCAnMmVSIZE) are defined in line units.

**Note**

The back and front porch of the video signal must have at least a width of one pixel.

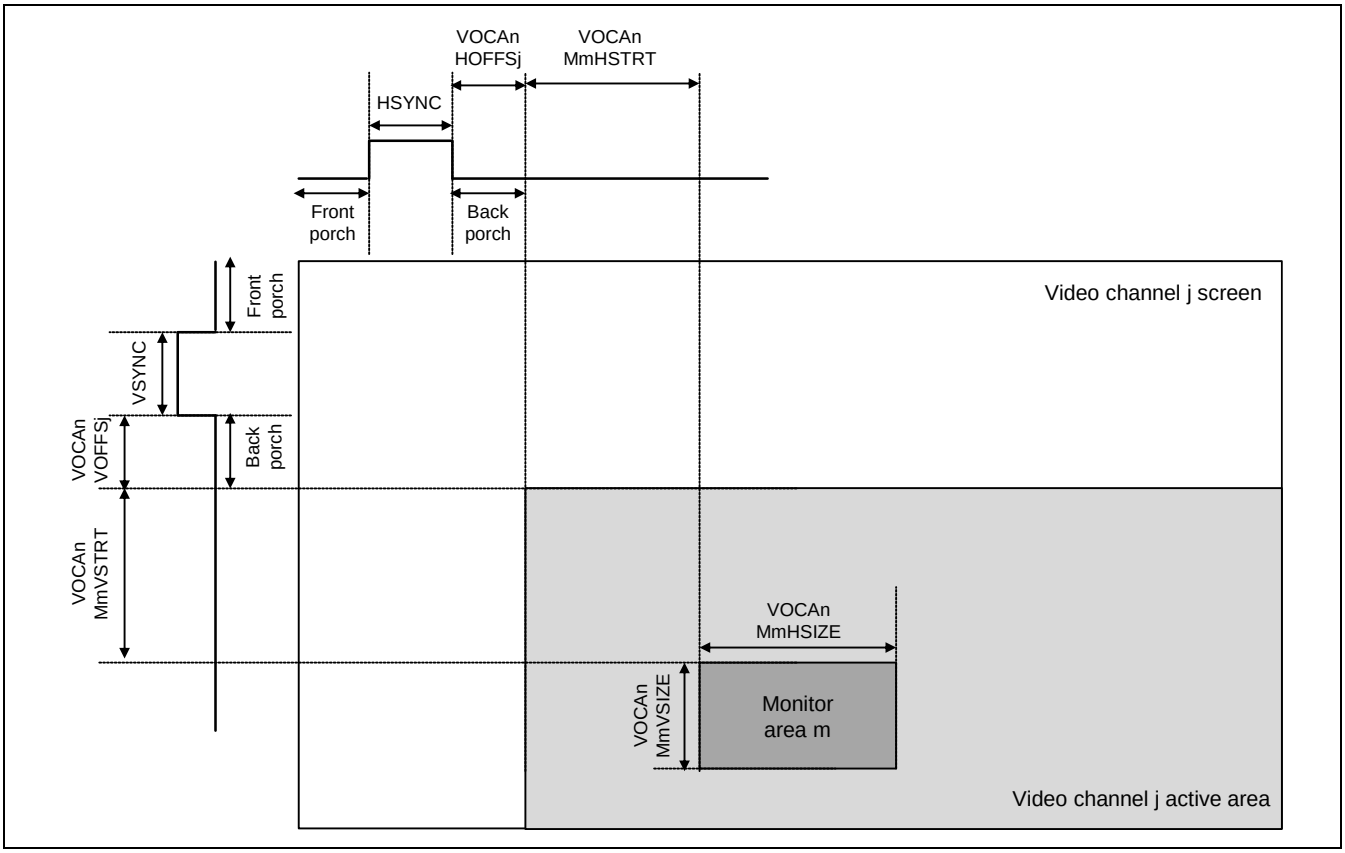



**Figure 65.9 Monitor area definition with high active HSYNC and VSYNC**

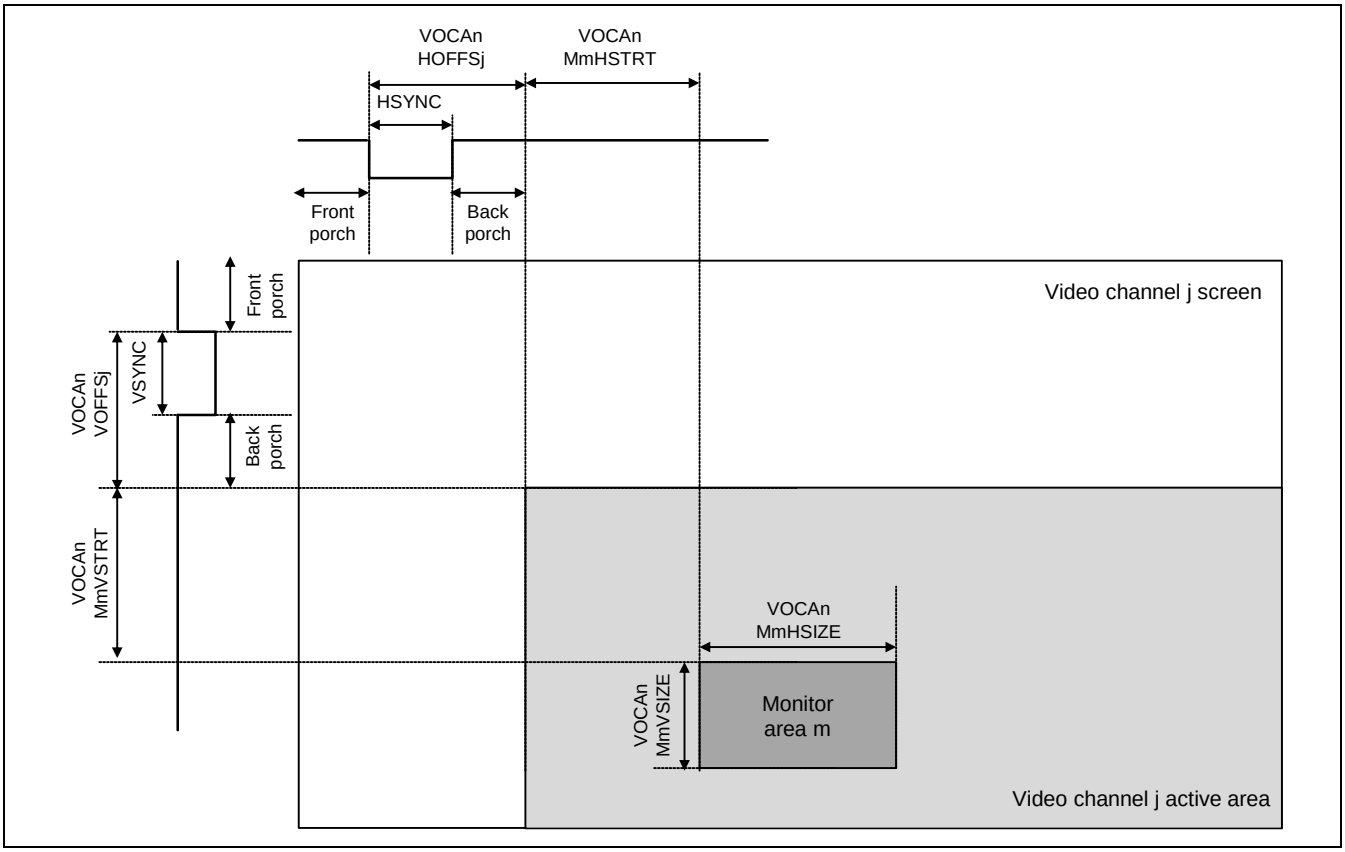



**Figure 65.10 Monitor area definition with low active HSYNC and VSYNC**

**(2) Start Address on RAM**

Start address on RAM is set by DOC0MmCFG2 (m = 15 to 0) registers. The register assigns the start address for expected data of each monitor on Expected RAM. It is required to specify the start address for expected data of each monitor before storing the expected data.

New expected data on Expected RAM can be written if there is vacant area on it. Resizing data on new vacant area on Expected RAM if there is not vacant area on it.

**(3) Threshold Value of Monitor**

Threshold value of monitor is set by DOC0MmCFG3 (m = 15 to 0) registers. The register assigns acceptance threshold in sum of pixel of each monitor. The register size is decided by next equation.

The range of threshold is from 0 to 262143.

- Min. allowed value: 0. Video Check Monitor errors always occur when DOC0MmCFG3.VOCAnMmTHSH=0.
- Max. allowed value: 262143.

**(4) CLUT of Monitor**

The reference colors for the discriminator calculation are stored as 2-bit CLUT2 indices, that select one out of four 12-bpp RGB444 colors from a color look-up table. CLUT of monitor is set by DOC0MmCFG4/CFG5/CFG6/CFG7 (m = 15 to 0).

**(5) Reference Color Definition**

Each RGB444 reference color describes an acceptance range by an upper and lower limit for each of the RGB components. The limit values for each of the four reference colors for each monitor area m are defined by the registers DOC0MmCFG4 to DOC0MmCFG7 with following content:

- VOCAnMmRUP[3:0]/RLO[3:0]: upper/lower limit of red component
- VOCAnMmGUP[3:0]/GLO[3:0]: upper/lower limit of green component
- VOCAnMmBUP[3:0]/BLO[3:0]: upper/lower limit of blue component

**(6) Reference Color Selection**

The 2-bpp CLUT2 indices are stored in the reference color RAM. The RAM is organized as 4096 x 32 bit.

The address of the reference color CLUT2 index of the first pixel of the monitor area m is defined in the Video Output Monitor m reference color RAM address register DOC0MmCFG2.VOCAnMmADDR[12:0] in the range of 0 to 8191, i.e. H’000 to H’FFF.

The 2-bpp CLUT2 indices are stored in the color RAM via the registers DOC0EXPD0000 to DOC0EXPD8191.

The addresses of the monitor areas reference color indices can be freely assigned in the RAM. The Table 65.4 below gives an examples for three monitored areas (m = 0, 2, 10) of different size.

**Note**

1. The total number of monitored pixel is limited to 65,536 pixels.
2. The RAM areas, assigned to different monitor areas, must not overlap.

**Table 65.4 Reference color RAM example**

| Monitor area | DOC0MmCFG2 | RAM address | Bit*1 | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| not used | | H'000 | | | | | | | | | | | | | | | | |
| | | H'001 | | | | | | | | | | | | | | | | |
| | | … | | | | | | | | | | | | | | | | |
| m = 0 | VOCAnM0ADDR | H'010 | I0 | | I1 | | I2 | | I3 | | I4 | | I5 | | I6 | | I7 | |
| 32 x 32 pixel | VOCAnM0ADDR + 1 | H'011 | I8 | | I9 | | I10 | | I11 | | I12 | | I13 | | I14 | | I15 | |
| | … | … | … | | … | | … | | … | | … | | … | | … | | … | |
| | VOCAnM0ADDR + 127 | H'08F | I1016 | | I1017 | | I1018 | | I1019 | | I1020 | | I1021 | | I1022 | | I1023 | |
| m = 10 | VOCAnM10ADDR | H'090 | I0 | | I1 | | I2 | | I3 | | I4 | | I5 | | I6 | | I7 | |
| 60 x 35 pixel | VOCAnM10ADDR + 1 | H'091 | I8 | | I9 | | I10 | | I11 | | I12 | | I13 | | I14 | | I15 | |
| | … | … | … | | … | | … | | … | | … | | … | | … | | … | |
| | VOCAnM10ADDR + 262 | H'196 | I2096 | | I2097 | | I2098 | | I2099 | | — | | — | | — | | — | |
| not used | | … | | | | | | | | | | | | | | | | |
| m = 2 | VOCAnM2ADDR | H'800 | I0 | | I1 | | I2 | | I3 | | I4 | | I5 | | I6 | | I7 | |
| 128 x 128 pixel | VOCAnM2ADDR+1 | H'801 | I8 | | I9 | | I10 | | I11 | | I12 | | I13 | | I14 | | I15 | |
| | … | … | … | | … | | … | | … | | … | | … | | … | | … | |
| | VOCAnM2ADDR + 2047 | H'FFF | I16376 | | I16377 | | I16378 | | I16379 | | I16380 | | I16381 | | I16382 | | I16383 | |

Note 1. "I" stands for a 2-bpp CLUT2 color index.

#### (7) Expected Data Per Pixel

Expected data per pixel is set by DOC0EXPD registers. The register is directly mapped to store the Expected data on Expected RAM. The Expected data is stored per 16bit. However, these data need to be aligned per 8pixel (16bit).

#### (8) Other Setting

When DU (Display Unit) output the image with YUV422, DOC register setting should be done as the following.

#### (a) YUV422

The following register setting for DU is required. In detail, refer to the register DEF10R of Section 64.2 in DU.

### 65.3.2.2 Register Setting (2) Active Monitor Check

The Activity Monitor controls whether the Video Output Monitor has completed the check of a monitor area within a certain time window. If the Video Output Monitor does not report completion of the monitor area check within this time window, the Activity Monitor notifies an error.

**Note**

Since the Activity Monitor requires active Video Output Monitors on the video channel to check activity, at least one Video Output Monitor must be enabled for the respective video channel. That means at least one Video Output Monitor with

- DOC0EN.VOCAnENm = 1 and DOC0CH.VOCAnCHm = 0 (assigned to video channel 0)
- DOC0EN.VOCAnENm = 1 and DOC0CH.VOCAnCHm = 1 (assigned to video channel 1)

must be set up, if the activity of both video channels shall be monitored.

If only the Activity Monitor is to be used, but no Video Output Monitor on a certain video channel, enable a Video Output Monitor for that video channel and set it up in such a way that it does not generate any Video Output Monitor errors. This can be achieved by selecting the range of all reference colors to maximum, i.e.

DOC0MmCFG4 = DOC0MmCFG5 = DOC0MmCFG6 = DOC0MmCFG7 = H'0FFF 0000

**(1) Activity Monitors Activation**

The Activity Monitors for both video channels can be enabled and disabled all together:

- DOC0EN.VOCAnEN16 = 0: Activity Monitors inactive
- DOC0EN.VOCAnEN16 = 1: Activity Monitors active

**(2) Active Monitors Set-up**

The time window for valid Video Output Monitor process completion is determined by a counter, that is operated with the clock of the Low Speed (32.768kHz = 0.0305ms per one cycle). The time window's duration and location on the time axis is set up via the timing register DOC0TIMEj for each of both video channels (j = 0, 1):

- DOC0TIMEj.VOCAnMINj: minimum detection time of Activity Monitors for video channel j
- DOC0TIMEj.VOCAnMAXj: maximum detection time of Activity Monitors for video channel j

**Note**

1. Subject to the Activity Monitor check in a certain video channel frame is always the current monitor area for that frame, i.e. the area m, indicated by DOC0STR.VOCAnSELMON[3:0].

The time window definitions DOC0TIMEj.VOCAnMINj/MAXj are automatically selected for the video channel j, that is assigned to the current monitor area via the Video Output Monitor channel assignment VOCAnCHm.

### 65.3.3 DOC Behavior

This section describes the behavior of DOC. Error notification algorithm of Video Monitor Check and Active Monitor is summarized.

#### 65.3.3.1 Video Output Monitor Error Notification

An error during check of the Video Output Monitor area m can be detected for two reasons:

**(1) Monitor Area-m Content Mismatch Error**

Such error is detected, if the content check (with respect to discriminator vs. acceptance threshold) fails. Upon detection of such error

- the monitor m area error bit DOC0STR.VOCAnEm is set
- the Video Output Monitor interrupt DOC0.doc is asserted.

**(2) Monitor Area-m Parameter Error**

Such error is detected, if the definition of a monitored area m (area m start point in DOC0MmCFG0 and size in DOC0MmCFG1) does not fit to the active screen size of the related video channel. Upon detection of such error

- the parameter error DOC0STR.VOCAnE18 flag is set
- the monitor m area error bit DOC0STR.VOCAnEm is set

**Note**

1. The error flagsin the status register DOC0STR remain set until they are cleared via the control register DOC0CTL. Refer to “Video Output Monitor error flag clearing procedure” below for details.
2. The DOC0.doc remains asserted as long as any of the VOCAnEm are set.

**(3) Video Output Monitor Error Flag Clearing Procedure**

In order to clear a Video Output Monitor error flags, that has been set due to a content mismatch or parameter error detection, proceed as follows:

1. Disable all Video Output Monitors by DOC0EN.VOCAnENm = 0.
2. Wait until Video Output Monitors are inactive, i.e. until DOC0STR.VOCAnVOCSTR = 0.
3. Evaluate the error source via the error flags in the status register DOC0STR.
4. Clear all Video Output Monitor error flags by
   - DOC0CTL.VOCAnCLm = 1 for content mismatch errors
   - DOC0CTL.VOCAnCL18 = 1 for parameter error
5. Release all Video Output Monitor error flags clear by
   - DOC0CTL.VOCAnCLm = 0 for content mismatch errors
   - DOC0CTL.VOCAnCL18 = 0 for parameter error
6. Verify that all Video Output Monitor error flags are reset, i.e. that
   - DOC0STR.VOCAnEm = 0: no content mismatch error
   - DOC0STR.VOCAnE18 = 0: no parameter error
7. Enable required Video Output Monitor m by DOC0EN.VOCAnENm = 1.

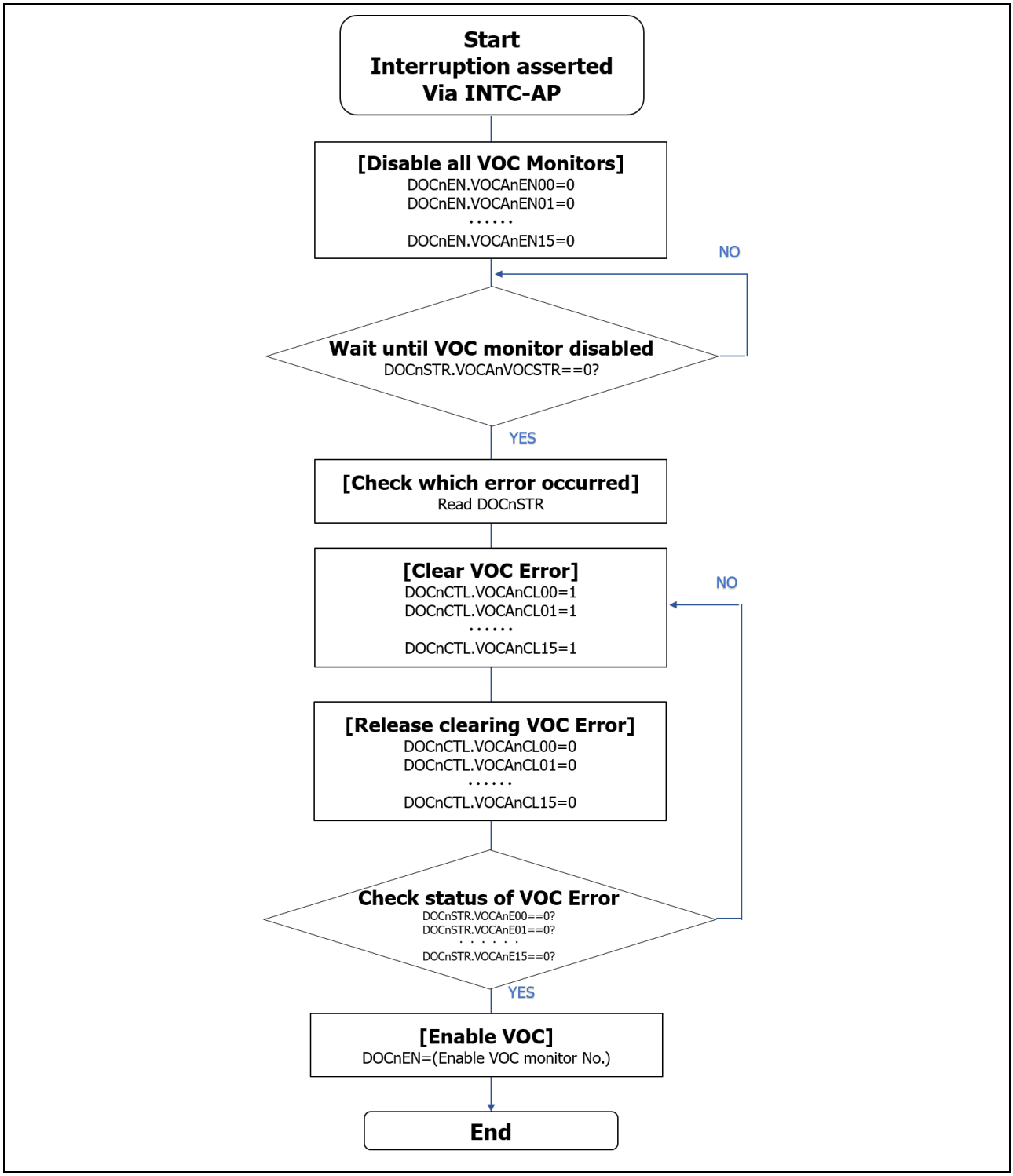



**Figure 65.11 Video Output Monitor error flag clearing procedure**

### (4) Video Output Monitor Behavior After Error Detection

The behaviour of the Video Output Monitor after an error detection can be selected from two different modes:

- DOC0CTL.VOCAnMKVOC = 0:
  Not mask Video Output Monitor
- DOC0CTL.VOCAnMKVOC = 1:

Mask all Video Output Monitor

**(5) Interrupt Masking**

Assertion of an error interrupt DOC0.doc can be disabled:

- DOC0CTL.VOCAnMKINT = 0:
  DOC0.doc is generated in case of a Video Output Monitor error detection
- DOC0CTL.VOCAnMKINT = 1:
  no DOC0.doc is generated in case of a Video Output Monitor error detection

Note that DOC0CTL.VOCAnMKINT affects also the Activity Monitor interrupt DOC0.act.

**(6) Monitoring Sequence**

In each video output frame only a single Video Output Monitor area m is examined.

### Priority of monitor areas

If several monitor areas are enabled the monitoring order follows a fixed priority:

priority of m = 0 > m = 1 > ... > m = 15

### Detection timing

The time for a content mismatch detection depends on the total number of monitored areas. The maximum detection time of an erroneous area is calculated by (number of monitored areas x 2) / framerate.

**Note**

The maximum detection time of an erroneous frame assumes that the VOC needs to switch between the video channels after each Video Output Monitor frame and that the switching to the other video channel takes one frame for synchronization.

It is recommended to setup the Video Output Monitors in a way, that video channel switching is minimized. I.e. set up Video Output Monitors related to the same video channel consecutively. (example: M0 to M7: video channel 0, M8 to M15: video channel 1) This will shorten the detection time to

(number of monitoring area + 2) / frame rate.

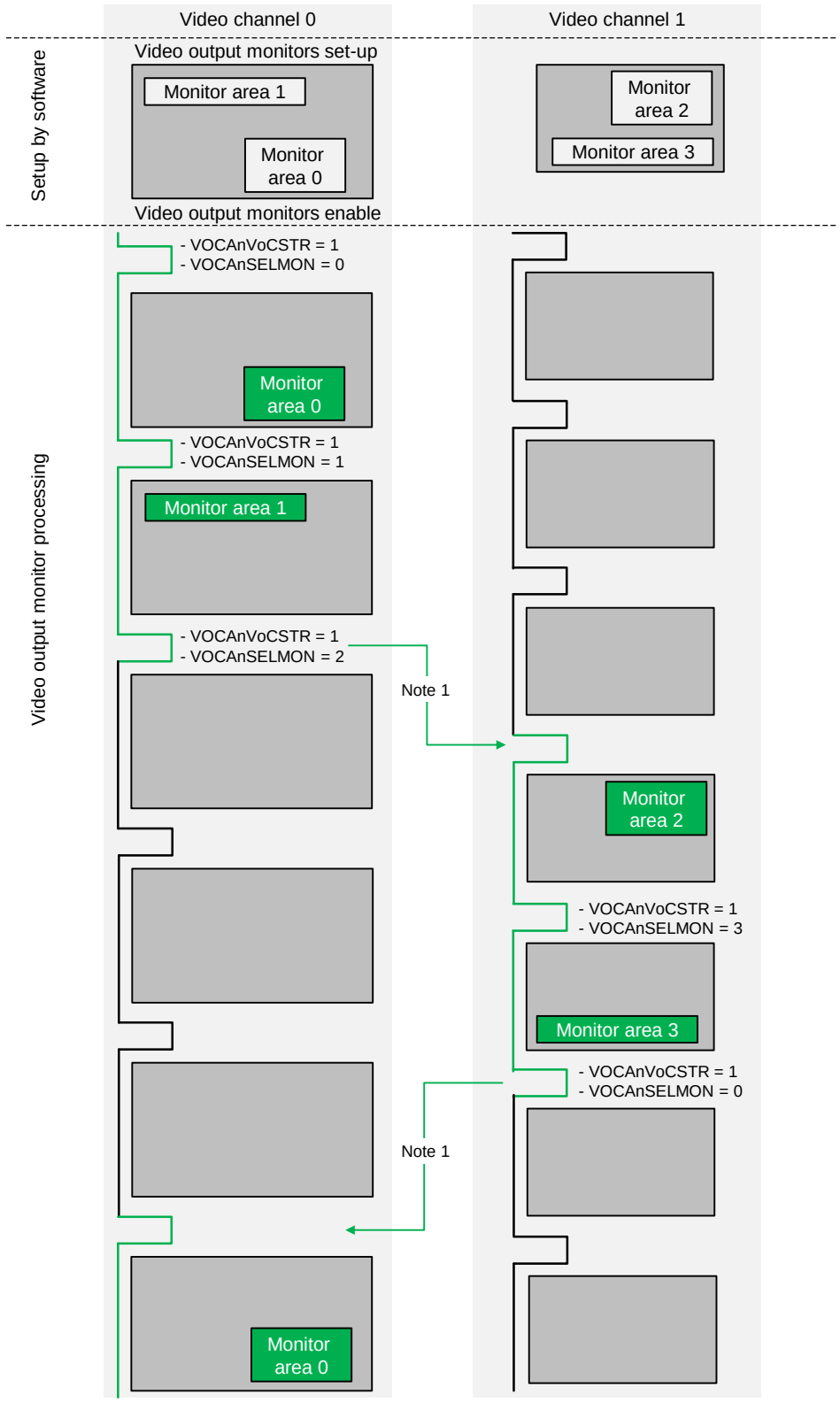
**(7) Monitor Process Status**

The current Video Output Monitor process status is reflected in two status bits in the status register DOC0STR:

- VOCAnVOCSTR: Video Output Monitor active in the current video channel frame
- VOCAnSELMON[3:0]: monitor number for current video channel frame

These status bits are valid for the current video channel frame.

The following diagram shows an example for sequence, with monitor areas for both video channels.




Note: 1. During switching between the video channels the Video Output Monitor status indicator VOCAnVOCSTR becomes 0 meanwhile.

**Figure 65.12 Video Output Monitor processing sequence example**

### 65.3.3.2 Activity Monitors Error Notification

The Activity Monitor notifies an error at the end of the current video channel frame, if the active Video Output Monitor does not complete area m processing within the allowed time window. Upon detection of such error

- the Activity Monitor error bit
  - DOC0STR.VOCAnE16 for video channel 0 error
  - DOC0STR.VOCAnE17 for video channel 1 error is set
- the Activity Monitor error interrupt DOC0.act asserted

**Note**

1. The error flags in the status register DOC0STR remain set until they are cleared via the control register DOC0CTL. Refer to “Activity Monitors error flag clearing procedure” below for details.
2. The DOC0.act remains asserted as long as any of the Activity Monitor error flags VOCAnE16 and VOCAnE17 are set.

**(1) Activity Monitors Error Flag Clearing Procedure**

In order to clear an Activity Monitor error flag proceed as follows:

1. Disable the Activity Monitors by DOC0EN.VOCAnEN16 = 0.
2. Wait until Activity Monitors are inactive, i.e. until DOC0STR.VOCAnACTSTR = 0.
3. Evaluate the error source via the error flags in the status register DOC0STR.
4. Clear all Activity Monitor error flags by DOC0CTL.VOCAnCL[17:16] = 1.
5. Release all Activity Monitor error flags clear by DOC0CTL.VOCAnCL[17:16] = 0.
6. Verify that all Activity Monitor error flags are reset, i.e. that DOC0STR.VOCAnE[17:16] = 0. If any of the above error flags are not cleared, return to 5.
7. Enable the Activity Monitors by DOC0EN.VOCAnEN16 = 1.

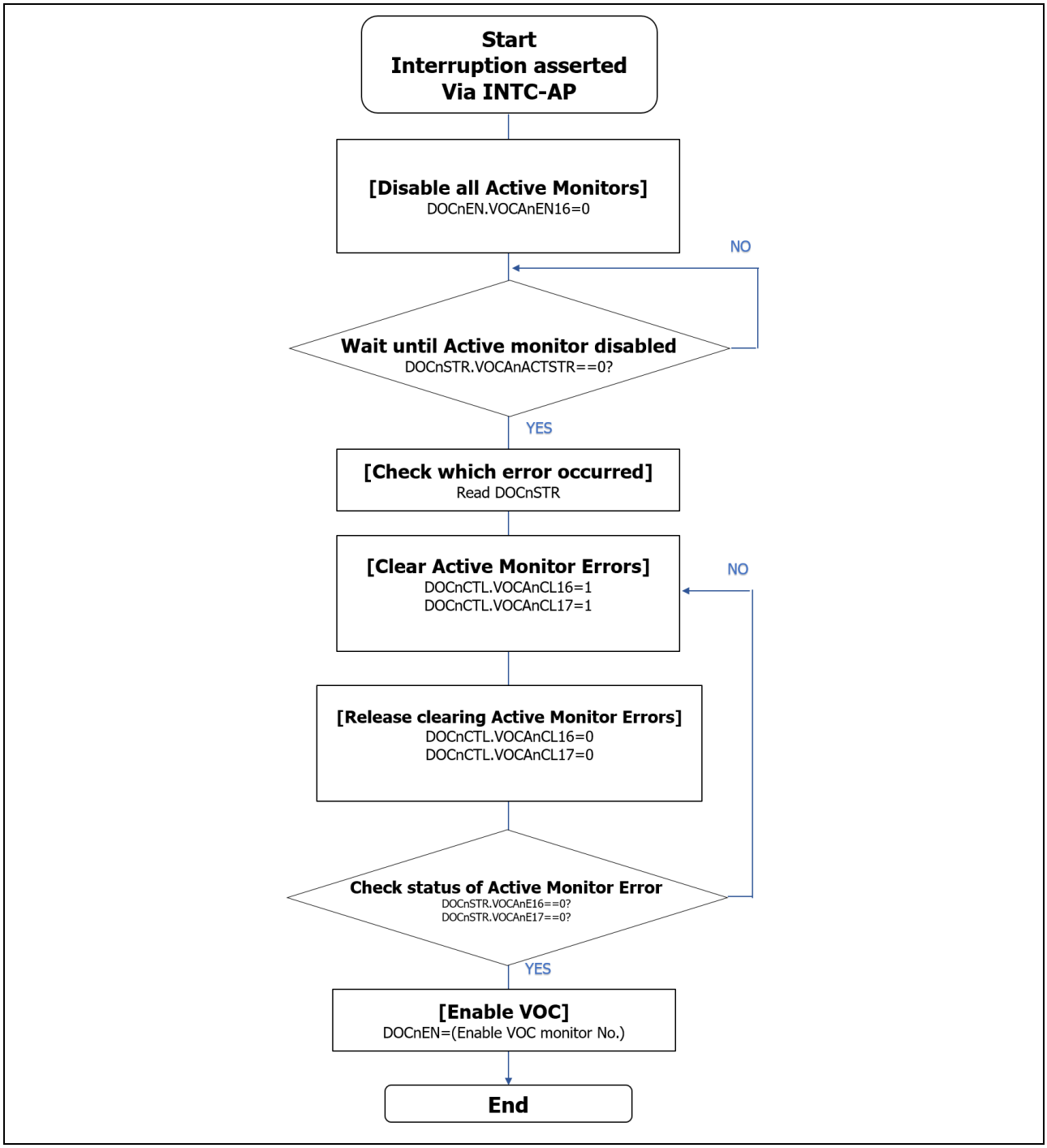



**Figure 65.13 Active Monitor error flag clearing procedure**

### (2) Interrupt Masking

Assertion of an error interrupt DOC0.act can be disabled:

- DOC0CTL.VOCAnMKINT = 0:
  DOC0.act is generated in case of an Activity Monitor error detection
- DOC0CTL.VOCAnMKINT = 1:
  no DOC0.act is generated in case of an Activity Monitor error detection
  Note that DOC0CTL.VOCAnMKINT affects also the Video Output Monitor interrupt DOC0.doc.

Figure 65.14 shows an example with correct detection of the Video Output Monitor completion signal.
Thus no error is reported.

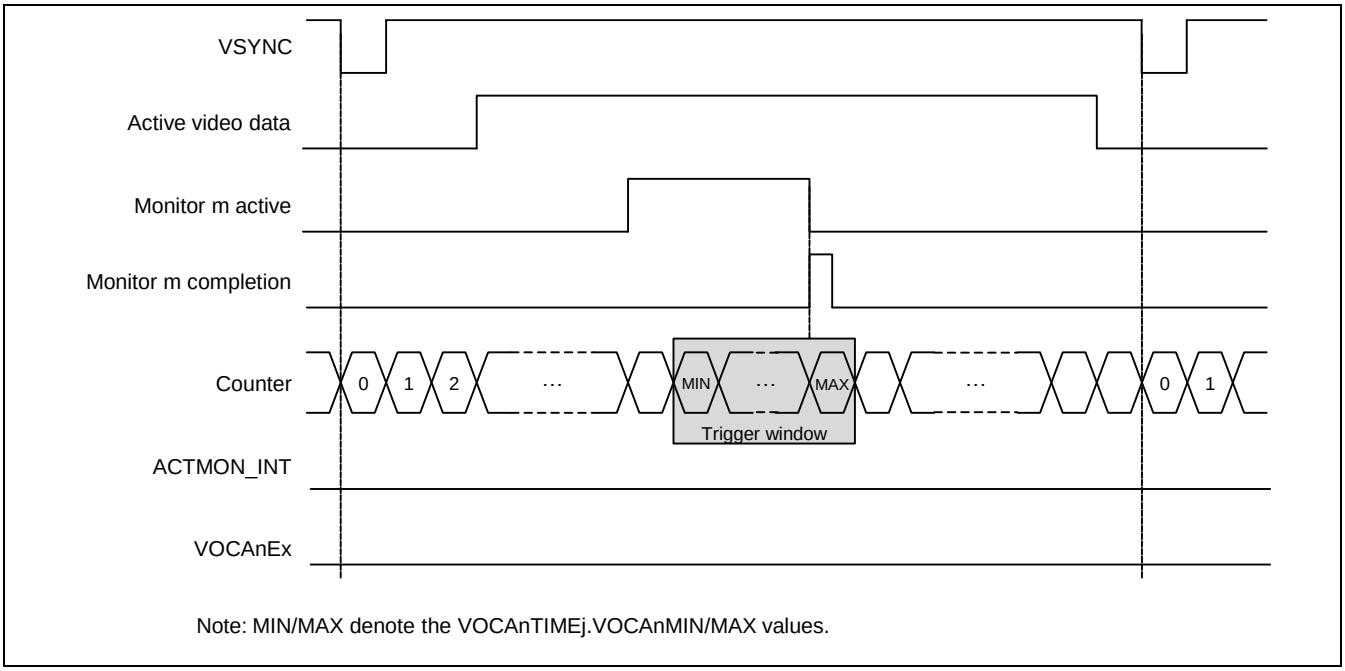



**Figure 65.14 Active Monitor process without error detection**

Figure 65.15 shows an example with detection of an incorrect Video Output Monitor completion signal. The Activity Monitor error flag VOCAn.VOCAnE16/E17 for video channel 0/1 is set and the DOC0.act interrupt is asserted.

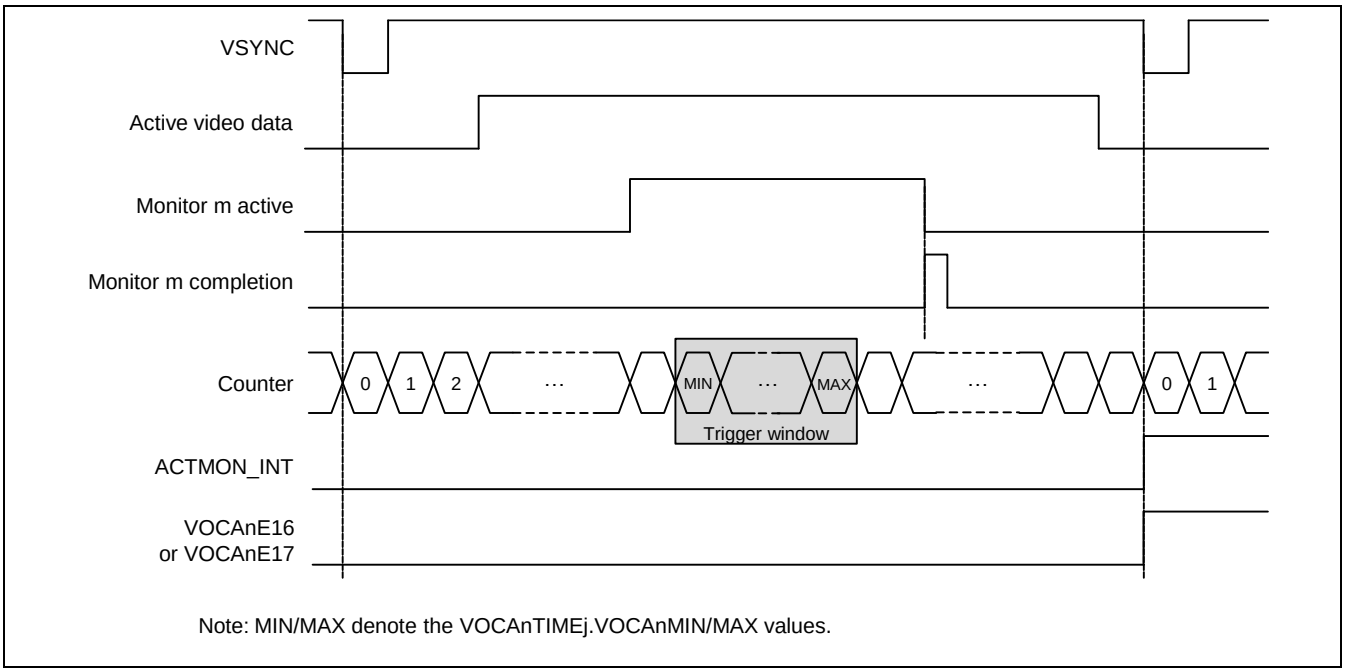



**Figure 65.15 Active Monitor process with error detection**

**(3) Safety Aspects**

By use of the Activity Monitor the operation of several video channel functions are indirectly supervised:

- VSYNC/HSYNC signals of the video channel
- video channel pixel clock
- operation of the Video Output Monitor

If any of the above modules operations fail, the Activity Monitor is able to detect such malfunction and issues error notifications.

## 65.4 Usage Notes

- The DOC has internal control registers which are writable / readable by the APB protocol via the APB Bus. However, accessing the registers is allowed only when the respective Video Monitor m (m = 15 to 1) is disabled (DOC0EN.VOCAnENm = 1’b0).

## 65.5 Safety Mechanism

No description on this section.

# 67. DSI/CSI-2 TX

## 67.1 Overview

This DSI/CSI-2-TX module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The DSI/CSI-2-TX module has a Transmitter function for MIPI Alliance Specification for Display Serial Interface (DSI) and MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2). The DSI/CSI-2-TX module supports MIPI Alliance Specification for Display Serial Interface (DSI) Specification and MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2) Specification, and works with MIPI Alliance Specification for D-PHY Specification. The DSI/CSI-2-TX module provide a solution for transmitting MIPI DSI/CSI-2 compliant digital video and packets.
Normative References are following

Normative References

- MIPI Alliance Specification for Display Serial Interface Version 1.3.1 (Errata01 28-Jun-2017)
- MIPI Alliance Specification for Camera Serial Interface 2 Version 1.3
- MIPI Alliance Specification for D-PHY Version 1.2

Note: the wording about "Host" and "Peripheral" in this document

Mainly, this document is described based on the point of view of a DSI Transmitter.

The meaning of the word "Host" is "a device included DSI Transmitter", and the meaning of the word "Peripheral" is "a device included DSI Receiver (e.g. Display)".

In case of using this Transmitter as CSI-2, the meaning of the word "Host" is "a device included CSI-2 Transmitter (e.g. Camera)", and the meaning of the word "Peripheral" is "a device included CSI-2 Receiver".

### 67.1.1 Features

The following is Key features of the DSI/CSI-2-TX module.

#### 67.1.1.1 DSI

- Supports MIPI DSI Version 1.3.1
  - ➢ Supports MIPI D-PHY 1.5Gbps without de-skew function (for 1.5G)
  - ➢ Supports MIPI D-PHY 2.5Gbps with de-skew function (for 2.5G)

- Supports D-PHY specification version 1.2

- Protocol Interface
  - ➢ Video mode and Packet Generation from Input Video signal through DU (Display Unit).
    - Supports RGB format (16bpp, 18bpp, 18bpp Loosely and 24bpp)
    - Supports YCbCr 4:2:2 format (16bpp)
    - Not supports YCbCr 4:2:0 format
    - Supports Blanking Packet or LP-11 selection during each of HSA, HBP and HFP Blanking interval

  - Not supports Burst transfer mode
  - Not supports Interlaced Video
  - Not supports 3D control payload
- Supports Command Transfer mode which supports Packet Tx/Rx by Register
- Supports automated insertion of EoTp
- Supports ULPS (TX).
- Not supports ULPS(RX)
- Not supports TE

- Link Layer
  - Supports 1,2,3 and 4 lane configuration
  - Supports unidirectional high-speed mode Tx
  - Supports Bidirectional LP mode Tx/Rx (Only Lane0)
  - Supports ECC/Checksum Generation for Tx Packet
  - Supports ECC/Checksum Verification and ECC Error correction for Rx Packet
  - Supports one Virtual Channel number selected 0,1,2 and 3 (Static and specified by Register, Not supports interleave)
  - Not supports Non-Continuous Clock Mode
  - Supports Detection for PHY Contention Error and Timeout Error

- Supports Interleaving: Line/Column/Side-by-Side. Two videos per one DSI port can be possible to output using DU/VSPD. Refer to the USECASE section

#### 67.1.1.2 CSI-2

- Supports MIPI Alliance Specification for Camera Serial Interface 2 version1.3 Specification
  - Supports MIPI D-PHY 1.5Gbps without de-skew function
  - Supports MIPI D-PHY 2.5Gbps with de-skew function

- Supports D-PHY specification version 1.2

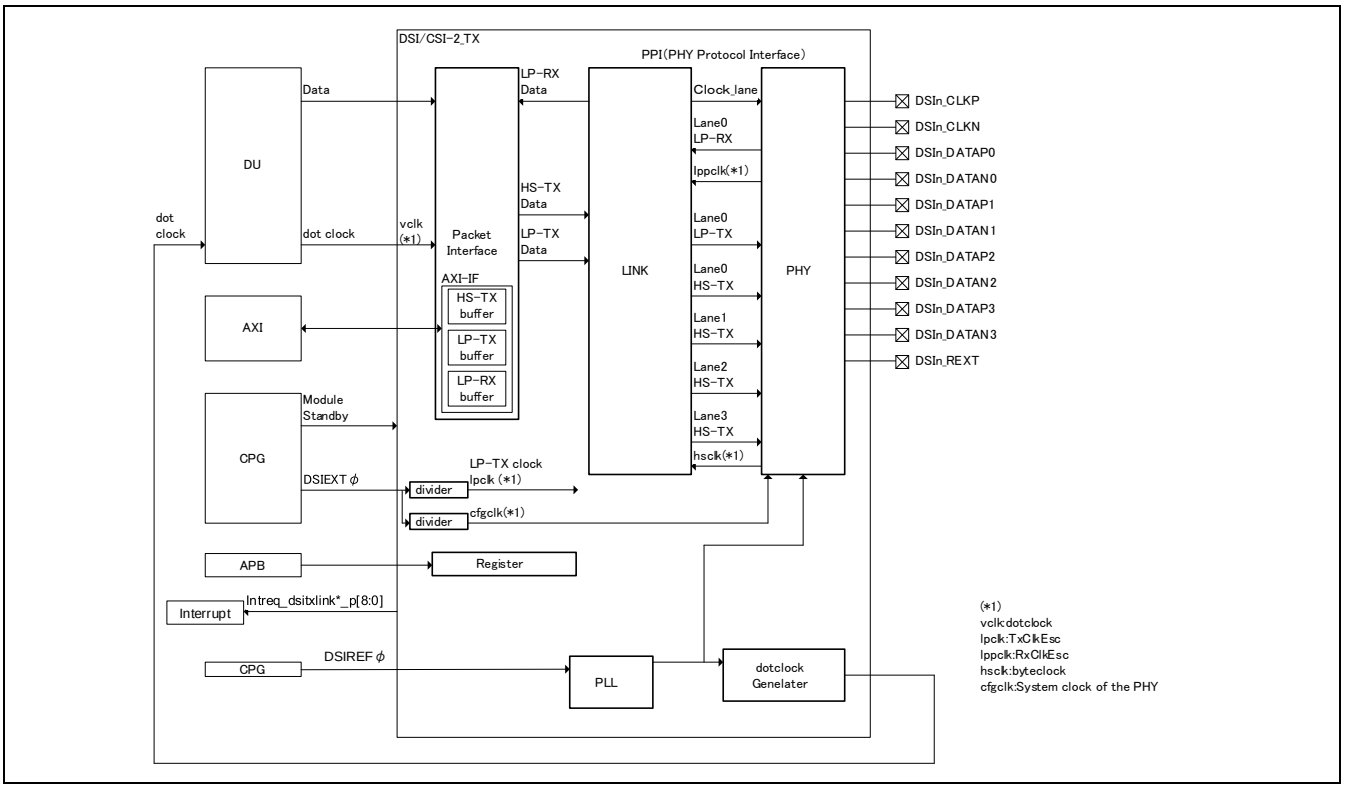
- Protocol Interface
  - Video mode and Packet Generation from Input Video signal through DU (Display Unit).
    - Supports RGB565/666/888
    - Supports YUV422 8bit
    - Not supports YUV420
    - Not supports Interlaced Video
    - Not supports Line Start Packet and Line End Packet
  - Supports Command Transfer mode which supports Packet Tx by Register
  - Supports ULPS

- Link Layer
  - Supports 1,2 and 4 lane configuration
  - Supports unidirectional high-speed mode Tx
  - Supports ECC/Checksum Generation for Tx Packet
  - Supports 4 Virtual Channel number selected from 0,1,2 and 3 (Supports frame level interleave)
  - Not support DT(Data Type) interleave transmission in same VC
  - Not supports Non-Continuous Clock mode

- Supports Interleaving: Line/Column/Side-by-Side. Two videos per one CSI port can be possible to output using DU/VSPD. Refer to the USECASE section.

### 67.1.2 Block Diagram

Figure 67.1 shows the block diagrams of the DSI/CSI-2-TX module.




Note:DSIEXTϕ and DSIREFϕ from CPG are common in each channel.

**Figure 67.1 Block Diagram of DSI/CSI-2-TX module**

Figure 67.2 shows Configuration of DSI/CSI-2-TX module.

The transfer of Video stream is only through VSPD/DU.

The transmitter of AXI Bus supports Tx/Rx when only Command Transfer mode.

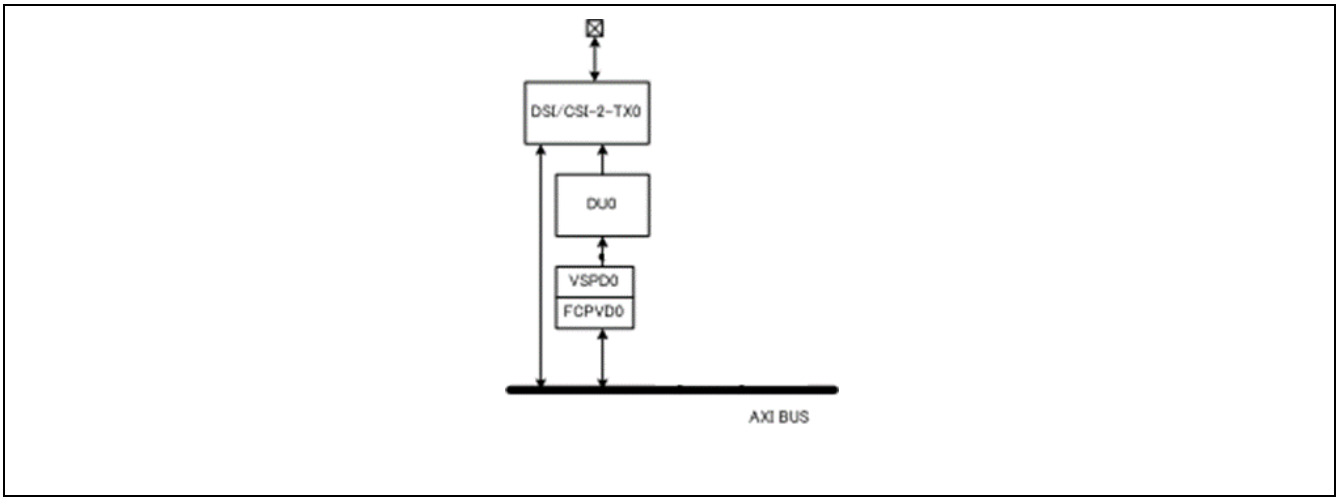



**Figure 67.2 Configuration of DSI/CSI-2-TX module**

### 67.1.3 External Pins

**Table 67.1 External Pins**

| Pin Name | Abbreviation | I/O | Function |
|---|---|---|---|
| DSI0 clock lane positive data pin | DSI0_CLKP | output | Positive D-PHY Differential Clock Line |
| DSI0 clock lane negative data pin | DSI0_CLKN | output | Negative D-PHY Differential Clock Line |
| DSI0 lane 0 positive data pin | DSI0_DATAP0 | I/O | Positive D-PHY Differential Data Line0 |
| DSI0 lane 0 negative data pin | DSI0_DATAN0 | I/O | Negative D-PHY Differential Data Line0 |
| DSI0 lane 1 positive data pin | DSI0_DATAP1 | output | Positive D-PHY Differential Data Line1 |
| DSI0 lane 1 negative data pin | DSI0_DATAN1 | output | Negative D-PHY Differential Data Line1 |
| DSI0 lane 2 positive data pin | DSI0_DATAP2 | output | Positive D-PHY Differential Data Line2 |
| DSI0 lane 2 negative data pin | DSI0_DATAN2 | output | Negative D-PHY Differential Data Line2 |
| DSI0 lane 3 positive data pin | DSI0_DATAP3 | output | Positive D-PHY Differential Data Line3 |
| DSI0 lane 3 negative data pin | DSI0_DATAN3 | output | Negative D-PHY Differential Data Line3 |
| DSI0 external resistor connection pin | DSI0_REXT | I/O | 200 Ω External Resistor Connection for DSI/CSI-2-TX0 |

Notes: Immediately after reset assertion, the differential lines of datan/datap and clkn/clkp are high impedance (Hi-Z).

### 67.1.4 Register Configuration

The following lists the base address of each DSI/CSI-2-TX module.

| Module name | Address |
|---|---|
| DSI0 | H'FED8_0000 |

Table 67.2 shows Registers of the DSI/CSI-2-TX module. These Registers are fixed by 32 bits length. Unaligned access is prohibited.

**Table 67.2 Registers**

| Register Name | Abbr | Offset Address | Initial Value |
|---|---|---|---|
| Interrupt Status Register | ISR | H'000 | H'0000 0000 |
| Link Status Register | LINKSR | H'010 | H'0000 0000 |

| Register Name | Abbr | Offset Address | Initial Value |
|---|---|---|---|
| Tx Set Register | TXSETR | H'100 | H'0F0F 1003 |
| Tx Command transfer Mode Set Register | TXCMSETR | H'110 | H'0000 0000 |
| Tx Command transfer Mode Control Register | TXCMCR | H'120 | H'0000 0000 |
| Tx Command transfer Mode Status Register | TXCMSR | H'130 | H'0000 0000 |
| Tx Command transfer Mode Status Clear Register | TXCMSCR | H'134 | H'0000 0000 |
| Tx Command transfer Mode Interrupt Enable Register | TXCMIER | H'138 | H'0000 0000 |
| Tx Command transfer Mode Address Set 0 Register | TXCMADDRSET0R | H'140 | H'0000 0000 |
| Tx Command transfer Mode Header Data Register | TXCMPHDR | H'150 | H'0000 0000 |
| Tx Command transfer Mode Packet Payload Data0 Register | TXCMPPD0R | H'160 | H'0000 0000 |
| Tx Command transfer Mode Packet Payload Data1 Register | TXCMPPD1R | H'164 | H'0000 0000 |
| Tx Command transfer Mode Packet Payload Data2 Register | TXCMPPD2R | H'168 | H'0000 0000 |
| Tx Command transfer Mode Packet Payload Data3 Register | TXCMPPD3R | H'16C | H'0000 0000 |
| Tx Video Mode Set Register | TXVMSETR | H'180 | H'0000 0100 |
| Tx Video Mode Set 2 Register | TXVMSET2R | H'184 | H'0000 0000 |
| Tx Video Mode Control Register | TXVMCR | H'190 | H'0000 1000 |
| Tx Video Mode Status Register | TXVMSR | H'1A0 | H'0000 0000 |
| Tx Video Mode Status Clear Register | TXVMSCR | H'1A4 | H'0000 0000 |
| Tx Video Mode Interrupt Enable Register | TXVMIER | H'1A8 | H'0000 0000 |
| Tx Video Mode Pixel Stream Packet Header Set Register | TXVMPSPHSETR | H'1C0 | H'0000 0000 |
| Tx Video Mode Parameter Set 0 Register | TXVMVPRMSET0R | H'1D0 | H'0000 0000 |
| Tx Video Mode Parameter Set 1 Register | TXVMVPRMSET1R | H'1D4 | H'0000 0000 |
| Tx Video Mode Parameter Set 2 Register | TXVMVPRMSET2R | H'1D8 | H'0000 0000 |
| Tx Video Mode Parameter Set 3 Register | TXVMVPRMSET3R | H'1DC | H'0000 0000 |
| Tx Video Mode Parameter Set 4 Register | TXVMVPRMSET4R | H'1E0 | H'0000 0000 |
| Rx Set Register | RXSETR | H'200 | H'0F0F 0000 |
| Rx Packet Set Register | RXPSETR | H'210 | H'0000 0000 |
| Rx Packet Status Register | RXPSR | H'220 | H'0000 0000 |
| Rx Packet Status Clear Register | RXPSCR | H'224 | H'0000 0000 |
| Rx Packet Interrupt Enable Register | RXPIER | H'228 | H'0000 0000 |
| Rx Packet Address Set 0 Register | RXPADDRSET0R | H'230 | H'0000 0000 |
| Rx Packet Size Set Register | RXPSIZESETR | H'238 | H'0000 0000 |
| Rx Packet Header Data Register | RXPHDR | H'240 | H'0000 0000 |
| Rx Packet Payload Data0 Register | RXPPD0R | H'250 | H'0000 0000 |
| Rx Packet Payload Data1 Register | RXPPD1R | H'254 | H'0000 0000 |
| Rx Packet Payload Data2 Register | RXPPD2R | H'258 | H'0000 0000 |
| Rx Packet Payload Data3 Register | RXPPD3R | H'25C | H'0000 0000 |
| Rx Ack and Error Packet parameter Register | AKEPR | H'300 | H'0000 0000 |
| Rx Response Timeout Set Register | RXRESPTOSETR | H'400 | H'0000 0000 |
| Turn-Around Control Register | TACR | H'500 | H'0000 0000 |
| Turn-Around Status Register | TASR | H'510 | H'0000 0000 |

| Register Name | Abbr | Offset Address | Initial Value |
|---|---|---|---|
| Turn-Around Status Clear Register | TASCR | H'514 | H'0000 0000 |
| Turn-Around Interrupt Enable Register | TAIER | H'518 | H'0000 0000 |
| Timeout Set 0 Register | TOSET0R | H'600 | H'0000 0000 |
| Timeout Set 1 Register | TOSET1R | H'604 | H'0000 0000 |
| Timeout Set 2 Register | TOSET2R | H'608 | H'0000 0000 |
| Timeout Status Register | TOSR | H'610 | H'0000 0000 |
| Timeout Status Clear Register | TOSCR | H'614 | H'0000 0000 |
| Timeout Interrupt Enable Register | TOIER | H'618 | H'0000 0000 |
| PPI Set Register | PPISETR | H'700 | H'0000 0000 |
| PPI Clock Lane Control Register | PPICLCR | H'710 | H'0000 0000 |
| PPI Clock Lane Status Register | PPICLSR | H'720 | (*1) |
| PPI Clock Lane Status Clear Register | PPICLSCR | H'724 | H'0000 0000 |
| PPI Clock Lane Interrupt Enable Register | PPICLIER | H'728 | H'0000 0000 |
| PPI Data Lane0 Control Register | PPIDL0CR | H'730 | H'0000 0000 |
| PPI Data Lane0 Status Register | PPIDL0SR | H'740 | (*1) |
| PPI Data Lane0 Status Clear Register | PPIDL0SCR | H'744 | H'0000 0000 |
| PPI Data Lane0 Interrupt Enable Register | PPIDL0IER | H'748 | H'0000 0000 |
| PPI Data Lane Control Register | PPIDLCR | H'750 | H'0000 0000 |
| PPI Data Lane Status Register | PPIDLSR | H'760 | (*1) |
| PPI Data Lane Status Clear Register | PPIDLSCR | H'764 | H'0000 0000 |
| PPI Data Lane Interrupt Enable Register | PPIDLIER | H'768 | H'0000 0000 |
| Skew Calibration Control Register | SCALCR | H'900 | H'0000 0000 |
| Skew Calibration Status Register | SCALSR | H'910 | H'0000 0000 |
| Skew Calibration Status Clear Register | SCALSCR | H'914 | H'0000 0000 |
| Skew Calibration Interrupt Enable Register | SCALIER | H'918 | H'0000 0000 |
| Skew Calibration Pattern Size Register | SCALPSR | H'920 | H'0000 0000 |
| Skew Calibration Pattern Size 1 Register | SCALPS1R | H'924 | H'0000 0000 |
| IP Set Register | IPSETR | H'A00 | H'0000 0000 |
| Tx Video Mode Frame Type Control Register | TXVMFTCR | H'A80 | H'0000 0000 |
| Tx Video Mode Set (Frame Type 1) Register | TXVMSETF1R | H'AA0 | H'0000 0000 |
| Tx Video Mode Set 2 (Frame Type 1) Register | TXVMSET2F1R | H'AA4 | H'0000 0000 |
| Tx Video Mode Pixel Stream Packet Header Set (Frame Type 1) Register | TXVMPSPHSETF1R | H'AA8 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 0 (Frame Type 1) Register | TXVMVPRMSET0F1R | H'AAC | H'0000 0000 |
| Tx Video Mode Video Parameter Set 1 (Frame Type 1) Register | TXVMVPRMSET1F1R | H'AB0 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 2 (Frame Type 1) Register | TXVMVPRMSET2F1R | H'AB4 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 3 (Frame Type 1) Register | TXVMVPRMSET3F1R | H'AB8 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 4 (Frame Type 1) Register | TXVMVPRMSET4F1R | H'ABC | H'0000 0000 |
| Tx Video Mode Set (Frame Type 2) Register | TXVMSETF2R | H'AC0 | H'0000 0000 |
| Tx Video Mode Set 2 (Frame Type 2) Register | TXVMSET2F2R | H'AC4 | H'0000 0000 |

| Register Name | Abbr | Offset Address | Initial Value |
|---|---|---|---|
| Tx Video Mode Pixel Stream Packet Header Set (Frame Type 2) Register | TXVMPSPHSETF2R | H'AC8 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 0 (Frame Type 2) Register | TXVMPSPHSET0F2R | H'ACC | H'0000 0000 |
| Tx Video Mode Video Parameter Set 1 (Frame Type 2) Register | TXVMPSPHSET1F2R | H'AD0 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 2 (Frame Type 2) Register | TXVMPSPHSET2F2R | H'AD4 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 3 (Frame Type 2) Register | TXVMPSPHSET3F2R | H'AD8 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 4 (Frame Type 2) Register | TXVMPSPHSET4F2R | H'ADC | H'0000 0000 |
| Tx Video Mode Set (Frame Type 3) Register | TXVMSETF3R | H'AE0 | H'0000 0000 |
| Tx Video Mode Set 2 (Frame Type 3) Register | TXVMSET2F3R | H'AE4 | H'0000 0000 |
| Tx Video Mode Pixel Stream Packet Header Set (Frame Type 3) Register | TXVMPSPHSETF3R | H'AE8 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 0 (Frame Type 3) Register | TXVMPSPHSET0F3R | H'AEC | H'0000 0000 |
| Tx Video Mode Video Parameter Set 1 (Frame Type 3) Register | TXVMPSPHSET1F3R | H'AF0 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 2 (Frame Type 3) Register | TXVMPSPHSET2F3R | H'AF4 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 3 (Frame Type 3) Register | TXVMPSPHSET3F3R | H'AF8 | H'0000 0000 |
| Tx Video Mode Video Parameter Set 4 (Frame Type 3) Register | TXVMPSPHSET4F3R | H'AFC | H'0000 0000 |
| Fault detection Set Register | FDSETR | H'B00 | H'0000 0000 |
| Fault detection Control Register | FDCR | H'B10 | H'0000 0000 |
| Fault detection Status Register | FDSR | H'B20 | H'0000 0000 |
| Fault detection Status Clear Register | FDSCR | H'B24 | H'0000 0000 |
| Fault detection Interrupt Enable Register | FDIER | H'B28 | H'0000 0000 |
| Fault detection PPI Timer Set Register | FDPPITMSETR | H'B30 | H'0000 0000 |
| Fault detection Data Type Timer Set Register | FDDTTMSETR | H'B40 | H'0000 0000 |
| Fault detection Data Type Timer Enable0 Register | FDDTEN0R | H'B44 | H'0000 0000 |
| Fault detection Data Type Timer Enable1 Register | FDDTEN1R | H'B48 | H'0000 0000 |
| Fault detection Control Register 2 | FDCR2 | H'1100 | H'0000 0000 |
| Fault detection Status Register 2 | FDSR2 | H'1104 | H'0000 0000 |
| Fault detection Status Clear Register 2 | FDSCR2 | H'1108 | H'0000 0000 |
| Fault detection Interrupt Enable Register 2 | FDIER2 | H'110C | H'0000 0000 |
| lpclk frequency Set Register | LPCLKSET | H'1000 | H'0000 0014 |
| cfgclk frequency Set Register | CFGCLKSET | H'1004 | H'0000 001F |
| vclk output Set Register | VCLKSET | H'100C | H'0000 000F |
| vclk output Enable Register | VCLKEN | H'1010 | H'0000 0000 |
| PHY SETUP Register | PHYSETUP | H'1014 | H'0000 2000 |
| Clock Set 1 Register | CLOCKSET1 | H'101C | H'2000 0100 |
| Clock Set 2 Register | CLOCKSET2 | H'1020 | H'0000 0000 |
| Clock Set 3 Register | CLOCKSET3 | H'1024 | H'0000 0000 |

| Register Name | Abbr | Offset Address | Initial Value |
|---|---|---|---|
| PHY Test Interface Write Register | PHTW | H'1034 | H'0000 0000 |
| PHY Test Interface Read Register | PHTR | H'1038 | H'0000 0000 |
| PHY Test Interface Clear Register | PHTC | H'103C | H'0000 0001 |

All access except for the address listed above is prohibited, and these addresses are Reserved.

Note: *1 Initial value is depended on input signal. Refer to 67.2.2 Detail of Registers for detail.

(include Safety Mechanism related registers)

### 67.1.5 Connected Module

## 67.2 Register Description

(include Safety Mechanism related register description)

### 67.2.1 Register Access

The access propriety of Registers and Fields on each mode is shown on following table.

○: Full Access

△: Limited access. Refer to 67.2.2 Detail of Registers for detail.

×: Not permitted to access. When it needs to write to these registers, write the initial value.

| Register | Field | DSI mode | CSI-2 mode |
|---|---|---|---|
| ISR | | | |
| | SCAL | ○ | ○ |
| | PPIDL | ○ | ○ |
| | PPIDL0 | ○ | ○ |
| | PPICL | ○ | ○ |
| | TO | ○ | ○ |
| | TA | ○ | × |
| | RXP | ○ | × |
| | TXVM | ○ | ○ |
| | TXCM | ○ | ○ |
| LINKSR | LPBUSY | ○ | × |
| | HSBUSY | ○ | ○ |
| TXSETR | | | |
| | EOTPEN | ○ | × |
| | LANECNT | ○ | △ |
| TXCMSETR | SPDTYP | ○ | × |
| | LPPDACC | ○ | ○ |
| TXCMCR | BTATYP | ○ | × |
| | BTAREQ | ○ | × |
| | TXREQ | ○ | ○ |
| TXCMSR | CLSNERR | ○ | ○ |
| | AXIERR | ○ | ○ |
| | TXREQEND | ○ | ○ |
| TXCMSCR | CLSNERR | ○ | ○ |
| | AXIERR | ○ | ○ |
| | TXREQEND | ○ | ○ |
| TXCMIER | CLSNERR | ○ | ○ |
| | AXIERR | ○ | ○ |
| | TXREQEND | ○ | ○ |
| TXCMADDRSET0R | ADDRL | ○ | ○ |
| TXCMPHDR | FMT | ○ | ○ |

| Register | Field | DSI mode | CSI-2 mode |
|---|---|---|---|
| | VC | ○ | ○ |
| | DT | ○ | ○ |
| | DATA1 | ○ | ○ |
| | DATA0 | ○ | ○ |
| TXCMPPD0R | DATA | ○ | ○ |
| TXCMPPD1R | DATA | ○ | ○ |
| TXCMPPD2R | DATA | ○ | ○ |
| TXCMPPD3R | DATA | ○ | ○ |
| TXVMSETR | | | |
| | SYNSEQ | ○ | × |
| | VSTPM | ○ | ○ |
| | PIXWDTH | ○ | ○ |
| | VSEN | ○ | ○ |
| | HFPBPEN | ○ | × |
| | HBPBPEN | ○ | × |
| | HSABPEN | ○ | × |
| TXVMSET2R | FNUM | × | ○ |
| TXVMCR | VFCLR | ○ | ○ |
| | EN | ○ | ○ |
| TXVMSR | | | |
| | VBPDET | ○ | ○ |
| | VSADET | ○ | ○ |
| | VFERR | ○ | ○ |
| | PDIS | ○ | ○ |
| | STP | ○ | ○ |
| | STR | ○ | ○ |
| | VFRDY | ○ | ○ |
| | ACT | ○ | ○ |
| | RDY | ○ | ○ |
| TXVMSCR | | | |
| | VBPDET | ○ | ○ |
| | VSADET | ○ | ○ |
| | VFERR | ○ | ○ |
| | PDIS | ○ | ○ |
| | STP | ○ | ○ |
| | STR | ○ | ○ |
| TXVMIER | | | |
| | VBPDET | ○ | ○ |
| | VSADET | ○ | ○ |
| | VFERR | ○ | ○ |
| | PDIS | ○ | ○ |
| | STP | ○ | ○ |
| | STR | ○ | ○ |
| TXVMPSPHSETR | VC | ○ | ○ |

| Register | Field | DSI mode | CSI-2 mode |
|---|---|---|---|
| | DT | ○ | ○ |
| TXVMVPRMSET0R | HSPOL | ○ | ○ |
| | VSPOL | ○ | ○ |
| | CSPC | ○ | ○ |
| | BPP | ○ | ○ |
| TXVMVPRMSET1R | VACTIVE | ○ | ○ |
| | VSAVW | × | ○ |
| | VSA | ○ | ○ |
| TXVMVPRMSET2R | VFPVW | × | ○ |
| | VFP | ○ | ○ |
| | VBPVW | × | ○ |
| | VBP | ○ | ○ |
| TXVMVPRMSET3R | HACTIVE | ○ | ○ |
| | HSBYTE | ○ | ○ |
| | HSA | ○ | ○ |
| TXVMVPRMSET4R | HFP | ○ | ○ |
| | HBP | ○ | ○ |
| RXSETR | | | |
| | CRCEN | ○ | × |
| | ECCEN | ○ | × |
| RXPSETR | LPPDACC | ○ | × |
| RXPSR | ECCERR1B | ○ | × |
| | RESPTOERR | ○ | × |
| | OVRERR | ○ | × |
| | AXIERR | ○ | × |
| | CRCERR | ○ | × |
| | WCERR | ○ | × |
| | UEXDTERR | ○ | × |
| | UEXPKTERR | ○ | × |
| | ECCERR | ○ | × |
| | MLFERR | ○ | × |
| | RCVACK | ○ | × |
| | RCVEOT | ○ | × |
| | RCVAKE | ○ | × |
| | RCVRESP | ○ | × |
| | UEXTRGERR | ○ | × |
| | BTAREQEND | ○ | × |
| RXPSCR | ECCERR1B | ○ | × |
| | RESPTOERR | ○ | × |
| | OVRERR | ○ | × |
| | AXIERR | ○ | × |
| | CRCERR | ○ | × |
| | WCERR | ○ | × |
| | UEXDTERR | ○ | × |

| Register | Field | DSI mode | CSI-2 mode |
|---|---|---|---|
| | UEXPKTERR | ○ | × |
| | ECCERR | ○ | × |
| | MLFERR | ○ | × |
| | RCVACK | ○ | × |
| | RCVEOT | ○ | × |
| | RCVAKE | ○ | × |
| | RCVRESP | ○ | × |
| | UEXTRGERR | ○ | × |
| | BTAREQEND | ○ | × |
| RXPIER | ECCERR1B | ○ | × |
| | RESPTOERR | ○ | × |
| | OVRERR | ○ | × |
| | AXIERR | ○ | × |
| | CRCERR | ○ | × |
| | WCERR | ○ | × |
| | UEXDTERR | ○ | × |
| | UEXPKTERR | ○ | × |
| | ECCERR | ○ | × |
| | MLFERR | ○ | × |
| | RCVACK | ○ | × |
| | RCVEOT | ○ | × |
| | RCVAKE | ○ | × |
| | RCVRESP | ○ | × |
| | UEXTRGERR | ○ | × |
| | BTAREQEND | ○ | × |
| RXPADDRSET0R | ADDRL | ○ | × |
| RXPSIZESETR | SIZE | ○ | × |
| RXPHDR | FMT | ○ | × |
| | VC | ○ | × |
| | DT | ○ | × |
| | DATA1 | ○ | × |
| | DATA0 | ○ | × |
| RXPPD0R | DATA | ○ | × |
| RXPPD1R | DATA | ○ | × |
| RXPPD2R | DATA | ○ | × |
| RXPPD3R | DATA | ○ | × |
| AKEPR | VC | ○ | × |
| | DT | ○ | × |
| | ERRRPT | ○ | × |
| RXRESPTOSETR | RESPTOVL | ○ | × |
| TACR | TRNREQ | ○ | × |
| TASR | ECCERR1B | ○ | × |
| | UEXDTERR | ○ | × |
| | UEXPKTERR | ○ | × |

| Register | Field | DSI mode | CSI-2 mode |
|---|---|---|---|
| | ECCERR | ○ | × |
| | MLFERR | ○ | × |
| | RCVACK | ○ | × |
| | RCVEOT | ○ | × |
| | RCVAKE | ○ | × |
| | TRNREQEND | ○ | × |
| TASCR | ECCERR1B | ○ | × |
| | UEXDTERR | ○ | × |
| | UEXPKTERR | ○ | × |
| | ECCERR | ○ | × |
| | MLFERR | ○ | × |
| | RCVACK | ○ | × |
| | RCVEOT | ○ | × |
| | RCVAKE | ○ | × |
| | TRNREQEND | ○ | × |
| TAIER | ECCERR1B | ○ | × |
| | UEXDTERR | ○ | × |
| | UEXPKTERR | ○ | × |
| | ECCERR | ○ | × |
| | MLFERR | ○ | × |
| | RCVACK | ○ | × |
| | RCVEOT | ○ | × |
| | RCVAKE | ○ | × |
| | TRNREQEND | ○ | × |
| TOSET0R | HTXTOVL | ○ | ○ |
| TOSET1R | LRXHTOVL | ○ | × |
| TOSET2R | TATOVL | ○ | × |
| TOSR | TATO | ○ | × |
| | LRXHTO | ○ | × |
| | HTXTO | ○ | ○ |
| TOSCR | TATO | ○ | × |
| | LRXHTO | ○ | × |
| | HTXTO | ○ | ○ |
| TOIER | TATO | ○ | × |
| | LRXHTO | ○ | × |
| | HTXTO | ○ | ○ |
| PPISETR | ULPSWKUP | ○ | ○ |
| | DL0TRNDIS | ○ | ○ |
| | CLEN | ○ | ○ |
| | DLEN | ○ | ○ |
| PPICLCR | TXREQHS | ○ | ○ |
| | TXULPSEXT | ○ | ○ |
| | TXULPSCLK | ○ | ○ |
| PPICLSR | HSTOLP | ○ | ○ |

| Register | Field | DSI mode | CSI-2 mode |
|---|---|---|---|
| | TOHS | ○ | ○ |
| | FROMULPS | ○ | ○ |
| | TOULPS | ○ | ○ |
| | ULPSACTN | ○ | ○ |
| | STPST | ○ | ○ |
| PPICLSCR | HSTOLP | ○ | ○ |
| | TOHS | ○ | ○ |
| | FROMULPS | ○ | ○ |
| | TOULPS | ○ | ○ |
| PPICLIER | HSTOLP | ○ | ○ |
| | TOHS | ○ | ○ |
| | FROMULPS | ○ | ○ |
| | TOULPS | ○ | ○ |
| PPIDL0CR | FCERX | ○ | × |
| | TXTRGESC | ○ | × |
| PPIDL0SR | ERRCLP1 | ○ | ○ |
| | ERRCLP0 | ○ | ○ |
| | ERRCTRL | ○ | ○ |
| | ERRSYNESC | ○ | ○ |
| | ERRESC | ○ | ○ |
| | DIR | ○ | ○ |
| | TX2RX | ○ | × |
| | RX2TX | ○ | × |
| | ULPSACTN | ○ | ○ |
| | STPST | ○ | ○ |
| | RXULPSESC | ○ | × |
| | RXLPDTESC | ○ | × |
| | RXTRGESC | ○ | × |
| PPIDL0SCR | ERRCLP1 | ○ | ○ |
| | ERRCLP0 | ○ | ○ |
| | ERRCTRL | ○ | ○ |
| | ERRSYNESC | ○ | ○ |
| | ERRESC | ○ | ○ |
| | TX2RX | ○ | × |
| | RX2TX | ○ | × |
| | RXTRGESC | ○ | × |
| PPIDL0IER | ERRCLP1 | ○ | ○ |
| | ERRCLP0 | ○ | ○ |
| | ERRCTRL | ○ | ○ |
| | ERRSYNESC | ○ | ○ |
| | ERRESC | ○ | ○ |
| | TX2RX | ○ | × |
| | RX2TX | ○ | × |
| | RXTRGESC | ○ | × |

| Register | Field | DSI mode | CSI-2 mode |
|---|---|---|---|
| PPIDLCR | | | |
| | TXULPSEXT | ○ | ○ |
| | TXULPSESC | ○ | ○ |
| PPIDLSR | FROMULPS | ○ | ○ |
| | TOULPS | ○ | ○ |
| | ULPSACTN | ○ | ○ |
| | STPST | ○ | ○ |
| PPIDLSCR | FROMULPS | ○ | ○ |
| | TOULPS | ○ | ○ |
| PPIDLIER | FROMULPS | ○ | ○ |
| | TOULPS | ○ | ○ |
| SCALCR | CALTYP | ○ | ○ |
| | CALREQ | ○ | ○ |
| SCALSR | CALREQEND | ○ | ○ |
| SCALSCR | CALREQEND | ○ | ○ |
| SCALIER | CALREQEND | ○ | ○ |
| SCALPSR | PTNSZ | ○ | ○ |
| SCALPS1R | PTNSZ | ○ | ○ |
| IPSETR | CSI2M | ○ | ○ |
| TXVMFTCR | FTCUR | × | ○ |
| | FTNXT | × | ○ |
| TXVMSETF(i)R (i=1～3) | | | |
| | VSEN | × | ○ |
| TXVMSET2F(i)R (i=1～3) | FNUM | × | ○ |
| TXVMPSPHSETF(i)R (i=1～3) | VC | × | ○ |
| | DT | × | ○ |
| TXVMVPRMSET0F(i)R (i=1～3) | CSPC | × | ○ |
| TXVMVPRMSET1F(i)R (i=1～3) | VACTIVE | × | ○ |
| | VSAVW | × | ○ |
| | VSA | × | ○ |
| TXVMVPRMSET2F(i)R (i=1～3) | VFPVW | × | ○ |
| | VFP | × | ○ |
| | VBPVW | × | ○ |
| | VBP | × | ○ |
| TXVMVPRMSET3F(i)R (i=1～3) | HACTIVE | × | ○ |
| | HSBYTE | × | ○ |
| | HSA | × | ○ |
| TXVMVPRMSET4F(i)R (i=1～3) | HFP | × | ○ |
| | HBP | × | ○ |
| FDSETR | VFCENF3 | × | ○ |

| Register | Field | DSI mode | CSI-2 mode |
|---|---|---|---|
| | VFCENF2 | × | ○ |
| | VFCENF1 | × | ○ |
| | VFCEN | ○ | ○ |
| FDCR | DTTEN | ○ | ○ |
| | PPITEN | ○ | ○ |
| FDSR | VINERR | ○ | ○ |
| | DTTO | ○ | ○ |
| | PPITO | ○ | ○ |
| FDSCR | VINERR | ○ | ○ |
| | DTTO | ○ | ○ |
| | PPITO | ○ | ○ |
| FDIER | VINERR | ○ | ○ |
| | DTTO | ○ | ○ |
| | PPITO | ○ | ○ |
| FDPPITMSETR | PPITCSCL | ○ | ○ |
| | PPITCVL | ○ | ○ |
| | SPDTYP | ○ | × |
| | PPITOSCL | ○ | ○ |
| | PPITOVL | ○ | ○ |
| FDDTTMSETR | SPDTYP | ○ | × |
| | DTTOSCL | ○ | ○ |
| | DTTOVL | ○ | ○ |
| FDDTEN0R | DTENL | ○ | ○ |
| FDDTEN1R | DTENH | ○ | ○ |
| FDCR2 | EDCEN | ○ | ○ |
| FDSR2 | EDCERR | ○ | ○ |
| FDSCR2 | EDCERR | ○ | ○ |
| FDIER2 | EDCERR | ○ | ○ |

Notes: Other Registers listed above are the Common Register related to PHY configuration.

### 67.2.2 Detail of Registers

Legend for Register Description

Initial value: Register value after a reset.

—: Undefined value

R/W: Readable/writable. The written value can be read.

R0/W: Readable/writable. The read value always is 0.

Note: Do not access locations that appear empty in the address map. Operation is not guaranteed in case of access to change the value at any location other than those of the registers listed.

#### 67.2.2.1 Interrupt Status Register (ISR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SCAL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PPIDL | PPIDL0 | PPICL | — | — | — | TO | — | — | — | TA | — | RXP | TXVM | TXCM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 27 to 17 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0.. |
| 16 | SCAL | B'0 | R | (for 2.5G)<br>When this value is “1”, an interrupt of Skew Calibration exist.<br>Refer to the description of SCALSR register for more information. |
| 15 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | PPIDL | B'0 | R | When this value is “1”, an interrupt overall of D-PHY PPI Data Lane exists.<br>Refer to the description of PPIDLSR register for more information. |
| 13 | PPIDL0 | B'0 | R | When this value is “1”, an interrupt of D-PHY PPI Data Lane0 exists.<br>Refer to the description of PPIDL0SR register for more information. |
| 12 | PPICL | B'0 | R | When this value is “1”, an interrupt of D-PHY PPI Clock Lane exists.<br>Refer to the description of PPICLSR register for more information. |
| 11 to 9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | TO | B'0 | R | When this value is “1”, an interrupt of Timeout exists.<br>Refer to the description of TOSR register for more information. |
| 7 to 5 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 4 | TA | B'0 | R | When this value is “1”, an interrupt of Turn-Around exists.<br>Refer to the description of TASR register for more information |
| 3 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 2 | RXP | B'0 | R | When this value is “1”, an interrupt of Rx Packet exists.<br>Refer to the description of RXPSR register for more information. |
| 1 | TXVM | B'0 | R | When this value is “1”, an interrupt of Tx Video mode exists.<br>Refer to the description of TXVMSR register for more information. |
| 0 | TXCM | B'0 | R | When this value is “1”, an interrupt of Tx Command Transfer mode exists.<br>Refer to the description of TXCMSR register for more information. |

### 67.2.2.2 Link Status Register (LINKSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | LPBUSY | HSBUSY |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | LPBUSY | B'0 | R | When this value is “1”, LP is running.<br>Check this value to confirm to finish of LP behavior. |
| 0 | HSBUSY | B'0 | R | When this value is “1”, HS is running.<br>Check this value to confirm to finish of HS behavior. |

### 67.2.2.3 Tx Set Register (TXSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SET1 | | | | — | — | — | — | SET2 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EOT PEN | — | — | — | — | — | — | — | — | — | — | LANECNT | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 27 to 24 | SET1 | H'F | R/W | Prohibit to change. Keep the Initial Value. |
| 23 to 20 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 19 to 16 | SET2 | H'F | R | Prohibit to change. Keep the Initial Value. |
| 15 to 13 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 12 | EOTPEN | B'1 | R/W | Enable HS Tx EoTp<br>B'0: Disable Tx EoTp<br>B'1: Enable Tx EoTp<br>Notice, EoTp is always disable in LP Tx. |
| 11 to 2 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1, 0 | LANECNT | H'3 | R/W | Set Lane count.<br>H'0: Lane count is 1 (Use of Lane0)<br>H'1: Lane count is 2 (Use of Lane0 and Lane1)<br>H'2: Lane count is 3 (Use of Lane0, Lane1 and Lane2)<br>H'3: Lane count is 4 (Use of Lane0, Lane1, Lane2 and Lane3)<br>When PPISETR.DLEN=H'1, set to H'0.<br>When PPISETR.DLEN=H'3, set to H'1.<br>When PPISETR.DLEN=H'7, set to H'2.<br>When PPISETR.DLEN=H'F, set to H'3.<br>Notice, prohibit to set to H'2 in CSI-2 mode. |

### 67.2.2.4 Tx Command transfer Mode Set Register (TXCMSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SPD TYP | — | — | — | — | — | — | — | LPP DAC C |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | SPDTYP | B'0 | R/W | Set the speed type in Command Transfer mode.<br>B'0: High Speed<br>B'1: Low Power |
| 7 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | LPPDACC | B'0 | R/W | Set the method to access to Tx long Packet payload data.<br>B'0: Register access(by using TXCMPPD[n]R)<br>B'1: AXI access (by using AXI read access) |

### 67.2.2.5 Tx Command transfer Mode Control Register (TXCMCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | BTA TYP | BTA REQ | TXR EQ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W | R0/W | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 2 | BTATYP | B'0 | R0/W | Tx Request of Command Transfer mode with Bus Turn-Around (BTA).<br>B'0: Tx Request is Read Command when BTAREQ=1<br>B'1: Tx Request is Non-Read Command when BTAREQ=1<br>When BTAREQ=0, any value of this field is No meaning and ignored. |
| 1 | BTAREQ | B'0 | R0/W | Tx Request of Command Transfer mode with Bus Turn-Around.<br>B'0: No meaning, ignored<br>B'1: Tx Request of Command Transfer mode with Bus Turn-Around.*1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | TXREQ | B'0 | R0/W | Tx Request of Command Transfer mode without Bus Turn-Around.<br>B'0: No meaning, ignored<br>B'1: Tx Request of Command Transfer mode without Bus Turn-Around.*1 |

Note: *1 It is prohibited to set '1' at the same time to TXCMCR.BTAREQ and TXCMCR.TXREQ.

### 67.2.2.6 Tx Command transfer Mode Status Register (TXCMSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLSNERR | — | AXIERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TXREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 18 | CLSNERR | B'0 | R | When this value is “1”, Tx Error of HS Command Transfer Packet is occurred by the abnormal transmission.<br>Write “1” to TXCMSCR.CLSNERR register to clear this field. |
| 17 | — | B’0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | AXIERR | B'0 | R | When this value is “1”, Tx Error of HS Command Transfer Packet is occurred by the Error of AXI Read.<br>Write “1” to TXCMSCR.AXIERR register to clear this field. |
| 15 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | TXREQEND | B'0 | R | When this value is “1”, completion of the request for HS Command Transfer Packet is detected.<br>Write “1” to TXCMSCR.TXREQEND register to clear this field. |

### 67.2.2.7 Tx Command transfer Mode Status Clear Register (TXCMSCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLSNERR | — | AXIERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W | R | R0/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TXREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 18 | CLSNERR | B'0 | R0/W | Set to "1" to clear the TXCMSR.CLSNERR.<br>B'0: No meaning, ignored<br>B'1: Clear TXCMSR.CLSNERR |
| 17 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | AXIERR | B'0 | R0/W | Set to "1" to clear the TXCMSR.AXIERR.<br>B'0: No meaning, ignored<br>B'1: Clear TXCMSR.AXIERR |
| 15 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | TXREQEND | B'0 | R0/W | Set to "1" to clear the TXCMSR.TXREQEND.<br>B'0: No meaning, ignored<br>B'1: Clear TXCMSR.TXREQEND |

### 67.2.2.8 Tx Command transfer Mode Interrupt Enable Register (TXCMIER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | CLSNERR | — | AXIERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TXREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 18 | CLSNERR | B'0 | R/W | Interrupt enable for TXCMSR.CLSNERR<br>B'0: Disable interrupt of dsi_int_txcm when TXCMSR.CLSNERR=1<br>B'1: Enable interrupt of dsi_int_txcm when TXCMSR.CLSNERR=1 |
| 17 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | AXIERR | B'0 | R/W | Interrupt enable for TXCMSR.AXIERR<br>B'0: Disable interrupt of dsi_int_txcm when TXCMSR.AXIERR=1<br>B'1: Enable interrupt of dsi_int_txcm when TXCMSR.AXIERR=1 |
| 15 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | TXREQEND | B'0 | R/W | Interrupt enable for TXCMSR.TXREQEND<br>B'0: Disable interrupt of dsi_int_txcm when TXCMSR.TXREQEND=1<br>B'1: Enable interrupt of dsi_int_txcm when TXCMSR.TXREQEND=1 |

### 67.2.2.9 Tx Command transfer Mode Address Set 0 Register (TXCMADDRSET0R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDRL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDRL | | | | | | | | | | | | | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | ADDRL | H'0000 0000 | R/W | Set the Start Address for Payload Data of Tx Long Packet.<br>Bit[31:3] of the address can be set. Bit[2:0] is always fixed to "0".<br>AXI master can read the payload data by read access when the access of the payload data for Tx Command Transfer mode is used by AXI access (TXCMSET0R.LPPDACC=1) and is a Long Packet for Tx Command Transfer mode (TXCMPHDR.FMT=1). |
| 2 to 0 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.10 Tx Command transfer Mode Header Data Register (TXCMPHDR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | FMT | VC | | DT | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA1 | | | | | | | | DATA0 | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 24 | FMT | B'0 | R/W | Packet format of Tx Packet Header.<br>This field should be set by the data type.<br>B'0: Short Packet<br>B'1: Long Packet |
| 23, 22 | VC | H'0 | R/W | Identifier of virtual channel of Tx Packet Header. This value is limited to 3 from 0. |
| 21 to 16 | DT | H'00 | R/W | Data type of Tx Packet Header. |
| 15 to 8 | DATA1 | H'00 | R/W | Data1 of Tx Packet Header.*1<br>When Tx Packet format is Long Packet, this value is upper 8 bits of the word count. |
| 7 to 0 | DATA0 | H'00 | R/W | Data0 of Tx Packet Header. *1<br>When Tx Packet format is Long Packet, this value is lower 8 bits of the word count. |

Note: *1

・The maximum size for HS Command Transfer mode Long Packet by using AXI is 1024 Byte.
(TXCMSET0R.LPPDACC=1, SPDTYP=0 and TXCMPHDR.FMT=1)

・The maximum size for LP Command Transfer mode Long Packet by using AXI is 128 Byte.
(TXCMSET0R.LPPDACC=1, SPDTYP=1 and TXCMPHDR.FMT=1)

・The maximum size for Command Transfer mode Long Packet by using Register is 16 Byte.
(TXCMSET0R.LPPDACC=0 and TXCMPHDR.FMT=1)

### 67.2.2.11 Tx Command transfer Mode Packet Payload Data0 Register (TXCMPPD0R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DATA | H'0000 0000 | R/W | Payload Data0 for Tx Long Packet.<br>The data is transferred as Payload Data0 when the access of the payload data for Tx Long Packet is used by Register access (TXCMSET0R.LPPDACC=0) and is a Long Packet of Tx Command Transfer mode (TXCMPHDR.FMT=1).<br>Bit[7:0] : Data byte 0<br>Bit[15:8] : Data byte 1<br>Bit[23:16]: Data byte 2<br>Bit[31:24]: Data byte 3 |

### 67.2.2.12 Tx Command transfer Mode Packet Payload Data1 Register (TXCMPPD1R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DATA | H'0000 0000 | R/W | Payload Data1 for Tx Long Packet.<br>The data is transferred as Payload Data1 when the access of the payload data for Tx Long Packet is used by Register access (TXCMSET0R.LPPDACC=0) and is a Long Packet of Tx Command Transfer mode (TXCMPHDR.FMT=1).<br>Bit[7:0] : Data byte4<br>Bit[15:8] : Data byte 5<br>Bit[23:16]: Data byte 6<br>Bit[31:24]: Data byte 7 |

### 67.2.2.13 Tx Command transfer Mode Packet Payload Data2 Register (TXCMPPD2R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DATA | H'0000 0000 | R/W | Payload Data2 for Tx Long Packet.<br>The data is transferred as Payload Data2 when the access of the payload data for Tx Long Packet is used by Register access (TXCMSET0R.LPPDACC=0) and is a Long Packet of Tx Command Transfer mode (TXCMPHDR.FMT=1).<br>Bit[7:0] : Data byte8<br>Bit[15:8] : Data byte9<br>Bit[23:16]: Data byte10<br>Bit[31:24]: Data byte11 |

### 67.2.2.14 Tx Command transfer Mode Packet Payload Data3 Register (TXCMPPD3R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DATA | H'0000 0000 | R/W | Payload Data3 for Tx Long Packet.<br>The data is transferred as Payload Data3 when the access of the payload data for Tx Long Packet is used by Register access (TXCMSET0R.LPPDACC=0) and is a Long Packet of Tx Command Transfer mode (TXCMPHDR.FMT=1).<br>Bit[7:0] : Data byte12<br>Bit[15:8] : Data byte13<br>Bit[23:16]: Data byte14<br>Bit[31:24]: Data byte15 |

### 67.2.2.15 Tx Video Mode Set Register (TXVMSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SYNSEQ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VSTPM | — | — | — | — | PIXWDTH | | | — | — | — | VSEN | — | HFPBPEN | HBPBPEN | HSABPEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R/W | R/W | R/W | R | R | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | SYNSEQ | B'0 | R/W | Sequence of Tx Video mode.<br>B'0: Sync Pulse mode (Non-Burst Mode with Sync Pulses)<br>B'1: Sync Event mode (Non-Burst Mode with Sync Events) |
| 15 | VSTPM | B'0 | R/W | Set to B'1 |
| 14 to 11 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 10 to 8 | PIXWDTH | B’001 | R/W | Prohibits to change from Initial Value |
| 7 to 5 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 4 | VSEN | B'0 | R/W | (for 2.5G)<br>Enable transmission of Skew Calibration at every frame in Tx Video Mode.<br>B'0: Disable<br>B'1: Enable |
| 3 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 2 | HFPBPEN | B'0 | R/W | Enable transmission of Blanking Packet during the HFP interval. *1<br>B'0: Disable (LP-11)<br>B'1: Enable (Data Type=H’19)<br>If the period of HFP exceeds (packet maximum length (WC=65535)+4Byte), be sure to set this bit to B’0. |
| 1 | HBPBPEN | B'0 | R/W | Enable transmission of Blanking Packet during the HBP interval. *1<br>B'0: Disable (LP-11)<br>B'1: Enable (Data Type=H’19)<br>If the period of HBP exceeds (packet maximum length (WC=65535)+4Byte), be sure to set this bit to B’0. |
| 0 | HSABPEN | B'0 | R/W | Enable transmission of Blanking Packet during the HSA interval. *1<br>B'0: Disable (LP-11)<br>B'1: Enable (Data Type=H’19)<br>If the period of HSA exceeds (packet maximum length (WC=65535)+4Byte), be sure to set this bit to B’0. |

Note: *1 there is sufficient time to transition from HS to LP mode and back again, a timed interval in LP mode may substitute for a Blanking Packet, thus saving power.

The host processor should return to LP state once per scanline during the horizontal blanking time.

### 67.2.2.16 Tx Video Mode Set 2 Register (TXVMSET2R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FNUM | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 15 to 0 | FNUM | D'0 | R/W | Maximum value of the Frame Number field both of Frame Start Packet and Frame End Packet.<br>D'0: Set to "0" to Frame Number field.:<br>Other: Frame number is 1 at the first frame after starting Video mode.<br>Frame number is incremented automatically by every frame.<br>When the Frame number is same as this FNUM value, the value is set to 1 at the next frame. |

### 67.2.2.17 Tx Video Mode Control Register (TXVMCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | VFCLR | — | — | — | — | — | — | — | — | — | — | — | EN |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 12 | VFCLR | B'1 | R/W | Clear FIFO *1 for Tx Video mode.<br>When write “1”, request Clear FIFO.<br>When write “0”, withdraw the request of Clear FIFO. |
| 11 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | EN | B'0 | R/W | When write “1”, enable Video mode Tx. This field should write “1” before starting Video Input.<br>When write “0”, disable Video mode Tx. Refer to Figure 67.13 for the procedure. |

Note: *1 This FIFO is a translator of Tx Video Data to hsclk (byteclock) domain from vclk (dotclock) domain.

### 67.2.2.18 Tx Video Mode Status Register (TXVMSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | VBP DET | VSA DET | — | — | — | VFE RR | — | — | — | PDIS | — | — | STP | STR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | VFR DY | — | — | — | ACT | — | — | — | — | — | — | — | RDY |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 29 | VBPDET | B’0 | R | "1" is set when the start of the VBP period (0→1 when VSYNC is high polarity and 1→0 for Low polarity) is detected.<br>Write “1” to TXVMSCR.VBPDET register to clear this field. |
| 28 | VSADET | B’0 | R | "1" is set when the start of the VSA period (0→1 when VSYNC is high polarity and 1→0 for Low polarity) is detected.<br>Write “1” to TXVMSCR.VSADET register to clear this field. |
| 27 to 25 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 24 | VFERR | B'0 | R | When this value is “1”, Overrun or Underrun of Tx Video mode FIFO is occurred.<br>Write “1” to TXVMSCR.VFERR register to clear this field. |
| 23 to 21 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 20 | PDIS | B'0 | R | When this value is “1”, the Tx Packet is discarded in Video mode.<br>Write “1” to TXVMSCR.PDIS register to clear this field. |
| 19, 18 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 17 | STP | B'0 | R | When this value is “1”, the End of Tx is detected in Video mode.<br>Write “1” to TXVMSCR.STP register to clear this field. |
| 16 | STR | B'0 | R | When this value is “1”, the Start of Tx is detected in Video mode.<br>Write “1” to TXVMSCR.STR register to clear this field. |
| 15 to 13 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 12 | VFRDY | B'0 | R | Status of Tx Video FIFO.<br>B'0: Not ready<br>B'1: Ready for use |
| 11 to 9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | ACT | B'0 | R | Status of Video mode Tx<br>B'0: stopped<br>B'1: running |
| 7 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | RDY | B'0 | R | Status of Video mode Tx ready<br>Input Video is permitted to start after the confirmation of this value of “1”.<br>B'0: Not ready of Tx Video mode (Input Video inhibited)<br>B'1: Ready for use of Tx Video mode (Input Video permitted) |

### 67.2.2.19 Tx Video Mode Status Clear Register (TXVMSCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | — | — | VBP DET | VSA DET | — | — | — | VFE RR | — | — | — | PDIS | — | — | STP | STR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R0/W | R0/W | R | R | R | R0/W | R | R | R | R0/W | R | R | R0/W | R0/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | — | B'00 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 29 | VBPDET | B’0 | R0/W | Set to "1" to clear the TXVMSR.VBPDET.<br>B'0: No meaning, ignored<br>B'1: Clear TXVMSR.VBPDET |
| 28 | VSADET | B’0 | R0/W | Set to "1" to clear the TXVMSR.VSADET.<br>B'0: No meaning, ignored<br>B'1: Clear TXVMSR.VSADET |
| 27 to 25 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 24 | VFERR | B'0 | R0/W | Set to "1" to clear the TXVMSR.VFERR.<br>B'0: No meaning, ignored<br>B'1: Clear TXVMSR.VFERR |
| 23 to 21 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 20 | PDIS | B'0 | R0/W | Set to "1" to clear the TXVMSR.PDIS.<br>B'0: No meaning, ignored<br>B'1: Clear TXVMSR.PDIS |
| 19, 18 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 17 | STP | B'0 | R0/W | Set to "1" after detection of the End of Tx in Video mode to detect the next one.<br>B'0: No meaning, ignored<br>B'1: Clear TXVMSR.STP |
| 16 | STR | B'0 | R0/W | Set to "1" after detection of the Start of Tx in Video mode to detect the next one.<br>B'0: No meaning, ignored<br>B'1: Clear TXVMSR.STR |
| 15 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.20 Tx Video Mode Interrupt Enable Register (TXVMIER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | VBPDET | VSADET | — | — | — | VFERR | — | — | — | PDIS | — | — | STP | STR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R | R/W | R | R | R | R/W | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | — | B’00 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 29 | VBPDET | B’0 | R/W | Interrupt enable for TXVMSR.VBPDET<br>B'0: Disable interrupt of dsi_int_txvm when TXVMSR.VBPDET=1<br>B'1: Enable interrupt of dsi_int_txvm when TXVMSR.VBPDET=1 |
| 28 | VSADET | B’0 | R/W | Interrupt enable for TXVMSR.VFERR<br>B'0: Disable interrupt of dsi_int_txvm when TXVMSR.VSADET=1<br>B'1: Enable interrupt of dsi_int_txvm when TXVMSR.VSADET=1 |
| 27 to 25 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 24 | VFERR | B'0 | R/W | Interrupt enable for TXVMSR.VFERR<br>B'0: Disable interrupt of dsi_int_txvm when TXVMSR.VFERR=1<br>B'1: Enable interrupt of dsi_int_txvm when TXVMSR.VFERR=1 |
| 23 to 21 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 20 | PDIS | B'0 | R/W | Interrupt enable for TXVMSR.PDIS<br>B'0: Disable interrupt of dsi_int_txvm when TXVMSR.PDIS=1<br>B'1: Enable interrupt of dsi_int_txvm when TXVMSR.PDIS=1 |
| 19, 18 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 17 | STP | B'0 | R/W | Interrupt enable for detection of End of Tx in Video mode.<br>B'0: Disable interrupt of dsi_int_txvm when TXVMSR.STP=1<br>B'1: Enable interrupt of dsi_int_txvm when TXVMSR.STP=1 |
| 16 | STR | B'0 | R/W | Interrupt enable for detection of Start of Tx in Video mode.<br>B'0: Disable interrupt of dsi_int_txvm when TXVMSR.STR=1<br>B'1: Enable interrupt of dsi_int_txvm when TXVMSR.STR=1 |
| 15 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.21 Tx Video Mode Pixel Stream Packet Header Set Register (TXVMPSPHSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | VC | | DT | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 23, 22 | VC | H'0 | R/W | Identifier for virtual channel of Tx Video Mode Pixel Stream Packet Header. |
| 21 to 16 | DT | H'00 | R/W | Data Type of Tx Video Mode Pixel Stream Packet Header. It can be set following values.<br>(for DSI mode)<br>H'0E:Packed Pixel Stream, 16bit RGB<br>H'1E:Packed Pixel Stream, 18bit RGB<br>H'2E:Loosely Packed Pixel Stream, 18bit RGB<br>H'3E:Packed Pixel Stream, 24bit RGB<br>H'2C:Packed Pixel Stream, 16bit YCbCr 4:2:2<br>(for CSI-2 mode)<br>H'22:RGB565<br>H'23:RGB666<br>H'24:RGB888<br>H'1E:YUV422 8-bit<br>Operation is prohibited to write any value except for listed above. |
| 15 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.22 Tx Video Mode Parameter Set 0 Register (TXVMVPRMSET0R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HS POL | VS POL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CSPC | | — | BPP | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 17 | HSPOL | B'0 | R/W | Set polarity of HSYNC signal.<br>B'0: High Active<br>B'1: Low Active. |
| 16 | VSPOL | B'0 | R/W | Set polarity of VSYNC signal.<br>B'0: High Active<br>B'1: Low Active |
| 15 to 6 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 5 to 4 | CSPC | B'00 | R/W | Set color space of input video pixel.<br>B'00:RGB<br>B'01:YCbCr(DSI)/YUV(CSI-2)<br>B’10:Reserved<br>B’11:Reserved |
| 3 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 2 to 0 | BPP | B'000 | R/W | Set VideoColorDepth of input video pixel.<br>B'000:16bpp<br>B'001:18bpp<br>B'010:24bpp*1<br>Operation is prohibited to write any value except for listed above.<br>Note: *1 Use this value for Loosely Packed Pixel Stream, 18bit RGB. |

### 67.2.2.23 Tx Video Mode Parameter Set 1 Register (TXVMVPRMSET1R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | VACTIVE | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | VSAVW | — | — | VSA | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 30 to 16 | VACTIVE | H'0000 | R/W | Set Line count of VACTIVE (Vertical Active lines) |
| 15 | | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | VSAVW | B'0 | R/W | Allows the number of lines in a VSA period to vary.<br>0b : Not allowed<br>1b : Allow<br>Do not change the initial value in DSI mode. |
| 13 to 12 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 11 to 0 | VSA | H'000 | R/W | Set Line count of VSA (Vertical Sync Active) |

### 67.2.2.24 Tx Video Mode Parameter Set 2 Register (TXVMVPRMSET2R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | VFPVW | — | VFP | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | VBPVW | — | VBP | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | | B’0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 30 | VFPVW | B’0 | R/W | Allows the number of lines in a VFP period to vary.<br>0b : Not allowed<br>1b : Allow<br>Do not change the initial value in DSI mode. |
| 29 | — | B’0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 28 to 16 | VFP | H'0000 | R/W | Set Line count of VFP (Vertical Front Porch) |
| 15 | | B’0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | VBPVW | B’0 | R/W | Allows the number of lines in a VBP period to vary.<br>0b : Not allowed<br>1b : Allow<br>Do not change the initial value in DSI mode. |
| 13 | — | B’0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 12 to 0 | VBP | H'0000 | R/W | Set Line count of VBP (Vertical Back Porch) |

### 67.2.2.25 Tx Video Mode Parameter Set 3 Register (TXVMVPRMSET3R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | HACTIVE | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSBYTE | | | | HSA | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 30 to 16 | HACTIVE | H'0000 | R/W | Set pixel count of HACTIVE (Horizontal Active pixels) |
| 15 to 12 | HSBYTE | H'0 | R/W | Reserved |
| 11 to 0 | HSA | H'000 | R/W | Set pixel count of HSA (Horizontal Sync Active) |

### 67.2.2.26 Tx Video Mode Parameter Set 4 Register (TXVMVPRMSET4R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | HFP | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | HBP | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 28 to 16 | HFP | H'0000 | R/W | Set pixel count of HFP (Horizontal Front Porch) |
| 15 to 13 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 12 to 0 | HBP | H'0000 | R/W | Set pixel count of HBP (Horizontal Back Porch) |

### 67.2.2.27 Rx Set Register (RXSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | CRCEN | | | | — | — | — | — | ECCEN | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 27 to 24 | CRCEN | H'F | R/W | Enable for Received CRC check for virtual channel<br>[0]:Enable Received CRC for virtual channel 0 (B'0:Disable, B'1:Enable)<br>[1]:Enable Received CRC for virtual channel 1 (B'0:Disable, B'1:Enable)<br>[2]:Enable Received CRC for virtual channel 2 (B'0:Disable, B'1:Enable)<br>[3]:Enable Received CRC for virtual channel 3 (B'0:Disable, B'1:Enable) |
| 23 to 20 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 19 to 16 | ECCEN | H'F | R/W | Enable Received ECC check for virtual channel<br>[0]:Enable Received ECC for virtual channel 0 (B'0:Disable, B'1:Enable)<br>[1]:Enable Received ECC for virtual channel 1 (B'0:Disable, B'1:Enable)<br>[2]:Enable Received ECC for virtual channel 2 (B'0:Disable, B'1:Enable)<br>[3]:Enable Received ECC for virtual channel 3 (B'0:Disable, B'1:Enable)<br>Notice, it is recommended that the value of Enable Received ECC should be same for all virtual channel. |
| 15 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.28 Rx Packet Set Register (RXPSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | LPPDACC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | LPPDACC | B'0 | R/W | Set access method for Received Long packet payload data.<br>0:Register access mode (using RXPPD[n]R register access)<br>1:AXI access mode (using AXI write access) |

### 67.2.2.29 Rx Packet Status Register (RXPSR)

This register shows Rx Packet Status by TXCMCR.BTAREQ.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECCERR1B | — | — | UEXTRGERR | RESPTOERR | OVRERR | AXIERR | CRCERR | WCERR | UEXDTERR | UEXPKTERR | ECCERR | MLFERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RCVACK | — | — | — | RCVEOT | RCVAKE | RCVRESP | — | — | — | — | — | — | — | BTAREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 28 | ECCERR1B | B'0 | R | When this value is “1”, the Rx ECC Error (1bit) is detected.<br>Write “1” to RXPSCR.ECC1BERR register to clear this field. |
| 27, 26 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25 | UEXTRGERR | B'0 | R | When this value is “1”, the Rx Unexpected Trigger is detected.<br>Write “1” to RXPSCR.UEXTRGERR register to clear this field. |
| 24 | RESPTOERR | B'0 | R | When this value is “1”, the Rx Timeout is detected after changing to Rx mode along with Bus Turn-Around and within the time specified by RESPTOSETR.RESPTOVL.<br>Write “1” to RXPSCR.RESPTOERR register to clear this field. |
| 23 | OVRERR | B'0 | R | When this value is “1”, the Buffer Overrun Error is detected at Rx Long Packet.<br>Write “1” to RXPSCR.OVRERR register to clear this field. |
| 22 | AXIERR | B'0 | R | When this value is “1”, the Response except for OKAY is received at AXI write.<br>Write “1” to RXPSCR.AXIERR register to clear this field. |
| 21 | CRCERR | B'0 | R | When this value is “1”, the Rx CRC Error is detected.<br>Write “1” to RXPSCR.CRCERR register to clear this field. |
| 20 | WCERR | B'0 | R | When this value is “1”, the Word Count Error of Rx Packet length below word count is detected.<br>Write “1” to RXPSCR.WCERR register to clear this field. |
| 19 | UEXDTERR | B'0 | R | When this value is “1”, the unexpected Packet Type is received.<br>Write “1” to RXPSCR.UEXDTERR register to clear this field. |
| 18 | UEXPKTERR | B'0 | R | When this value is “1”, the unexpected Packet is received.<br>Write “1” to RXPSCR.UEXPKTERR register to clear this field. |
| 17 | ECCERR | B'0 | R | When this value is “1”, the Rx ECC Error (over 1bit) is detected.<br>Write “1” to RXPSCR.ECCERR register to clear this field. |
| 16 | MLFERR | B'0 | R | When this value is “1”, the Rx Packet under 4bit length is received.<br>Write “1” to RXPSCR.MLFERR register to clear this field. |
| 15 | Reserved | B'0 | R | This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 14 | RCVACK | B'0 | R | When this value is "1", the ACK trigger is received.<br>When unexpected ACK trigger is received, this value does not set to "1" and UEXTRGERR is set to "1". Refer to Section 67.3.2.10.3.5 for detail.<br>Write "1" to RXPSCR.RCVACK register to clear this field. |
| 13 to 11 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 10 | RCVEOT | B'0 | R | When this value is “1”, the EoTp Packet is received.<br>When unexpected EoTp Packet is received, this value does not set to "1" and UEXPKTERR is set to "1". Refer to Section 67.3.2.10.3.4 for detail.<br>Write “1” to RXPSCR.RCVEOT register to clear this field. |
| 9 | RCVAKE | B'0 | R | When this value is “1”, the Acknowledge and Error Report Packet is received.<br>When unexpected Acknowledge and Error Report Packet is received, this value does not set to "1" and UEXPKTERR is set to "1". Refer to Section 67.3.2.10.3.4 for detail.<br>Write “1” to RXPSCR.RCVAKE register to clear this field. |
| 8 | RCVRESP | B'0 | R | When this value is “1”, the packet except for Acknowledge and Error Report Packet is received.<br>When any packet except for EoTP Packet and Acknowledge and Error Report Packet is received, this value does not set to "1" and UEXPKTERR is set to "1". Refer to Section 67.3.2.10.3.4 for detail.<br>Write “1” to RXPSCR.RCVRESP register to clear this field. |
| 7 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | BTAREQEND | B'0 | R | When this value is “1”, the completion of Transmission request include Rx by TXCMCR.BTAREQ Packet is detected.<br>Write “1” to RXPSCR.BTAREQEND register to clear this field. |

### 67.2.2.30 Rx Packet Status Clear Register (RXPSCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECCERR1B | — | — | UEXTRGERR | RESPTOERR | OVRERR | AXIERR | CRCERR | WCERR | UEXDTERR | UEXPKTERR | ECCERR | MLFERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R0/W | R | R | R0/W | R0/W | R0/W | R0/W | R0/W | R0/W | R0/W | R0/W | R0/W | R0/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RCVACK | — | — | — | RCVEOT | RCVAKE | RCVRESP | — | — | — | — | — | — | — | BTAREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R0/W | R | R | R | R0/W | R0/W | R0/W | R | R | R | R | R | R | R | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 28 | ECCERR1B | B'0 | R0/W | Set to "1" to clear the RXPSR.ECCERR1B.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.ECCERR1B |
| 27, 26 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25 | UEXTRGERR | B'0 | R0/W | Set to "1" to clear the RXPSR.UEXTRGERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.UEXTRGERR |
| 24 | RESPTOERR | B'0 | R0/W | Set to "1" to clear the RXPSR.RESPTOERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.RESPTOERR |
| 23 | OVRERR | B'0 | R0/W | Set to "1" to clear the RXPSR.OVRERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.OVRERR |
| 22 | AXIERR | B'0 | R0/W | Set to "1" to clear the RXPSR.AXIERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.AXIERR |
| 21 | CRCERR | B'0 | R0/W | Set to "1" to clear the RXPSR.CRCERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.CRCERR |
| 20 | WCERR | B'0 | R0/W | Set to "1" to clear the RXPSR.WCERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.WCERR |
| 19 | UEXDTERR | B'0 | R0/W | Set to "1" to clear the RXPSR.UEXDTERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.UEXDTERR |
| 18 | UEXPKTERR | B'0 | R0/W | Set to "1" to clear the RXPSR.UEXPKTERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.UEXPKTERR |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | ECCERR | B'0 | R0/W | Set to "1" to clear the RXPSR.ECCERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.ECCERR |
| 16 | MLFERR | B'0 | R0/W | Set to "1" to clear the RXPSR.MLFERR.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.MLFERR |
| 15 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | RCVACK | B'0 | R0/W | Set to "1" to clear the RXPSR.RCVACK.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.RCVACK |
| 13 to 11 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 10 | RCVEOT | B'0 | R0/W | Set to "1" to clear the RXPSR.RCVEOT.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.RCVEOT |
| 9 | RCVAKE | B'0 | R0/W | Set to "1" to clear the RXPSR.RCVAKE.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.RCVAKE |
| 8 | RCVRESP | B'0 | R0/W | Set to "1" to clear the RXPSR.RCVRESP.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.RCVRESP |
| 7 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | BTAREQEND | B'0 | R0/W | Set to "1" to clear the RXPSR.BTAREQEND.<br>B'0: No meaning, ignored<br>B'1: Clear RXPSR.BTAREQEND |

### 67.2.2.31 Rx Packet Interrupt Enable Register (RXPIER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECCERR1B | — | — | UEXTRGERR | RESPTOERR | OVRERR | AXIERR | CRCERR | WCERR | UEXDTERR | UEXPKTERR | ECCERR | MLFERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RCVACK | — | — | — | RCVEOT | RCVAKE | RCVRESP | — | — | — | — | — | — | — | BTAREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 28 | ECCERR1B | B'0 | R/W | Interrupt enable for RXPSR.ECCERR1B<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.ECCERR1B=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.ECCERR1B=1 |
| 27, 26 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25 | UEXTRGERR | B'0 | R/W | Interrupt enable for RXPSR.UEXTRGERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.UEXTRGERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.UEXTRGERR=1 |
| 24 | RESPTOERR | B'0 | R/W | Interrupt enable for RXPSR.RESPTOERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.RESPTOERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.RESPTOERR=1 |
| 23 | OVRERR | B'0 | R/W | Interrupt enable for RXPSR.OVRERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.OVRERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.OVRERR=1 |
| 22 | AXIERR | B'0 | R/W | Interrupt enable for RXPSR.AXIERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.AXIERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.AXIERR=1 |
| 21 | CRCERR | B'0 | R/W | Interrupt enable for RXPSR.CRCERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.CRCERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.CRCERR=1 |
| 20 | WCERR | B'0 | R/W | Interrupt enable for RXPSR.WCERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.WCERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.WCERR=1 |
| 19 | UEXDTERR | B'0 | R/W | Interrupt enable for RXPSR.UEXDTERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.UEXDTERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.UEXDTERR=1 |
| 18 | UEXPKTERR | B'0 | R/W | Interrupt enable for RXPSR.UEXPKTERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.UEXPKTERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.UEXPKTERR=1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | ECCERR | B'0 | R/W | Interrupt enable for RXPSR.ECCERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.ECCERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.ECCERR=1 |
| 16 | MLFERR | B'0 | R/W | Interrupt enable for RXPSR.MLFERR<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.MLFERR=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.MLFERR=1 |
| 15 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | RCVACK | B'0 | R/W | Interrupt enable for RXPSR.RCVACK<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.RCVACK=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.RCVACK=1 |
| 13 to 11 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 10 | RCVEOT | B'0 | R/W | Interrupt enable for RXPSR.RCVEOT<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.RCVEOT=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.RCVEOT=1 |
| 9 | RCVAKE | B'0 | R/W | Interrupt enable for RXPSR.RCVAKE<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.RCVAKE=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.RCVAKE=1 |
| 8 | RCVRESP | B'0 | R/W | Interrupt enable for RXPSR.RCVRESP<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.RCVRESP=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.RCVRESP=1 |
| 7 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | BTAREQEND | B'0 | R/W | Interrupt enable for RXPSR.BTAREQEND<br>B'0: Disable interrupt of dsi_int_rxp when RXPSR.BTAREQEND=1<br>B'1: Enable interrupt of dsi_int_rxp when RXPSR.BTAREQEND=1 |

### 67.2.2.32 Rx Packet Address Set 0 Register (RXPADDRSET0R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDRL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDRL | | | | | | | | | | | | | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | ADDRL | H'0000 0000 | R/W | Set start address for Payload data of Rx Long Packet.<br>Bit[31:3] of the address can be set. Bit[2:0] is fixed to "0".<br>AXI master can read the payload data by read access when the access of the payload data for Rx Command Transfer mode is used by AXI access (RXPSET0R.LPPDACC=1) and is a Long Packet for Tx Command Transfer mode (RXPHDR.FMT=1). |
| 2 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.33 Rx Packet Size Set Register (RXPSIZESETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | SIZE | | | | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-7 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 6-3 | SIZE | H'0 | R/W | Set memory size of AXI by Byte Unit used for payload data of Rx Long Packet.<br>AXI master can write the payload data by write access when the access of the payload data for Rx Command Transfer mode is used by AXI access (RXPSET0R.LPPDACC=1) and is a Long Packet for Tx Command Transfer mode (RXPHDR.FMT=1).<br>The memory size setting is allowed by 8 byte unit because Bit[2:0] of the size is always fixed to "0".<br>The size of Rx Long Packet should be within 128 Byte. The value of 0 Byte and over 129 Byte is prohibited.<br>Without using Rx Long Packet, the Initial Value is permitted to set for this register. |
| 2 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.34 Rx Packet Header Data Register (RXPHDR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | FMT | VC | | DT | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA1 | | | | | | | | DATA0 | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 24 | FMT | B'0 | R | Packet format of Rx Packet Header.<br>B'0: Short Packet<br>B'1: Long Packet |
| 23, 22 | VC | H'0 | R | Identifier of virtual channel of Rx Packet Header. This value is limited to 3 from 0. |
| 21 to 16 | DT | H'00 | R | Data type of Rx Packet Header. |
| 15 to 8 | DATA1 | H'00 | R | Data1 of Rx Packet Header.<br>When Rx Packet format is Long Packet, this value is upper 8 bits of the word count. |
| 7 to 0 | DATA0 | H'00 | R | Data0 of Rx Packet Header.<br>When Rx Packet format is Long Packet, this value is lower 8 bits of the word count. |

### 67.2.2.35 Rx Packet Payload Data0 Register (RXPPD0R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DATA | H'0000 0000 | R | Payload Data0 for Rx Long Packet.<br>The data is received as Payload Data when the access of the payload data for Rx Long Packet is used by Register access (RXPSETR.LPPDACC=0) and is a Long Packet of Rx Command Transfer mode (RXPHDR.FMT=1).<br>Valid Payload data is identified by word count of Rx Packet Header (RXPHDR.DATA0 and RXPHDR.DATA1).<br>Bit[7:0] : Data byte0<br>Bit[15:8] : Data byte1<br>Bit[23:16]: Data byte2<br>Bit[31:24]: Data byte3 |

### 67.2.2.36 Rx Packet Payload Data1 Register (RXPPD1R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DATA | H'0000 0000 | R | Payload Data1 for Rx Long Packet.<br>The data is received as Payload Data when the access of the payload data for Rx Long Packet is used by Register access (RXPSETR.LPPDACC=0) and is a Long Packet of Rx Command Transfer mode (RXPHDR.FMT=1).<br>Valid Payload data is identified by word count of Rx Packet Header (RXPHDR.DATA0 and RXPHDR.DATA1).<br>Bit[7:0] : Data byte4<br>Bit[15:8] : Data byte5<br>Bit[23:16]: Data byte6<br>Bit[31:24]: Data byte7 |

### 67.2.2.37 Rx Packet Payload Data2 Register (RXPPD2R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DATA | H'0000 0000 | R | Payload Data2 for Rx Long Packet.<br>The data is received as Payload Data when the access of the payload data for Rx Long Packet is used by Register access (RXPSETR.LPPDACC=0) and is a Long Packet of Rx Command Transfer mode (RXPHDR.FMT=1).<br>Valid Payload data is identified by word count of Rx Packet Header (RXPHDR.DATA0 and RXPHDR.DATA1).<br>Bit[7:0] : Data byte8<br>Bit[15:8] : Data byte9<br>Bit[23:16]: Data byte10<br>Bit[31:24]: Data byte11 |

### 67.2.2.38 Rx Packet Payload Data3 Register (RXPPD3R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DATA | H'0000 0000 | R | Payload Data3 for Rx Long Packet.<br>The data is received as Payload Data when the access of the payload data for Rx Long Packet is used by Register access (RXPSETR.LPPDACC=0) and is a Long Packet of Rx Command Transfer mode (RXPHDR.FMT=1).<br>Valid Payload data is identified by word count of Rx Packet Header (RXPHDR.DATA0 and RXPHDR.DATA1).<br>Bit[7:0] : Data byte12<br>Bit[15:8] : Data byte13<br>Bit[23:16]: Data byte14<br>Bit[31:24]: Data byte15 |

### 67.2.2.39 Rx Ack and Error Packet parameter Register (AKEPR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | VC | | DT | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ERRRPT | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 23, 22 | VC | H'0 | R | Identifier of virtual channel of Rx Acknowledge and Error Report Packet. This value is limited to 3 from 0. |
| 21 to 16 | DT | H'00 | R | Data type of Rx Acknowledge and Error Report Packet. |
| 15 to 0 | ERRRPT | H'0000 | R | Error report of Rx Acknowledge and Error Report Packet. |

### 67.2.2.40 Rx Response Timeout Set Register (RXRESPTOSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESPTOVL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESPTOVL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | RESPTOVL | H'0000 0000 | R/W | Set Timeout value to measure the interval between Start of Rx mode and Start of Packet or Trigger Receive along with Bus Turn-Around with TXCMCR.BTAREQ.<br>The limit Time for Timeout is calculated by formula of RESPTOVL * (period of lpclk).<br>Set a value greater than Peripherai provides. If Peripheral offers more than one value, set the highest value.<br>When RXPSR.RESPTOERR is "1", the Timeout is detected. Notice, the Timeout is not detected when RESPTOVL=0. |

### 67.2.2.41 Turn-Around Control Register (TACR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TRN REQ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | TRNREQ | B'0 | R0/W | Bus Turn-Around request to Rx from Tx.<br>Write "0": No meaning, ignored<br>Write "1": Bus Turn-Around request to Rx from Tx by assertion TurnRequest of PPI Data Lane0. |

### 67.2.2.42 Turn-Around Status Register (TASR)

This register shows Turn-Around Status by TACR.TRNREQ.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECC ERR 1B | — | — | — | — | — | — | — | — | UEX DT ERR | UEX PKT ERR | ECC ERR | MLF ERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RCV ACK | — | — | — | RCV EOT | RCV AKE | — | — | — | — | — | — | — | — | TRN REQ END |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 28 | ECCERR1B | B'0 | R | When this value is “1”, the Rx ECC Error (1bit) is detected.<br>Write “1” to TASCR.ECC1BERR register to clear this field. |
| 27 to 20 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 19 | UEXDTERR | B'0 | R | When this value is “1”, the Packet with unexpected Data Type is received.<br>Write “1” to TASCR.UEXDTERR register to clear this field. |
| 18 | UEXPKTERR | B'0 | R | When this value is “1”, the unexpected Packet is received.<br>Write “1” to TASCR.UEXPKTERR register to clear this field. |
| 17 | ECCERR | B'0 | R | When this value is “1”, the Rx ECC Error (over 1bit) is detected.<br>Write “1” to TASCR.ECCERR register to clear this field. |
| 16 | MLFERR | B'0 | R | When this value is “1”, the Packet with under 4 byte length is received.<br>Write “1” to TASCR.MLFERR register to clear this field. |
| 15 | — | B'0 | R | This bit is always read as 0. The write value should always be 0. |
| 14 | RCVACK | B'0 | R | When this value is "1", the ACK trigger is received.<br>Write "1" to TASCR.RCVACK register to clear this field. |
| 13 to 11 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 10 | RCVEOT | B'0 | R | When this value is “1”, the EoTp Packet is detected.<br>When unexpected EoTp Packet is received, this value does not set to "1" and UEXPKTERR is set to "1". Refer to Section 67.3.2.10.3.4 for detail.<br>Write “1” to TASCR.RCVEOT register to clear this field. |
| 9 | RCVAKE | B'0 | R | When this value is “1”, the Acknowledge and Error Report Packet is detected.<br>When unexpected Acknowledge and Error Report Packet is received, this value does not set to "1" and UEXPKTERR is set to "1".Refer to Section 67.3.2.10.3.4 for detail.<br>Write “1” to TASCR.RCVAKE register to clear this field. |
| 8 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | TRNREQEND | B'0 | R | When this value is “1”, the completion of Turn-Around by TACR.TRNREQ and the confirmation of the Direction as "1" (Rx) are detected.<br>Write “1” to TASCR.TRNREQEND register to clear this field. |

### 67.2.2.43 Turn-Around Status Clear Register (TASCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECCERR1B | — | — | — | — | — | — | — | — | UEXDTERR | UEXPKTERR | ECCERR | MLFERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R0/W | R | R | R | R | R | R | R | R | R0/W | R0/W | R0/W | R0/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RCVACK | — | — | — | RCVEOT | RCVAKE | — | — | — | — | — | — | — | — | TRNREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R0/W | R | R | R | R0/W | R0/W | R | R | R | R | R | R | R | R | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 28 | ECCERR1B | B'0 | R0/W | Set to "1" to clear the TASR.ECCERR1B.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.ECCERR1B |
| 27 to 24 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 23 to 20 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 19 | UEXDTERR | B'0 | R0/W | Set to "1" to clear the TASR.UEXDTERR.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.UEXDTERR |
| 18 | UEXPKTERR | B'0 | R0/W | Set to "1" to clear the TASR.UEXPKTERR.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.UEXPKTERR |
| 17 | ECCERR | B'0 | R0/W | Set to "1" to clear the TASR.ECCERR.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.ECCERR |
| 16 | MLFERR | B'0 | R0/W | Set to "1" to clear the TASR.MLFERR.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.MLFERR |
| 15 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | RCVACK | B'0 | R0/W | Set to "1" to clear the TASR.RCVACK.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.RCVACK |
| 13 to 11 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 10 | RCVEOT | B'0 | R0/W | Set to "1" to clear the TASR.RCVEOT.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.RCVEOT |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | RCVAKE | B'0 | R0/W | Set to "1" to clear the TASR.RCVAKE.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.RCVAKE |
| 8 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | TRNREQEND | B'0 | R0/W | Set to "1" to clear the TASR.TRNREQEND.<br>B'0: No meaning, ignored<br>B'1: Clear TASR.TRNREQEND |

### 67.2.2.44 Turn-Around Interrupt Enable Register (TAIER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | ECCERR1B | — | — | — | — | — | — | — | — | UEXDTERR | UEXPKTERR | ECCERR | MLFERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RCVACK | — | — | — | RCVEOT | RCVAKE | — | — | — | — | — | — | — | — | TRNREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 28 | ECCERR1B | B'0 | R/W | Interrupt enable for TASR.ECCERR1B<br>B'0: Disable interrupt of dsi_int_ta when TASR.ECCERR1B=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.ECCERR1B=1 |
| 27to 24 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 23 to 20 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 19 | UEXDTERR | B'0 | R/W | Interrupt enable for TASR.UEXDTERR<br>B'0: Disable interrupt of dsi_int_ta when TASR.UEXDTERR=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.UEXDTERR=1 |
| 18 | UEXPKTERR | B'0 | R/W | Interrupt enable for TASR.UEXPKTERR<br>B'0: Disable interrupt of dsi_int_ta when TASR.UEXPKTERR=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.UEXPKTERR=1 |
| 17 | ECCERR | B'0 | R/W | Interrupt enable for TASR.ECCERR<br>B'0: Disable interrupt of dsi_int_ta when TASR.ECCERR=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.ECCERR=1 |
| 16 | MLFERR | B'0 | R/W | Interrupt enable for TASR.MLFERR<br>B'0: Disable interrupt of dsi_int_ta when TASR.MLFERR=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.MLFERR=1 |
| 15 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | RCVACK | B'0 | R/W | Interrupt enable for TASR.RCVACK<br>B'0: Disable interrupt of dsi_int_ta when TASR.RCVAC=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.RCVAC=1 |
| 13 to 11 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 10 | RCVEOT | B'0 | R/W | Interrupt enable for TASR.RCVEOT<br>B'0: Disable interrupt of dsi_int_ta when TASR.RCVEOT=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.RCVEOT=1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | RCVAKE | B'0 | R/W | Interrupt enable for TASR.RCVAKE<br>B'0: Disable interrupt of dsi_int_ta when TASR.RCVAKE=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.RCVAKE=1 |
| 8 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | TRNREQEND | B'0 | R/W | Interrupt enable for TASR.TRNREQEND<br>B'0: Disable interrupt of dsi_int_ta when TASR.TRNREQEND=1<br>B'1: Enable interrupt of dsi_int_ta when TASR.TRNREQEND=1 |

### 67.2.2.45 Timeout Set 0 Register (TOSET0R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HTXTOVL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HTXTOVL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | HTXTOVL | H'0000 0000 | R/W | Set Host Tx Timeout value (HTX_TO).<br>The limit Time for Timeout is calculated by formula of HTXTOVL * (period of hsclk).<br>When TOSR.HTXTO is "1", the Timeout is detected.<br>Notice, the Timeout is not detected when HTXTOVL=0. |

### 67.2.2.46 Timeout Set 1 Register (TOSET1R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LRXHTOVL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LRXHTOVL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | LRXHTOVL | H'0000 0000 | R/W | Set Host LP Rx Timeout value (LRX_H_TO).<br>The limit Time for Timeout is calculated by formula of LRXHTOVL * (period of lpclk).<br>When TOSR.LRXHTO is "1", the Timeout is detected.<br>Notice, the Timeout is not detected when LRXHTOVL=0. |

### 67.2.2.47 Timeout Set 2 Register (TOSET2R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TATOVL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TATOVL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | TATOVL | H'0000 0000 | R/W | Set Turn-Around Timeout value (TA_TO).<br>The limit Time for Timeout is calculated by formula of TATOVL * (period of lpclk).<br>When TOSR.TATO is "1", the Timeout is detected.<br>Notice, the Timeout is not detected when TATOVL=0. |

### 67.2.2.48 Timeout Status Register (TOSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TA TO | LRX HTO | HTX TO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 2 | TATO | B'0 | R | When this value is "1", the Turn-Around Timeout (TA_TO) is detected.<br>Write "1" to TOSCR.TATO register to clear this field. |
| 1 | LRXHTO | B'0 | R | When this value is "1", the Host LP Rx Timeout (LRX_H_TO) is detected.<br>Write "1" to TOSCR.LRXHTO register to clear this field. |
| 0 | HTXTO | B'0 | R | When this value is "1", the Host HS Tx Timeout (HTX_TO) is detected.<br>Write "1" to TOSCR.HTXTO register to clear this field. |

### 67.2.2.49 Timeout Status Clear Register (TOSCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TA TO | LRX HTO | HTX TO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W | R0/W | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 2 | TATO | B'0 | R0/W | Set to "1" after detection of the Turn-Around Timeout (TA_TO) to detect the next one.<br>B'0: No meaning, ignored<br>B'1: Clear TOSR.TATO |
| 1 | LRXHTO | B'0 | R0/W | Set to "1" after detection of the Host LP Rx Timeout (LRX_H_TO) to detect the next one.<br>B'0: No meaning, ignored<br>B'1: Clear TOSR.LRXHTO |
| 0 | HTXTO | B'0 | R0/W | Set to "1" after detection of the Host HS Tx Timeout (HTX_TO) to detect the next one.<br>B'0: No meaning, ignored<br>B'1: Clear TOSR.HTXTO |

### 67.2.2.50 Timeout Interrupt Enable Register (TOIER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TA TO | LRX HTO | HTX TO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 2 | TATO | B'0 | R/W | Interrupt enable for Turn-Around Timeout (TA_TO).<br>B'0: Disable interrupt of dsi_int_to when TOSR.TATO=1<br>B'1: Enable interrupt of dsi_int_to when TOSR.TATO=1 |
| 1 | LRXHTO | B'0 | R/W | Interrupt enable for Host LP Rx Timeout (LRX_H_TO).<br>B'0: Disable interrupt of dsi_int_to when TOSR.LRXHTO=1<br>B'1: Enable interrupt of dsi_int_to when TOSR.LRXHTO=1 |
| 0 | HTXTO | B'0 | R/W | Interrupt enable for Host HS Tx Timeout (HTX_TO).<br>B'0: Disable interrupt of dsi_int_to when TOSR.HTXTO=1<br>B'1: Enable interrupt of dsi_int_to when TOSR.HTXTO=1 |

### 67.2.2.51 PPI Set Register (PPISETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ULPSWKUP | | | — | — | — | — | — | — | — | DL0TRNDIS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CLEN | — | — | — | — | DLEN | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 26 to 24 | ULPS WKUP | H'0 | R/W | Set Twake-up period of ULPS.<br>This period is specified by count value of lpclk.<br>This period should be more than 1ms.<br>H'0: 20,480 count<br>H'1: 10,240 count<br>H'2: 5,120 count<br>H'3: 2,560 count<br>H'4: 1,280 count<br>H'5: 640 count<br>H'6: 320 count<br>H'7: 160 count |
| 23 to 17 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | DL0 TRNDIS | B'0 | R/W | Turn-Around is disabled by PPI Data Lane0 TurnDisable.<br>All actions listed below are prohibited when this DL0TRNDIS="1".<br>- Write "1" to TXCMCR.BTAREQ register<br>- Write "1" to TACR.TRNREQ register |
| 15 to 9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | CLEN | B'0 | R/W | Clock Lane is enabled by PPI Clock Lane Enable. |
| 7 to 4 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 3 to 0 | DLEN | H'0 | R/W | Data Lane is enabled by each PPI Data Lane Enable.<br>[0]: Data Lane0 Enable<br>[1]: Data Lane1 Enable<br>[2]: Data Lane2 Enable<br>[3]: Data Lane3 Enable<br>The value for this register is permitted for H'0, H'1, H'3, H'7 and H'F. Others are prohibited |

### 67.2.2.52 PPI Clock Lane Control Register (PPICLCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | TX REQ HS | — | — | — | — | — | — | TXUL PSE XT | TXUL PSC LK |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R0/W | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | TXREQHS | B'0 | R/W | Start HS Clock on Clock Lane by PPI Clock Lane TxRequestHS.<br>B'0: No HS transmission (TxRequestHS=0)<br>B'1: Request for HS transmission (TxRequestHS=1) |
| 7 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | TXULPS EXT | B'0 | R0/W | When write "1", request to exit from Ultra-low Power State (ULPS) by PPI Clock Lane TxUlpsExit. |
| 0 | TXULPS CLK | B'0 | R0/W | When write "1", request to transition to Ultra-low Power State (ULPS) by PPI Clock Lane TxUlpsClk. |

### 67.2.2.53 PPI Clock Lane Status Register (PPICLSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | HSTOLP | TOHS | FROMULPS | TOULPS | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ULPSACTN | STPST |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 27 | HSTOLP | B'0 | R | When this value is "1", the transition to LP from HS is occurred.<br>Write "1" to PPICLSCR.HSTOLP register to clear this field. |
| 26 | TOHS | B'0 | R | When this value is "1", the transition to HS is occurred.<br>Write "1" to PPICLSCR.TOHS register to clear this field. |
| 25 | FROMULPS | B'0 | R | When this value is "1", the Exit from ULSP is detected.<br>Write "1" to PPICLSCR.FROMULPS register to clear this field. |
| 24 | TOULPS | B'0 | R | When this value is "1", the Enter to ULSP is detected.<br>Write "1" to PPICLSCR.TOULPS register to clear this field. |
| 23 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | ULPSACTN | B'1 | R | Status of UlpsActiveNot on PPI Clock Lane. |
| 0 | STPST | *1 | R | Status of Stopstate on PPI Clock Lane. |

Note: *1 This Initial Value is depend on ppi_cl_stopstate.

### 67.2.2.54 PPI Clock Lane Status Clear Register (PPICLSCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | HSTOLP | TOHS | FROMULPS | TOULPS | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R0/W | R0/W | R0/W | R0/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 27 | HSTOLP | B'0 | R0/W | Set to "1" to clear PPICLSR.HSTOLP.<br>B'0: No meaning, ignored<br>B'1: Clear PPICLSR.HSTOLP。 |
| 26 | TOHS | B'0 | R0/W | Set to "1" to clear PPICLSR.TOHS.<br>B'0: No meaning, ignored<br>B'1: Clear PPICLSR.TOHS |
| 25 | FROMULPS | B'0 | R0/W | Set to "1" to clear PPICLSR.FROMULPS.<br>B'0: No meaning, ignored<br>B'1: Clear PPICLSR.FROMULPS |
| 24 | TOULPS | B'0 | R0/W | Set to "1" to clear PPICLSR.TOULPS.<br>B'0: No meaning, ignored<br>B'1: Clear PPICLSR.TOULPS |
| 23 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.55 PPI Clock Lane Interrupt Enable Register (PPICLIER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | HSTOLP | TOHS | FROMULPS | TOULPS | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 27 | HSTOLP | B'0 | R/W | Interrupt enable for the transition to LP from HS.<br>B'0: Disable interrupt of dsi_int_ppicl when PPICLSR.HSTOLP=1<br>B'1: Enable interrupt of dsi_int_ppicl when PPICLSR.HSTOLP=1 |
| 26 | TOHS | B'0 | R/W | Interrupt enable for the transition to HS.<br>B'0: Disable interrupt of dsi_int_ppicl when PPICLSR.TOHS=1<br>B'1: Enable interrupt of dsi_int_ppicl when PPICLSR.TOHS=1 |
| 25 | FROMULPS | B'0 | R/W | Interrupt enable for the transition from ULPS.<br>B'0: Disable interrupt of dsi_int_ppicl when PPICLSR.FROMULPS=1<br>B'1: Enable interrupt of dsi_int_ppicl when PPICLSR.FROMULPS=1 |
| 24 | TOULPS | B'0 | R/W | Interrupt enable for the transition to ULPS.<br>B'0: Disable interrupt of dsi_int_ppicl when PPICLSR.TOULPS=1<br>B'1: Enable interrupt of dsi_int_ppicl when PPICLSR.TOULPS=1 |
| 23 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.56 PPI Data Lane0 Control Register (PPIDL0CR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TXTRGESC | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R0/W | R0/W | R0/W | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 3 to 0 | TXTRGESC | H'0 | R0/W | Set to "1" to request the Each TxTriggerEsc on PPI Data Lane0.<br>H'0: TxTriggerEsc is not asserted<br>H'1: TxTriggerEsc[0] asserted<br>H'2: TxTriggerEsc[1] asserted<br>H'4: TxTriggerEsc[2] asserted<br>H'8: TxTriggerEsc[3] asserted<br>Others are prohibited. |

### 67.2.2.57 PPI Data Lane0 Status Register (PPIDL0SR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | ERRCLP1 | ERRCLP0 | ERRCTRL | ERRSYNESC | ERRESC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | DIR | TX2RX | RX2TX | ULPSACTN | STPST | RXULPSESC | RXLPDTESC | RXTRGESC | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | *1 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 20 | ERRCLP1 | B'0 | R | When this value is "1", the Contention Error of LP1 (ErrContentionLP1) is detected on PPI Data Lane0.<br>Write "1" to PPIDL0SCR.ERRCLP1 register to clear this field. |
| 19 | ERRCLP0 | B'0 | R | When this value is "1", the Contention Error of LP0 (ErrContentionLP0) is detected on PPI Data Lane0.<br>Write "1" to PPIDL0SCR.ERRCLP0 register to clear this field. |
| 18 | ERRCTRL | B'0 | R | When this value is "1", the Control Error (ErrControl) is detected on PPI Data Lane0.<br>Write "1" to PPIDL0SCR.ERRCTRL register to clear this field. |
| 17 | ERRSYNESC | B'0 | R | When this value is "1", the Sync Error of LPDT (ErrSyncEsc) is detected on PPI Data Lane0.<br>Write "1" to PPIDL0SCR.ERRSYNESC register to clear this field. |
| 16 | ERRESC | B'0 | R | When this value is "1", the Escape mode Entry Error (ErrEsc) is detected on PPI Data Lane0.<br>Write "1" to PPIDL0SCR.ERRESC register to clear this field. |
| 15 to 11 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 10 | DIR | B'0 | R | Direction (Tx or Rx) of PPI Data Lane0 |
| 9 | TX2RX | B'0 | R | When this value is "1", the the transition of Direction from 0 (Tx) to 1(Rx) is detected on PPI Data Lane0.<br>Write "1" to PPIDL0SCR.TX2RX register to clear this field. |
| 8 | RX2TX | B'0 | R | When this value is "1", the the transition of Direction from 1(Rx) to 0 (Tx) is detected on PPI Data Lane0.<br>Write "1" to PPIDL0SCR.RX2TX register to clear this field. |
| 7 | ULPSACTN | B'1 | R | Status of UlpsActiveNot on PPI Data Lane0. |
| 6 | STPST | *1 | R | Status of Stopstate on Data Lane0. |
| 5 | RXULPSESC | B'0 | R | Status of RxULPSEsc on PPI Data Lane0. |
| 4 | RXLPDTESC | B'0 | R | Status of RxLpdtEsc on Data Lane0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 to 0 | RXTRGESC | H'0 | R | When this value is "1", the Event of Escape mode Rx Trigger is detected on PPI Data Lane0.<br>Write "1" to PPIDL0SCR.RXTRGESC[n] register to clear this field. |

Note: *1 This Initial Value is depend on ppi_dl0_stopstate.

### 67.2.2.58 PPI Data Lane0 Status Clear Register (PPIDL0SCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | ERRCLP1 | ERRCLP0 | ERRCTRL | ERRSYNESC | ERRESC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R0/W | R0/W | R0/W | R0/W | R0/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TX2RX | RX2TX | — | — | — | — | RXTRGESC | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R0/W | R0/W | R | R | R | R | R0/W | R0/W | R0/W | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 20 | ERRCLP1 | B'0 | R0/W | Set to "1" to clear PPIDL0SR.ERRCLP1.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDL0SR.ERRCLP1 |
| 19 | ERRCLP0 | B'0 | R0/W | Set to "1" to clear PPIDL0SR.ERRCLP0.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDL0SR.ERRCLP0 |
| 18 | ERRCTRL | B'0 | R0/W | Set to "1" to clear PPIDL0SR.ERRCTRL.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDL0SR.ERRCTRL |
| 17 | ERRSYNESC | B'0 | R0/W | Set to "1" to clear PPIDL0SR.ERRSYNESC.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDL0SR.ERRSYNESC |
| 16 | ERRESC | B'0 | R0/W | Set to "1" to clear PPIDL0SR.ERRESC.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDL0SR.ERRESC |
| 15 to 10 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 9 | TX2RX | B'0 | R0/W | Set to "1" to clear PPIDL0SR.TX2RX.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDL0SR.TX2RX |
| 8 | RX2TX | B'0 | R0/W | Set to "1" to clear PPIDL0SR.RX2TX.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDL0SR.RX2TX |
| 7 to 4 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 3 to 0 | RXTRGESC | H'0 | R0/W | Set to "1" to clear PPIDL0SR.RXTRGESC[3:0] for each bit.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDL0SCR.RXTRGESC[n] |

### 67.2.2.59 PPI Data Lane0 Interrupt Enable Register (PPIDL0IER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | ERRCLP1 | ERRCLP0 | ERRCTRL | ERRSYNESC | ERRESC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TX2RX | RX2TX | — | — | — | — | RXTRGESC | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 20 | ERRCLP1 | B'0 | R/W | Interrupt enable for the Contention Error of LP1 on PPI Data Lane0 (ErrContentionLP1)<br>B'0: Disable interrupt of dsi_int_ppidl0 when PPIDL0SR.ERRCLP1=1<br>B'1: Enable interrupt of dsi_int_ppidl0 when PPIDL0SR.ERRCLP1=1 |
| 19 | ERRCLP0 | B'0 | R/W | Interrupt enable for the Contention Error of LP0 on PPI Data Lane0 (ErrContentionLP0)<br>B'0: Disable interrupt of dsi_int_ppidl0 when PPIDL0SR.ERRCLP0=1<br>B'1: Enable interrupt of dsi_int_ppidl0 when PPIDL0SR.ERRCLP0=1 |
| 18 | ERRCTRL | B'0 | R/W | Interrupt enable for the Control Error on PPI Data Lane0 (ErrControl)<br>B'0: Disable interrupt of dsi_int_ppidl0 when PPIDL0SR.ERRCTRL=1<br>B'1: Enable interrupt of dsi_int_ppidl0 when PPIDL0SR.ERRCTRL=1 |
| 17 | ERRSYNESC | B'0 | R/W | Interrupt enable for the LPDT Sync Error on PPI Data Lane0 (ErrSyncEsc)<br>B'0: Disable interrupt of dsi_int_ppidl0 when PPIDL0SR.ERRSYNESC=1<br>B'1: Enable interrupt of dsi_int_ppidl0 when PPIDL0SR.ERRSYNESC=1 |
| 16 | ERRESC | B'0 | R/W | Interrupt enable for the Escape mode Error on PPI Data Lane0 (ErrEsc)<br>B'0: Disable interrupt of dsi_int_ppidl0 when PPIDL0.ERRESC=1<br>B'1: Enable interrupt of dsi_int_ppidl0 when PPIDL0.ERRESC=1 |
| 15 to 10 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 9 | TX2RX | B'0 | R/W | Interrupt enable for PPIDL0SR.TX2RX<br>B'0: Disable interrupt of dsi_int_ppidl0 when PPIDL0SR.TX2RX=1<br>B'1: Enable interrupt of dsi_int_ppidl0 when PPIDL0SR.TX2RX=1 |
| 8 | RX2TX | B'0 | R/W | Interrupt enable for PPIDL0SR.RX2TX<br>B'0: Disable interrupt of dsi_int_ppidl0 when PPIDL0SR.RX2TX=1<br>B'1: Enable interrupt of dsi_int_ppidl0 when PPIDL0SR.RX2TX=1 |
| 7 to 4 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 to 0 | RXTRGESC | H'0 | R/W | Interrupt enable for Escape mode Rx Trigger Event on PPI Data Lane0<br>B'0: Disable interrupt of dsi_int_ppidl0 when PPIDL0SR.RXTRGESC[n]=1<br>B'1: Enable interrupt of dsi_int_ppidl0 when PPIDL0SR.RXTRGESC[n]=1 |

### 67.2.2.60 PPI Data Lane Control Register (PPIDLCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TXULPSEXT | TXULPSESC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | TXULPSEXT | B'0 | R0/W | Set to "1" to request the UPLS Exit by TxUlpsExit on all PPI Data Lane.<br>B'0: No meaning, ignored<br>B'1: Request the Exit from UPLS |
| 0 | TXULPSESC | B'0 | R0/W | Set to "1" to request the transition to ULPS by TxUlpsEsc on all PPI Data Lane.<br>B'0: No meaning, ignored<br>B'1: Request the transition to ULPS |

### 67.2.2.61 PPI Data Lane Status Register (PPIDLSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | FROMULPS | TOULPS | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ULPSACTN | | | | STPST | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | *1 | *1 | *1 | *1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25 | FROMULPS | B'0 | R | When this value is "1", the transition to StopState from ULPS by ULPS Exit Request is detected on all PPI Data Lane enabled by PPISETR.DLEN.<br>Write "1" to PPIDLSCR.FROMULPS register to clear this field. |
| 24 | TOULPS | B'0 | R | When this value is "1", the transition to ULPS by PPIDLCR.TXULPSESC is detected on all PPI Data Lane enabled by PPISETR.DLEN.<br>Write "1" to PPIDLSCR.TOMULPS register to clear this field. |
| 23 to 8 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 7 to 4 | ULPSACTN | H'F | R | Status of UlpsActiveNot on PPI Data Lane.<br>[0]: UlpsActiveNot of PPI Data Lane0<br>[1]: UlpsActiveNot of PPI Data Lane1<br>[2]: UlpsActiveNot of PPI Data Lane2<br>[3]: UlpsActiveNot of PPI Data Lane3 |
| 3 to 0 | STPST | *1 | R | Status of Stopstate on PPI Data Lane.<br>[0]: Stopstate of PPI Data Lane0<br>[1]: Stopstate of PPI Data Lane1<br>[2]: Stopstate of PPI Data Lane2<br>[3]: Stopstate of PPI Data Lane3 |

Note: *1 This Initial Value is depend on the value of ppi_dl0_stopstate, ppi_dl1_stopstate, ppi_dl2_stopstate and ppi_dl3_stopstate in Input Signal.

### 67.2.2.62 PPI Data Lane Status Clear Register (PPIDLSCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | FROMULPS | TOULPS | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R0/W | R0/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25 | FROMULPS | B'0 | R0/W | Set to "1" to clear PPIDLSR.FROMULPS.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDLSR.FROMULPS |
| 24 | TOULPS | B'0 | R0/W | Set to "1" to clear PPIDLSR.TOULPS.<br>B'0: No meaning, ignored<br>B'1: Clear PPIDLSR.TOULPS |
| 23 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.63 PPI Data Lane Interrupt Enable Register (PPIDLIER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | FROMULPS | TOULPS | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25 | FROMULPS | B'0 | R/W | Interrupt enable for the transition from ULPS on all PPI Data Lane<br>B'0: Disable interrupt of dsi_int_ppidl when PPIDLSR.FROMULPS=1<br>B'1: Enable interrupt of dsi_int_ppidl when PPIDLSR.FROMULPS=1 |
| 24 | TOULPS | B'0 | R/W | Interrupt enable for the transition to ULPS on all PPI Data Lane<br>B'0: Disable interrupt of dsi_int_ppidl when PPIDLSR.TOULPS=1<br>B'1: Enable interrupt of dsi_int_ppidl when PPIDLSR.TOULPS=1 |
| 23 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.64 Skew Calibration Control Register (SCALCR)

(for 2.5G)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CAL TYP | CAL REQ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | CALTYP | B'0 | R0/W | Choice of effective value for the period of TxSkewCalHS.<br>B'0: SCALPSR is effective when CALREQ=1<br>B'1: SCALPS1R is effective when CALREQ=1<br>This register is not effective when CALREQ=0. |
| 0 | CALREQ | B'0 | R0/W | Enable TxSkewCalHS on all Data Lane.<br>B'0: No meaning, ignored<br>B'1: Enable TxSkewCalHS with period based on the value of SCALPSR.PTNSZ or SCALPS1R.PTNSZ. |

### 67.2.2.65 Skew Calibration Status Register (SCALSR)

(for 2.5G)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CALREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | CALREQEND | B'0 | R | When this value is "1", the Completion of Skew Calibration by SCALCR.CALREQ is detected or the Completion of Skew Calibration by TXVMSET(F(i))R.VSEN=1 is detecteded.<br>Write "1" to SCALSCR.CALREQEND register to clear this field. |

### 67.2.2.66 Skew Calibration Status Clear Register (SCALSCR)

(for 2.5G)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CALREQEND |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | CALREQEND | B'0 | R0/W | Set to "1" to clear SCALSR.CALREQEND.<br>B'0: No meaning, ignored<br>B'1: Clear SCALSR.CALREQEND |

### 67.2.2.67 Skew Calibration Interrupt Enable Register (SCALIER)

(for 2.5G)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CAL REQ END |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | CALREQEND | B'0 | R/W | Interrupt enable for SCALSR.CALREQEND<br>B'0: Disable interrupt of dsi_int_scal when SCALSR.CALREQEND=1<br>B'1: Enable interrupt of dsi_int_scal when SCALSR.CALREQEND=1 |

### 67.2.2.68 Skew Calibration Pattern Size Register (SCALPSR)

(for 2.5G)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTNSZ | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTNSZ | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | PTNSZ | H'0000 0000 | R/W | Period for transmission of Skew Calibration Pattern on Skew Calibration. (Unit: hsclk) Asserts TxSkewCalHS for the duration of (field value +1) cycle period.<br>Refer to section 67.3.2.11.1 for detail.<br>This register is used for following.<br>Case 1: Set to "0" to SCALCR.CALTYP with Skew Calibration Request (SCALCR.CALREQ=1) by F/W<br>Case 2: Skew Calibration on Video mode (TXVMSET(F(i))R.VSEN=1)<br>It is inhibited to write "0" to this field at the case described above. |

### 67.2.2.69 Skew Calibration Pattern Size 1 Register (SCALPS1R)

(for 2.5G)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTNSZ | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PTNSZ | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | PTNSZ | H'0000 0000 | R/W | Period for transmission of Skew Calibration Pattern on Skew Calibration. (Unit: hsclk) Asserts TxSkewCalHS for the duration of (field value +1) cycle period.<br>Refer to section 67.3.2.11.1 for detail.<br>This register is used for following.<br>Case 1: Set to "1" to SCALCR.CALTYP with Skew Calibration Request (SCALCR.CALREQ=1) by F/W<br>It is inhibited to write "0" to this field at the case described above. |

### 67.2.2.70 IP Set Register (IPSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CSI2M |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | CSI2M | B'0 | R/W | Tx mode configuration for this Transmitter<br>B'0: DSI mode<br>B'1: CSI-2 mode |

### 67.2.2.71 Tx Video Mode Frame Type Control Register (TXVMFTCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FTCUR | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FTNXT | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 17, 16 | FTCUR | H'0 | R | Indicates the type of outbound frame you are currently sending while in video mode.<br>00b: Frame type 0 (set with TXVM*R)<br>01b: Frame type 1 (set with TXVM*F1R)<br>10b: Frame type 2 (set with TXVM*F2R)<br>11b: Frame type 3 (set with TXVM*F3R)<br>This field is updated to the frame boundary (at the start of the VSA period). |
| 15 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1, 0 | FTNXT | H'0 | R/W | During video mode operation, set the frame type to send to the next frame.<br>00b: Frame type 0 (set with TXVM*R)<br>01b: Frame type 1 (set with TXVM*F1R)<br>10b: Frame type 2 (set with TXVM*F2R)<br>11b: Frame type 3 (set with TXVM*F3R)<br>This field should be set before the currently transmitting frame is completed (at least before 10 usec before the next VSYNC start edge occurs).<br>See Chapter 5.2.4.3 for details. |

### 67.2.2.72 Tx Video Mode Set (Frame Type(i)) Register (TXVMSETF(i)R) (i=1,2,3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | VSEN | — | — | — | — |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 4 | VSEN | B'0 | R/W | Refer to 67.2.2.18 |
| 3 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.73 Tx Video Mode Set 2 (Frame Type(i)) Register (TXVMSET2F(i)R) (i=1,2,3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FNUM | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 15 to 0 | FNUM | H’0000 | R/W | Refer to 67.2.2.19 |

### 67.2.2.74 Tx Video Mode Pixel Stream Packet Header Set (Frame Type(i)) Register (TXVMPSPHSETF(i)R) (i=1,2,3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | VC | | DT | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/w |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 23, 22 | VC | B'00 | R/W | Refer to 67.2.2.24<br>Set a different VC value for each frame type. It is prohibited to set the same VC value for multiple frame types to use. |
| 21 to 16 | DT | H'00 | R/W | Refer to 67.2.2.24 |
| 15 to 0 | — | All 0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.75 Tx Video Mode Video Parameter Set 0 (Frame Type(i)) Register (TXVMVPRMSET0F(i)R) (i=1,2,3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CSPC | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | — | All 0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 5 to 4 | CSPC | B'00 | R/W | Refer to 67.2.2.25<br>It is possible to set for each frame type, but only CSPC with the same BPP can be mixed. |
| 3 to 0 | — | H'0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.76 Tx Video Mode Video Parameter Set 1 (Frame Type(i)) Register (TXVMVPRMSET1F(i)R) (i=1,2,3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | VACTIVE | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | — | VSA VW | — | — | VSA | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | B’0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 30 to 16 | VACTIVE | H’0000 | R/W | Refer to 67.2.2.26 |
| 15 | | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | VSAVW | B'0 | R/W | Allows the number of lines in a VSA period to vary.<br>0b : Not allowed<br>1b : Allow<br>Do not change the initial value in DSI mode. |
| 13 to 12 | — | H’0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 11 to 0 | VSA | H’000 | R/W | Refer to 67.2.2.26 |

### 67.2.2.77 Tx Video Mode Video Parameter Set 2 (Frame Type(i)) Register (TXVMVPRMSET2F(i)R) (i=1,2,3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | VFP VW | — | VFP | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | VBP VW | — | VBP | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | | B’0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 30 | VFPVW | B’0 | R/W | Allows the number of lines in a VFP period to vary.<br>0b : Not allowed<br>1b : Allow<br>Do not change the initial value in DSI mode. |
| 29 | — | B’0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 28 to 16 | VFP | H’0000 | R/W | Refer to 67.2.2.27 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | | B’0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 | VBPVW | B’0 | R/W | Allows the number of lines in a VBP period to vary.<br>0b : Not allowed<br>1b : Allow<br>Do not change the initial value in DSI mode. |
| 13 | — | All 0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 12 to 0 | VBP | H’0000 | R/W | Refer to 67.2.2.27 |

### 67.2.2.78 Tx Video Mode Video Parameter Set 3 (Frame Type(i)) Register (TXVMVPRMSET3F(i)R) (i=1,2,3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | HACTIVE | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSBYTE | | | | HSA | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | B’0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 30 to 16 | HACTIVE | H’0000 | R/W | Refer to 67.2.2.28 |
| 15 to 12 | HSBYTE | H’0 | R/W | Refer to 67.2.2.28 |
| 11 to 0 | HSA | H’000 | R/W | Refer to 67.2.2.28 |

### 67.2.2.79 Tx Video Mode Video Parameter Set 4 (Frame Type(i)) Register (TXVMVPRMSET4F(i)R) (i=1,2,3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | HFP | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | HBP | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 28 to 16 | HFP | H'0000 | R/W | Refer to 67.2.2.29 |
| 15 to 13 | — | All 0 | R | Reserved.<br>The read value is undefined. The write value should always be 0. |
| 12 to 0 | HBP | H'0000 | R/W | Refer to 67.2.2.29 |

### 67.2.2.80 Fault detection Set Register (FDSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | VFCENF3 | | — | — | — | — | — | — | VFCENF2 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | VFCENF1 | | — | — | — | — | — | — | VFCEN | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25, 24 | VFCENF3 | H'0 | R/W | Enable Tx Frame CRC at every Frame in Video mode for FrameType3 (TXVMFTCR.FTNXT=H'3)<br>[1]: Reserved<br>[0]: Tx Frame CRC of VSPD0 (B'0: Disable, B'1: Enable) |
| 23 to 18 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 17, 16 | VFCENF2 | H'0 | R/W | Enable Tx Frame CRC at every Frame in Video mode for FrameType2 (TXVMFTCR.FTNXT=H'2)<br>[1]: Reserved<br>[0]: Tx Frame CRC of VSPD0 (B'0: Disable, B'1: Enable) |
| 15 to 10 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 9 to 8 | VFCENF1 | H'0 | R/W | Enable Tx Frame CRC at every Frame in Video mode for FrameType1 (TXVMFTCR.FTNXT=H'1)<br>[1]: Reserved<br>[0]: Tx Frame CRC of VSPD0 (B'0: Disable, B'1: Enable) |
| 7 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1, 0 | VFCEN | H'0 | R/W | Enable Tx Frame CRC at every Frame in Video mode for FrameType0 (TXVMFTCR.FTNXT=H'0)<br>[1]: Reserved<br>[0]: Tx Frame CRC of VSPD0 (B'0: Disable, B'1: Enable) |

### 67.2.2.81 Fault detection Control Register (FDCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DTTEN | PPITEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | DTTEN | B'0 | R/W | Enable Tx Video Packet Inspection Timer<br>B'0: Disable<br>B'1: Enable<br>Refer to section 67.5 for detail. |
| 0 | PPITEN | B'0 | R/W | Enable Tx PPI Inspection Timer<br>B'0: Disable<br>B'1: Enable<br>Refer to section 67.5 for detail. |

### 67.2.2.82 Fault detection Status Register (FDSR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VINERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DTTACT | PPITACT | — | — | — | — | — | — | DTTO | PPITO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | VINERR | B'0 | R | When this value is “1”, unexpected Video signal input is detected.<br>Write “1” to FDSCR.VINERR register to clear this field.<br>Refer to section 67.5 for detail. |
| 15 to 10 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 9 | DTTACT | B'0 | R | Status of the Timer for Tx Packet Inspection at the internal of this module<br>B'0: Timer is disabled<br>B'1: Timer is Enabled<br>After the configuration of FDCR.DTTEN register, this Status is updated after some period which is related to the frequency of pclk and hs2clk[*1]/lpclk.<br>Following procedure is needed for re-start of the Timer after stopping.<br>1. Confirm DTTACT=1<br>2. Write "0" to FDCR.DTTEN register<br>3. Wait for DTTACT=0<br>4. Configure the Timer<br>5. Write "1" to FDCR.DTTEN register |
| 8 | PPITACT | B'0 | R | Status of the Timer for Tx Packet Inspection at the PPI end<br>B'0: Timer is disabled<br>B'1: Timer is Enabled<br>After the configuration of FDCR.PPITEN register, this Status is updated after some period which is related to the frequency of pclk and hs2clk[*1]/lpclk.<br>Following procedure is needed for re-start of the Timer after stopping.<br>1. Confirm PPITACT=1<br>2. Write "0" to FDCR.PPITEN register<br>3. Wait for PPITACT=0<br>4. Configure the Timer<br>5. Write "1" to FDCR.PPITEN register |
| 7 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | DTTO | B'0 | R | When this value is “1”, the transmission is not executed within the period specified by FDDTTMSETR register.<br>Write “1” to FDSCR.DTTO register to clear this field.<br>Refer to section 67.5 for detail. |
| 0 | PPITO | B'0 | R | When this value is “1”, the Tx PPI is not executed within the period specified by FDDTTMSETR register.<br>Write “1” to FDSCR.PPITO register to clear this field.<br>Refer to section 67.5 for detail. |

Note: *1 hs2clk: hsclk(byteclock)*1/2 (internal clock of Link)

### 67.2.2.83 Fault detection Status Clear Register (FDSCR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VINERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DTTO | PPITO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | VINERR | B'0 | R0/W | Set to "1" to clear the FDSR.VINERR.<br>B'0: No meaning, ignored<br>B'1: Clear FDSR.VINERR |
| 15 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | DTTO | B'0 | R0/W | Set to "1" to clear the FDSR.DTTO.<br>B'0: No meaning, ignored<br>B'1: Clear FDSR.DTTO |
| 0 | PPITO | B'0 | R0/W | Set to "1" to clear the FDSR.PPITO.<br>B'0: No meaning, ignored<br>B'1: Clear FDSR.PPITO |

### 67.2.2.84 Fault detection Interrupt Enable Register (FDIER)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VINERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DTTO | PPITO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | VINERR | B'0 | R/W | Interrupt enable for FDSR.VINERR<br>B'0: Disable interrupt of dsi_int_fd when FDSR.VINERR=1<br>B'1: Enable interrupt of dsi_int_fd when FDSR.VINERR=1 |
| 15 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | DTTO | B'0 | R/W | Interrupt enable for FDSR.DTTO<br>B'0: Disable interrupt of dsi_int_fd when FDSR.DTTO=1<br>B'1: Enable interrupt of dsi_int_fd when FDSR.DTTO=1 |
| 0 | PPITO | B'0 | R/W | Interrupt enable for FDSR.PPITO<br>B'0: Disable interrupt of dsi_int_fd when FDSR.PPITO=1<br>B'1: Enable interrupt of dsi_int_fd when FDSR.PPITO=1 |

### 67.2.2.85 Fault detection PPI Timer Set Register (FDPPITMSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | PPITCSCL | | — | — | PPITCVL | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPD TYP | — | PPITOSCL | | — | — | PPITOVL | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 29, 28 | PPITCSCL | H'0 | R/W | Multiplier for Tx PPI Inspection timer<br>B'00: multiplied by 1<br>B'01: multiplied by 1024 (2^10)<br>B'10: multiplied by 1048576 (2^20)<br>Others described above are prohibited. |
| 27, 26 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25 to 16 | PPITCVL | H'000 | R/W | Cycle count value for Tx PPI Inspection timer<br>At the following two cases, the Tx PPI is correct.<br>Case1: At SPDTYP=0, the cycle count per Tx PPI is over the value which is PPITCVL*PPITCSCL by hsclk.<br>Case2: At SPDTYP=1, the cycle count per Tx PPI is over the value which is PPITCVL*PPITCSCL by lpclk.<br>At FDCR.PPITEN=1, it is prohibited to set "0". |
| 15 | SPDTYP | B'0 | R/W | Transfer speed of the Packet<br>B'0: High Speed<br>B'1: Low Power |
| 14 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 13, 12 | PPITOSCL | H'0 | R/W | Multiplier for Timeout period of Tx PPI Inspection timer<br>B'00: multiplied by 1<br>B'01: multiplied by 1024 (2^10)<br>B'10: multiplied by 1048576 (2^20)<br>Others are prohibited. |
| 11, 10 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 to 0 | PPITOVL | H'000 | R/W | Timeout count for Tx PPI Inspection timer<br>At SPDTYP=0, the timeout count is calculated by PPITOVL*PPITOSCL*(hsclk period).<br>At SPDTYP=1, the timeout count is calculated by PPITOVL*PPITOSCL*(lpclk period).<br>At FDCR.PPITEN=1, it is prohibited to set "0". |

### 67.2.2.86 Fault detection Data Type Timer Set Register (FDDTTMSETR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPD TYP | — | DTTOSCL | | — | — | DTTOVL | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 15 | SPDTYP | B'0 | R/W | Transfer speed of the Packet<br>B'0: High Speed<br>B'1: Low Power |
| 14 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 13, 12 | DTTOSCL | H'0 | R/W | Multiplier for Timeout period of Tx Packet Inspection timer<br>B'00: multiplied by 1<br>B'01: multiplied by 1024 (2^10)<br>B'10: multiplied by 1048576 (2^20)<br>Others are prohibited. |
| 11, 10 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 9 to 0 | DTTOVL | H'000 | R/W | Timeout count for Tx Packet Inspection timer.<br>At SPDTYP=0, the timeout count is calculated by DTTOVL*DTTOSCL*(hsclk period). The value at SPDTYP=0 should be an even number, because the timer works by hs2clk[*1].<br>At SPDTYP=1, the timeout count is calculated by DTTOVL*DTTOSCL*(lpclk period).<br>At FDCR.DTTEN=1, it is prohibited to set "0". |

Note: *1 hs2clk: hsclk(byteclock)*1/2 (internal clock of Link)

### 67.2.2.87 Fault detection Data Type Timer Enable0 Register (FDDTEN0R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTENL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTENL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DTENL | H'0000 0000 | R/W | Configuration for Data Type of Tx Packet Inspection timer. (to H’1F from H’00)<br>[31]: Data Type H’1F (B'0: Disable, B'1: Enable)<br>[n]: Data Type n (B'0: Disable, B'1: Enable)<br>[0]: Data Type H’00 (B'0: Disable, B'1: Enable)<br>The character "n" is the number of the Data Type. |

### 67.2.2.88 Fault detection Data Type Timer Enable1 Register (FDDTEN1R)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DTENH | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTENH | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-0 | DTENH | H'0000 0000 | R/W | Configuration for Data Type of Tx Packet Inspection timer. (to H’3F from H’20)<br>[31]: Data Type H’3F (B'0: Disable, B'1: Enable)<br>[m]: Data Type m+H’20 (B'0: Disable, B'1: Enable)<br>[0]: Data Type H’20 (B'0: Disable, B'1: Enable)<br>The character "m" is the number related to the Data Type. |

### 67.2.2.89 lpclk frequency Set Register (LPCLKSET)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CKEN | LPCLKDIV | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | CKEN | B'0 | R/W | Enable for lpclk output<br>0: STOP<br>1: Enable |
| 7 to 0 | LPCLKDIV | H'14 | R/W | Divider configuration for frequency of lpclk=$T_{LPX\ (H)}$.<br>The value for $T_{LPX\ (H)}$ / $T_{LPX\ (P)}$ should be to 3/2 form 2/3 within 2MHz<=lpclk<20MHz.<br>The frequency of lpclk: DSIEXTϕ /((LPCLKDIV+1)*2) |

### 67.2.2.90 cfgclk frequency Set Register (CFGCLKSET)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CKEN | — | — | CFGCLKDIV | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | CKEN | B'0 | R/W | Enable for cfgclk output<br>0: STOP<br>1: Enable |
| 7, 6 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 5 to 0 | CFGCLKDIV | H'1F | R/W | Divider configuration for frequency of cfgclk of PHY.<br>This value should be used with Initial Value and is prohibited to change from Initial Value.<br>Frequency of cfgclk: DSIEXTϕ/( CFGCLKDIV +1) |

### 67.2.2.91 vclk output Set Register (VCLKSET)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CKEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | COLOR | — | DIV | | | BPP | | LANE | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31-17 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 16 | CKEN | B'0 | R/W | Clock output enable to Dotclk Generator from PLL<br>B'0: Disable<br>B'1: Enable |
| 15-9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | COLOR | B'0 | R/W | Set color space.<br>B'0:RGB<br>B'1:YCbCr(DSI)/YUV(CSI-2) |
| 7 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 6-4 | DIV | B'00 | R/W | Divider configuration for generating vclk (dotclk ) in Dotclock Generator.<br>This value should be same as the CLOCKSET2.vco_cntrl[5:3].<br>B'000: Divided by 2<br>B'001: Divided by 4<br>B'010: Divided by 8<br>B'011: Divided by 16<br>B’100: Divided by 32<br>B’101: Divided by 64 |
| 3, 2 | BPP | H'3 | R/W | ColorDepth for generating vclk ( dotclk ) in Dotclock Generator.<br>B'00: RGB 16Bit, YCbCr 4:2:2 16Bit /RGB 565, YUV422 8-bit<br>B'01: RGB 18Bit /RGB 666<br>B'10: RGB 18Bit Loosely<br>B'11: RGB 24Bit/RGB 888 |
| 1, 0 | LANE | H'3 | R/W | Tx Lane configuration for generating vclk ( dotclk ) in Dotclock Generator.<br>H'0: 1 Lane (Lane0 enabled)<br>H'1: 2 Lane (Lane0 and Lane1 enabled)<br>H'2: 3 Lane (Lane0, Lane1 and Lane2 enabled)<br>H'3: 4 Lane (Lane0, Lane1, Lane2 and Lane3 enabled)<br>The value of H'2 is prohibited in CSI-2 mode.<br>This value should be same as TXSETR.LANECNT. |

### 67.2.2.92 vclk output Enable Register (VCLKEN)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CKEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | CKEN | B'0 | R/W | Output enable for vclk(dotclk )<br>Set to "1" to enable for vclk after the configuration of VCLKSET.DIV, VCLKSET.BPP and VCLKSET.LANE.<br>H'0: STOP<br>H'1: Output enable |

### 67.2.2.93 PHY SETUP Register (PHYSETUP)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | | | hsfreqrange | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | SET | | | | | | — | — | — | — | — | — | shutdownz | rstz |
| Initial value: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 22 to 16 | hsfreqrange | H'00 | R/W | High-speed Frequency Range Selection.<br>Refer to<br>Table 67.14 for Operating Frequency Range selection.<br>The write access should be permitted for the case of Section 67.3.5 Startup Sequence. |
| 15, 14 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 13 to 8 | SET | B'100000 | R/W | To change from Initial Value is prohibited. |
| 7 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | shutdownz | B'0 | R/W | This line is used to place the PHY in shutdown. All analog blocks are in shutdown mode and the digital logic is reset.<br>The write access should be permitted for the case of Section 67.3.5 Startup Sequence. |
| 0 | rstz | B'0 | R/W | This line is used to place the digital section of the PHY in reset state.<br>The write access should be permitted for the case of Section 67.3.5 Startup Sequence. |

### 67.2.2.94 Clock Set 1 Register (CLOCKSET1)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | wakeuppll | — | — | — | — | — | — | — | — | — | — | — | lock | — |
| Initial value: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | clksel | | — | — | — | — | — | | shadow_clear | updateplll |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

—

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 29 | wakeuppll | 1 | R/W | <br>B’0: disable PLL<br>B’1: enable PLL |
| 28 to 18 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 17 | lock | B'0 | R | Lock state of PLL<br>B'0: PLL is not locked<br>B'1: PLL is locked |
| 16 to 10 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 9, 8 | clksel | B'01 | R/W | This value should be Initial Value. To change from Initial Value is prohibited.<br>Control for clock output of PLL<br>B'01: Clock generation |
| 7 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | shadow_clear | B'0 | R/W | This register is used for clearing of the PLL parameters.<br>The write access should be permitted for the case of Section 67.3.5 Startup Sequence. |
| 0 | updatepll | B'0 | R/W | This register is used for configuring of the PLL parameters.<br>The write access should be permitted for the case of Section 67.3.5 Startup Sequence and Section 67.3.2.8 Exit of ULPS. |

### 67.2.2.95 Clock Set 2 Register (CLOCKSET2)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | m | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | vco_cntrl | | | | | | — | — | — | — | n | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 25 to 16 | m | H'000 | R/W | m as parameter for clock control *1 |
| 15, 14 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 13 to 8 | vco_cntrl | H'00 | R/W | vco_cntrl as parameter for clock control *1 |
| 7 to 4 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 3 to 0 | n | H'0 | R/W | n as parameter for clock control *1 |

Note: *1 Refer to Section67.3.4 Clock configuration for detail.

### 67.2.2.96 Clock Set 3 Register (CLOCKSET3)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | prop_cntrl | | | | | | — | — | int_cntrl | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | cpbias_cntrl | | | | | | | — | — | — | — | — | — | gmp_cntrl | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 29 to 24 | prop_cntrl | H'00 | R/W | prop_cntrl as parameter for clock control *1 |
| 23, 22 | — | H'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 21 to 16 | int_cntrl | H'00 | R/W | int_cntrl as parameter for clock control *1 |
| 15 | — | B'0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 14 to 8 | cpbias_cntrl | H'00 | R/W | ChargePump bias control, as parameter for clock control *1 |
| 7 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1, 0 | gmp_cntrl | H'0 | R/W | gmp_cntrl as parameter for clock control *1 |

Note: *1 Refer to Section3.4 Clock configuration for detail.

### 67.2.2.97 PHY Test Interface Write Register (PHTW)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DWEN | TESTDIN_DATA | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CWEN | TESTDIN_CODE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 24 | DWEN | B'0 | R/W | The write access should be permitted for the case of Section 67.3.5 Startup Sequence. Others are prohibited.<br><br>TESTDIN_DATA enable<br>B'0: Disable<br>B'1: Enable<br>This bit is auto cleared when value of TESTDIN_DATA sets to PHY.<br>If DWEN is set to B’1, it should be read PHTW to confirm that DWEN is auto cleared to B’0. |
| 23 to 16 | TESTDIN_DATA | H'00 | R/W | TESTDIN_DATA Bit Set Values. |
| 15 to 9 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 8 | CWEN | B'0 | R/W | TESTDIN_CODE enable<br>B'0: Disable<br>B'1: Enable<br>This bit is auto cleared when value of TESTDIN_CODE sets to PHY.<br>If CWEN is set to B’1, it should be read PHTW to confirm that CWEN is auto cleared to B’0. |
| 7 to 0 | TESTDIN_CODE | H'0000 | R/W | TESTDIN_CODE Bit Set Values. |

### 67.2.2.98 PHY Test Interface Read Register (PHTR)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TESTDOUT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 23 to 16 | TESTDOUT | H’00 | R | Those bits are TESTDIN_DATA loopback value from PHY.<br>READ PHTR immediately after the following processing.<br>-PHTW.DWEN/CWEN is auto cleared after writing to PHTW.<br>Example)<br>Set PHTW to H’01020114.<br>Wait DWEN and CWEN becomes B’00.<br>READ PHTR = H’00020000. |
| 15 to 0 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |

### 67.2.2.99 PHY Test Interface Clear Register (PHTC)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TEST CLR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | TESTCLR | B'1 | R/W | This bit sets TESTCLR of the PHY.<br>This bit should be permitted to access only for the case of Section 67.3.5 Startup Sequence. Others are prohibited. |

### 67.2.2.100 Fault detection Control Register 2 (FDCR2)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EDCERRIN | EDCEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 1 | EDCERRIN | B'0 | R/W | Error injection to EDC Check function.B'0: NormalB'1: Error injection<br>Error injection can only be inserted when the DE signal from the DU module is asserted. |
| 0 | EDCEN | B'0 | R/W | Enable EDC Check<br>B'0: Disable<br>B'1: Enable<br>Check the EDC value added at VSPDWhen the display data becomes values set with the DOORn by DU, an error is output, please turn it off.<br>Refer to section 67.5 for detail. |

### 67.2.2.101 Fault detection Status Register 2 (FDSR2)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EDCERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | EDCERR | B'0 | R | When this value is “1”, EDC Error is detected.<br>Write “1” to FDSCR2.EDCERR register to clear this field.<br>Refer to section 67.5 for detail. |

### 67.2.2.102 Fault detection Status Clear Register 2 (FDSCR2)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EDC ERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R0/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | EDCERR | B'0 | R0/W | Set to "1" to clear the FDSR2.EDCERR.<br>B'0: No meaning, ignored<br>B'1: Clear FDSR2.EDCERR. |

### 67.2.2.103 Fault detection Interrupt Enable Register 2 (FDIER2)

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EDC ERR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value is undefined. The write value should always be 0. |
| 0 | EDCERR | B'0 | R/W | Interrupt enable for FDSR2.EDCERR<br>B'0: Disable interrupt of dsi_int_fd when FDSR2.EDCERR=1<br>B'1: Enable interrupt of dsi_int_fd when FDSR2.EDCERR=1 |

## 67.3 Operation

### 67.3.1 Interrupt

**Table 67.3 Interrupt Signals**

| Interrupt Signal | description | Interrupt Condition | Interrupt source Register |
|---|---|---|---|
| intreq_dsitxlink*_p[8] (called “dsi_int_txcm” in this document) | Interrupt of DSI Tx Command Transfer mode | Completion of Tx Request by TXCMCR.TXREQ | TXCMSR.TXREQEND |
| | | Detection of AXI Error at AXI Read | TXCMSR.AXIERR |
| | | Detection of discarded Command Transfer mode Packet in Video mode | TXCMSR.CLSNERR |
| intreq_dsitxlink*_p[7] (called “dsi_int_txvm” in this document) | Interrupt of DSI Tx Video mode | Start of HS Video Mode Tx | TXVMSR.STR |
| | | End of HSVideo Mode Tx | TXVMSR.STP |
| | | Detection of Discarded Tx Packet | TXVMSR.PDIS |
| | | Overrun/Underrun Error at Tx Video Mode FIFO | TXVMSR.VFERR |
| | | Detection of VSA period start | TXVMSR.VSADET |
| | | Detection of VBP period start | TXVMSR.VBPDET |
| intreq_dsitxlink*_p[6] (called “dsi_int_rxp” in this document) | Interrupt of DSI Rx Packet | Completion of request by TXCMCR.BTAREQ | RXPSR.BTAREQEND |
| | | Detection of Packet except for Acknowledge and Error Report Packet and EoTp Packet | RXPSR.RCVRESP |
| | | Received Acknowledge and Error Report Packet | RXPSR.RCVAKE |
| | | Received EoTp Packet | RXPSR.RCVEOT |
| | | Received ACK trigger | RXPSR.RCVACK |
| | | Received Packet with under 4Byte length | RXPSR.MLFERR |
| | | Detection of Rx ECC Error (over 1bit) | RXPSR.ECCERR |
| | | Received unexpected Packet | RXPSR.UEXPKTERR |
| | | Received unexpected Rx DT Packet | RXPSR.UEXDTERR |
| | | WC Error of Rx Long Packet | RXPSR.WCERR |
| | | CRC Error of Rx Long Packet | RXPSR.CRCERR |
| | | Detection of AXI Error at AXI Write | RXPSR.AXIERR |
| | | Overrun in Rx Buffer | RXPSR.OVRERR |
| | | ECC Error (1Bit) of Rx Packet | RXPSR.ECCERR1B |
| intreq_dsitxlink*_p[5] (called “dsi_int_ta” in this document) | Interrupt of DSI Turn-Around | Completion of BTA request by TACR.TRNREQ | TASR.TRNREQEND |
| | | Received Acknowledge and Error Report Packet | TASR.RCVAKE |
| | | Received EoTp Packet | TASR.RCVEOT |
| | | Received ACK trigger | TASR.RCVACK |
| | | Received Packet with under 4Byte length | TASR.MLFERR |
| | | Detection of Rx ECC Error (over 1bit) | TASR.ECCERR |
| | | Received unexpected Packet | TASR.UEXPKTERR |
| | | Received unexpected Rx DT Packet | TASR.UEXDTERR |
| | | ECC Error (1Bit) of Rx Packet | TASR.ECCERR1B |

| Interrupt Signal | description | Interrupt Condition | Interrupt source Register |
|---|---|---|---|
| intreq_dsitxlink*_p[4] (called “dsi_int_to” in this document) | Interrupt of DSI Timeout | Host Tx Timeout (HTX_TO) | TOSR.HTXTO |
| | | Host LP Rx Timeout (LRX_H_TO) | TOSR.LRXHTO |
| | | Turn-Around Timeout (TA_TO) | TOSR.TATO |
| intreq_dsitxlink*_p[3] (called “dsi_int_ppicl” in this document) | Interrupt of DSI D-PHY PPI Clock Lane | PPI Clock Lane Transition to ULPS | PPICLSR.TOULPS |
| | | PPI Clock Lane Transition from ULPS | PPICLSR.FROMULPS |
| | | PPI Clock Lane Transition to HS | PPICLSR.TOHS |
| | | PPI Clock Lane Transition to LP from HS | PPICLSR.HSTOLP |
| intreq_dsitxlink*_p[2] (called “dsi_int_ppidl0” in this document) | Interrupt of DSI D-PHY PPI Data Lane0 | PPI Data Lane0 Escape mode Rx Trigger Event[0] (Reset Trigger) | PPIDL0SR.RXTRGESC[0] |
| | | PPI Data Lane0 Escape mode Rx Trigger Event[1] | PPIDL0SR.RXTRGESC[1] |
| | | PPI Data Lane0 Escape mode Rx Trigger Event[2] (ACK Trigger) | PPIDL0SR.RXTRGESC[2] |
| | | PPI Data Lane0 Escape mode Rx Trigger Event[3] | PPIDL0SR.RXTRGESC[3] |
| | | PPI Data Lane0 Transition to Tx from Rx | PPIDL0SR.RX2TX |
| | | PPI Data Lane0 Transition to Rx from Tx | PPIDL0SR.TX2RX |
| | | PPI Data Lane0 Escape mode Entry Error (ErrEsc) | PPIDL0SR.ERRESC |
| | | PPI Data Lane0 LPDT Sync Error (ErrSyncEsc) | PPIDL0SR.ERRSYNESC |
| | | PPI Data Lane0 Control Error (ErrControl) | PPIDL0SR.ERRCTRL |
| | | PPI Data Lane0 LP0 Contention Error | PPIDL0SR.ERRCLP0 |
| | | PPI Data Lane0 LP1 Contention Error | PPIDL0SR.ERRCLP1 |
| intreq_dsitxlink*_p[1] ( called “dsi_int_ppidl” in this document) | DSI D-PHY All PPI Data Lane Interrupt | All PPI Data Lane Transition to ULPS | PPIDLSR.TOULPS |
| | | All PPI Data Lane Transition from ULPS | PPIDLSR.FROMULPS |
| intreq_dsitxlink*_p[0] (called “dsi_int_scal” in this document) | DSI Skew Calibration Interrupt | Skew Calibration Complete | SCALSR.CALREQEND (for 2.5G) |

The Interrupt signal consists of combination logic. The Interrupt Signal from a Status Register is enabled by the signal related Interrupt Enable Register. These interrupt signals are bound by OR logic.

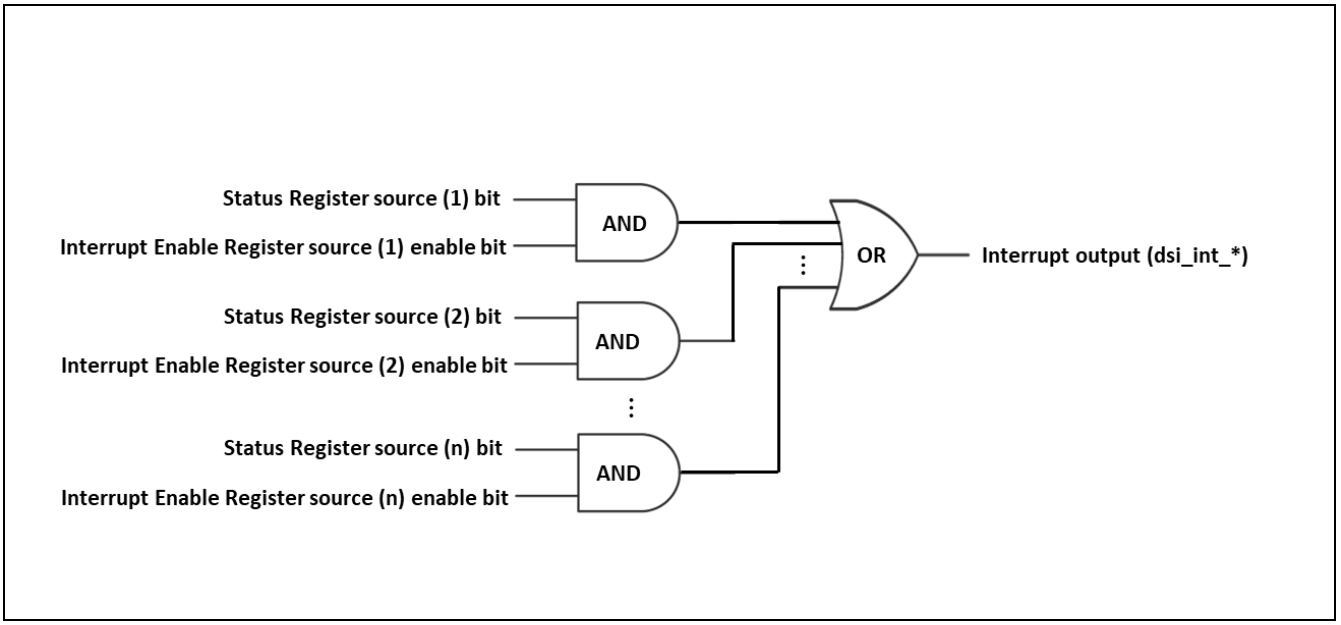



**Figure 67.3 Interrupt signal output**

### 67.3.2 Operation flow

Operation flow for DSI mode and CSI-2 mode are shown on Table 67.4.

○: Support

×: Not Support

-: Excluded

**Table 67.4 Operation flow for DSI/CSI-2 mode**

| Category | | DSI mode | CSI-2 mode |
|---|---|---|---|
| RESET | | ○ | ○ |
| Start/StopTx HS Clock/STOP HS Clock | | ○ | ○ |
| Command Transfer mode | | - | - |
| | Tx Short Packet | ○ | ○ |
| | Tx Long Packet | ○ | ○ |
| | Command Tx to request to some response of Peripheral device (Read Command) | ○ | × |
| | Tx Non-Read Command along with Bus Turn-Around | ○ | × |

| | | | | |
|---|---|---|---|---|
| Video mode | | | – | – |
| | Start of Video mode | | ○ | ○ |
| | End of Video mode | | ○ | ○ |
| | Different types of video data transmission (CSI-2 mode) | | × | ○ |
| | Compressed Video data transmission | | × | × |
| | Register Configuration (DSI mode) | | ○ | × |
| | Register Configuration (CSI-2 mode) | | × | ○ |
| Tx Reset Trigger | | | ○ | × |
| Bus Turn-Around Request | | | ○ | × |
| Error information of Peripheral device | | | ○ | × |
| Enter/Exit ULPS | | | ○ | ○ |
| Force transition to StopState | | | ○ | ○ |
| Error Handling procedure | | | – | – |
| | Error detection in D-PHY | | ○ | ○ |
| | Timeout Error | | – | – |
| | | Tx HS Timeout Error | ○ | ○ |
| | | Rx LP Timeout Error | ○ | × |
| | | Turn-Around Timeout Error | ○ | × |
| | | Rx Response Timeout Error | ○ | × |
| | Rx Packet Error | | ○ | × |
| | Error of handling for Rx Data Payload | | ○ | × |
| | Tx Command Transfer mode Error | | ○ | ○ |
| | Tx Video mode Error | | ○ | ○ |
| Tx Skew Calibration pattern | | | ○ | ○ |
| Fault detection | | | ○ | ○ |

#### 67.3.2.1 Software Reset

A software reset can be set by controlling the software reset register 4 (SRCR4) and software reset clear register 4 (SRSTCLR4) of the clock pulse generator (CPG) module. DSI/CSI-2-TX channels DSI/CSI-2-TX0, DSI/CSI-2-TX1 correspond to bits 15 and 16 in the SRCR4 and SRSTCLR4 registers, respectively. To apply a software reset to any of the DSI/CSI-2-TX channels, set the corresponding bit in the SRCR4 register. Enabling the bit initializes all the registers of the corresponding DSI/CSI-2-TX channel. To cancel a software reset of any of the DSI/CSI-2-TX channels, set the corresponding bit in the SRSTCLR4 register.

To apply a software reset to any of the DSI/CSI-2-TX channels, execute the following procedure using the register corresponding to the channel.

1. Set the corresponding bit in the SRCR4 register of the Clock Pulse Generator (CPG) module.

To cancel a software reset of any of the DSI/CSI-2-TX channels and start the channel, execute the following procedure using the register corresponding to the channel.

2. Set the corresponding bit in the SRSTCLR4 register of the Clock Pulse Generator (CPG) module.

This reset is used in Force transition to StopState transition (described in section 67.3.2.9) or Error Handling (described in section 67.3.2.10), Shutdown sequence (described in section 67.3.6).

#### 67.3.2.2 Start/Stop of HS clock

HS Tx Clock Start/Stop sequence is described in this section. HS Tx of PPI Clock Lane is controlled by Software.

Before HS Tx procedure, Start HS Tx Clock.

When write "1" to the PPICLCR.TXREQHS register, to request by TxRequestHS=1 to start the HS Tx of Clock Lane for D-PHY.

When transition to HS from LP of PPI Clock Lane is caused, dsi_int_ppicl interrupt is detected by PPICLSR.TOHS=1.

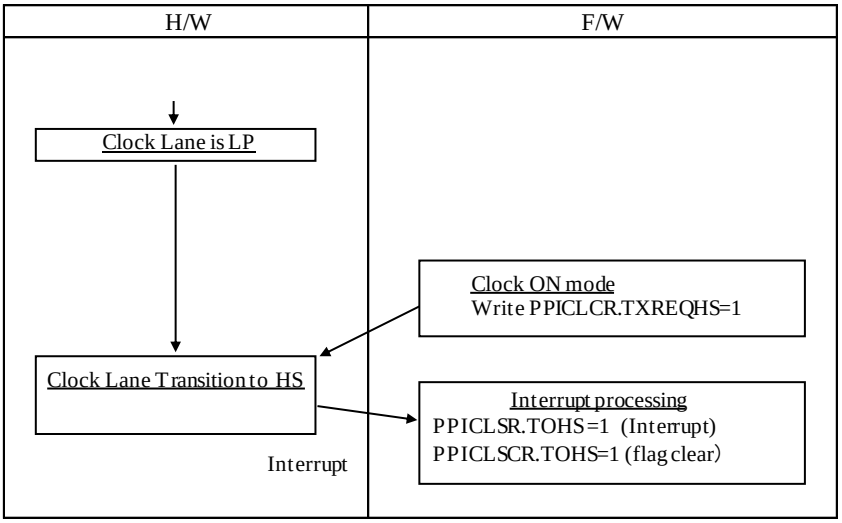
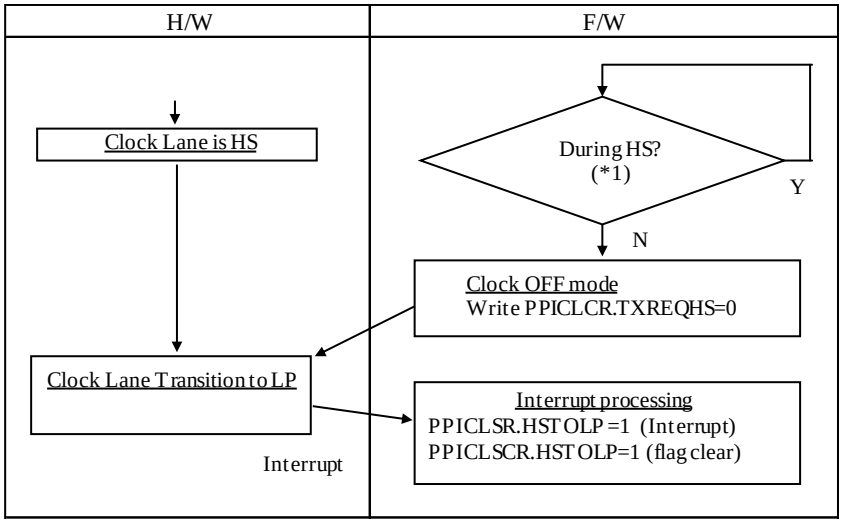



**Figure 67.4 Start of HS clock**

After the confirmation that it is not in HS Tx procedure, write "0" to the PPICLCR.TXREQHS register to request by TxRequestHS=0 to stop the HS Tx of Clock Lane for D-PHY.

When transition to LP from HS of PPI Clock Lane is caused, dsi_int_ppicl interrupt is detected by PPICLSR.HSTOLP=1.




(*1) One of the conditions described below is confirmed as "During HS".
- During processing the Command Transfer mode Packet in HS
- LINKSR.HSBUSY=1
- TXVMSR.ACT=1
- During Transmission of the Skew Calibration pattern

**Figure 67.5 Stop of HS clock**

### 67.3.2.3 Command transfer mode

#### 67.3.2.3.1 Tx Short Packet

Tx Short Packet sequence by using Command Transfer mode is described in this section.

As the initial configuration, set the Tx speed to TXCMSETR.SPDTYP register and set "1" to TXCMIER.TXREQEND register.

Then write VC, DT, DATA0 and DATA1 value to TXCMPHDR register as Packet Header. And write "0" to TXCMPHDR.FMT register as Short Packet. After that, write "1" to the TXCMCR.TXREQ register to start Tx Short Packet.

When completion of Tx request by TXCMCR.TXREQ is detected, the dsi_int_txcm interrupt is detected by TXCMSR.TXREQEND=1. This dsi_int_txcm interrupt is indicated to ready for next request.

Before confirmation of this interrupt, it is prohibited to start the other request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Rx Bus Turn-Around (described in section 67.3.2.6), Error information of peripheral device (described in section 67.3.2.7), Transition to ULPS (described in section 67.3.2.8) or Skew Calibration Pattern Tx (described in section 67.3.2.11).

The registers listed below are prohibited to configure during the period which is after writing to "1" to the TXCMCR.TXREQ register and is before changing to TXCMSR.TXREQEND=1.

TXCMSETR, TXCMPHDR

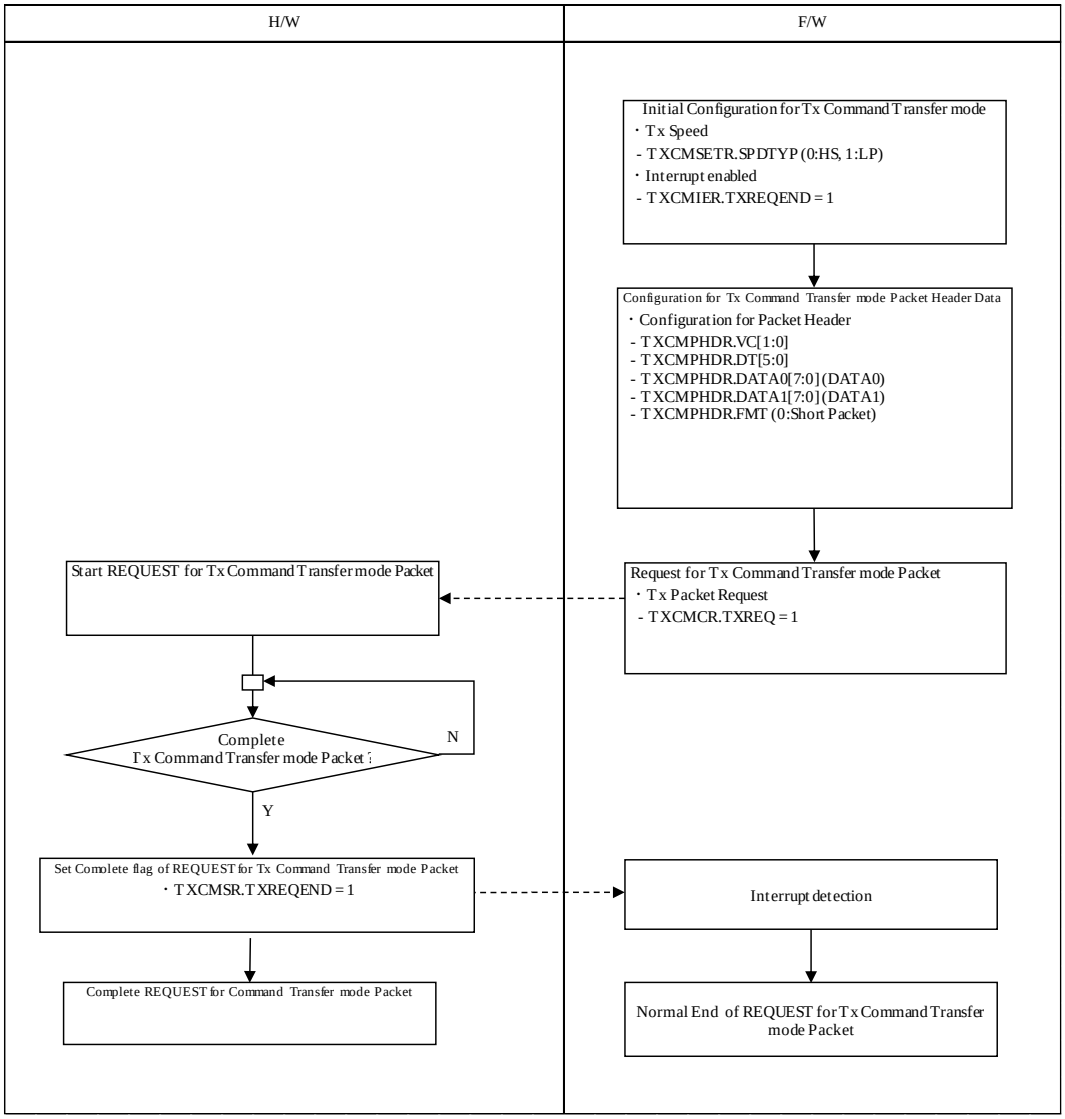



**Figure 67.6 Tx Short Packet**

#### 67.3.2.3.2 Tx Long Packet

##### 67.3.2.3.2.1 Register Access mode

Long Packet Tx by using Register Access mode supports the Payload Data only to 16 Byte from 0 Byte. Long Packet Tx with over 16 Byte as the Payload Data is supported by AXI Access mode.

Long Packet Tx sequence by using Register Access mode is described in this section.

As initial configuration, set the Tx speed to TXCMSETR.SPDTYP register, set "0" to the TXCMSETR.LPPDACC register as Register Access mode, and set "1" to TXCMIER.TXREQEND register.

Then write VC, DT, DATA0 (WC[7:0]) and DATA1 (WC[15:8]) value to the TXCMPHDR register as Packet Header. Write "1" to the TXCMPHDR.FMT register as Long Packet Tx.

Payload Data is written in the TXCMPPD[n]R(n=0,1,2,3) register with specified Byte size defined by the WC[15:0].

After that, write "1" to the TXCMCR.TXREQ register to start Tx Long Packet.

When completion of Tx request by TXCMCR.TXREQ is detected, dsi_int_txcm interrupt is detected by TXCMSR.TXREQEND=1.

This dsi_int_txcm interrupt is indicated to ready for next request. Before confirmation of this interrupt, it is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Rx Bus Turn-Around (described in section 67.3.2.6), Error information of peripheral device (described in section 67.3.2.7), Transition to ULPS (described in section 67.3.2.8) or Skew Calibration Pattern Tx (described in section 67.3.2.11).

The registers listed below are prohibited to configure during the period which is after writing to "1" to the TXCMCR.TXREQ register and is before changing to TXCMSR.TXREQEND=1.

TXCMSETR, TXCMPHDR, TXCMPPD0, TXCMPPD1, TXCMPPD2, TXCMPPD3R

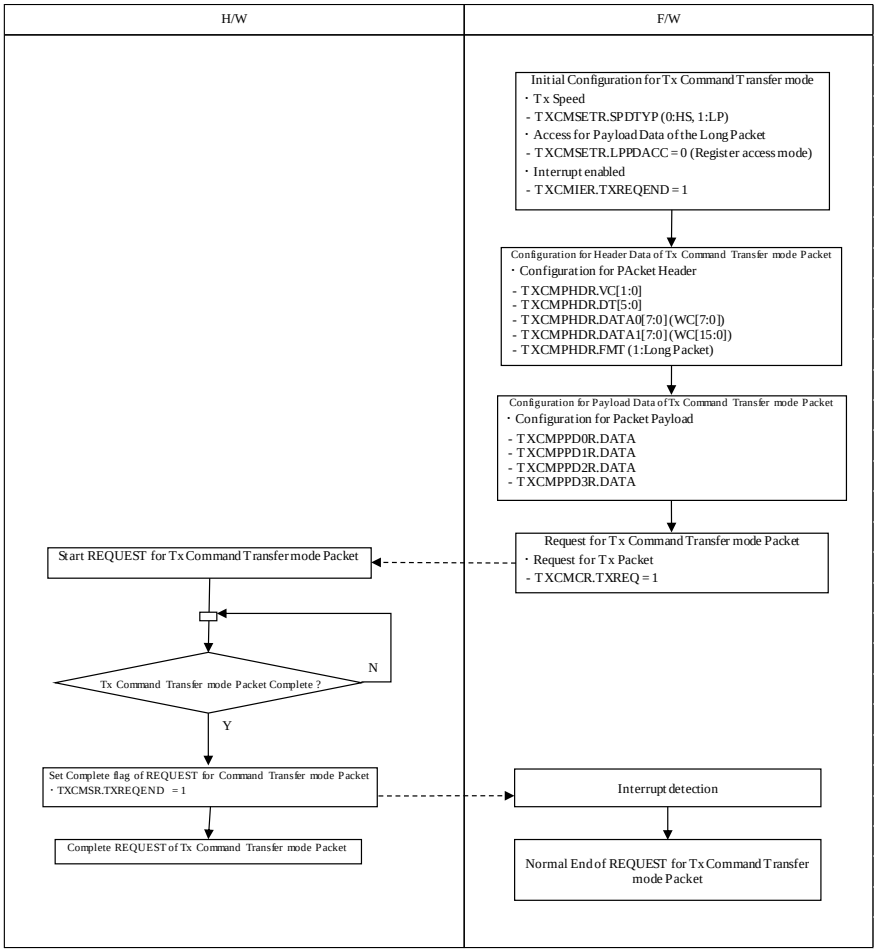



**Figure 67.7 Tx Long Packet (Register Access mode)**

#### 67.3.2.3.2.2 AXI Access mode

Long Packet Tx by using AXI Access mode supports the Payload Data up to 128 Byte in LP mode and 1024 Byte in HS mode from 0 Byte.

Long Packet Tx sequence by using AXI Access mode is described in this section.
As initial configuration, set the Tx speed to TXCMSETR.SPDTYP register, set "1" to the TXCMSETR.LPPDACC register as AXI Access mode, set the start address for Payload Data Memory to the TXCMADDRSET0R.ADDRL register, and set "1" to TXCMIER.TXREQEND register.

Then write VC, DT, DATA0(WC[7:0]) and DATA1(WC[15:8]) value to the TXCMPHDR register as Packet Header.

Write "1" to the TXCMPHDR.FMT register as Long Packet.

Payload Data is written in the Memory with specified Byte size defined by the WC[15:0]. Refer to Figure 67.8 for the alignment of the Payload Data in the Memory.

After that, write "1" to the TXCMCR.TXREQ register to start Tx Long Packet. When completion of Tx request by TXCMCR.TXREQ is detected, dsi_int_txcm interrupt is detected by TXCMSR.TXREQEND=1. This dsi_int_txcm interrupt is indicated to ready for next request.

After this interrupt, the result of the Tx Request is indicated by the TXCMSR register.

The failure of the Tx Long Packet is indicated by the TXCMSR.AXIERR register, the TXCMSR.UNRERR register or the TXCMSR.CLSNERR register. The Payload Data may re-transmit if needed.

Before confirmation of this interrupt, it is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Rx Bus Turn-Around (described in section 67.3.2.6), Error information of peripheral device (described in section 67.3.2.7), Transition to ULPS (described in section 67.3.2.8) or Skew Calibration Pattern Tx (described in section 67.3.2.11).

The registers listed below are prohibited to configure during the period which is after writing to "1" to the TXCMCR.TXREQ register and is before changing to TXCMSR.TXREQEND=1.

TXCMSETR, TXCMADDRSET0R, TXCMPHDR

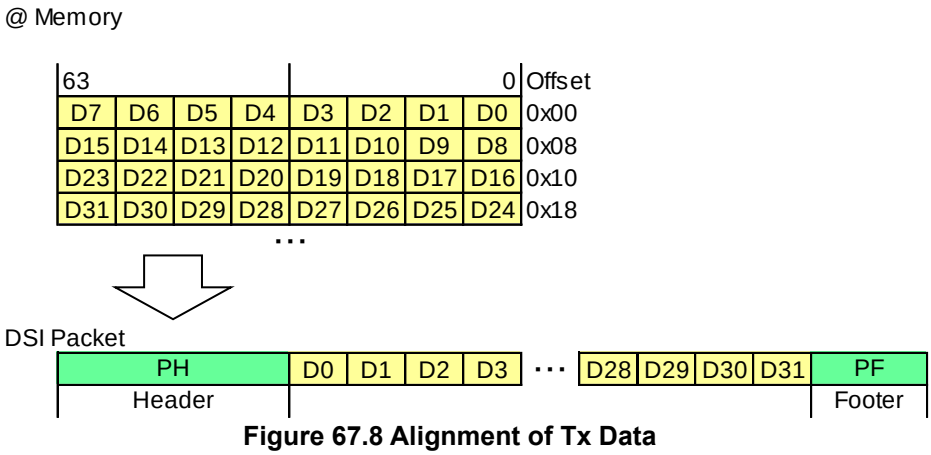
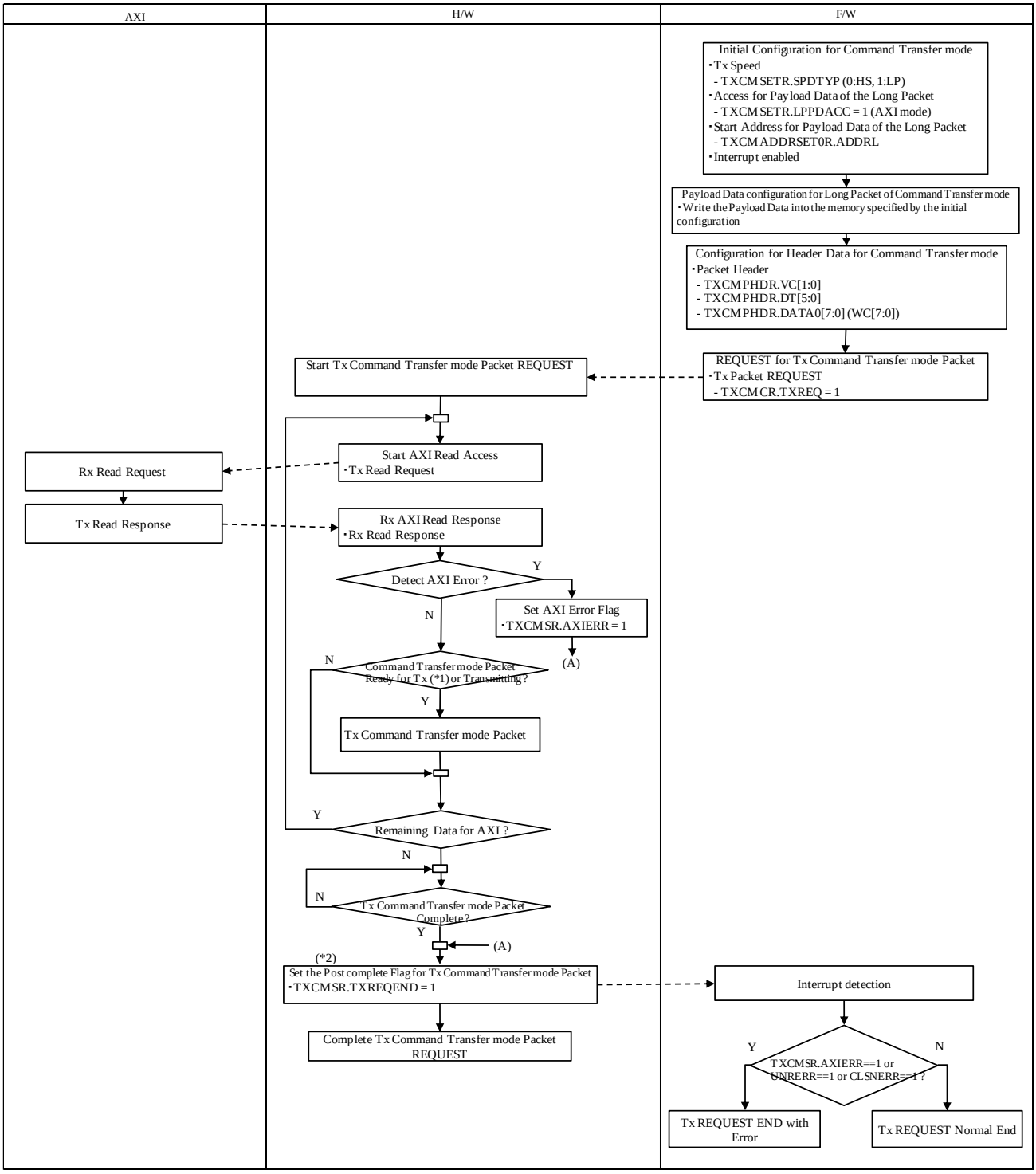



**Figure 67.8 Alignment of Tx Data**




(*1) Tx Command Transfer mode Packet is started by the Data over the WC or the threshold value specified by TXCMPHDR register for HS Tx Buffer.

(*2) The Flag is set after the completion of AXI Read for Outstanding and after the completion of Post for Tx Command Transfer mode Packet .

**Figure 67.9 Tx Long Packet (AXI Access mode)**

#### 67.3.2.3.3 Command Tx to request some response of Peripheral device (Read Command)

When a Host Processor sends the command to request some response of a peripheral device in Command Transfer mode as Read Command, the Transmitter requests the Bus Turn-Around after the completion of the Read Command and hands over the Bus Control to the peripheral device.

After the Bus handover, the peripheral device sends “Response Packet” or “Acknowledge and Error Report Packet” by using LP mode. After this Packet transfer is completed, the peripheral device requests the Bus Turn-Around to hand over the Bus Control. And the Bus is controlled by the Host Processor.

Command Tx sequence by Host Processor to request some response of Peripheral device is described in this section. Following is needed as initial configuration.

・Set the Tx speed to TXCMSETR.SPDTYP register

・Select the place for received Payload Data to storage by the RXPSETR.LPPDACC register (0: Register, 1: Memory on AXI Bus)

In case of Memory on AXI Bus for received Payload Data:

・Set the Start Address to the RXPADDRSET0R.ADDRL register

・Set the Size of the Memory to the RXPSIZESETR.SIZE register. Maximum value is 128 Bytes.

・Set “1” to the RXPIER.BTAREQEND register and set “1” to the TOIER.TATO register.

In case of Register for received Payload Data by RXPSETR.LPPDACC=0, the Register for the Rx Payload Data supports up to 16 Byte from 0 Byte. If the Rx Payload Data over 16 Bytes is needed, Memory on AXI Bus should be selected by RXPSETR.LPPDACC=1. Memory on AXI Bus for Rx Payload Data supports up to 128 Byte from 0 Byte. The alignment of the Rx Payload Data in the Memory is same as Tx. (Figure 67.8).

Then write VC, DT, DATA0 (WC[7:0]) and DATA1 (WC[15:8]) value to the TXCMPHDR register as Packet Header.

Write "0" to the TXCMPHDR.FMT register as Short Packet. After that, write "1" to the TXCMCR.BTAREQ register with TXCMCR.BTATYP=0 to start the Command Tx using Short Packet to request some response of Peripheral device.

The Transmitter requests the Bus Turn-Around after the completion of the Command Tx and hands over the Bus Control to the peripheral device.

When the Bus Turn-Around request is not completed within the time specified by the TOSET2R.TATOVL register, dsi_int_to interrupt is detected by TOSR.TATO=1. After detection of this interrupt, software should assert Software RESET and should recover the procedure or should re-transmit of the Command Tx.

When the Bus Turn-Around request is completed within the time specified by the TOSET2R.TATOVL register, the Host Processor hands over the Bus Control to the peripheral device. And the Host Processor starts the Rx Response Timer. The Rx Response Timer continues to count until detection of the receipt of Packet.

When "the receipt of Response Packet", “the receipt of Acknowledge and Error Report Packet” or "Timeout of Rx Response Timer" is detected, dsi_int_rxp interrupt is detected by RXPSR.BTAREQEND=1.

After this interrupt, the result of the Rx Packet is indicated by the RXPSR register.

The following sequence is needed to enable next request.

1. When RXPSR.RCVRESP=1 is indicated, Packet Header of Rx Response Packet is correct. The Packet Header is

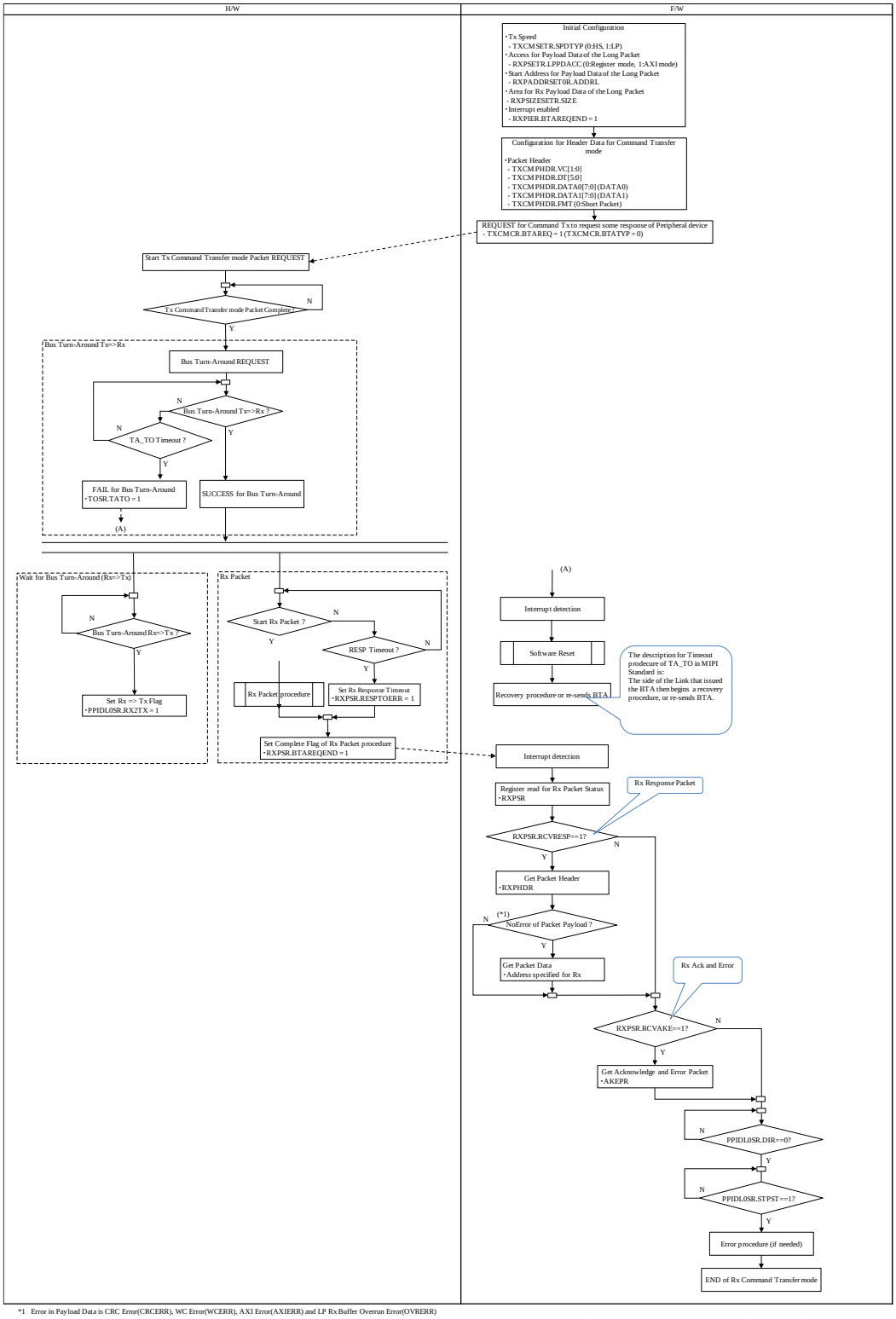
acquired by reading the RXPHDR register.
When all of RXPSR.CRCERR, RXPSR.WCERR, RXPSR.AXIERR and RXPSR.OVRERR are "0", Payload Data is correct. The Payload Data is acquired by reading the storage specified by RXPSETR.LPPDACC register.
When RXPSR.RCVRESP=0 is indicated, Packet Header of Rx Response Packet is not received.

2. When RXPSR.RCVAKE=1 is indicated, the Acknowledge and Error Report is received. The Data is acquired by reading the AKEPR register. When RXPSR.RCVAKE=0 is indicated, the Acknowledge and Error Report is not received.
3. Confirm the Bus handover to the Host Processor by reading PPIDL0SR.DIR register. Wait until finish of the handover procedure to the Host Processor, if that is not yet.
4. Confirm the StopState (LP11) of the Data Lane by reading PPIDL0SR.STPST register.
5. When this Error is detected about the Rx Packet, Error Handling procedure may be needed by referring to section 67.3.2.10 if needed.

Before confirmation of next request to be enabled, it is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Rx Bus Turn-Around (described in section 67.3.2.6), Error information of peripheral device (described in section 67.3.2.7), Transition to ULPS (described in section 67.3.2.8) or Skew Calibration Pattern Tx (described in section 67.3.2.11).

The registers listed below are prohibited to configure during the period which is after writing to "1" to the TXCMCR.BTAREQ register and is before changing to RXPSR.BTAREQEND=1.

TXCMSETR, TXCMPHDR, RXPSETR, RXPADDRSET0, RXPSIZESETR




*1 Error in Payload Data is CRC Error(CRCERR), WC Error(WCERR), AXI Error(AXIERR) and LP Rx Buffer Overrun Error(OVRERR)

**Figure 67.10 Tx Command request with response from peripheral device**

#### 67.3.2.3.4 Tx Non-Read Command along with Bus Turn-Around

Until Bus handover along with Bus Turn-Around request, any Error information is stored in the Peripheral device. When the Peripheral device gets the Bus control by handover after some Rx Command (except for the Read Command), the Peripheral device sends ACK Trigger by the reason of no Error and sends Acknowledge and Error Report Packet by the reason of some Error.

When the Peripheral device gets the Bus control after Rx Read Command, the Peripheral device sends Acknowledge and Error Report Packet if some Error is stored in the Peripheral device. Refer to section 67.3.2.3.3 for detail of the Bus Turn-Around by Read Command.

When the Host Processor sends Non-Read Command with Bus Turn-Around in the Command Transfer mode, the Transmitter requests the Bus Turn-Around and hands over the Bus Control to Peripheral device after this Tx Non-Read Command.

After the handover to the Peripheral device, the Peripheral device sends ACK Trigger or Acknowledge and Error Report Packet by LP mode. After these Tx, the Bus Turn-Around is requested by the Peripheral device. The Host Processor gets the Bus Control by the handover.

The Procedure for Non-Read Command Tx with Bus Turn-Around is described in this section.

Following is needed as initial configuration.

- Set Tx Command speed to TXCMSETR.SPDTYP register
- Set "1" to RXPIER.BTAREQEND register and TOIER.TATO register

Refer to section 67.3.2.3.1 for Short Packet Tx, section 67.3.2.3.2.1 for Long Packet Tx by Register Access mode or 67.3.2.3.2.2 for Long Packet Tx by AXI Access mode. Notice, to write “1” to the TXCMCR.BTAREQ register with TXCMCR.BTATYP=1 is needed instead of using the TXCMCR.TXREQ register.

After this Tx, The Transmitter requests Bus Turn-Around as Host Processor, and waits for the handover of Bus Control to the Peripheral device.

When the Bus Turn-Around request is not completed within the time specified by the TOSET2R.TATOVL register, dsi_int_to interrupt is detected by TOSR.TATO=1. After this interrupt, the Recovery procedure or re-transmission of the Command is needed after Software RESET.

When the Bus Turn-Around request is completed within the time specified by the TOSET2R.TATOVL register, the Host Processor hands over the Bus Control to the Peripheral device. And the Host Processor starts the Rx Response Timer.

The Rx Response Timer continues to count until detection of the receipt of Trigger or Packet.

When Rx of "ACK Trigger", Rx of "Acknowledge and Error Report Packet" or "Timeout of the Rx Response Timer" is detected, dsi_int_rxp interrupt is indicated by RXPSR.BTAREQEND=1.

After this interrupt, the result of the Rx Packet is indicated by the RXPSR register.

The following sequence is needed to enable next request.

1. (A) When RXPSR.RCVACK=1 is indicated, the ACK Trigger is received. This means that No Error is detected by the Peripheral device.

(B) When RXPSR.RCVAKE=1 is indicated, the Acknowledge and Error Report Packet is received. Read the AKEPR register to know the Error Information of the Peripheral device.

2. Confirm the Bus handover to the Host Processor by reading PPIDL0SR.DIR register. Wait until finish of the handover procedure to the Host Processor, if that is not yet.
3. Confirm the StopState (LP11) of the Data Lane by reading PPIDL0SR.STPST register.
4. When this Error is detected about the Rx Packet, Error Handling procedure may be needed by referring to section 67.3.2.10 if needed.

Before confirmation of next request to be enabled, it is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Rx Bus Turn-Around (described in section 67.3.2.6), Error information of peripheral device (described in section 67.3.2.7), Transition to ULPS (described in section 67.3.2.8) or Skew Calibration Pattern Tx (described in section 67.3.2.11).

The registers listed below are prohibited to configure during the period which is after writing to "1" to the TXCMCR.BTAREQ register and is before changing to RXPSR.BTAREQEND=1.

TXCMSETR, TXCMADDRSET0, TXCMPHDR, TXCMPPD0, TXCMPPD1, TXCMPPD2, TXCMPPD3R

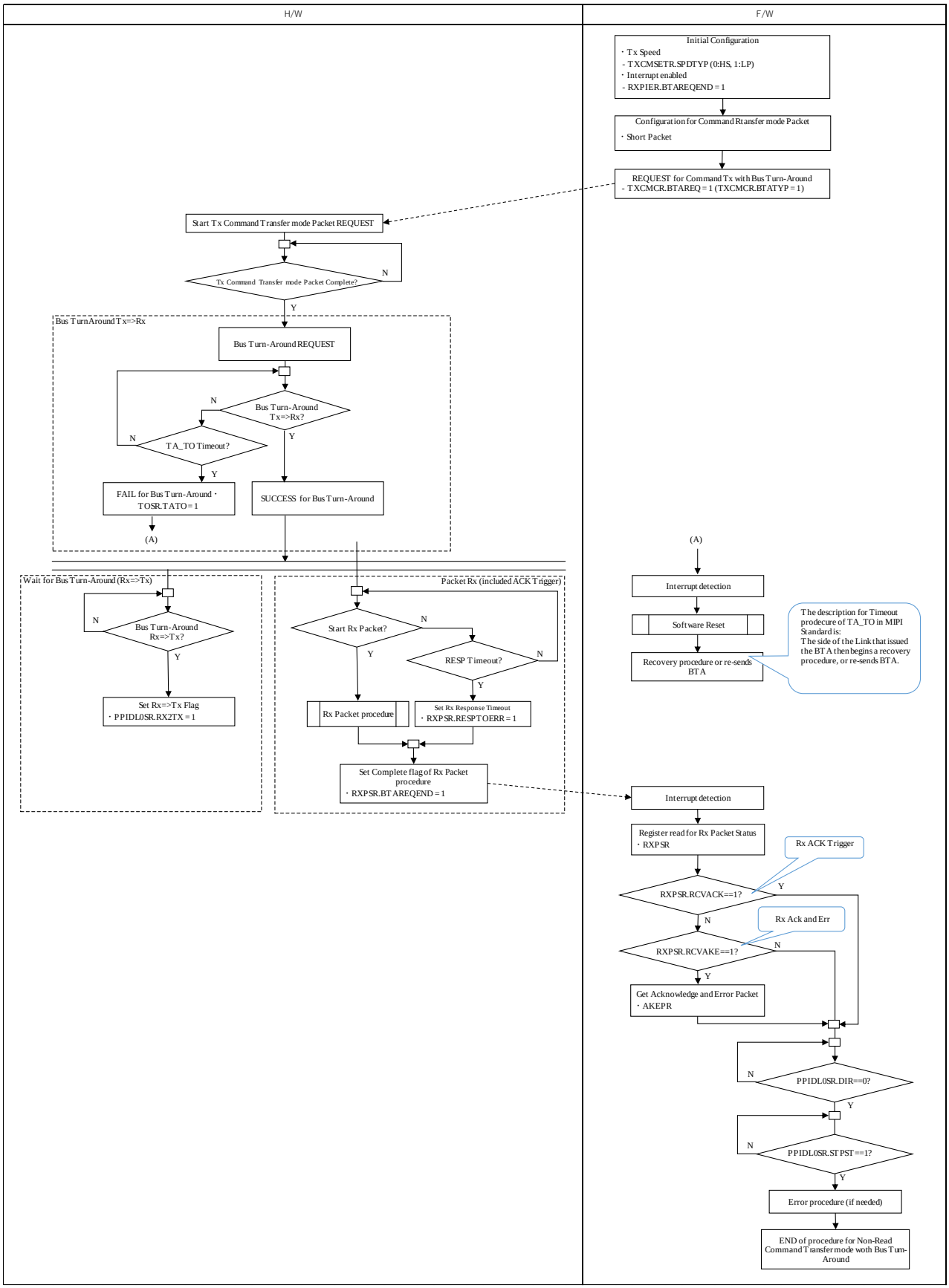



**Figure 67.11 Tx Non-Read Command with Bus Turn-Around**

#### 67.3.2.4 Video mode

##### 67.3.2.4.1 Start of Video mode

The procedure for Start of Video mode is shown in Figure 67.12. Before start of video mode operation, follow the procedure in Figure 67.4 to set HS clock transmission state.

After the initial configuration of Video mode, confirm that HS/LP operation is not running, and set "0" to TXVMCR.VFCLR register to withdraw the request of Clear FIFO. Wait until TXVMSR.VFRDY=1. Then set "1" to TXVMCR.EN register. Wait until TXVMSR.RDY=1. After that, start the Video signal input. When Video signal input is started, TXVMSR.ACT=1 is indicated.

Confirm the completion of the transaction of Command Transfer mode before start of Video mode. Following is prohibited until the start of Video mode indicated by TXVMSR.ACT=1.

- End of the Video mode
- Start of Command Transfer mode

The registers listed below are prohibited to configure during the period of TXVMCR.VFCLR=0.

TXVMSETR, TXVMSET2R, TXVMPSPHSETR, TXVMVPRMSET0, TXVMVPRMSET1, TXVMVPRMSET2, TXVMVPRMSET3, TXVMVPRMSET4R, TXVMSETF(i)R, TXVMSET2F(i)R, TXVMPSPHSETF(i)R, TXVMVPRMSET0,1,2,3,4F(i)R, SCALPSR, FDSETR

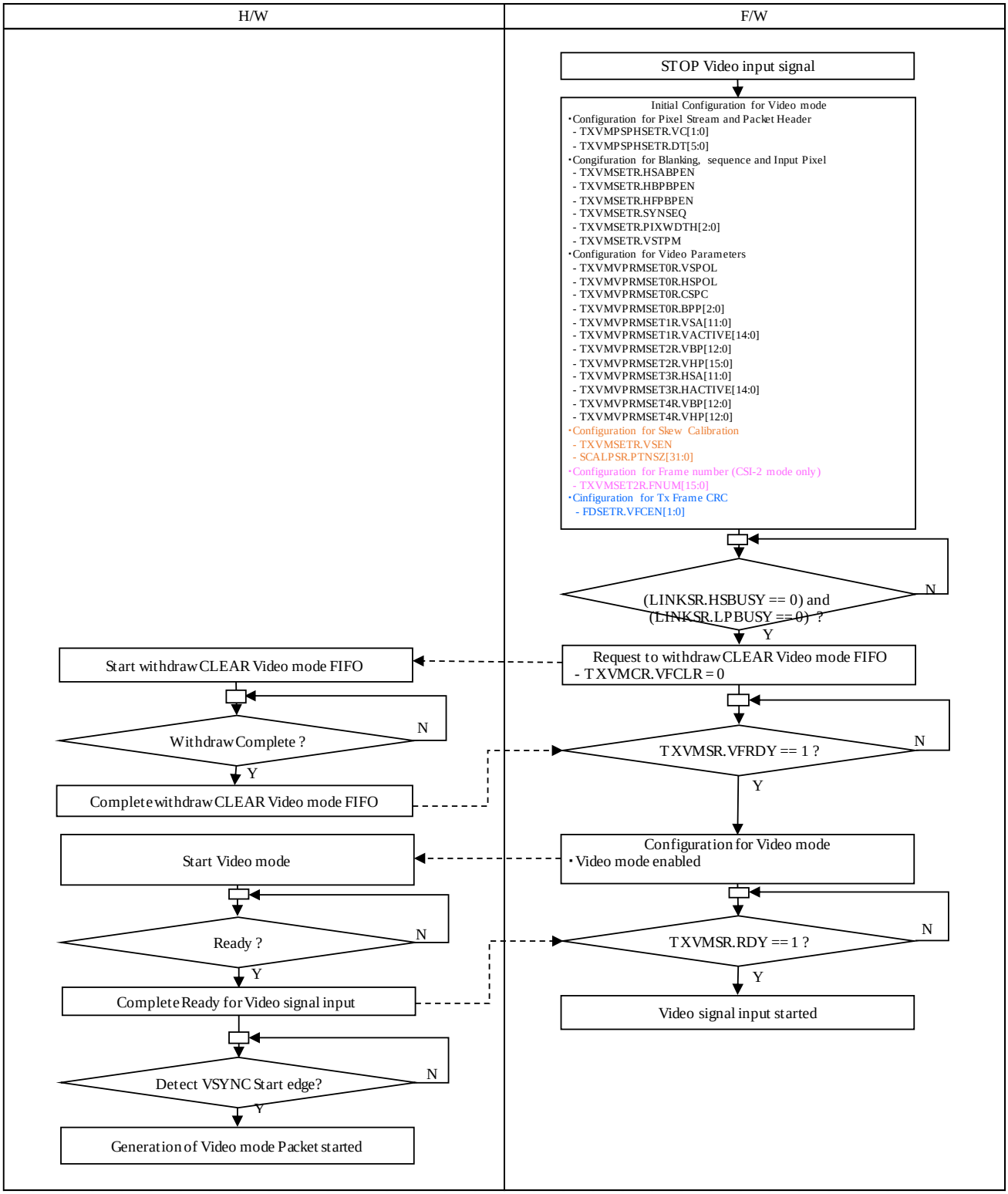



**Figure 67.12 Start of Vide mode**

#### 67.3.2.4.2 End of Video mode

The procedure for End of Video mode is shown in Figure 67.13.

When both of the start of first Frame (VSYNC assertion) and the disabling Video mode (TXVMCR.EN=0) are detected, the Transmitter stops Tx Packet in Video mode and clears TXVMSR.ACT register to "0".

Stop the Video mode after the confirmation of TXVMSR.ACT=0. After the detection of TXVMSR.ACT=0, the Timing for stopping of Video signal input is free. After the stop of the Video signal input, set "1" to TXVMCR.VFCLR register to request Clear FIFO and wait for TXVMSR.VFRDY=0.

Following is permitted to change the configuration only during the state of TXVMSR.VFRDY=0 described above.

- Change configuration of Video mode
- Change Lane Count

Command mode packets can be sent and received during video mode operation (see Chapters 67.4.1.1.1.1 and 67.4.1.1.2.3).

If you request command mode packet transmission (light 1 to TXCMCR.TXREQ or TXBTAREQ) during video mode operation, perform the video mode operation termination process after confirming that the send completion flag (TXCMSR.TXREQEND) has been set.

It is prohibited to terminate video mode operation during command mode packet requests.

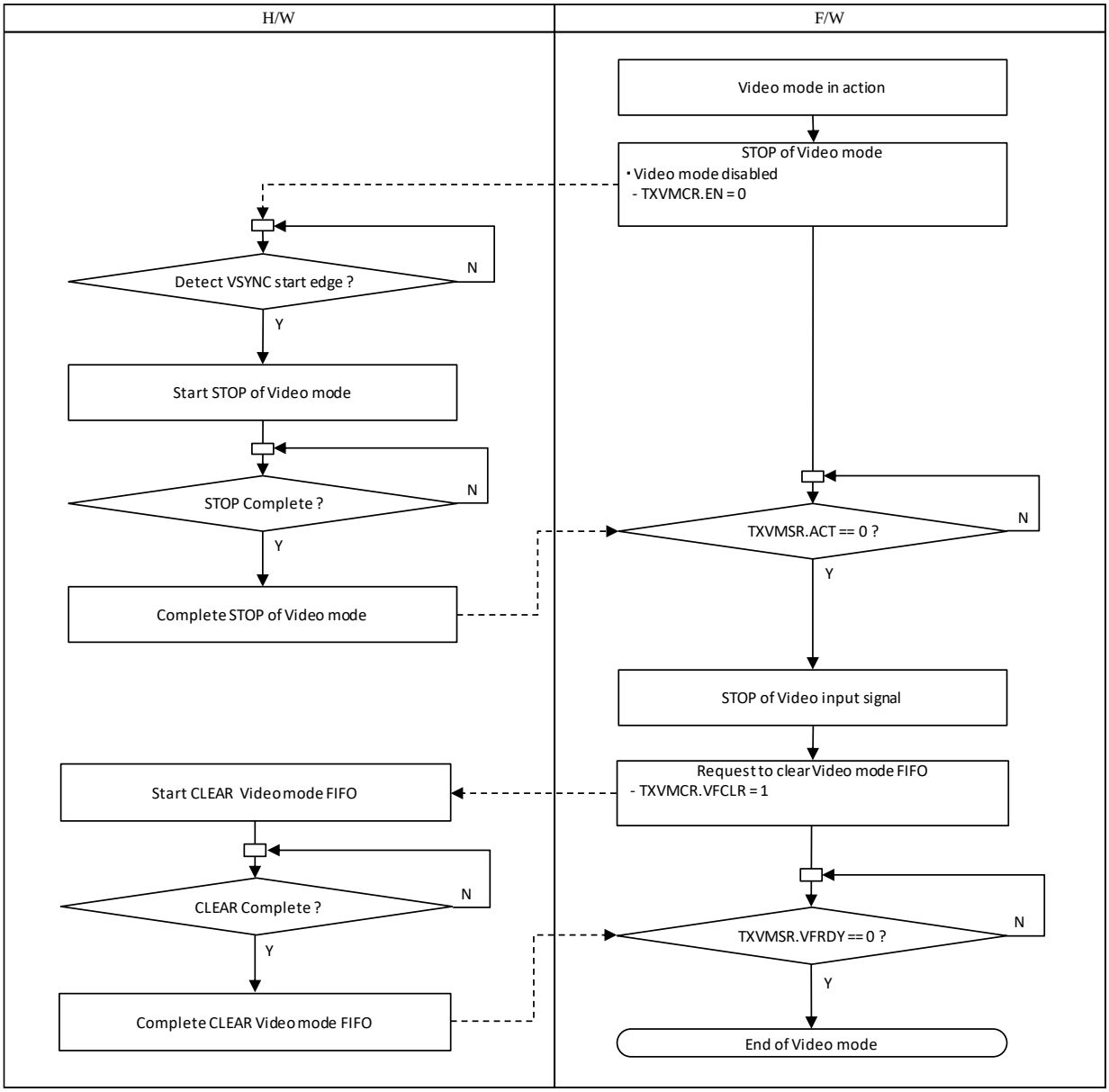



**Figure 67.13 End of Video mode (TXVMSETR.VSTPM=1)**

#### 67.3.2.4.3 Video data transmission of different types

In CSI-2 mode, different types of video data (up to four types) can be switched and transmitted per frame.

Table 5-3 shows whether video parameters can be switched.

| Parameters | Register Name | Switching (*1) |
|---|---|---|
| dotclock (vclk) | - | × |
| Pixel data width | TXVMSETR.PIXWDTH | × |
| BPP | TXVMVPRMSET0R.BPP | × |
| Color Space (RGB, YCbCr/YUV | TXVMVPRMSET0R.CSPC | ○(*2) |
| Data Type | TXVMPSPHSETR.DT | ○(*2) |

| | | |
|---|---|---|
| Virtual Channel | TXVMPSPHSETR.VC | ○ |
| Polarity of VSYNC | TXVMVPRMSET0R.VSPOL | × |
| VTOTAL | - (Sum of each period) | × |
| VSA, VACTIVE, VBP, VFP | TXVMVPRMSET1R.VSA<br>TXVMVPRMSET1R.VACTIVE<br>TXVMVPRMSET2R.VBP<br>TXVMVPRMSET2R.VFP | ○ |
| Polarity of HSYNC | TXVMVPRMSET0R.HSPOL | × |
| HTOTAL | - (Sum of each period) | × |
| HSA, HACTIVE, HBP, HFP | TXVMVPRMSET3R.HSA<br>TXVMVPRMSET3R.HACTIVE<br>TXVMVPRMSET4R.HBP<br>TXVMVPRMSET4R.HFP | ○ |
| Periodic Skew Calibration enable | TXVMSETR.VSEN | ○ |
| Maximum value of the Frame Number | TXVMSET2R.FNUM | ○ |
| FrameCRC packet enabl | FDSETR.VFCEN | ○ |

(*1)「○」support、「×」not support
(*2) Only the same BPP can be mixed

Figure 67.14 shows a different type of video data transmission behavior.
When "Start of Video mode" described in Chapter 67.3.2.4.1, please set the frame type (up to 4 types) to be used in the following registers.
TXVMSETR、TXVMSET2R、TXVMPSPHSETR、TXVMVPRMSET0,1,2,3,4R、TXVMSETF(i)R、TXVMSET2F(i)R、TXVMPSPHSETF(i)R、TXVMVPRMSET0,1,2,3,4F(i)R、FDSETR
In this case, set the frame type from the lower number to the continuous value. For example, if you want to use two frame types, use frame types 0 and 1. Other combinations (e.g. using frame types 0 and 3) are prohibited.
When a video signal is input and the first VSYNC start edge is detected, video data is transmitted according to the frame type parameters specified by TXVMFTCR.FTNXT. When the VSYNC start edge is detected, TXVMSR.VSADET sets 1, so specify the following frame type in TXVMFTCR.FTNXT before the frame transmission is complete (at least until 10usec before the next VSYNC start edge occurrence). Continue to set the TXVMFTCTR.FTNXT frame with TXVMSR.VSADET as the trigger.
※Note※
Setting the next frame type within one frame time is a requirement for S/W. Therefore, the Transmitter module does not detect errors due to the in-time setting.
Figure 67.15 shows a frame-by-frame video data switching image diagram.

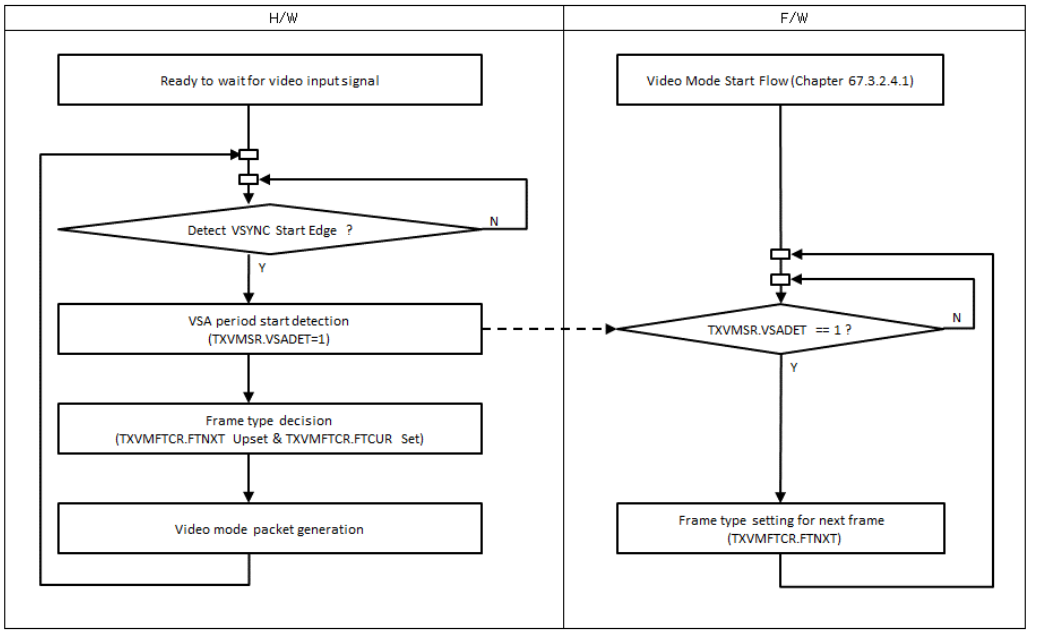



**Figure 67.14 video data transmission of different data type**

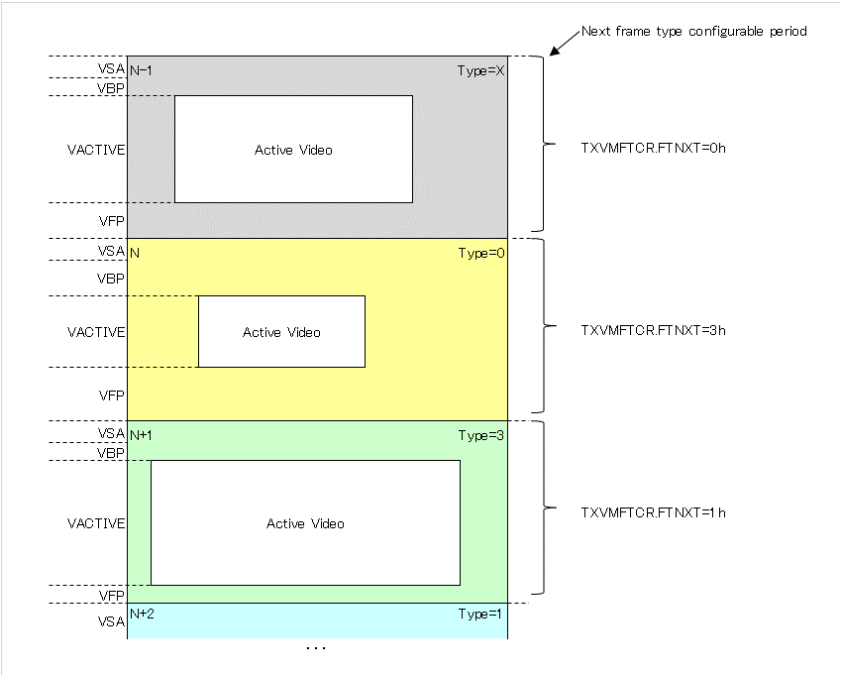



**Figure 67.15 switching video data of each frames**

#### 67.3.2.4.4 Register Configuration for DSI mode

##### 67.3.2.4.4.1 Configuration for Pixel Stream Packet Header

(1) Virtual Channel

Virtual Channel is configured for all output Packet in Video mode for Sync-Event (VSS, VSE, HSS and HSE), Blanking and Pixel Stream.

(2) Data Type
Configure the Data Type related to Video signal input.

Following Data Type is possible to configure in Video mode.

- RGB 16Bit (Packed Pixel Stream, 16bit RGB (H'0E))
- RGB 18Bit (Packed Pixel Stream, 18bit RGB (H'1E))
- RGB 18Bit Loosely (Loosely Packed Pixel Stream, 18bit RGB (H'2E))
- RGB 24Bit (Packed Pixel Stream 24bit RGB (H'3E))
- YCbCr 4:2:2 16Bit (Packed Pixel Stream 16bit YCbCr 4:2:2 (H'2C))

ColorDepth is related to the Data Type and should be configured by Table 67.6.

#### 67.3.2.4.4.2 Configuration of Blanking and the sequence

(1) Blanking
Blanking Configuration for HSA, HBP and HFP is shown in Table 67.5. LP11 (without Tx Packet) or Blanking Packet (Data Type=H'19) is selective for each Blanking Period of HSA,HBP and HFP. Only LP11 (without Tx Packet) should be used for HBLANK period.

**Table 67.5 Blanking Configuration for HSA, HBP and HFP**

| Blanking Configuration | Description |
|---|---|
| LP-11 | No Packet is transmitted during Blanking period.<br>By this behavior, the Transmitter exits from HS mode, and enters to LP-11. |
| Blanking Packet<br>(Data Type=H'19) | The HS mode Long Packet (Packet Header Data Type=H'19) is transmitted during Blanking period.<br>By this behavior, the Transmitter continues to be HS mode in Video mode. |

(2) Sequence
Configure the Sync-Pulse mode (Non-Burst Mode with Sync Pulses) or Sync-Event mode (Non-Burst Mode with Sync Events).

#### 67.3.2.4.4.3 Video Parameters

(1) Polarity of Vertical and Horizontal Signals (TXVMVPRMSET0R.VSPOL and HSPOL)
Configure the polarity of Vertical Sync (vsync) and Horizontal Sync (hsync) of Video signal input.

(2) Color Space (TXVMVPRMSET0R.CSPC)
Configure the Color Space parameters related to the Video signal input.

(3) ColorDepth (TXVMVPRMSET0R.BPP[2:0])
ColorDepth (TXVMVPRMSET0R.BPP[2:0]) is related to Data Type of Pixel Stream Header (TXVMPSPHSETR.DT[5:0]). Refer to Table 67.6 to determine the configuration of ColorDepth.

**Table 67.6 ColorDepth configurations of Pixel Stream Packet Header**

| Header Data Type of Pixel Stream Packet | ColorDepth | TXVMPSPHSETR.DT[5:0] | TXVMVPRMSETR.BPP[2:0] |
|---|---|---|---|
| Packed Pixel Stream, 16bit RGB | 16bpp | H'0E | B'000 |
| Packed Pixel Stream, 18bit RGB | 18bpp | H'1E | B'001 |
| Loosely Packed Pixel Stream, 18bit RGB | 24bpp *1 | H'2E | B'010 |
| Packed Pixel Stream 24bit RGB | 24bpp | H'3E | B'010 |
| Packed Pixel Stream 16bit YCbCr 4:2:2 | 16bpp | H'2C | B'000 |

Note: *1 24bpp should be selected for RGB 18Bit Loosely.

(4) Vertical Line count of Input Video (TXVMVPRMSET1R and TXVMVPRMSET2R)
Configure the VACTIVE, VSA, VFP and VBP period of Video signal input by the Line count.

(5) Horizontal Pixel count of Input Video (TXVMVPRMSET3R and TXVMVPRMSET4R)
Configure the HACTIVE, HSA, HFP and HBP period of Video signal input by the Pixel count.

#### 67.3.2.4.5 Register Configuration for CSI-2 mode

##### 67.3.2.4.5.1 Configuration for Pixel Stream Packet Header

(1) Virtual Channel

Virtual Channel is configured for all output Packet in Video mode (Frame Start Packet, Frame End Packet and Video Packet). When transmitting different types of video data by switching frame by frame (Chapter 67.3.2.4.3), set all the frame types to be used.

(2) Data Type

Configure the Data Type related to Video signal input. When transmitting different types of video data by switching frame by frame (Chapter 67.3.2.4.3), set all the frame types to be used.

Following Data Types are possible to configure in Video mode.

- RGB565 (H’22)
- RGB666 (H’23)
- RGB888 (H’24)
- YUV422 8-bit (H’1E)

#### 67.3.2.4.5.2 Video Parameters

(1) Polarity of the Vertical Sync and Horizontal Sync (TXVMVPRMSET0R.VSPOL and HSPOL)
Configure the Polarity of Vertical Sync (vsync) and Horizontal Sync (hsync).

(2) Color Space (TXVMVPRMSET0R.CSPC)
Configure the ColorSpace related to the Video signal input. When transmitting different types of video data by switching frame by frame (Chapter 67.3.2.4.3), set all the frame types to be used.

(3) ColorDepth (TXVMVPRMSET0R.BPP[2:0])
ColorDepth (TXVMVPRMSET0R.BPP[2:0]) is related to Data Type of Pixel Stream Header (TXVMPSPHSETR.DT[5:0]). Refer to Table 67.8 to determine the configuration of ColorDepth.

**Table 67.8 ColorDepth configurations of Pixel Stream Packet Header**

| Header Data Type of Pixel Stream Packet | ColorDepth | TXVMPSPHSET(F(i))R.DT[5:0] | TXVMVPRMSETR.BPP[2:0] |
|---|---|---|---|
| RGB 565 | 16bpp | H'22 | B'000 |
| RGB 666 | 18bpp | H'23 | B'001 |
| RGB 888 | 24bpp | H'24 | B'010 |
| YUV422 8-bit | 16bpp | H'1E | B'000 |

(4) Configuration for Horizontal Video Line Count
Configure the VACTIVE, VSA, VFP and VBP period of Video signal input by the Line count. When transmitting different types of video data by switching frame by frame (Chapter 67.3.2.4.3), set all the frame types to be used.

(5) Configuration for Horizontal Video Pixel Count
Configure the HACTIVE, HSA, HFP and HBP period of Video signal input by the Pixel count. When transmitting different types of video data by switching frame by frame (Chapter 67.3.2.4.3), set all the frame types to be used.

### 67.3.2.5 Tx Reset Trigger

Tx Trigger procedure is described in this section. The Transmitter is enabled to only Tx RESET Trigger because of Host Processor.

After Tx RESET Trigger by Host Processor, the Peripheral device executes the initialization procedure. The access of the peripheral device is prohibited until the elapsed time specified by PR_TO register in the peripheral device after the Tx RESET Trigger. For the detail of the PR_TO register, refer to the related document of the Peripheral device.

The Transmitter does not have a specific counter for the elapsed time specified by the PR_TO. The system timer may be used for this purpose.

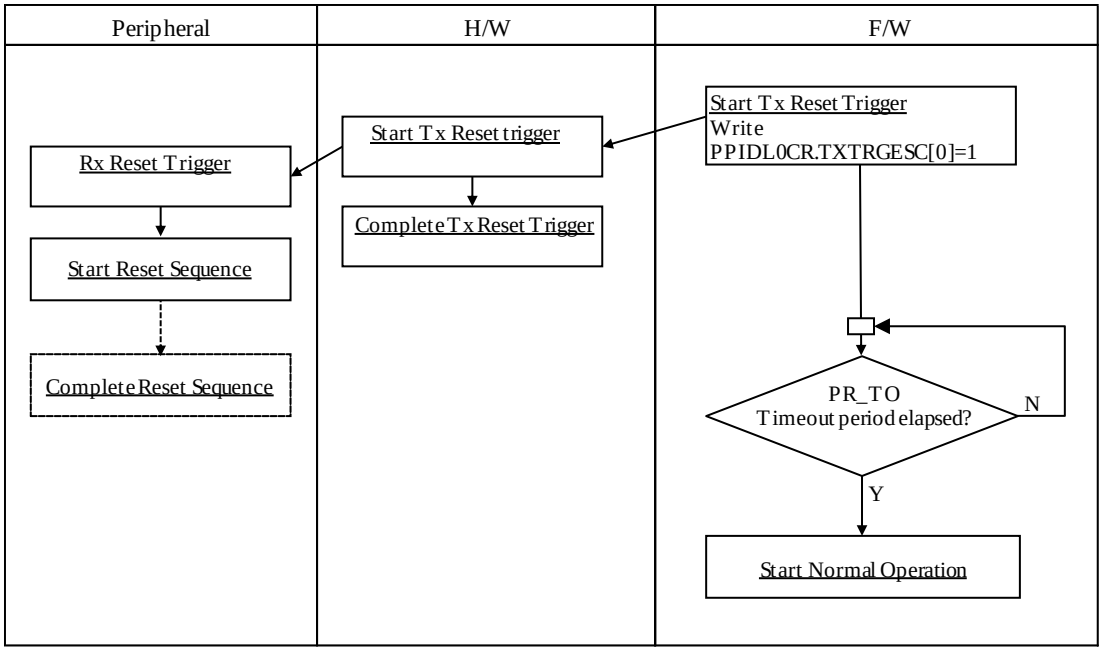



**Figure 67.17 Tx Reset Trigger**

### 67.3.2.6 Bus Turn-Around Request

Bus Turn-Around Request is described in this section. Bus Turn-Around Request may use to know the Error information of peripheral device.

Configure the appropriate bit of TAIER register and set "1" to TOIER.TATO register as Initial Configuration.

Write "1" to the TACR.TRNREQ register, then the Host Processor requests the Bus Turn-Around. Wait until the completion of the handover of Bus Control.

When the Bus Turn-Around Request is not completed within the time specified by TOSET2R.TATOVL register, dsi_int_to interrupt is occurred by TOSR.TATO =1. After this interrupt, the Recovery procedure is needed after Software RESET.

When the Bus Turn-Around Request is completed within the time specified by the TOSET2R.TATOVL register and handover of Bus Control to the Peripheral device is completed, dsi_int_ta interrupt is occurred by TASR.TRNREQEND=1.

Before the completion of the handover of the Bus Control to Host Processor (PPIDL0SR.DIR=0) by the Bus Turn-Around Request and the confirmation of the transition of Data Lane to StopState as LP11 (PPIDL0SR.STPST=1), it is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Error information of peripheral device (described in section 67.3.2.7), Transition to ULPS (described in section 67.3.2.8) or Skew Calibration Pattern Tx (described in section 67.3.2.11).

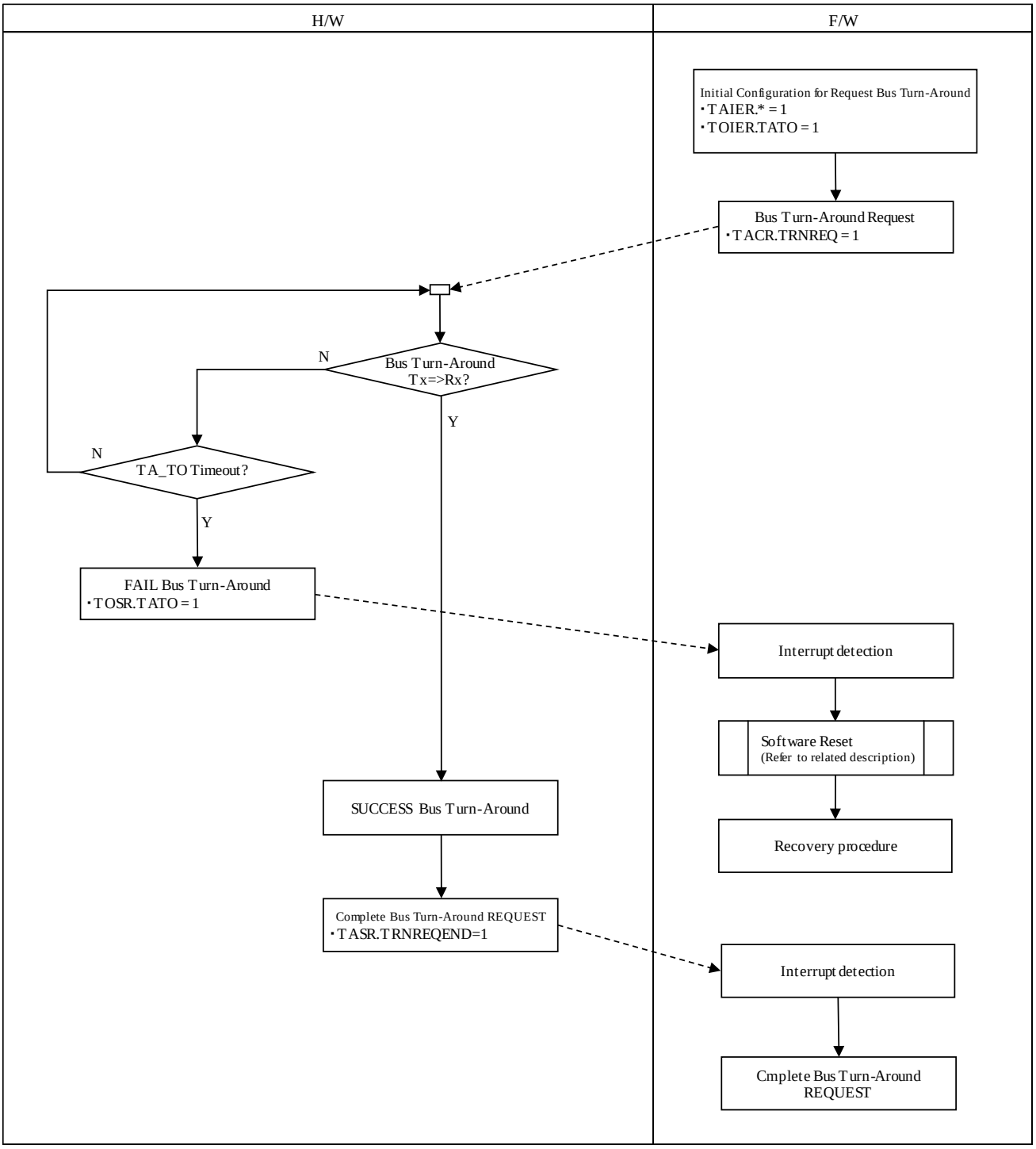



**Figure 67.18 Request of Bus Turn-Around**

#### 67.3.2.7 Error information of Peripheral device

Error information of Peripheral device is described in this section.

Bus Turn-Around Request to know the Error information of peripheral device is already described in previous section 67.3.2.6. The procedure to receive the Error information is described in next section 67.3.2.7.1.

Note that the Error information of Peripheral device is also enabled to know by Command Transfer with Bus Turn-Around. Refer to section 67.3.2.3.3 (Read Command) and section 67.3.2.3.4 (Non-Read Command) for detail.

Before the completion of the handover of the Bus Control to Host Processor (PPIDL0SR.DIR=0) by the Bus Turn-Around Request and the confirmation of the transition of Data Lane to StopState as LP11 (PPIDL0SR.STPST=1), it is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Rx Bus Turn-Around (described in section 67.3.2.6), Transition to ULPS (described in section 67.3.2.8) or Skew Calibration Pattern Tx (described in section 67.3.2.11).

##### 67.3.2.7.1 Rx Error information from Peripheral device

When ACK Trigger is received, dsi_int_ta interrupt is occurred by TASR.RCVACK=1. If Acknowledge and Error Report Packet is received, dsi_int_ta interrupt is also occurred by TASR.RCVAKE=1. To distinguish between these, confirm the cause of the interrupt by reading AKEPR register.

When some Error is detected or the Unexpected Trigger Packet is received, dsi_int_ta is occurred by the appropriate bit of TASR register. The Recovery procedure described in 67.3.2.10.2.4 should be needed.

As very rare case, because of the transmission accident etc., the transition to StopState (LP11) without any interrupt is occurred along with Bus Turn-Around (to Tx from Rx). If any interrupt does not occur within the expected Response time specified by RXRESPTOSETR register, the procedure described in just like section 67.3.2.10.2.4 should be needed.

The procedure if Rx Error information from Peripheral device is shown in Figure 67.19.

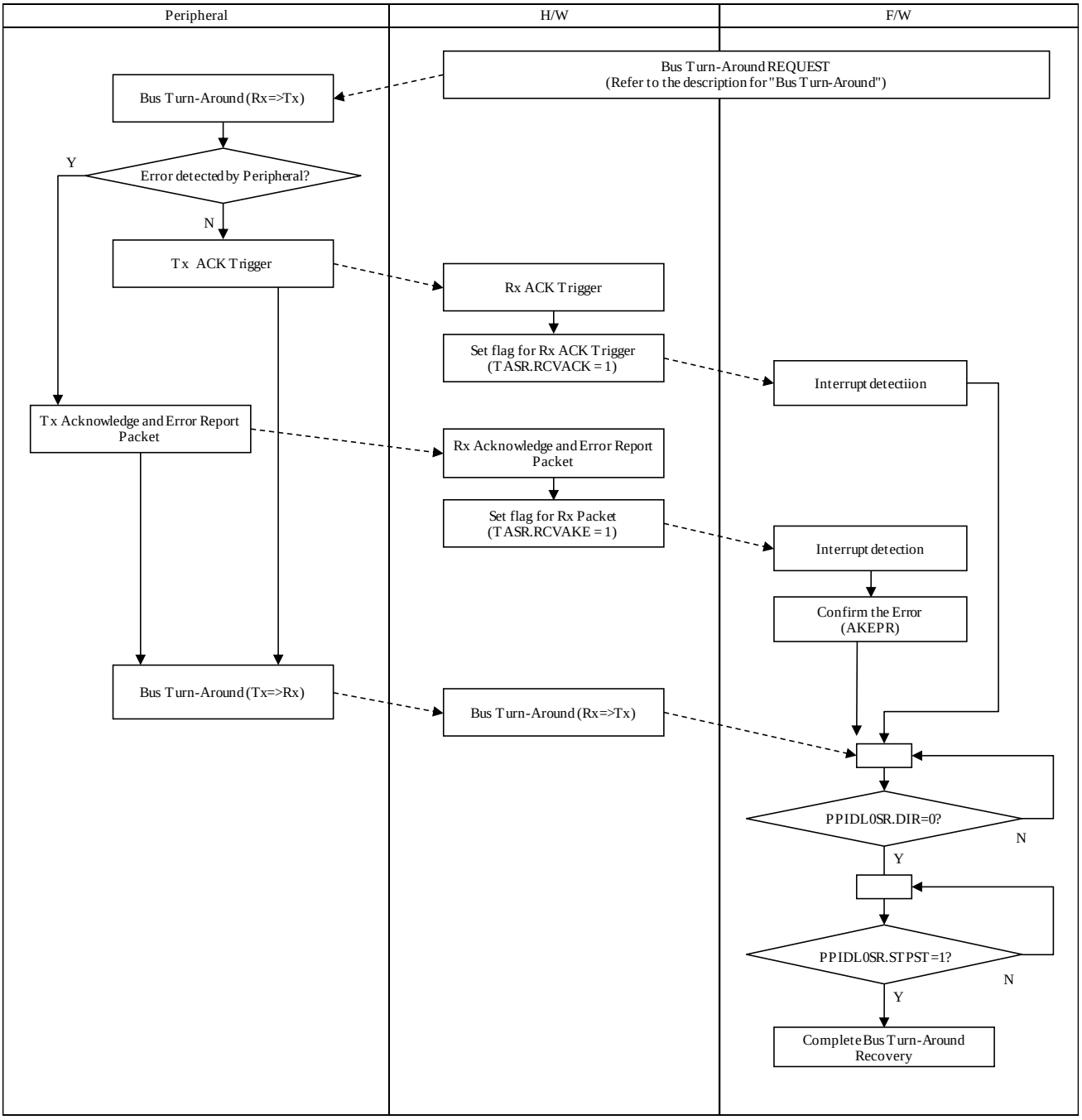



**Figure 67.19 Rx Error information from Peripheral device**

#### 67.3.2.8 Enter/Exit of ULPS

The procedure for Transition to ULPS and Exit from ULPS are described in this section.

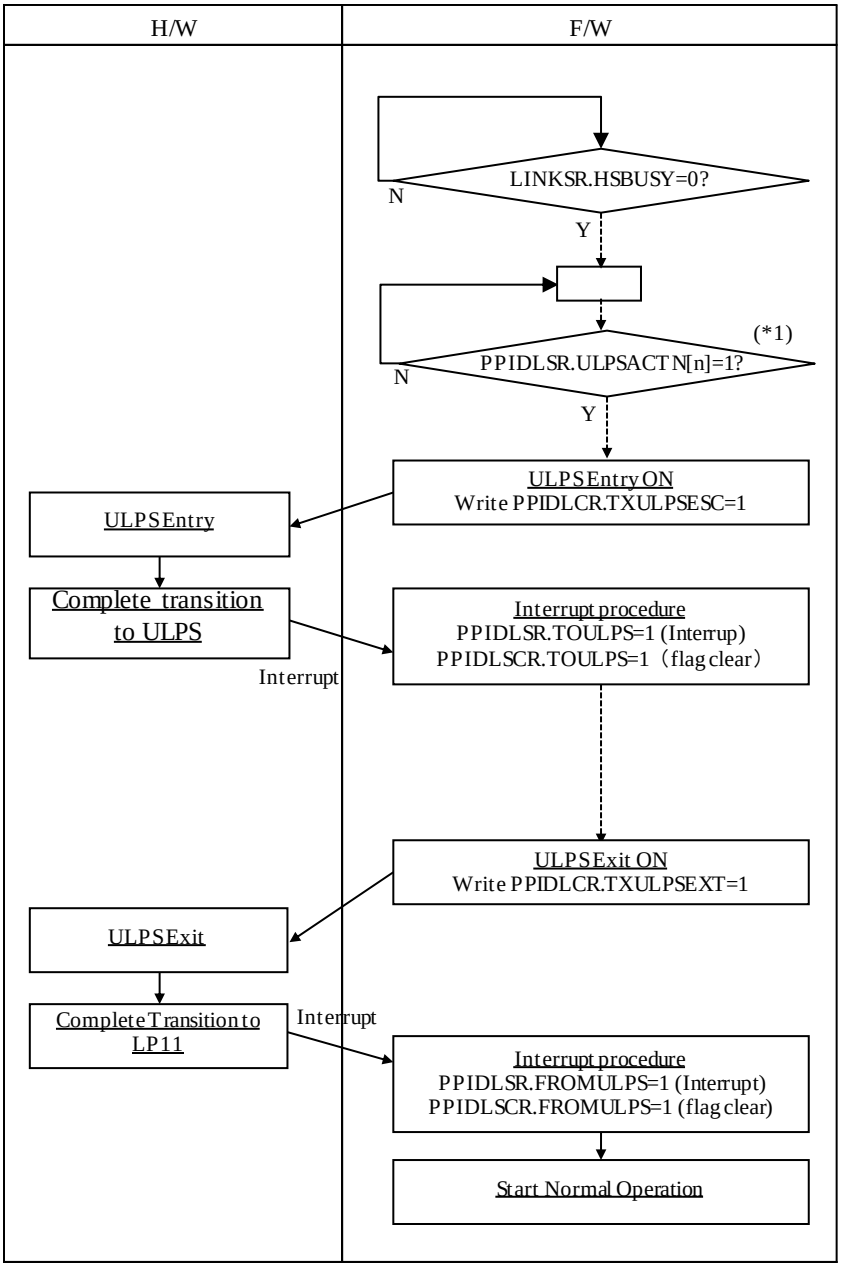
Clock Lane or Data Lane (Lane0, Lane1, Lane2 and Lane3) is enabled to transition to ULPS or to exit from ULPS individually. The Lane which is Enable=0 is disabled to transition to ULPS or to exit from ULPS.

All of Data Lane which is Enable=1 is enabled to transition to ULPS or to exit from ULPS at the same time.

Only the specific Data Lane of these is not possible to transition to ULPS or to exit from ULPS.

When both Clock Lane and Data Lane are enabled to transition to ULPS, at the same time it is necessary to exit from ULPS. (Refer to 67.22)

Once the transition to ULPS occurs and then in ULPS, it is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Video mode (described in section 67.3.2.4), Tx Reset trigger (described in section 67.3.2.5), Bus Turn-Around Request (described in section 67.3.2.6), Error information of peripheral device (described in section 67.3.2.7) or Skew Calibration Pattern Tx (described in section 67.3.2.11).




(*1) Confirm the Non-ULPS State of all Lanes specified by PPISETR.DLEN.

**Figure 67.20 Enter/Exit of ULPS (Data Lane)**

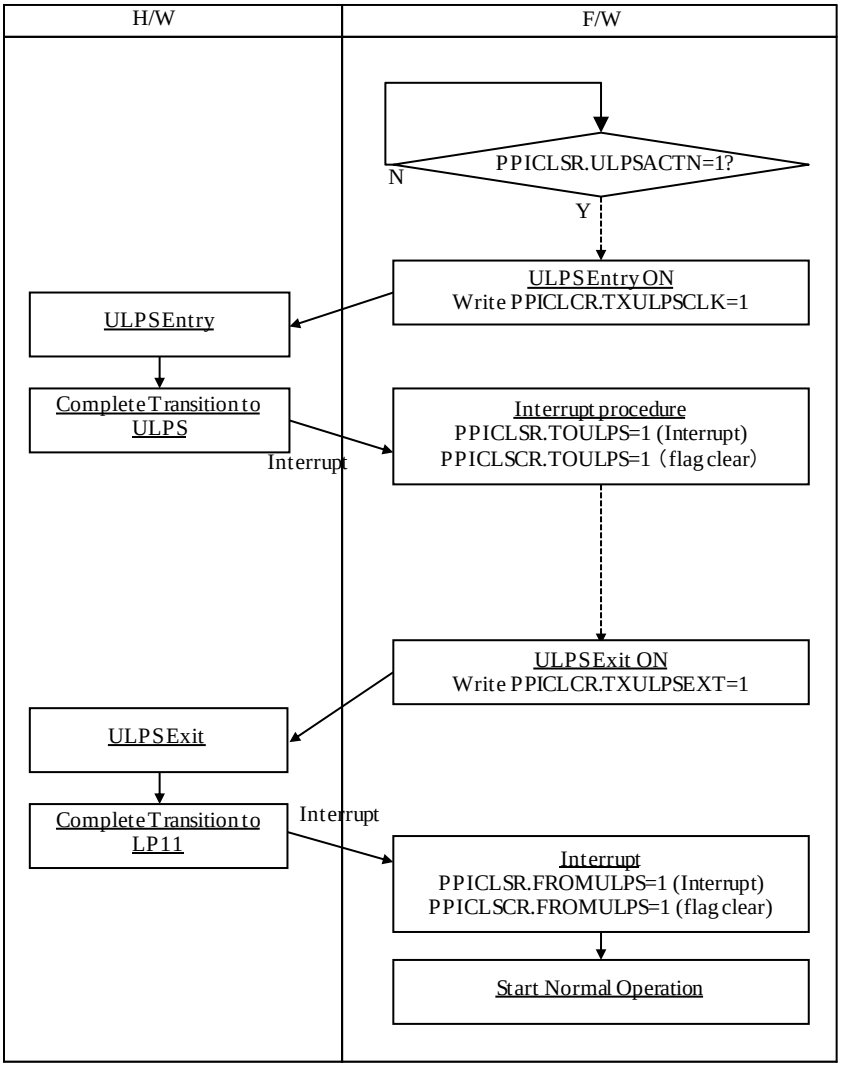



**Figure 67.21 Enter/Exit of ULPS (Clock Lane)**

#### 67.3.2.9 Force transition to StopState

The procedure of force transition to Stop State is described in this section.

When the D-PHY forces to be transitioned to Stop State (LP11) because of an Error detection, this procedure should be executed startup sequence of section 67.3.5 after the software reset of section 67.3.2.1.

| Force transition to StopState sequence |
|---|
| Set the corresponding bit in the SRCR4 register of the Clock Pulse Generator (CPG) module. |
| Set the corresponding bit in the SRSTCLR4 register of the Clock Pulse Generator (CPG) module. |
| Startup sequence.(refer to section 67.3.5) |

#### 67.3.2.10 Error Handling

##### 67.3.2.10.1 Error detection in D-PHY

When this Error is detected by D-PHY, the Transmitter indicates an interrupt by the Error. These interrupts are shown below. Refer to "Annex A Logical PHY-Protocol Interface Description" on "MIPI Alliance Specification for D-PHY Version 1.2" for detail.

- ErrEsc
- ErrSyncEsc
- ErrControl
- ErrContentionLP0
- ErrContentionLP1

When this Error is detected, Software RESET and Startup sequence (refer to section 67.3.5) should be needed to recover. And Tx Reset Trigger may be necessary by the needs.

The procedure for Recovery of Contention Error is described in Figure 67.23 for example.

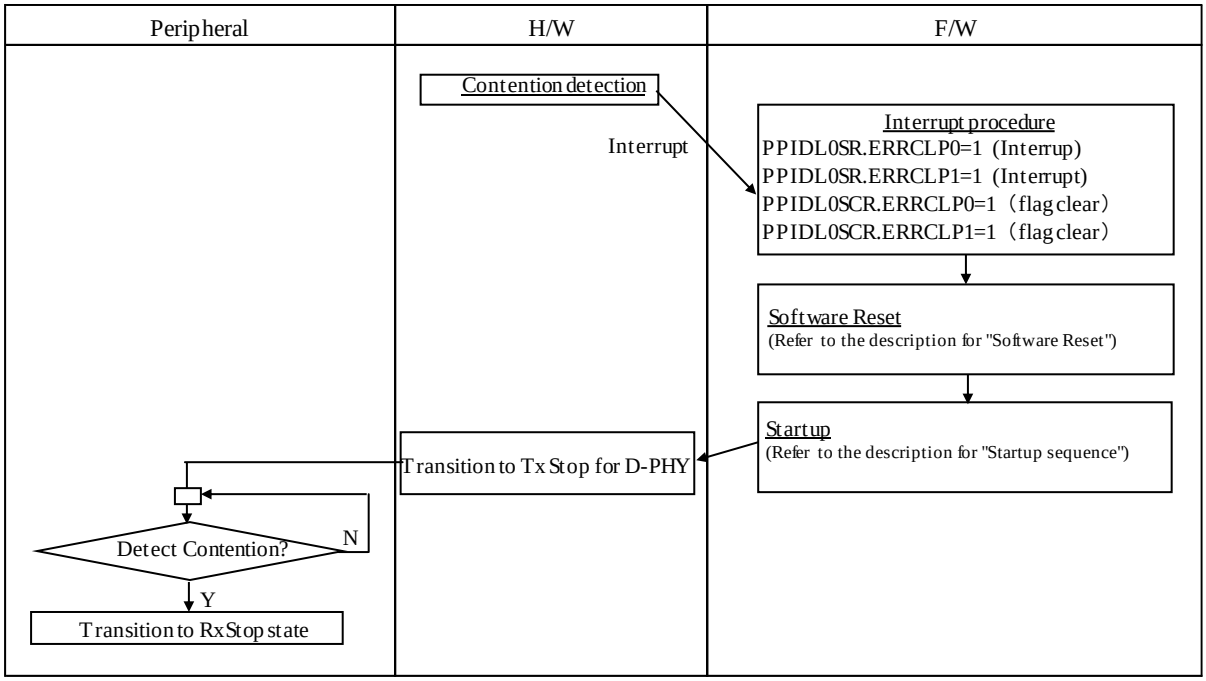



**Figure 67.23 Error Handling of D-PHY (e.g. Detect Contention)**

#### 67.3.2.10.2 Timeout Error

The Transmitter has two kinds of timers. One is defined by DSI Specification and another is an original timer. Refer to section 7.2.2 "Contention Recovery Using Timers" of the DSI Specification.

・Timers defied in DSI Specification

- Tx HS Timeout Error (HTX_TO)
- Rx LP Timeout Error (LRX_H_TO)
- Turn-Around Timeout Error (TA_TO)

・Original Timers

- Rx Response Timeout Error

#### 67.3.2.10.2.1 Tx HS Timeout Error

When this Error is detected, Software RESET and Startup sequence (refer to section 67.3.5) should be needed to recover. And Tx Reset Trigger may be necessary by the needs.

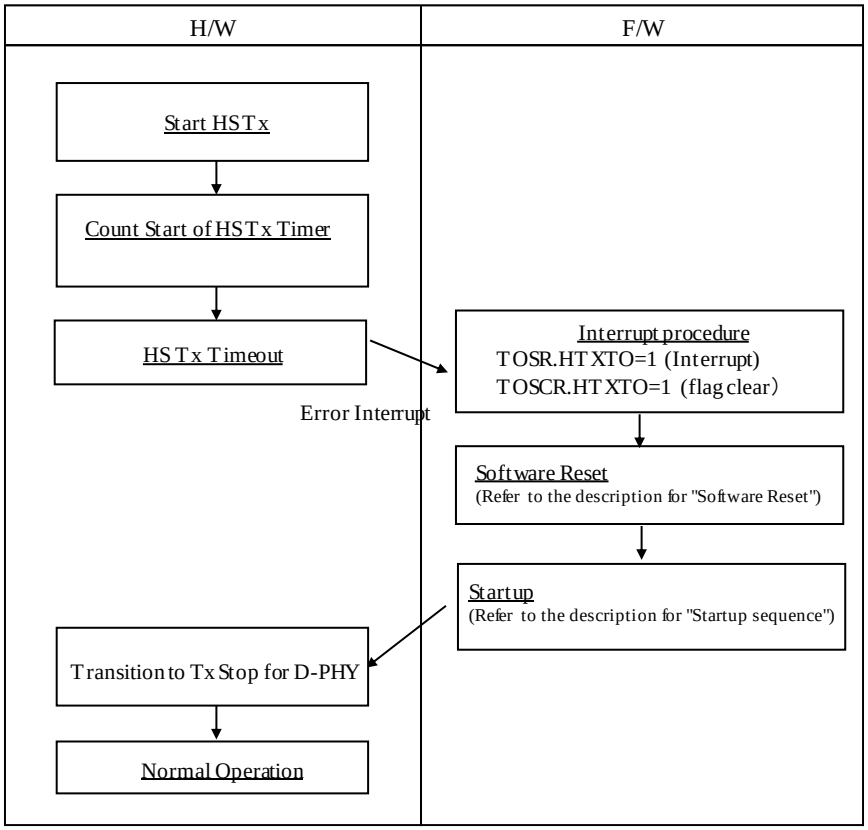



**Figure 67.24 Error Handling for Tx HS Timeout**

#### 67.3.2.10.2.2 Rx LP Timeout Error

When this Error is detected, Software RESET and Startup sequence (refer to section 67.3.5) should be needed to recover. And Tx Reset Trigger may be necessary by the needs.

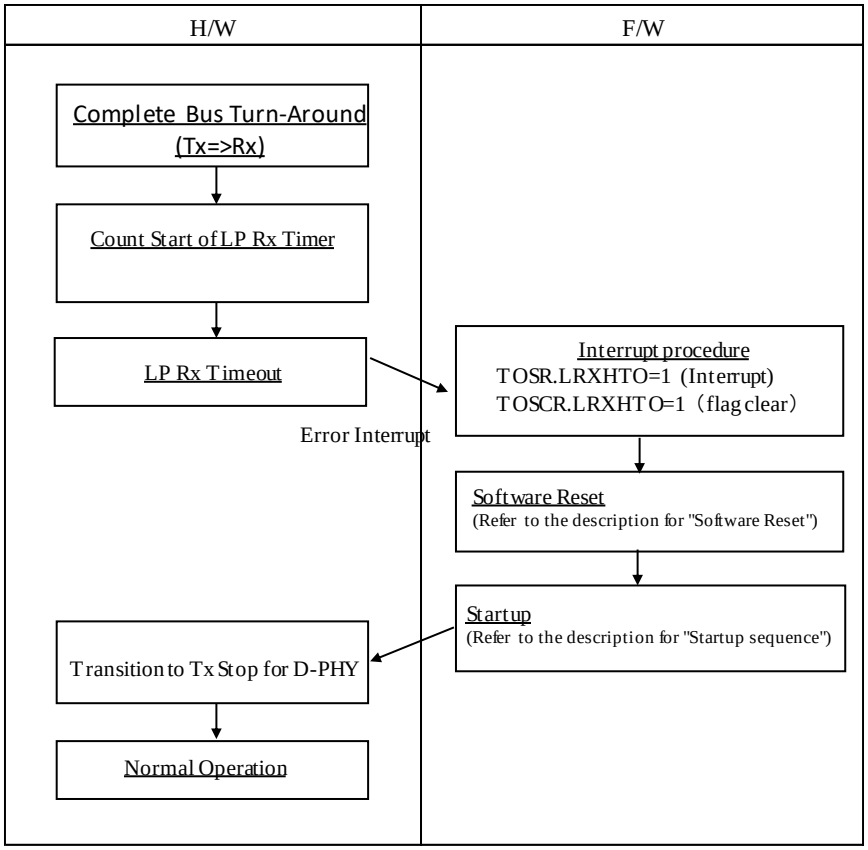



**Figure 67.25 Error Handling for Rx LP Timeout**

#### 67.3.2.10.2.3 Turn-Around Timeout Error

When this Error is detected, Software RESET and Startup sequence (refer to section 67.3.5) should be needed to recover. And Tx Reset Trigger may be necessary by the needs.

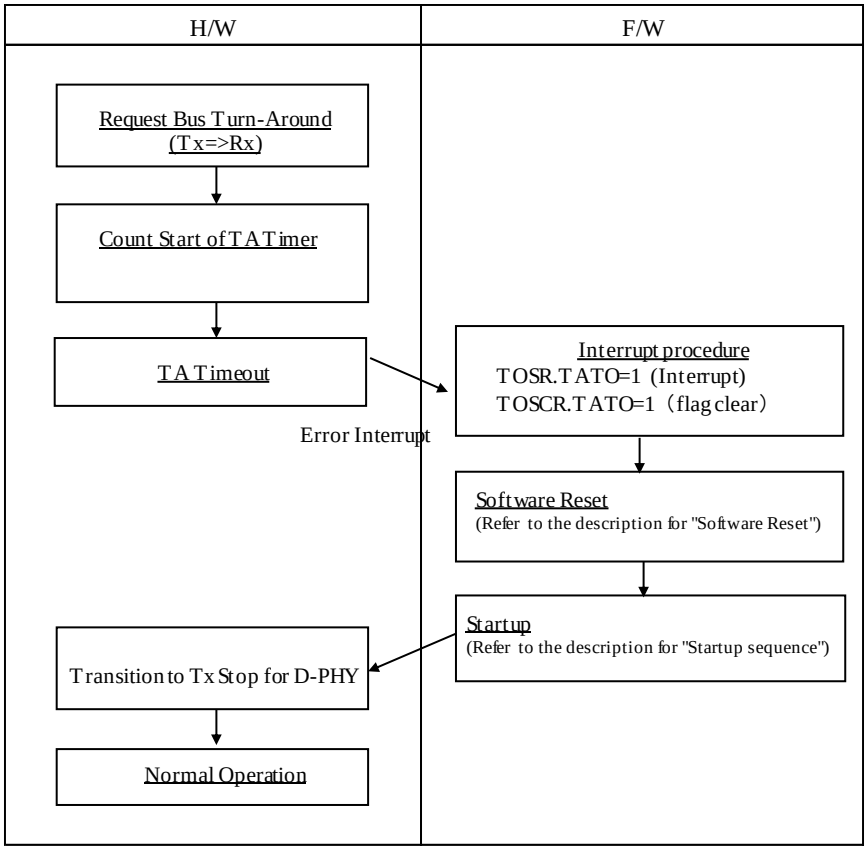



**Figure 67.26 Error Handling for Turn-Around Timeout**

#### 67.3.2.10.2.4 Rx Response Timeout Error

If Rx Packet is not started within the time specified by RXRESPTOSETRregister after the Bus Turn-Around by TXCMCR.BTAREQ register, then Rx Timeout is indicated by RXPSR.RESPTOERR=1.

Something to recover for this Error should be needed. For example it may be to resend the request, etc.

#### 67.3.2.10.3 Rx Packet Error

Rx Packet Error to be detected by this Transmitter is described in this section.

The Rx Packet Error has two kinds. One is the error which can be classified as a packet and which does not have an influence on other packets. This Error is related to the Packet. e.g. CEC Error or ECC 1bit Error.

The other is the error related to the Rx itself like fatal error.

A fatal error may be reported as only one error because the packet cannot be classified as a packet by the fatal error. A slight error like an ECC error (1bit Error) or CRC error has a possibility to be reported together with other errors. Thus an error report has the priority. The priority of the Errors which can detect by the Transmitter is the order listed below.

Priority order: Malformed Packet Error, ECC Error, Unexpected Data Type Error, Unexpected Packet Error, Unexpected Trigger Error, WC Error, CRC Error

These Errors are described below for detail.

##### 67.3.2.10.3.1 Malformed Packet Error

When the value of RXPSR/TASR.MLFERR is "1", the Error which is the Rx Packet with under 4 byte length is detected. Something to recover for this Error should be needed. For example, Re-transmission of the Command just before may be needed.

##### 67.3.2.10.3.2 ECC Error

Received ECC is enabled by RXSETR.ECCEN=1. If the Peripheral device does not support this ECC function, Received ECC should be disabled by RXSETR.ECCEN=0.

###### 67.3.2.10.3.2.1 ECC 1bit Error

If the ECC 1bit Error is detected by RXPSR/TASR.ECCERR1B=1, the Data of the Rx Packet with ECC Error is corrected automatically. After this automatic Error correction, the Rx Packet is continued.

###### 67.3.2.10.3.2.2 ECC Multi-bit Error

If the ECC Multi-bit Error is detected by RXPSR/TASR.ECCERR=1, the Rx Packet is discarded. Something to recover for this Error should be needed. For example, Re-transmission of the Command just before may be needed.

##### 67.3.2.10.3.3 Unexpected Data Type Error

If the reception of the Packet with "Reserved" Data Type described in Table 22 of DSI Specification is detected by RXPSR/TASR.UEXDTERR=1, the Rx Packet has an Error. Something to recover for this Error should be needed. For example, Re-transmission of the Command just before may be needed.

#### 67.3.2.10.3.4 Unexpected Packet Error

If the Rx Unexpected Packet is detected by RXPSR/TASR.UEXPKTERR=1, the Rx Packet has an Error. Something to recover for this Error should be needed. For example, Re-transmission of the Command just before may be needed.

The expected Rx Packet with Tx Command for requirement of the Response to the Peripheral device (described in section 67.3.2.3.3) is listed below. The Rx Packet except for the Packet listed below is Unexpected Packet.

- Response Packet (1 packet)
- Response Packet (1 packet) + EoTp Packet(1 packet)
- Response Packet (1 packet) + Acknowledge and Error Report Packet (1 packet)
- Response Packet (1 packet) + Acknowledge and Error Report Packet (1 packet) + EoTp Packet(1 packet)
- Acknowledge and Error Report Packet (1 packet)
- Acknowledge and Error Report Packet (1 packet) + EoTp Packet (1 packet)
- EoTp Packet (1 packet)

The expected Rx Packet for Tx Non-Read Command with Bus Turn-Around (described in section 67.3.2.3.4), for Bus Turn-Around Request (described in section 67.3.2.6) and for Error information from Peripheral device (described in section 67.3.2.7) is listed below. The Rx Packet except for the Packet listed below is Unexpected Packet.

- Acknowledge and Error Report Packet (1 packet)
- Acknowledge and Error Report Packet (1 packet) + EoTp Packet (1 packet)
- EoTp Packet (1 packet)

#### 67.3.2.10.3.5 Unexpected Trigger Error

If the Unexpected Rx Trigger is detected by RXPSR.UEXTRGERR=1, the Rx Trigger has an Error. Something to recover for this Error should be needed. For example, Re-transmission of the Command just before may be needed.

When the Command with Response to request to Peripheral device (described in section 67.3.2.3.3) is transmitted, the Unexpected Trigger Error is detected by the Rx Trigger listed below.

- ACK Trigger

#### 67.3.2.10.3.6 WC Error

When the Rx Long Packet with the Payload Data which is not enough for the WC value is detected by RXPSR.WCERR=1, the Payload Data of the Rx Long Packet has an Error. Something to recover for this Error should be needed. For example, Re-transmission of the Command just before may be needed.

#### 67.3.2.10.3.7 CRC Error

CRC check is enabled by RXSETR.CRCEN=1. If the Peripheral device does not support this CRC function, CRC function should be disabled by RXSETR.CRCEN=0. If the CRC Error is detected by RXPSR.CRCERR=1, the Rx Data has an Error. Something to recover for this Error should be needed. For example, Re-transmission of the Command just before may be needed.

### 67.3.2.10.4 Error of Handling for Rx Data Payload

#### 67.3.2.10.4.1 AXI Error

If the response of Master AXI Write is received except for OKAY, AXI Error is indicated by RXPSR.AXIERR=1. Something to recover for this Error should be needed. For example, the confirmation of the configuration for memory area may be needed.

#### 67.3.2.10.4.2 Rx Buffer Overrun Error

If at least one of the following condition is satisfied, the Rx Buffer Overrun Error is indicated by RXPSR.OVRERR=1.

In case of Register Access mode:

・The Rx Payload Data is over 16Byte length

In case of AXI Access mode:

・The Rx Payload Data is over the length specified by RXPSIZESETR.SIZE register

・RAM Overrun for the LP Rx is detected

The procedure described below should be needed by RXPSR.OVRERR=1.

In case of Register Access mode:

・Change to AXI Access mode

In case of AXI Access mode:

・When WordCount of Rx Packet is over the length specified by RXPSIZESETR.SIZE register, correct the RXPSIZESETR.SIZE register.

・When WordCount of Rx Packet is under the length specified by RXPSIZESETR.SIZE register, something to correct for this Error should be needed. For example, change the response time of AXI Slave faster.

### 67.3.2.10.5 Error in Tx Command Transfer mode

#### 67.3.2.10.5.1 AXI Error

If the response of Master AXI Read is received except for OKAY, AXI Error is indicated by TXCMSR.AXIERR=1. Something to recover for this Error should be needed. For example, the confirmation of the configuration for memory area may be needed.

#### 67.3.2.10.5.2 Collision Error

If the HS Command Transfer mode Packet is not transmitted correctly in Video mode, the Collision Error is indicated by TXCMSR.CLSNERR=1. Something to correct for this Error should be needed. For example, change the length of Command Transfer mode Packet shorter or, the packet is divided.

### 67.3.2.10.6 Error in Tx Video mode

#### 67.3.2.10.6.1 Packet Discarded Error

If the Tx Packet is not transmitted correctly in Video mode, the Packet Discarded Error is indicated by =1 TXVMSR.PDIS. For such case, check the Configurations for Video Input interface of Table 67.16 or Table 67.18.

#### 67.3.2.11 Tx Skew Calibration Pattern

The Procedure for Tx Skew Calibration Pattern by Firmware is described in this section. Refer to section 67.4.1.1.1.2 for detail of the Periodic de-skew in Video mode. The switching on/off of the Periodic de-skew is prohibited during Video mode.

##### 67.3.2.11.1 Periodic de-skew

The period for transmission of Skew Calibration Pattern is calculated by the following formula.

SCALPSR.PTNSZ – SKEWCALOFFSET

Refer to Table 67.9 for SKEWCALOFFSET.

Notice, the actual period of Tx Skew Calibration Pattern is depend on the D-PHY of attached Rx. The value of SCALPSR.PTNSZ should be determined by the behavior of the D-PHY of attached Rx with deep consideration. e.g. the actual delay of Tx Pattern from TxSkewCalHS=1, the actual period to stop the Pattern by TxSkewCalHS=0, etc.

For the Periodic de-skew, the Tx Skew Calibration Pattern is needed to continue for the specified period to 10 microseconds from $2^{10}$ UI (Unit Interval).

After the configuration of the SCALPSR.PTNSZ register, write "1" to the SCALCR.CALREQ register with SCALCR.CALTYP=0 to start the Periodic de-skew. When the next request is enabled by the completion of the Periodic de-skew, the dsi_int_scal interrupt is occurred by SCALSR.CALREQEN=1.

**Table 67.9 SKEWCALOFFSET**

| Bitrate (Mbps) | SKEWCALOFFSET |
|---|---|
| 2500 | 508 |
| 2450 | 507 |
| 2400 | 507 |
| 2350 | 503 |
| 2300 | 489 |
| 2250 | 481 |
| 2200 | 474 |
| 2150 | 461 |
| 2100 | 452 |
| 2050 | 440 |
| 2000 | 431 |
| 1950 | 419 |
| 1900 | 410 |
| 1850 | 399 |

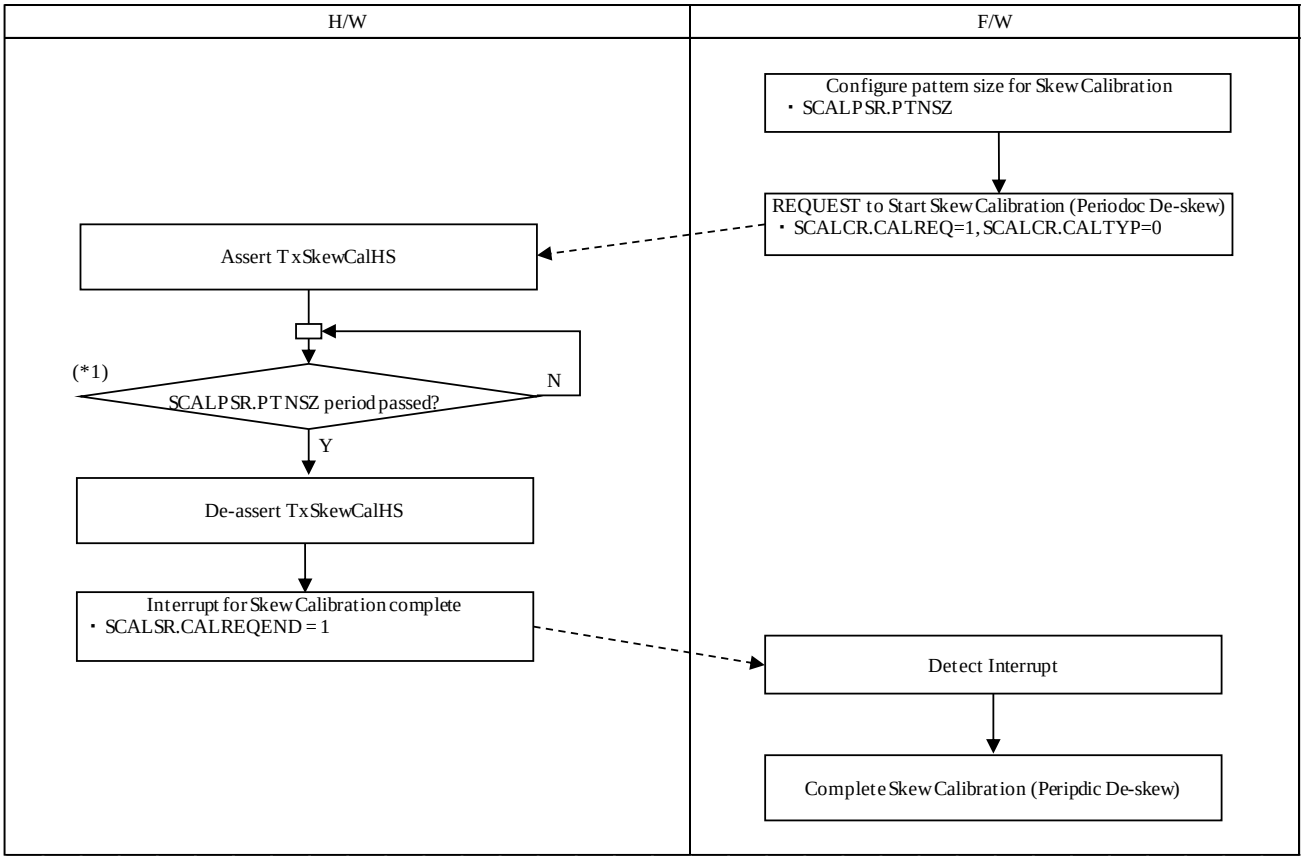
| | |
|---|---|
| 1800 | 389 |
| 1750 | 377 |
| 1700 | 367 |
| 1650 | 355 |
| 1600 | 346 |
| 1550 | 336 |
| 1500 | 326 |

Before Periodic de-skew, HS Tx Clock (described in 67.3.2.11.2) must be started.

It is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Bus Turn-Around Request (described in section 67.3.2.6), Error information of peripheral device (described in section 67.3.2.7) or Transition to ULPS (described in section 67.3.2.8).

The registers listed below are prohibited to configure during the period which is after writing to "1" to the SCALCR.CALREQ register and is before changing to SCALSR.CALREQEND=1.

SCALPSR



Note: Before Periodic de-skew, HS Tx Clock (described in 67.3.2.11.2) must be started.

**Figure 67.27 Skew Calibration (Periodic De-Skew)**

#### 67.3.2.11.2 Initial de-skew

When operating above 1.5 Gbps or changing to any rate above 1.5 Gbps, an initial deskew sequence shall be transmitted before HS Tx. When changing states, for example from ULPS to HS, transmission of any deskew sequence is optional,

provided HS operation resumes at a rate for which an initial deskew sequence has previously been transmitted. .

The period for transmission of Skew Calibration Pattern is calculated by the following formula.

SCALPS1R.PTNSZ – SKEWCALOFFSET

Refer to Table 67.9 for SKEWCALOFFSET.

Notice, the actual period of Tx Skew Calibration Pattern is depend on the D-PHY of attached Rx. The value of SCALPS1R.PTNSZ should be determined by the behavior of the D-PHY of attached Rx with deep consideration. e.g. the actual delay of Tx Pattern from TxSkewCalHS=1, the actual period to stop the Pattern by TxSkewCalHS=0, etc.

For the Initial de-skew, the Tx Skew Calibration Pattern is needed to continue for the specified period to 100 microseconds from $2^{15}$ UI(Unit Interval).

After the configuration of the SCALPS1R.PTNSZ register, write "1" to the SCALCR.CALREQ register with SCALCR.CALTYP=1 to start the Initial de-skew. When the next request is enabled by the completion of the Initial de-skew, the dsi_int_scal interrupt is occurred by SCALSR.CALREQEND=1.

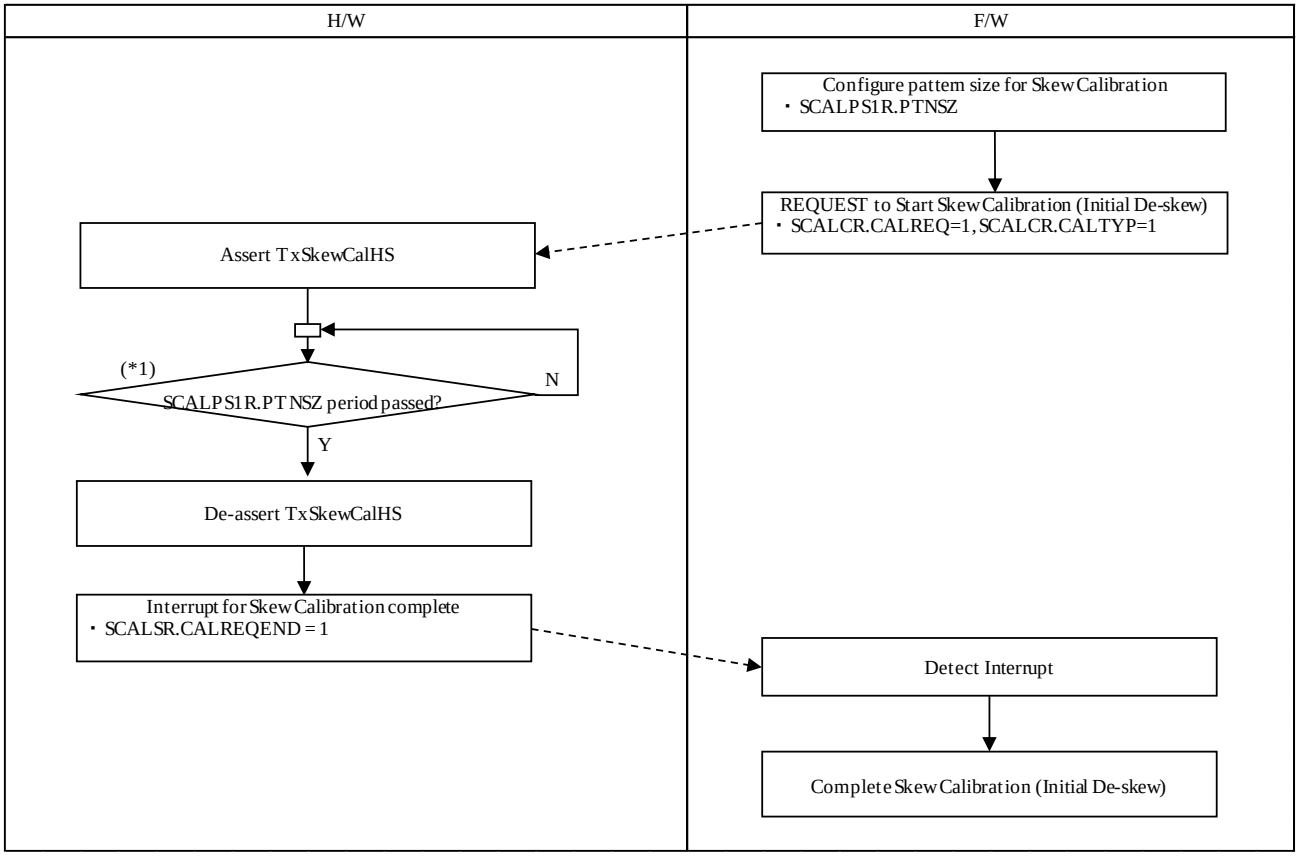
Before Initail de-skew, HS Tx Clock (described in 67.3.2.11.2) must be started.

It is prohibited to start another request which is Command Transfer mode (described in section 67.3.2.3), Tx Reset trigger (described in section 67.3.2.5), Bus Turn-Around Request (described in section 67.3.2.6), Error information of peripheral device (described in section 67.3.2.7) or Transition to ULPS (described in section 67.3.2.8).

The registers listed below are prohibited to configure during the period which is after writing to "1" to the SCALCR.CALREQ register and is before changing to SCALSR.CALREQEND=1.

SCALPS1R

If the D-PHY of attached Rx has an ability of automatically Initial de-skew, the following sequence may not need to execute.




Note: Before Initail de-skew, HS Tx Clock (described in 67.3.2.11.2) must be started.

**Figure 67.28 Skew-Calibration (Initial De-Skew)**

### 67.3.3 Data type

**Table 67.10 Support data type of DSI**

| Data Type (hex) | Description | Packet Size | Command Transfer mode*1 | Video mode |
|---|---|---|---|---|
| H'01 | Sync Event, V Sync Start | Short | — | ○ |
| H'11 | Sync Event, V Sync End | Short | — | ○ |
| H'21 | Sync Event, H Sync Start | Short | — | ○ |
| H'31 | Sync Event, H Sync End | Short | — | ○ |
| H'07 | Compression Mode Command | Short | — | — |
| H'08 | End of Transmission packet (EoTp) | Short | ○ | ○ |
| H'02 | Color Mode (CM) Off Command | Short | ○ | — |
| H'12 | Color Mode (CM) On Command | Short | ○ | — |
| H'22 | Shut Down Peripheral Command | Short | ○ | — |

| Data Type (hex) | Description | Packet Size | Command Transfer mode*1 | Video mode |
|---|---|---|---|---|
| H'32 | Turn On Peripheral Command | Short | ○ | — |
| H'03 | Generic Short WRITE, no parameters | Short | ○ | — |
| H'13 | Generic Short WRITE, 1 parameter | Short | ○ | — |
| H'23 | Generic Short WRITE, 2 parameters | Short | ○ | — |
| H'04 | Generic READ, no parameters | Short | ○ | — |
| H'14 | Generic READ, 1 parameter | Short | ○ | — |
| H'24 | Generic READ, 2 parameters | Short | ○ | — |
| H'05 | DCS Short WRITE, no parameters | Short | ○ | — |
| H'15 | DCS Short WRITE, 1 parameter | Short | ○ | — |
| H'06 | DCS READ, no parameters | Short | ○ | — |
| H'16 | Execute Queue | Short | ○ | — |
| H'37 | Set Maximum Return Packet Size | Short | ○ | — |
| H'27 | Scrambling Mode Command | Short | ○ | — |
| H'09 | Null Packet, no data | Long | ○ | — |
| H'19 | Blanking Packet, no data | Long | — | ○ |
| H'29 | Generic Long Write | Long | ○ | — |
| H'39 | DCS Long Write/write_LUT Command Packet | Long | ○ | — |
| H'0A | Picture Parameter Set | Long | ○ | — |
| H'0B | Compressed Pixel Stream | Long | — | — |
| H'0C | Loosely Packed Pixel Stream, 20-bit YCbCr, 4:2:2 Format | Long | — | — |
| H'1C | Packed Pixel Stream, 24-bit YCbCr, 4:2:2 Format | Long | — | — |
| H'2C | Packed Pixel Stream, 16-bit YCbCr, 4:2:2 Format | Long | — | ○ |
| H'0D | Packed Pixel Stream, 30-bit RGB, 10-10-10 Format | Long | — | — |
| H'1D | Packed Pixel Stream, 36-bit RGB, 12-12-12 Format | Long | — | — |
| H'3D | Packed Pixel Stream, 12-bit YCbCr, 4:2:0 Format | Long | — | — |
| H'0E | Packed Pixel Stream, 16-bit RGB, 5-6-5 Format | Long | — | ○ |

| Data Type (hex) | Description | Packet Size | Command Transfer mode*1 | Video mode |
|---|---|---|---|---|
| H'1E | Packed Pixel Stream, 18-bit RGB, 6-6-6 Format | Long | — | ○ |
| H'2E | Loosely Packed Pixel Stream, 18-bit RGB, 6-6-6 Format | Long | — | ○ |
| H'3E | Packed Pixel Stream, 24-bit RGB, 8-8-8 Format | Long | — | ○ |

Note: *1 Although any Data type can be transmitted, using Command Transfer mode in Videomode,

Consider MIPI DSI specification and the document of the Peripheral device.

**Table 67.11 Support data type of CSI-2**

| Data Type (hex) | Description | Packet Size | Command Transfer mode*1 | Video mode |
|---|---|---|---|---|
| H'00 | Frame Start Code | Short | — | ○ |
| H'01 | Frame End Code | Short | — | ○ |
| H'02 | Line Start Code (Optional) | Short | — | — |
| H'03 | Line End Code (Optional) | Short | — | — |
| H'08 | Generic Short Packet Code 1 | Short | ○ | — |
| H'09 | Generic Short Packet Code 2 | Short | ○ | — |
| H'0A | Generic Short Packet Code 3 | Short | ○ | — |
| H'0B | Generic Short Packet Code 4 | Short | ○ | — |
| H'0C | Generic Short Packet Code 5 | Short | ○ | — |
| H'0D | Generic Short Packet Code 6 | Short | ○ | — |
| H'0E | Generic Short Packet Code 7 | Short | ○ | — |
| H'0F | Generic Short Packet Code 8 | Short | ○ | — |
| H'10 | Null | Long | — | — |
| H'11 | Blanking Data | Long | — | — |
| H'12 | Embedded 8-bit non Image Data | Long | — | — |
| H'18 | YUV420 8-bit | Long | — | — |
| H'19 | YUV420 10-bit | Long | — | — |
| H'1A | Legacy YUV420 8-bit | Long | — | — |
| H'1C | YUV420 8-bit (Chroma Shifted Pixel Sampling) | Long | — | — |
| H'1D | YUV420 10-bit (Chroma Shifted Pixel Sampling) | Long | — | — |

| | | | | |
|---|---|---|---|---|
| H'1E | YUV422 8-bit | Long | — | ○ |
| H'1F | YUV422 10-bit | Long | — | — |
| H'20 | RGB444 | Long | — | — |
| H'21 | RGB555 | Long | — | — |
| H'22 | RGB565 | Long | — | ○ |
| H'23 | RGB666 | Long | — | ○ |
| H'24 | RGB888 | Long | — | ○ |
| H'28 | RAW6 | Long | — | — |
| H'29 | RAW7 | Long | — | — |
| H'2A | RAW8 | Long | — | — |
| H'2B | RAW10 | Long | — | — |
| H'2C | RAW12 | Long | — | — |
| H'2D | RAW14 | Long | — | — |
| H'2E | RAW16 (MIPI CSI-2 Version 2.0) | Long | — | — |
| H'2F | RAW20 (MIPI CSI-2 Version 2.0) | Long | — | — |
| H'30 | User Defined 8-bit Data Type 1 | Long | — | — |
| H'31 | User Defined 8-bit Data Type 2 | Long | — | — |
| H'32 | User Defined 8-bit Data Type 3 | Long | — | — |
| H'33 | User Defined 8-bit Data Type 4 | Long | — | — |
| H'34 | User Defined 8-bit Data Type 5 | Long | — | — |
| H'35 | User Defined 8-bit Data Type 6 | Long | — | — |
| H'36 | User Defined 8-bit Data Type 7 | Long | — | — |
| H'37 | User Defined 8-bit Data Type 8 | Long | — | — |

Note: *1 Although any Data type can be transmitted, using Command Transfer mode in Videomode,

Consider MIPI CSI-2 specification and the document of the Peripheral device.

### 67.3.4 Clock Configuration

The relationship should be needed for the dotclock and the data rate of DSI/CSI-2-TX. The frequency value can be obtained as follows:

Fout[MHz] = vclk [MHz] * ColorDepth [bpp]/(2*DataLaneCount)

hsclk = Fout/4

Fout: Frequency of clock lane of D-PHY

hsclk: byteclock

vclk: dotclock

The configuration of the clocks is shown on Figure 67.29.

Frequency of Fout is configured by setting of CLOCKSET2.m register and CLOCKSET2.n register. And DU select the dotclock output from the Dotclock Generator module.Refer to DU section for detail.

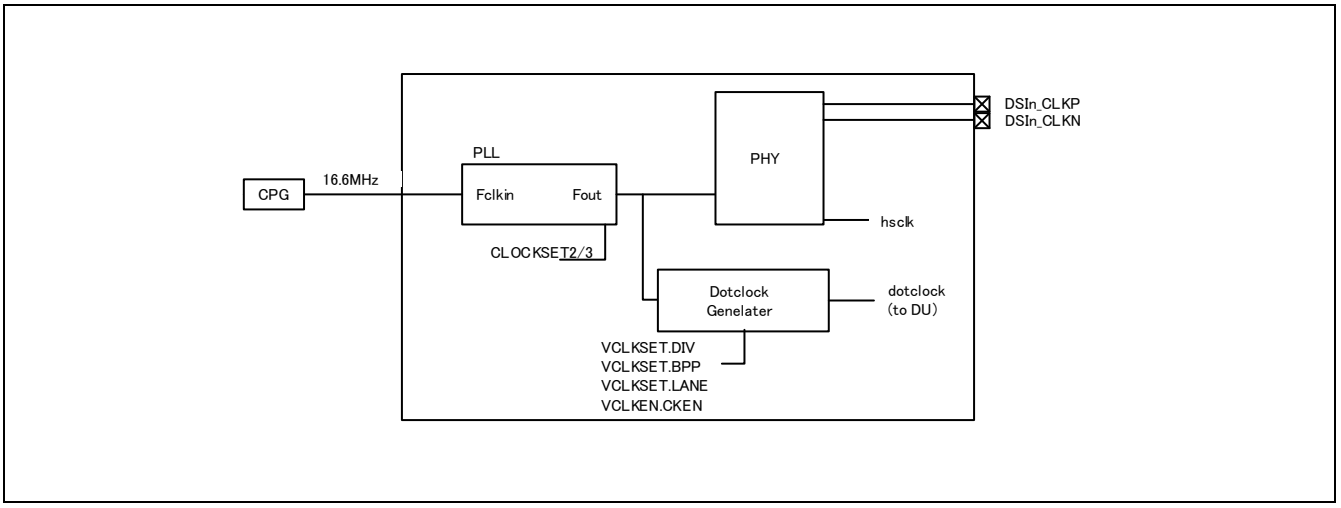



**Figure 67.29 Clock Configuration**

1250 MHz ≥ Fout ≥ 1000 MHz: Fout =M * Fclkin / (N *2 *2)

Where:

■ M: Feedback Multiplication Ratio M= CLOCKSET2.m

■ N: Input Frequency Division Ratio N= CLOCKSET2.n+1

Fclkin = DSIREFϕ

8 MHz <= Fclkin <= 64MHz

8 MHz <= Fclkin / N <= 24 MHz

2000 MHz <= M * Fclkin / (N * 2) <= 4000 MHz

1 <= N <= 8

167 <= M <= 1000

Output frequencies below 1000MHz are obtained through division (Refer to Table 67.13)

| | |
|---|---|
| 1000 MHz ≥ Fout ≥ 500 MHz: | Fout = M * Fclkin / (N *2 *4) |
| 500 MHz ≥ Fout ≥ 250 MHz: | Fout = M * Fclkin / (N *2 *8) |
| 250 MHz ≥ Fout ≥ 125 MHz: | Fout = M * Fclkin / (N *2 *16) |
| 125 MHz ≥ Fout ≥ 62.5 MHz: | Fout = M * Fclkin / (N *2 *32) |
| 62.5 MHz ≥ Fout ≥ 40 MHz: | Fout = M * Fclkin / (N *2 *64) |

**Table 67.12 Parameter Set**

| Fout [MHz] | CLOCKSET2. Vco_cntrl[5:0] | CLOCKSET3. Cpbias_cntrl[6:0] | CLOCKSET3. Gmp_cntrl[1:0] | CLOCKSET3. Int_cntrl[5:0] | CLOCKSET3. Prop_cntrl[5:0] |
|---|---|---|---|---|---|
| 1200 - 1250 | 000011 | 0000000 | 00 | 001000 | 001010 |
| 1000 - 1200 | 000111 | 0000000 | 00 | 001000 | 001010 |
| 875 - 1000 | 001000 | 0000000 | 00 | 001000 | 001010 |
| 725 - 875 | 001000 | 0000000 | 00 | 001000 | 001010 |
| 600 - 725 | 001011 | 0000000 | 00 | 001000 | 001010 |
| 500 - 600 | 001111 | 0000000 | 00 | 001000 | 001010 |
| 455.48 - 500 | 010000 | 0000000 | 00 | 001000 | 001010 |
| 362.5 - 455.48 | 010000 | 0000000 | 00 | 001000 | 001010 |
| 300 - 362.5 | 010011 | 0000000 | 00 | 001000 | 001010 |
| 250 - 300 | 010111 | 0000000 | 00 | 001000 | 001010 |
| 218.75 - 250 | 011000 | 0000000 | 00 | 001000 | 001010 |
| 181.25 - 218.75 | 011000 | 0000000 | 00 | 001000 | 001010 |
| 150 - 181.25 | 011011 | 0000000 | 00 | 001000 | 001010 |
| 125 - 150 | 011111 | 0000000 | 00 | 001000 | 001010 |
| 109.37 - 125 | 100000 | 0000000 | 00 | 001000 | 001010 |
| 90.63 - 109.37 | 100000 | 0000000 | 00 | 001000 | 001010 |
| 75 - 90.63 | 100011 | 0000000 | 00 | 001000 | 001010 |
| 62.5 - 75 | 100111 | 0000000 | 00 | 001000 | 001010 |
| 54.66 - 62.5 | 101000 | 0000000 | 00 | 001000 | 001010 |
| 45.31 - 54.66 | 101000 | 0000000 | 00 | 001000 | 001010 |
| 40 - 45.31 | 101011 | 0000000 | 00 | 001000 | 001010 |

**Table 67.13 Division factors based on vco_cntrl[5:3]**

| vco_cntrl[5:3] | division |
|---|---|
| 000 | 1/2 |
| 001 | 1/4 |
| 010 | 1/8 |
| 011 | 1/16 |
| 100 | 1/32 |
| 101 | 1/64 |

### 67.3.5 Startup sequence

The start-up procedure is shown below. Peripheral Device is needed to start-up after the start-up of this Tx.

It is possible to control the LINK as shown from 67.3.2.2 to 67.3.2.11 after the Startup sequence

<table>
<tr><th>Case 1 (operating rate <= 1.5 Gbps; cfgclk=25Mhz; slew rate calibration enabled):</th><th>Case 2 (operating rate > 1.5 Gbps; cfgclk=25Mhz; slew rate calibration disabled):</th></tr>
<tr><td colspan="2">Disable the corresponding bits in the MSTPCR4 registers of the Clock Pulse Generator (CPG) module.</td></tr>
<tr><td colspan="2">Enable DSI clock (DSIϕ)<br>Set DSIEXTCKCR=9’h000</td></tr>
<tr><td colspan="2">Set the frequency of LPCLK and enabled.<br>Set LPCLKSET.CKEN=1 and LPCLKSET.LPCLKDIV</td></tr>
<tr><td colspan="2">CFGCLK enabled<br>Set CFGCLKSET.CKEN=1</td></tr>
<tr><td colspan="2">Set PHYSETUP.rstz= 0</td></tr>
<tr><td colspan="2">Set PHYSETUP.shutdownz=0</td></tr>
<tr><td colspan="2">Set PHTC.TESTCLR = 1</td></tr>
<tr><td colspan="2">Set PHTC.TESTCLR = 0</td></tr>
<tr><td colspan="2">Set PHYSETUP.hsfreqrange[6:0] (Refer to Table 67.14)<br><br>< 450Mbps<br>Set PHTW=H’01010100<br>Set PHTW=H’011B01AC</td></tr>
<tr><td colspan="2">Set PHTW=H’01010100<br>Set PHTW=H’01030173<br>Set PHTW=H’01000174<br>Set PHTW=H’01500175<br>Set PHTW=H’01030176<br>Set PHTW=H’01040166<br>Set PHTW=H’010201AD</td></tr>
<tr><td>1Gbps < and <= 1.5Gbps<br>Set PHTW=H’01020100<br>Set PHTW=H’01980170<br>Set PHTW=H’01030171<br>Set PHTW=H’01100172<br><= 1Gbps<br>Set PHTW=H’01020100<br>Set PHTW=H’01910170<br>Set PHTW=H’01020171<br>Set PHTW=H’01110172</td><td rowspan="2">1.5Gbps < and <= 2.5Gbps<br>Set PHTW=H’01020100<br>Set PHTW=H’0144016B<br>Set PHTW=H’01000172</td></tr>
<tr><td>DataLaneCount = 1 case<br>Set PHTW=H'01070100<br>Set PHTW=H'010E010B<br>DataLaneCount = 2,1 case</td></tr>
</table>

| | |
|---|---|
| Set PHTW=H'01090100<br>Set PHTW=H'010E010B<br>**DataLaneCount = 3,2,1 case**<br>Set PHTW=H'010B0100<br>Set PHTW=H'010E010B<br>**DataLaneCount = 4 case**<br>Go to next step | |
| **<= 1.5Gbps**<br>Set PHTW=H'01010100<br>Set PHTW=H'01C0016e | |
| **For Clock Configuration**<br>1.Set CLOCKSET1.shadow_clear = 0<br>2.Set CLOCKSET1.shadow_clear = 1<br>3.Set CLOCKSET1.shadow_clear = 0<br>4.Set the PLL parameters below, refer to Clock Configuration of section 67.3.4<br>CLOCKSET2.m<br>CLOCKSET2.n<br>CLOCKSET2.vco_cntrl<br>CLOCKSET3.cpbias_cntrl<br>CLOCKSET3.gmp_cntrl<br>CLOCKSET3.int_cntrl<br>CLOCKSET3.prop_cntrl<br>5.Set CLOCKSET1.updatepll = 0<br>6.Set CLOCKSET1.updatepll = 1<br>Wait 2us.<br>7.Set CLOCKSET1.updatepll = 0 | |
| Set PPISETR.CLEN = 1 and PPISETR.DLEN[3:0] | |
| Set PHYSETUP.shutdownz = 1 | |
| Set PHYSETUP.rstz = 1 | |
| ・Wait until stopstatedata and stopstateclk outputs are asserted indicating PHY is driving LP11 in enabled datalanes and clocklane. (Consider 500us as maximum wait time)<br>LP11 is confirmed by PPICLSR.STPST register and PPIDLSR.STPST register.<br>The lock of PLL is enabled to confirm by CLOCKSET1.lock register. | |
| **<= 1.5Gbps**<br>Set PHTW=H'01020100<br>Set PHTW=H'00000180<br>Wait until PHTR[16] to be set to 1'b1<br>Set PHTW=H'01010100<br>Set PHTW=H'0100016e | **1.5Gbps < and <= 2.5Gbps**<br>- |
| **For Dotclock enabled.**<br>Set VCLKSET.CKEN= 1<br>Set VCLKSET.COLOR,DIV,BPP and LANE<br>Set VCLKEN = 1 after the configuration of VCLKSET. | |

Notes: After set PHTW, it is necessary wait DWEN and CWEN becomes B'00.

**Table 67.14 Frequency Ranges (hsfreqrange[6:0])**

| Range (Mbps) | Default Bit Rate (Mbps) | hsfreqrange[6:0] |
|---|---|---|
| 80 - 97.125 | 80 | 0000000 |
| 80 - 107.625 | 90 | 0010000 |
| 83.125 - 118.125 | 100 | 0100000 |
| 92.625 - 128.625 | 110 | 0110000 |
| 102.125 - 139.125 | 120 | 0000001 |
| 111.625 - 149.625 | 130 | 0010001 |
| 121.125 - 160.125 | 140 | 0100001 |
| 130.625 - 170.625 | 150 | 0110001 |
| 140.125 - 181.125 | 160 | 0000010 |
| 149.625 - 191.625 | 170 | 0010010 |
| 159.125 - 202.125 | 180 | 0100010 |
| 168.625 - 212.625 | 190 | 0110010 |
| 182.875 - 228.375 | 205 | 0000011 |
| 197.125 - 244.125 | 220 | 0010011 |
| 211.375 - 259.875 | 235 | 0100011 |
| 225.625 - 275.625 | 250 | 0110011 |
| 249.375 - 301.875 | 275 | 0000100 |
| 273.125 - 328.125 | 300 | 0010100 |
| 296.875 - 354.375 | 325 | 0100101 |
| 320.625 - 380.625 | 350 | 0110101 |
| 368.125 - 433.125 | 400 | 0000101 |
| 415.625 - 485.625 | 450 | 0010110 |
| 463.125 - 538.125 | 500 | 0100110 |
| 510.625 - 590.625 | 550 | 0110111 |
| 558.125 - 643.125 | 600 | 0000111 |
| 605.625 - 695.625 | 650 | 0011000 |
| 653.125 - 748.125 | 700 | 0101000 |
| 700.625 - 800.625 | 750 | 0111001 |
| 748.125 - 853.125 | 800 | 0001001 |
| 795.625 - 905.625 | 850 | 0011001 |
| 843.125 - 958.125 | 900 | 0101001 |
| 890.625 - 1010.625 | 950 | 0111010 |
| 938.125 - 1063.125 | 1000 | 0001010 |
| 985.625 - 1115.625 | 1050 | 0011010 |
| 1033.125 - 1168.125 | 1100 | 0101010 |
| 1080.625 - 1220.625 | 1150 | 0111011 |
| 1128.125 - 1273.125 | 1200 | 0001011 |
| 1175.625 - 1325.625 | 1250 | 0011011 |
| 1223.125 - 1378.125 | 1300 | 0101011 |
| 1270.625 - 1430.625 | 1350 | 0111100 |
| 1318.125 - 1483.125 | 1400 | 0001100 |
| 1365.625 - 1535.625 | 1450 | 0011100 |
| 1413.125 - 1588.125 | 1500 | 0101100 |

| Range (Mbps) | Default Bit Rate (Mbps) | hsfreqrange[6:0] |
|---|---|---|
| 1460.625 - 1640.625 | 1550 | 0111101 |
| 1508.125 - 1693.125 | 1600 | 0001101 |
| 1555.625 - 1745.625 | 1650 | 0011101 |
| 1603.125 - 1798.125 | 1700 | 0101110 |
| 1650.625 - 1850.625 | 1750 | 0111110 |
| 1698.125 - 1903.125 | 1800 | 0001110 |
| 1745.625 - 1955.625 | 1850 | 0011110 |
| 1793.125 - 2008.125 | 1900 | 0101111 |
| 1840.625 - 2060.625 | 1950 | 0111111 |
| 1888.125 - 2113.125 | 2000 | 0001111 |
| 1935.625 - 2165.625 | 2050 | 1000000 |
| 1983.125 - 2218.125 | 2100 | 1000001 |
| 2030.625 - 2270.625 | 2150 | 1000010 |
| 2078.125 - 2323.125 | 2200 | 1000011 |
| 2125.625 - 2375.625 | 2250 | 1000100 |
| 2173.125 - 2428.125 | 2300 | 1000101 |
| 2220.625 - 2480.625 | 2350 | 1000110 |
| 2268.125 - 2500 | 2400 | 1000111 |
| 2315.625 - 2500 | 2450 | 1001000 |
| 2363.125 - 2500 | 2500 | 1001001 |

### 67.3.6 Shutdown sequence and Standby mode

The DSI/CSI-2-TX module can gate the clock signals to reduce power consumption. Module standby mode can be set by controlling the module stop register 4 (MSTPCR4) of the clock pulse generator (CPG) module. DSI/CSI-2-TX channels DSI/CSI-2-TX0, DSI/CSI-2-TX1 correspond to bits 8 and 9 in the MSTPCR4 registers, respectively. To release any of the DSI/CSI-2-TX channels from standby mode, enable the corresponding the bits in the MSTPCR4 registers. Enabling the bits allows all types of access to the channel.

To set any of the DSI/CSI-2-TX channels to standby mode, execute the following procedure using the register corresponding to the channel.

| **1.Shutdown sequence (It is the same procedure when Forced termination of execution)** |
|---|
| Set the corresponding bit in the SRCR4 register of the Clock Pulse Generator (CPG) module. |
| **2.Stanby mode** |
| Disable the corresponding bits in the MSTPCR4 registers of the Clock Pulse Generator (CPG) module. |
| Set DSICKCR=9'h100 for disable the DSI clock (DSIϕ) |

To set any of the DSI/CSI-2-TX channels to operating mode from standby mode, execute the following procedure using the register corresponding to the channel.

| **3. Operating mode** |
|---|
| Set the corresponding bit in the SRSTCLR4 register of the Clock Pulse Generator (CPG) module. |
| For the startup sequence, see section 67.3.5 |

## 67.4 Usage Notes

### 67.4.1 Interfaces for Video mode

#### 67.4.1.1 Video Output

##### 67.4.1.1.1 DSI

This section describes the Video input interface on DSI mode (at IPSETR.CSI2M=0b).

Timings and Configurations for Video Output are shown below.

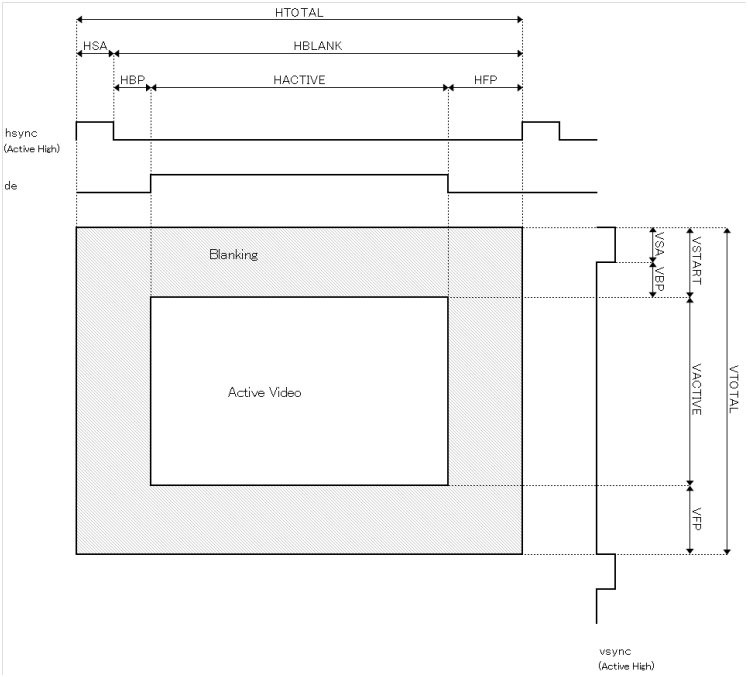



**Figure 67.30 Timing for Video Output (DSI)**

**Table 67.15 Video Output parameters (DSI)**

| category | Parameter | Description | remarks |
|---|---|---|---|
| The number of horizontal pixels | I | Horizontal sync signal active period pixels | hsync signal assert period number of pixels<br>High active: hsync signal“=”"1" period<br>Low active: hsync signal“=”"0" period |
| | HBLANK | Number of pixels for horizontal blending period | hsync negate period pixels for vertical blending period (inactive period) |
| | HBP | Horizontal back porch period pixels | Number of pixels between the end of the hsync signal assert signal and the start of the de signal assert during the vertical active period |
| | HACTIVE | Horizontal active period pixels | Number of pixels in de signal assert period<br>de signal“=”"1" period |

| category | Parameter | Description | remarks |
|---|---|---|---|
| | HFP | Horizontal front porch period pixels | Number of pixels in the period from the end of the de-signal assert to the start of the hsync signal assert |
| | HTOTAL | Total horizontal pixeIHSA+HBP+HACTIVEIP (HSA+HBLANK) | |
| Number of vertical lines | VSA | Number of vertical sync signal active period lines | vsync signal assert period<br>High active: vsync “i”nal = "1" period<br>Low active: vsync “i”nal = "0" period |
| | VBP | Number of vertical back porch period lines | Number of lines until the start of the vertical sync signal active period |
| | VACTIVE | Number of vertical active period lines | Number of lines being de-signaled |
| | VFP | Number of vertical front porch period lines | Number of lines from the end of the vertical active period to the start of the vertical synchronization signal active period |
| | VSTART | Number of vertical active period start lines | VSA + VBP |
| | VTOTAL | Total vertical lines | VSA+VBP+VACTIVE+VFP |

**Table 67.16 Configurations for Video Output interface (DSI)**

Common Rules for both Sync-Pulse mode and Sync-Event mode

| parameter | description |
|---|---|
| All | All parameters related to Input Video should be set to the value of not less than 1.<br>Exa“ple for "All p”raIers": HSA, HBP, HACTIVE, HFP, VSA, VBP, VACTIVE, VFP |
| HACTIVE | **TXVMPSPHSETR.DT** / **TXVMVPRMSET3R.HACTIVE**<br>RGB 16bit / Set 1 to 32767 because 1Pixel consists of 2Byte and the maximum length of the DSI packet is 65535Byte.<br>RGB 18bit / This value should be the multiple of 4. Set 4 to 29124 because the 4pixel consists of 9Byte and the maximum length of the DSI packet is 65535Byte.<br>RGB 24bit<br>RGB 18bit Loosely / Set 1 to 21845 because 1pixel consists of 3Byte and the maximum length of the DSI packet is 65535Byte.<br>YCbCr 4:2:2 16bit / This value should be the multiple of 2. Set 2 to 32766 because the 2pixel consists of 4Byte and the maximum length of the DSI packet is 65535Byte. |
| HTOTAL | HTOTAL should be the multiple of 4 when TXVMVPRMSET0R.BPP=001I18bpp). |
| HSA, HBP, HACTIVE, HFP | Each setting should be a multiple of 2 when TXVMSETR.PIXWDT =D’2.<br>Each setting should be a multiple of 4 when TXVMSETR.PIXWDT =D’4. |
| HACTIVE+HFP | ●No use of HFP Blanking Packet (TXVMSETR.HFPBPEN=0)<br>The period which is HACTIVE+HFP should be set to longer than the period which the transmitted time of Pixel Stream Packet and the transition time for HS-LP-HS are added.<br>(HACTIVE+HFP) > (period for the Tx Pixel Stream Packet)+(period for transition time as HS- |

| parameter | description |
|---|---|
| | LP-HS)<br>Low power consumption is possible to configure without using HFP Blanking Packet when the formula described above is satisfied. |
| | ●Use of HFP Blanking Packet (TXVMSETR.HFPBPEN=1)<br>Blanking Packet is transmitted at the period to Sync Event Packet from Pixel Stream Packet within the period of HACTIVE+HFP.<br>The period of HACTIVE+HFP should be within the range to maximum value from minimum value of the period which the transmitted time of Pixel Stream Packet and the transmitted time of Blanking Packet are added.<br>So, the period of HACTIVE+HFP should be set to the value within the range described below.<br>Actual period of HACTIVE+HFP = ((HACTIVE+HFP) * ColorDepth) / 8)<br>Transmitted time of Pixel Stream Packet = (6+ RoundUp((HACTIVE * ColorDepth)/8))<br>(Actual period of HACTIVE+HFP)-(period for the Tx Pixel Stream Packet) >= 6 |
| VFP+VSA+VBP,<br>HTOTAL | When sending Frame CRC packets (FDSETR.VFCEN≠0), set it in the following range.<br>・ ((VFP+VSA+period of VBP) * HTOTAL * pixel bit depth / 8) > 336 (*) |

Additional Rules for Sync Pulse mode

| parameter | description |
|---|---|
| Skew Calibration | When the Skew Calibration is used by TXVMSETR.VSEN=1, the period of HBP+HACTIVE+HFP should be set to longer than the period which the transmitted time of Sync Event Packet, the transition time for HS-LP-HS and period for Skew Calibration are added.<br>(HBP+HACTIVE+HFP) > (period for the Tx Sync Event Packet)+(period for transition time as HS-LP-HS)*2+(period for Skew Calibration) |
| Command Transfer mode Packet | When the Packet transmission of Command Transfer mode is used in the Tx Video mode (TXVMCR.EN=1), the period of HBP+HACTIVE+HFP should be set to the value within the range described below.<br>(HBP+HACTIVE+HFP) > (period for the Tx Sync Event Packet)+(period for the Tx Command Transfer Packet)+(Additional Time A)<br>(Additional Time A)=(period for transition time as HS-LP-HS) |
| HBP+HACTIVE+HFP | Transition of HS-LP-HS is caused within the period between current Sync Event Packet and next Sync Event Packet, the period of HBP+HACTIVE+HFP should be set to the value within the range described below.<br>(HBP+HACTIVE+HFP) > (period for the Tx Sync Event Packet)+(period for transition timeI HS-LP-HS) |
| HSA | ●No use of HSA Blanking Packet (TXVMSETR.HSABPEN=0)<br>Transition of HS-LP-HS is caused within the period between current Sync Event Packet and next Sync Event PackI the period of HSA should be set to the value within the raI described below.<br>HSA > (period for the Tx Sync Event Packet)+(period for transition time as HS-LP-HS)<br>Low power consumption is possible to Ifigure without using HSA Blanking Packet when the formula described aboIis satisfied. |
| | ●Use of HSA Blanking Packet (TXVMSETR.HSABPEN=1)<br>Transmission of Blanking Packet is caused within the period between current Sync Event Packet and next SynIvent Packet, the period of HSA should be set to the value wiIn the range described below.ISA * ColorDepth)/8 >= 10<br>Notice, HSA should be set to the value not less than minimum value of (period for the Tx Sync Event Packet) + (period for the Tx Blanking Packet). |
| HBP | ●No use of HBP Blanking Packet (TXVMSETR.HBPBPEN=0)<br>Transition of HS-LP-HS is caused within the period between current Sync Event Packet andIxt Sync Event Packet, the period of HSA should be set to thIalue within the range described below.<br>HSA > (period for the Tx Sync Event Packet)+(period for transition time as HS-LP-HS) |
| | ●Use of HBP Blanking Packet (TXVMSETR.HBPBPEN=1)<br>Transmission of Blanking Packet is caused within the period between current Sync Event Packet and next Sync Event Packet, the period of HPB should be set to the value within the range described below.<br>(HBP * ColorDepth)/8 >= 10<br>Notice, HPB should be set to the value not less than minimum value of (period for the Tx Sync Event Packet) + (period for the Tx Blanking Packet). |
| Frame CRC Packet | When sending Frame CRC packets (FDSETR.VFCEN(F(i))≠0), set the HBP+HACTIVE+HFP period to be longer than the following sum.<br>(Period for Sync Event Packet transmission)<br>(Period for FrameCRC packet (22Byte) transmission)<br>(Period for transition time as HS-LP-HS) |

Additional Rules for Sync-Event mode

| parameter | description |
|---|---|
| Skew Calibration | When the Skew Calibration is Id by TXVMSETR.VSEN=1, the period of HSA+HBP+HACTIVE+HFP should be set to longer than the period which the transmitted time of Sync Event Packet, the transition time for HS-LPI and period for Skew Calibration are added.<br>(HSA+HBP+HACTIVE+HFP) > (period for the Tx Sync Event Packet)+(period for transition time as HS-LP-HS)*2+(period for Skew Calibration) |
| Command Transfer mode Packet | When the Packet transmission of Command Transfer mode is usedI the Tx Video mode (TXVMCR.EN=1), the period of HSA+HBP+HACTIVE+HFP should bIet to the value within the range described below.<br>(HSA+HBP+HACTIVE+HFP) > (period for the Tx Sync Event Packet)+(period for the Tx Command Transfer Packet)+(Additional Time A)<br>(Additional Time A)=(period for transition time as HS-LP-HS) |
| HTOTAL | Transition of HS-LP-HS is caused within the period between current Sync Event Packet and next Sync Event Packet, the period of HTOTAL should be set to the value within the range described below.<br>HTOTAL > (period for the Tx SyncIent Packet)+(period for transition time as HS-LP-HS) |
| HSA+HBP | ●No use of HBP Blanking Packet (TXVMSETR.HBPBPEN=0)<br>Transition of HS-LP-HS is caused within the period between current Ic Event Packet and next Sync Event Packet, the period of HSA+HBPIould be set to the value within the range described below.<br>(HSA+HBP) > (period for the Tx Pixel Stream Packet)+(period for transition time as HS-LP-HS)<br>Low power consumption is possible at this configuration. |
| | ●Use of HBP Blanking Packet (TXVMSETR.HBPBPEN=1)<br>Transmission of Blanking Packet is caused within the period between cuInt Sync Event Packet and next Sync Event Packet, the period of HSIPB should be set to the value witI the range described below.<br>((HSA+HBP) * ColorDepth)/8 >= 10<br>Notice, HSA+HPB should be set to the value not less than minimum value of (period for the Tx Sync Event Packet) + (period for the Tx Blanking Packet). |

#### 67.4.1.1.1.1 Tx/Rx Command Transfer mode in Video mode

The Command Transfer mode Packet Tx is enabled only by HS, and is not supported by LP.

A request of Tx using Command Transfer mode in Video mode should be issued after confirmation of TXVMSR.ACT=1. Command Transfer mode Packet Tx is occurred in the BLLP period of the first Horizontal Line of Video mode Packet by the request of Tx using Command Transfer mode in Video mode. If the request of Tx using Command Transfer mode with requirement of response in Video mode, Bus Turn-Around with Packet Rx is occurred in the BLLP period of the first Horizontal Line of Video mode Packet. Refer to section 67.3.2.3 for detail. After the completion of Tx Command Transfer mode Packet without Bus Turn-Around, the Transmitter transitions to LP11. If the Packet is Tx Command Transfer mode Packet, Bus Turn-Around and Rx Packet are executed during the first Horizontal Line period.

All Command using Command Transfer mode described on Table 67.16" Command Transfer mode Packet" should be issued with limitation described in the Table 67.16.

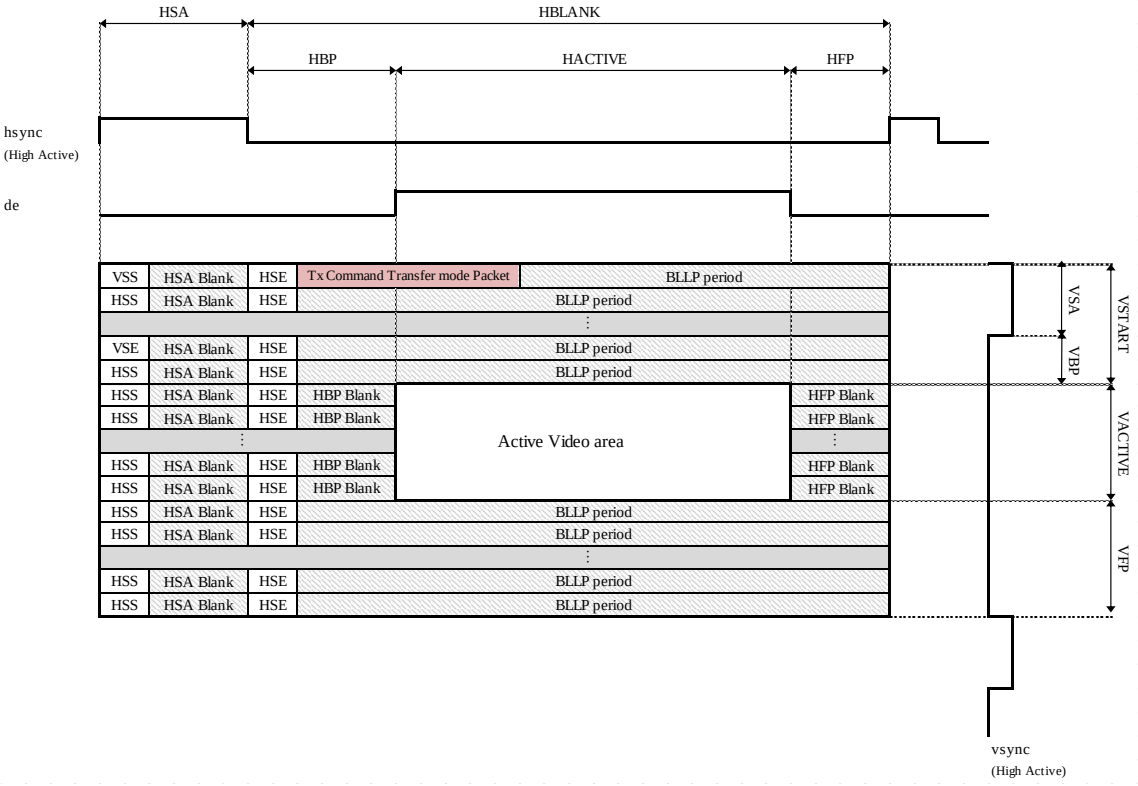



**Figure 67.31 Timings for Command Transfer mode in Video mode (Sync-Pulse mode)**

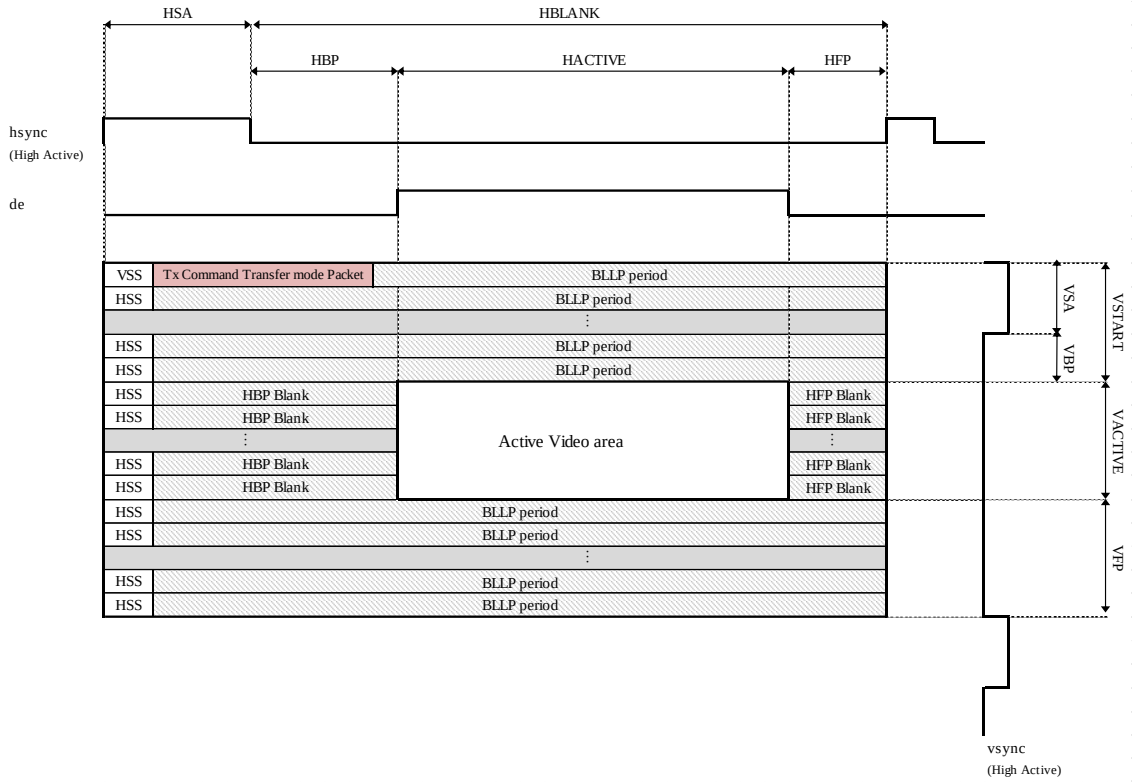



**Figure 67.32 Timings for Command Transfer mode in Video mode (Sync-Event mode)**

#### 67.4.1.1.1.2 Skew Calibration (Periodic de-skew) in Video mode

In Video mode, the Transmitter supports Skew Calibration at every Video Frame by H/W.

Skew Calibration should be enabled at the initial configuration phase described in 67.3.2.4.1. Skew Calibration is operated in the BLLP period of the Horizontal Line started VBP of Video mode Packet Skew Calibration Pattern is transmitted to connected device during the period specif“e” by SCALPSR.PTNSZ register. When the skew calibration pattern is sent, set "1" to SCALSR.CALLEQEND. The value of SCALPSR.PTNSZ should be within the range described on Table 67.16 ”Skew Calibration”.

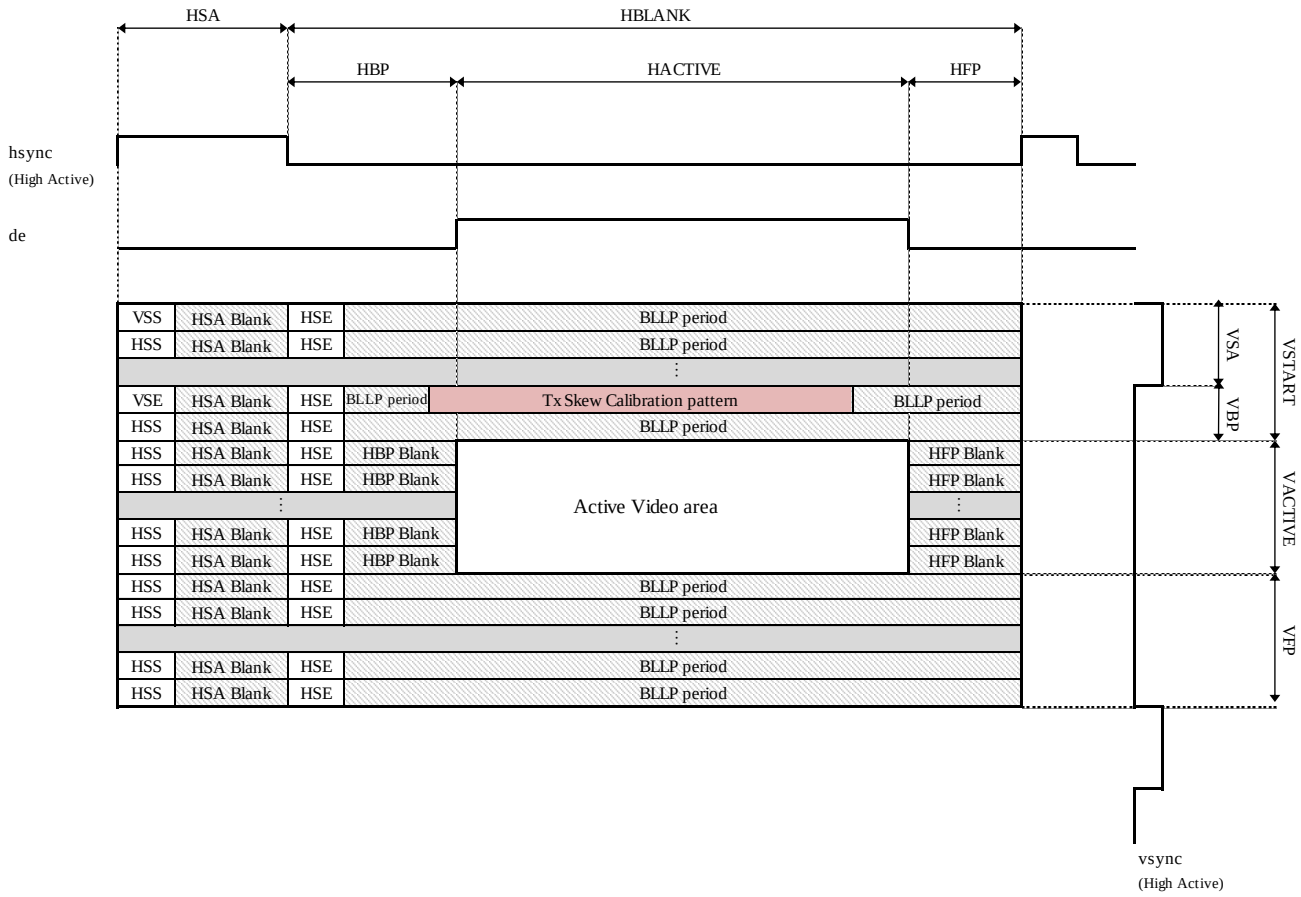



**Figure 67.33 Timings for Skew Calibration in Video mode (Sync-Pulse mode)**

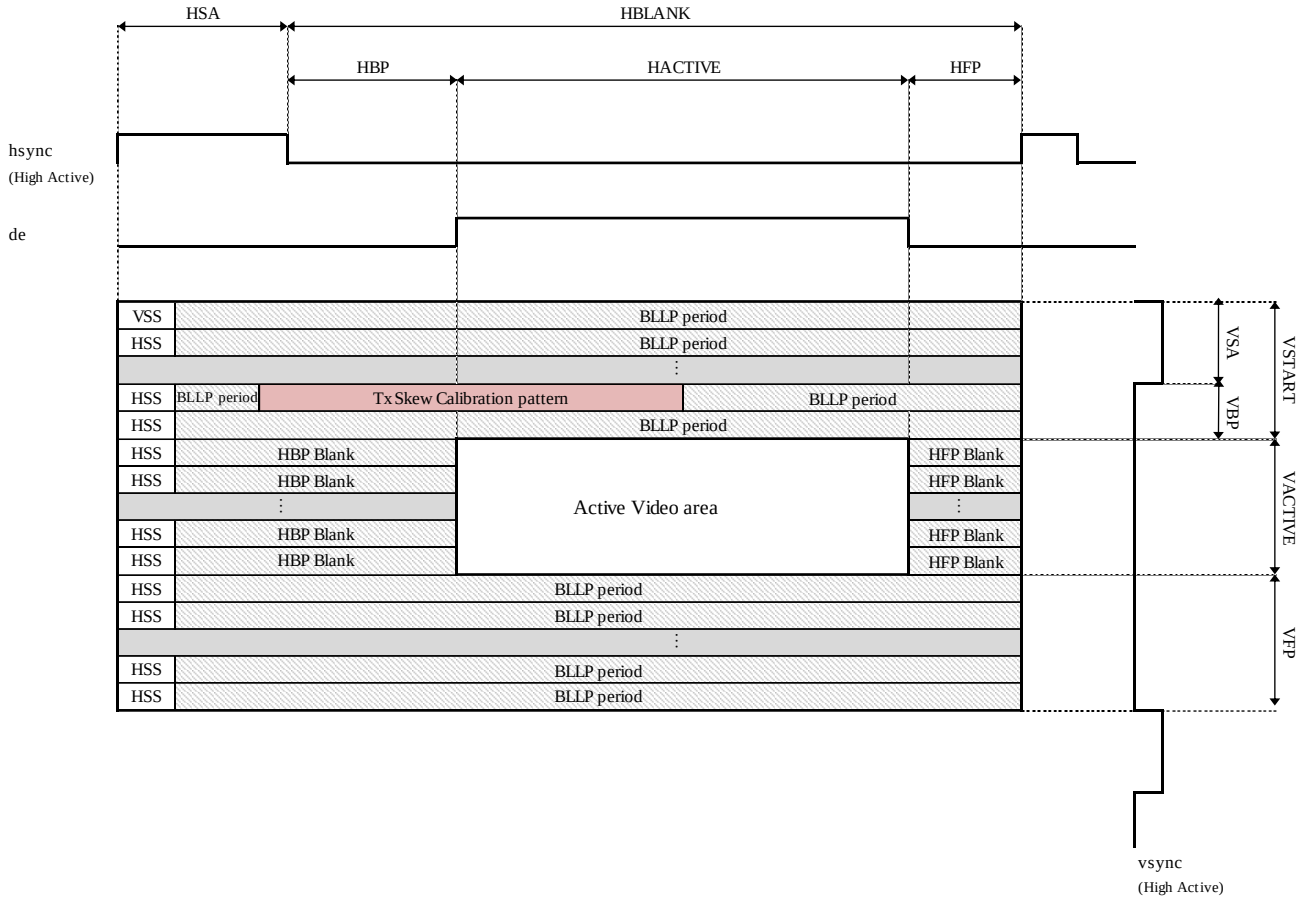



**Figure 67.34 Timings for Skew Calibration in Video mode (Sync-Event mode)**

#### 67.4.1.1.2 CSI-2

This section describes the Video Output interface on CSI-2 mode (IPSETR.CSI2M=1b).

##### 67.4.1.1.2.1 Timings for Video Output

Timings and Configurations for Video output are shown below.

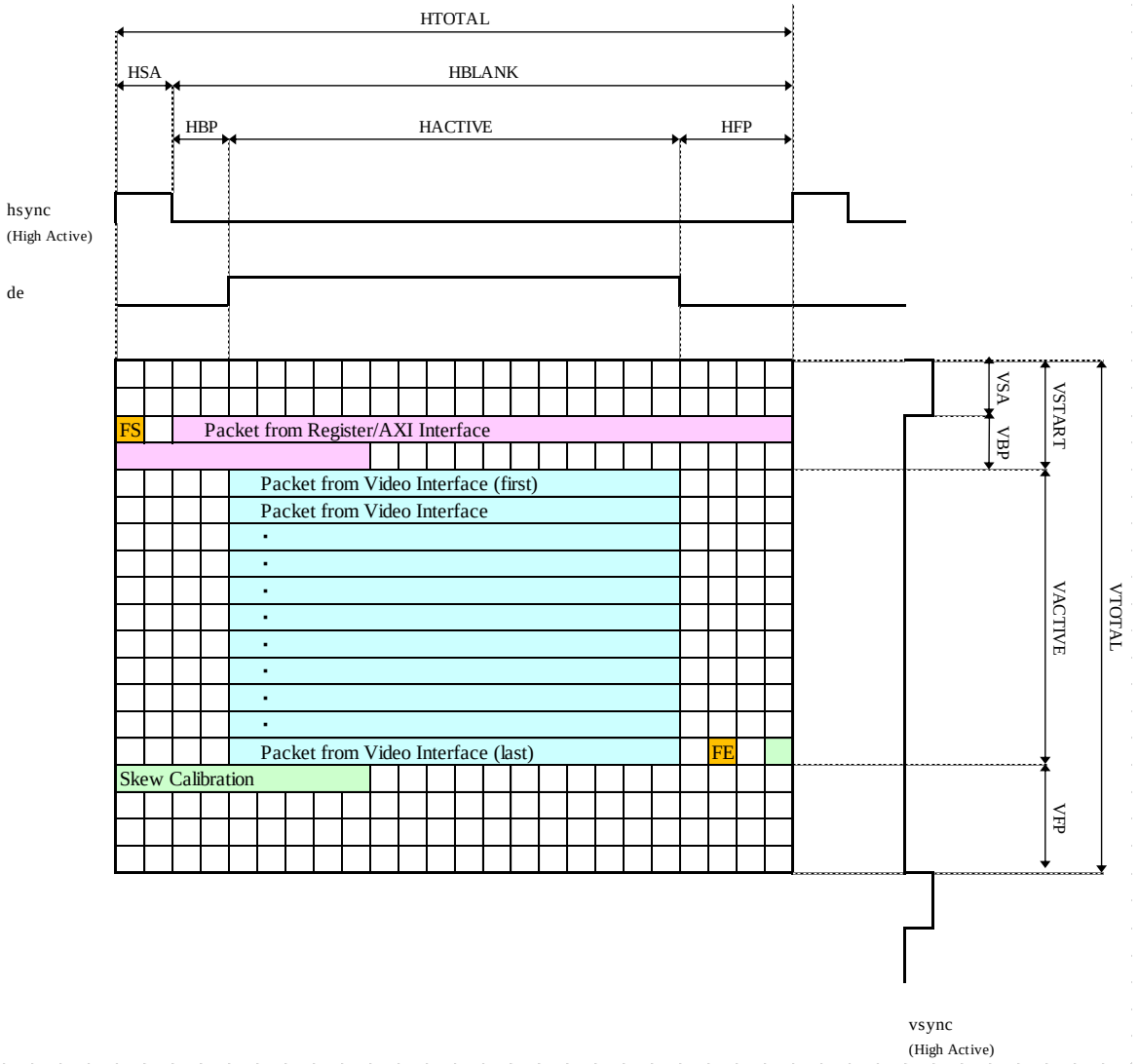



**Figure 67.35 Timings for Video Output (CSI-2)**

**Table 67.17 Video Output parametersISI-2)**

| category | Parameter | Description | remarks |
|---|---|---|---|
| Number of horizontal pixels | HSA | Horizontal sync signal active perio“ ”ixels | hsync signal assert period nu“b”r of pixels<br>High active: hsync signal = "1" period<br>Low active: hsync signal = "0" period |
| | HBLANK | Number of pixels for horizontal blending period | hsync negate period pixels for vertical blending period (inactive period) |
| | HBP | Horizontal back porch period pixels | Number of pixels between the end of the hsync signal assert signal and the start of the de signal assert during the vertical active period |
| | HACTIVE | Horizon“a” active period pixels | Number of pixels in de signal assert period<br>de signal = "1" period |
| | HFP | Horizontal front porch period pixels | Number of pixels in the period from the end of the de-signalIsert to the staIof the hsync signal assert |
| | HTOTAL | Total horizontal pixels | HSA+HBP+HACTIVE+HFP (HSA+HBLANK) |
| Number of vertical lines | VSA | Number of vertical “y”c signal active period lines | vsync “i”nal assert period<br>High active: vsync signal = "1" period<br>Low active: vsync signal = "0" period |
| | VBP | Number of vertical back porch period lines | Number of lines until the start of the vertical sync signal active period |
| | VACTIVE | Number of vertical active period lines | Number of lines being de-signaled |
| | VFP | Number of vertical front porch period lines | Number of lines from the end of the vertical active period to the start of the vertical synchronization signal active period |
| | VSTART | Number of vertical active period start lines | VSA + VBP |
| | VTOTAL | Total vertical lines | VSA+VBP+VACTIVE+VFP |

**Table 67.18 Configuration for Video Output interface (CSI-2)**

| Parameter | Description |
|---|---|
| All | All parame“ers related to”InI Video should be set to the value of not less In 1.<br>Example for "AIIarameters": HSA, HBP, HACTIVE, HFP, VSA, VBP, VACTIVE, VFP |
| HFP+HSA+HBP | The period of HFP+HSA+HBP should be set to longer than the period of tItransition time for HS-LP-HS of D-PHY. |
| Frame End Packet<br>Skew Calibration<br>Frame CRC packet | ((HSA+HBP+HACTIVE+HFP)× (VFP+VSA) + HFP) period(*1) should be longer than the following sum. |

| Parameter | Description |
|---|---|
| | ・(period for transition time as HS-LP-HS)<br>・（period of Frame CRC Packet generation < Frame CRC transmission setting enabled (FDSETR. VFCEN(F(i))≠00b)＞）<br>・（period of Frame CRC Packet transmission period < Frame CRC transmission setting enabled (FDSETR. VFCEN(F(i))≠00b)＞）<br>・(period for transition time as HS-LP-HS <Frame CRC transmission setting enabled (FDSETR. VFCEN(F(i))≠00b)＞）<br>・（period of Frame End Packet generation )<br>・(period of Frame End Packet transmission)<br>・(period for transition time as HS-LP-HS)<br>・(Send period of skew calibration < when the skew calibration transmission setting is enabled (TXVMSET(F(i))R.VSEN=1>)<br>・(period for transition time as HS-LP-HS < When skew calibration transmission transmission setting is enabled (TXVMSET(F(i)R.VSEN=1) >)<br>(*1) When transmitting different types of videIata by switching frames, the parameters used for calculating tI period are as follows.<br>HFP+ ((HSA+HBP+HACTIVE+HFP) ×VFP) : Parameter value of the current frame<br>((HSA+HBP+HACTIVE+HFP) ×VSA) : Parameter values for the next frame |
| Command Transfer mode Packet | When thelcket transmission of Command Transfer mode is used in the Tx Video mode (TXVMCR.EI), the period of (HSA+HBP+HACTIVE+HFP)*VBP should be set to the value within the range described below.<br>(HSA+HBP+HACTIVE+HFP)*VBP > (period for the Tx Pixel Stream Packet)+(period for transition time as HS-LP-HS) + (period for generation of Command Transfer mode Packet. ) +(period for Skew Calibration)+(period for transition time as HS-LP-HS) |
| HACTIVE | TXVMPSPHSET(F(i))R.DT — TXVMVPRMSET3(F(i))R.HACTIVE<br>RGB565 — Set 1 to 32767 because the 1Pixel consists of 2Byte and the maximum length of the CSI-2 packet is 65535Byte.<br>RGB666 — Set 4 to 29124 because the 4Pixel consists of 9Byte and the maximum length of the CSI-2 packet is 65535Byte. However, the setting value should be a multiple of 4.<br>RGB888 — Set 1 to 21845 because the 1Pixel consists of 3Byte and the maximum length of the CSI-2 packet is 65535Byte.<br>YUV422 8-bit — Set 2 to 32766 because the 2Pixel consists of 4Byte and the maximum length of the CSI-2 packet is 65535Byte. However, the setting value should be a multiple of 2. |
| VFP+VSA+VBP, HTOTAL | Tx of Frame CRC (FDSETR.VFCEN(F(i))≠0)<br>The values should be within the range by the following.<br>((VFP+VSA+VBP) * HTOTAL * ColorDepth / 8) > 336 (*)<br>(*) When transmitting different types of video data by switching frames, the parameters used for calculating this period are as follows.<br>(VFP×HTOTAL* Pixel Bit Depth / 8): Parameter value of the current frame<br>(VSA+VBP) * HTOTAL* pixel bit depth / 8): parameter value for the next frame |

#### 67.4.1.1.2.2 Packet generation from Input Video

The Transmitter supports Packet Generator for CSI-2 using input signal from Video Input interface. The Frame Start Packet, Frame End Packet and Video Packet are generated automatically by this Packet Generator.

##### 67.4.1.1.2.2.1 Frame Start Packet

Frame Start Packet (FS) is automatically generated and transmitted at the detection of vsync signal.

Figure 67.36 shows the format of Frame Start Packet and Table 67.19 shows parameters for Frame Start Packet.

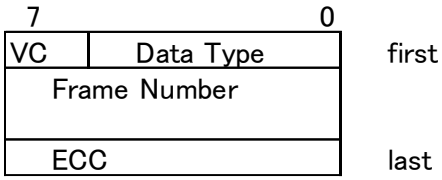



**Figure 67.36 Frame Start/Frame End packet format**

**Table 67.19 Parameters of Frame Start (FS) Packet**

| Field | Bit count | Description | |
|---|---|---|---|
| VC | 2 | Value of TXVMPSPHSET(F(i))R.VC | |
| Data Type | 6 | Frame Start Code (H'00) | |
| Frame Number | 16 | Frame count numbered | |
| | | TXVMSET2(F(i))R.FNUM=0 | Set to 0 to Frame Number |
| | | TXVMSET2(F(i))R.FNUM=others | Set to 1 to Frame Number in the Frame Start Packet to be transmitted first. When complete of transmission of the Frame End Packet *1, the Frame Number count is incremented by one internally. The Frame Number resets to 1 after the same value of TXVMSET2(F(i))R.FNUM transmitted.<br><br>Note: *1 Discarded Frame End Packet is included because of the failed transmission of the Frame End Packet. |
| ECC | 8 | This value is automatically calculated by the values of VC, Data Type and Frame Number. | |

##### 67.4.1.1.2.2.2 Frame End Packet

Frame End Packet (FE) is automatically transmitted as HS Tx after transmission of Video Packets specified by TXVMVPRMSET1(F(i))R.VACTIVE. Frame End Packet (FE) is automatically transmitted as HS Tx after the Tx Frame CRC Packet is used by FDSETR.VFCEN≠00b.

Frame End Packet is not transmitted when the Frame Start Packet is not transmitted at the Frame.

Figure 67.36 shows the format of Frame End Packet and Table 67.20 shows the parameters for Frame End Packet.

**Table 67.20 Parameters of Frame End (FE) Packet**

| Field | Bit count | Description |
|---|---|---|
| VC | 2 | Value of TXVMPSPHSET(F(i))R.VC |
| Data Type | 6 | Frame End Code(H'01) |
| Frame Number | 16 | Frame count numbered<br>TXVMSET2(F(i))R.FNUM=0: Set to 0 to Frame Number<br>TXVMSET2(F(i))R.FNUM=others: Set to 1 to Frame Number in the Frame Start Packet to be transmitted first. When complete of transmission of the Frame End Packet *1, the Frame Number count is incremented by one internally. The Frame Number resets to 1 after the same value of TXVMSET2(F(i))R.FNUM transmitted.<br>Note: *1 Discarded Frame End Packet is included because of the failed transmission of the Frame End Packet. |
| ECC | 8 | This value is automatically calculated by the values of VC, Data Type and Frame Number. |

67.4.1.1.2.2.3 Video Packet

Video Packet transfer is started by both condition of Asserted DE signal and Enabled Pixel Data from the video input interface. Video Packet without FS in the frame is not transferred.

Figure 67.37 shows video packet format and Table 67.21 shows parameters for video packet.

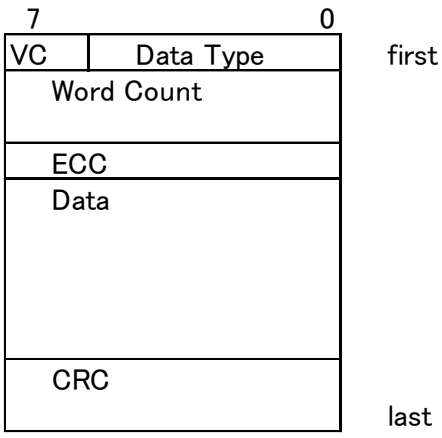



**Figure 67.37 Video Packet Format**

**Table 67.21 Video Packet Parameters**

| Field name | Bit count | Description for value |
|---|---|---|

| | | |
|---|---|---|
| VC | 2 | Value of TXVMPSPHSET(F(i))R.VC |
| Data Type | 6 | Value of TXVMPSPHSET(F(i))R.DT<br>The Transmitter module supports RGB565(H'22), RGB666(H'23), RGB888(H'24), YUV422 8-bit(H'1E). |
| Word Count | 16 | Value of Word Count which is automatically calculated by value of TXVMVPRMSET0(F(i))R.BPP, TXVMVPRMSET3(F(i))R.HACTIVE and TXVMVPRMSET3(F(i))R.HSBYTE. |
| ECC | 8 | Automatically calibrated value from VC, Data Type, and Word Count value. |
| Data | Word Count *8 | Pixel Data from video input interface. |
| CRC | 16 | Automatically calculated value from Data value. |

#### 67.4.1.1.2.3 Tx Command Transfer mode Packet in Video mode

In Video mode, the Transmitter supports Tx Command Transfer mode Packet. Tx Command Transfer mode Packet should be transmitted after Frame Start Packet by the request of Tx Command Transfer mode Packet described in section 67.3.2.3. The request of Tx Command Transfer mode Packet should be based on the condition described on Table 67.18 "Command Transfer mode Packet". The Command Transfer mode Packet can be transmitted once per Frame in Video mode by the request.

#### 67.4.1.1.2.4 Skew Calibration (Periodic de-skew) in Video mode

In Video mode, the Transmitter supports Skew Calibration at every Video Frame by H/W. Skew Calibration should be enabled at the initial configuration phase described in 67.3.2.4.1. Sen" "kew Calibration Pattern after Frame End packet transmission. When the skew calibration pattern is sent, set "1" to SCALSR.CALLEQEND. Skew Calibration Pattern is transmitted to connected device during the period specified by SCALPSR.PTNSZ register. The value of SCALPSR.PTNSZ should be within the range described on Table 67.18 "Skew Calibration".

### 67.4.2 USE CASE

### 67.4.2.1 Inter-Chip Connection Using MIPI CSI-2 I/F

The Data Transfer on the memory is enabled through CSI-2-Tx interface to another R-Car V4M device using CSI-2-Rx.

Refer to Figure 67.44 and Figure 67.45 for example.

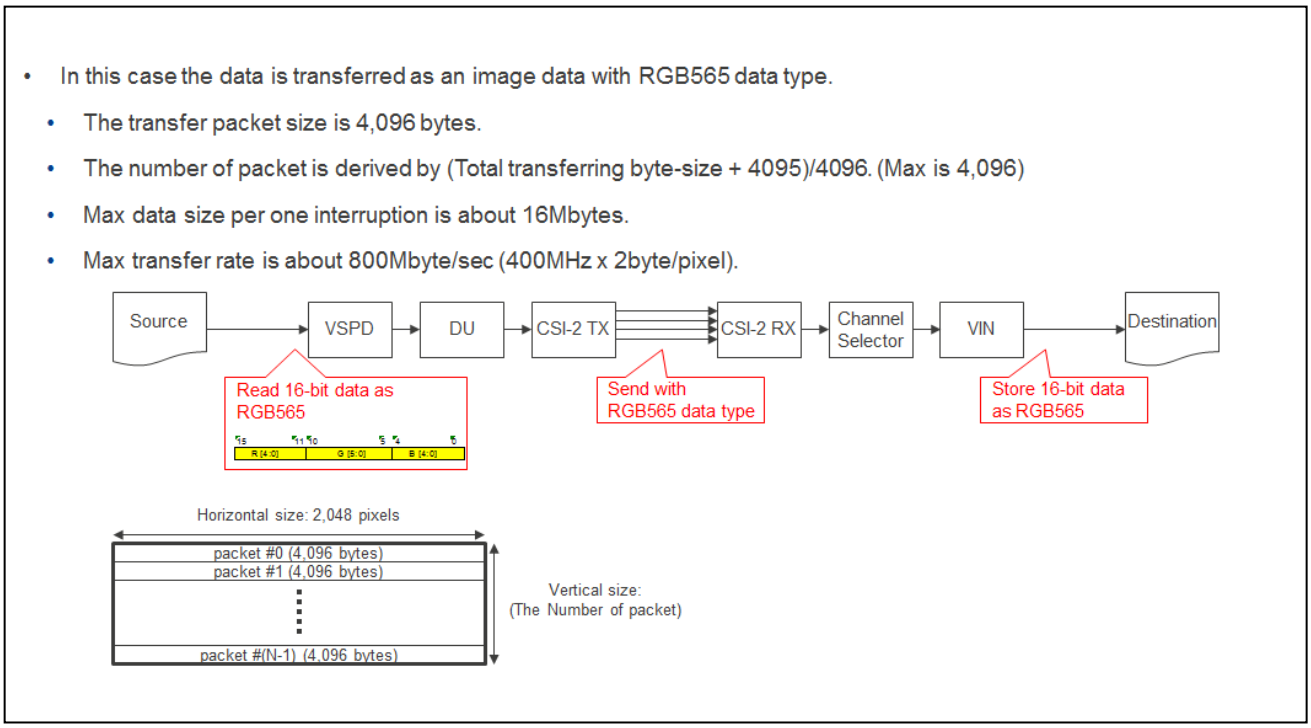



**Figure 67.44 transferred as an image data with RGB565 data type**

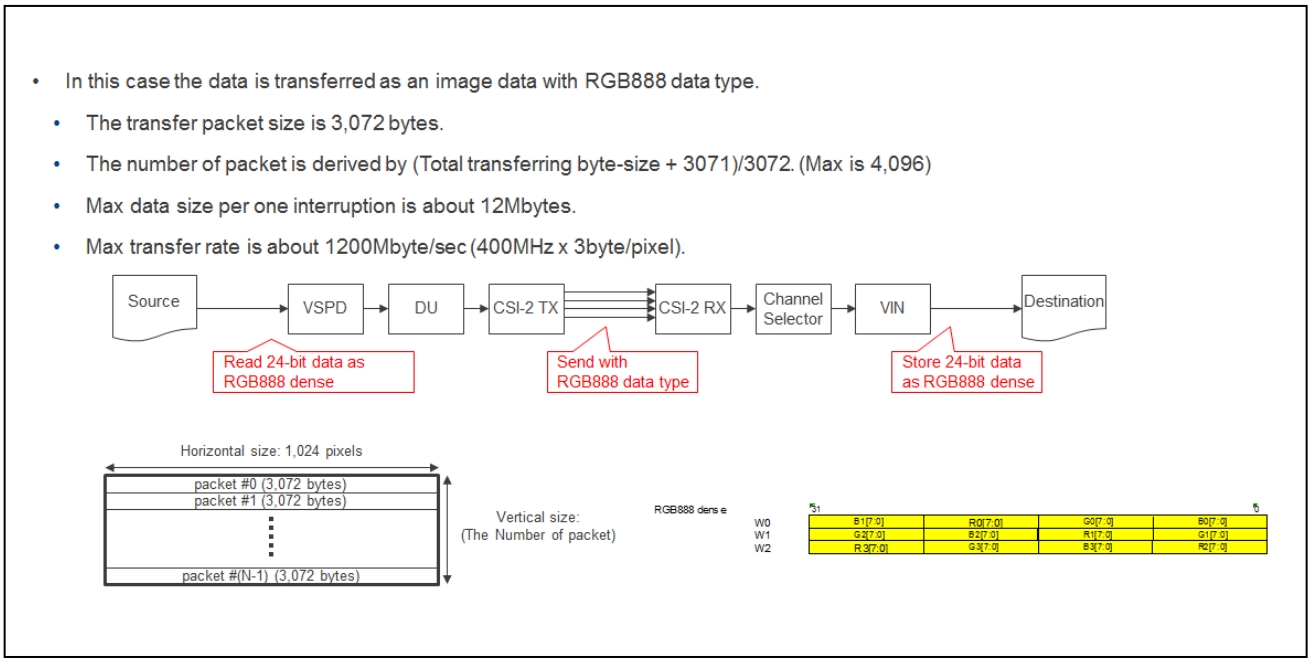



**Figure 67.45 transferred as an image data with RGB888 data type**

### 67.4.3 Example Configurations

This section describes the setting of Full-HD video transfer by using video mode as an example.

Format : RGB24bpp (DSI)

Data Lane Count : 4Lane

Over view for this example is following

1: Start the Start-up sequence (described in 67.3.5) with the parameters described on 67.4.3.1.

2: Start HS Tx Clock (described in 67.3.2.11.2) before HS Tx and initial de-skew.

3: Initial de-skew (described in 67.3.2.11.2) is needed for HS Tx over 1.5G before starting HS Tx.

4: Set some parameters to connected Peripheral device by using Command transfer mode. (described in 67.3.2.3)

5: Start the video mode transfer (described in 3.2.4 67.3.2.4 Figure 67.12 67.4.3.2

Periodic de-skew during video mode transfer is optional irrespective of data rate.

#### 67.4.3.1 Clock parameters

Fout = dotclock frequency * ColorDepth / 2 / DataLaneCount

= 148.5 * 24 / 2 / 4

= 445.5 MHz (891 Mbps)

Fclkin = 16.66 MHz

24 MHz ≥ Fclkin / N ≥ 8 MHz

4000 MHz ≥ M * Fclkin / (N * 2) ≥ 2000 MHz : Fout = M * Fclkin / (N * 2 * 8)

| CLOCKSET2.n | CLOCKSET2.m |
|---|---|
| D’1 | D’856 |

| Fout[MHz] | CLOCKSET2. vco_cntrl[5:0] | CLOCKSET3. cpbias_cntrl[6:0] | CLOCKSET3. gmp_cntrl[1:0] | CLOCKSET3. int_cntrl[5:0] | CLOCKSET3. prop_cntrl[5:0] |
|---|---|---|---|---|---|
| 426.56 - 500 | B' 010000 | B'0000000 | B'00 | B'001000 | B’001010 |

| Range (Mbps) | Default Bit Rate (Mbps) | hsfreqrange[6:0] |
|---|---|---|
| 843.125 – 958.125 | 900 | B’0101001 |

| LPCLKSET 19.05MHz configuration | CFGCLKSET 25MHz configuration | VCLKSET | VCLKEN |
|---|---|---|---|
| H'114 | H'11F | H'000102F | B'1 |

### 67.4.3.2 Video mode parameters

| TXVMPSPHSETR | |
|---|---|
| VC | DT |
| H'0 | H’3E |

| TXVMSETR | | | |
|---|---|---|---|
| **SYNSEQ** | **HFPBPEN** | **HBPBPEN** | **HSABPEN** |
| B'0 | B'1 | B'1 | B'1 |

| TXVMVPRMSET0R | | | |
|---|---|---|---|
| **VSPOL** | **HSPOL** | **CSPC** | **BPP** |
| *1 | *1 | B'0 | B'010 |

Note: *1 Set polarity of VSYNC/HSYNC. (Refer to section DU for polarity)

| HSA | HBP | HACTIVE | HFP | VSA | VBP | VACTIVE | VFP |
|---|---|---|---|---|---|---|---|
| D’44 | D’148 | D’1920 | D’88 | D’5 | D’36 | D'1080 | D'4 |

## 67.5 Safety Mechanisms

### 67.5.1 Safety Mechanism 1

Replace this subsection name to safety mechanism name (e.g. ECC for IP Memory Management Unit (IPMMU ). Duplicate this subsection for each safety mechanism.

#### 67.5.1.1 Overview

- Supports EDC Check
- Supports Video input Check (Vsync, Hsync, DE)
- Supports Tx Video Packet Inspection
- Supports Tx PPI Inspection
- Supports Tx Frame CRC (1 CRC per Frame)

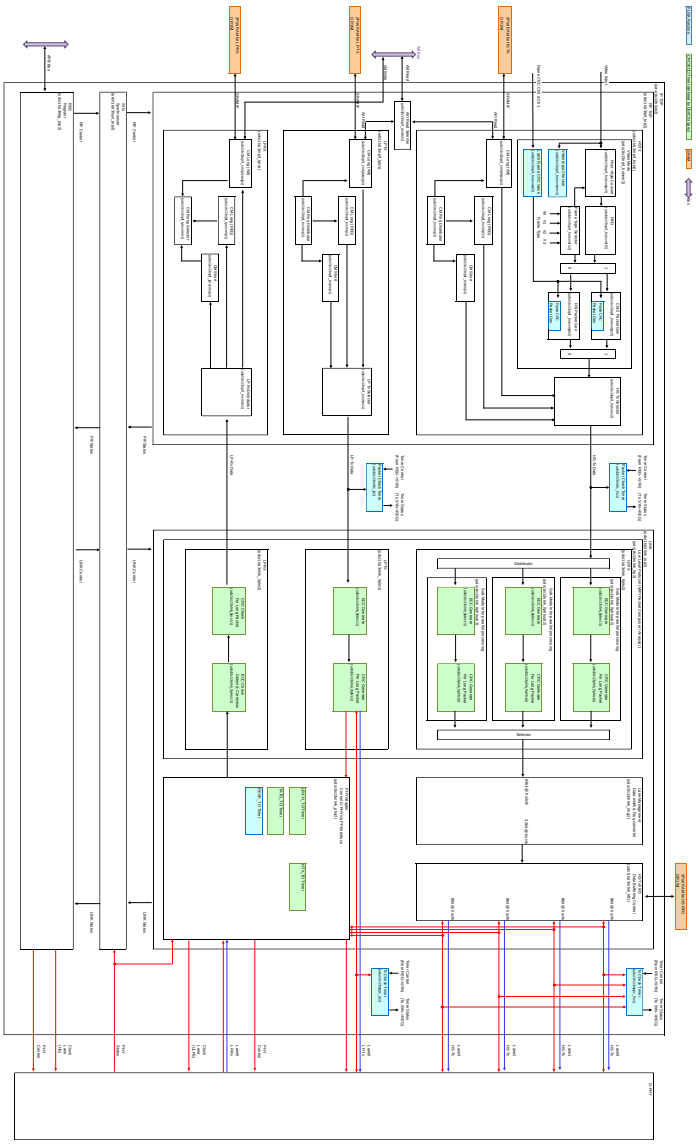


**Figure 67.46 Block Diagram of DSI/CSI-2-TX module (with FuSa)**

#### 67.5.1.2 Related Registers

Refer to 67.2 Register Description.

#### 67.5.1.3 Hardware Description

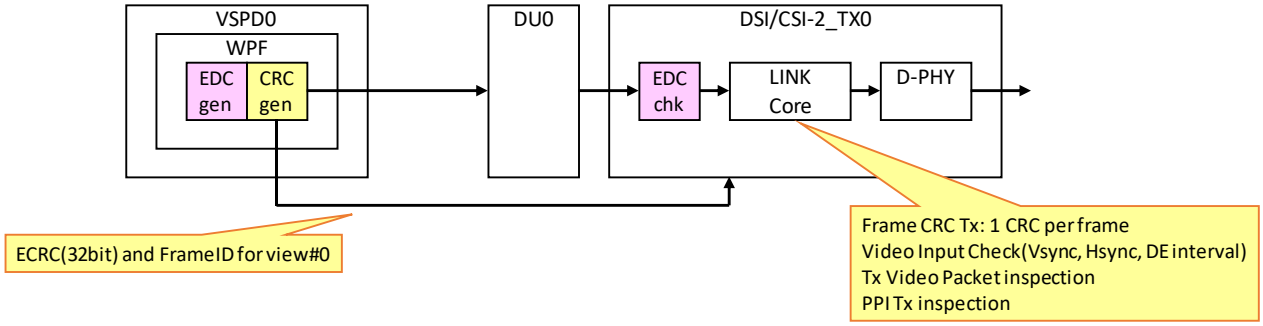



**Figure 67.47 Block Diagram (with FuSa)**

#### 67.5.1.3.1 Fault detection

##### 67.5.1.3.1.1 Tx Frame CRC (1 CRC per Frame)

When set to FDSETR-VFCEN(F(i))≠0, the CRC value for one frame of pixel data calculated by the upper layer is packetized and transmitted.Peripheral device can detect the presence or absence of fault to pixel data by comparing this CRC value with the CRC value calculated from the received pixel data.

DSI mode (IPSETR.CSI2M=B’0) :

Frame CRC can be transmit during video mode operation. When the video mode initially set in 67.3.2.4.1 is set, frame CRC transmission setting is enabled (FDSETR. VFCEN(F(i))≠00b) transmits the Generic Long Write Packet immediately after the "HSE packet (in sync pulse mode) or HSS packet (in sync event mode) on the VFP start horizontal line".Table 67.23 Generic Long Write packet parameters shows the parameters of the packet.

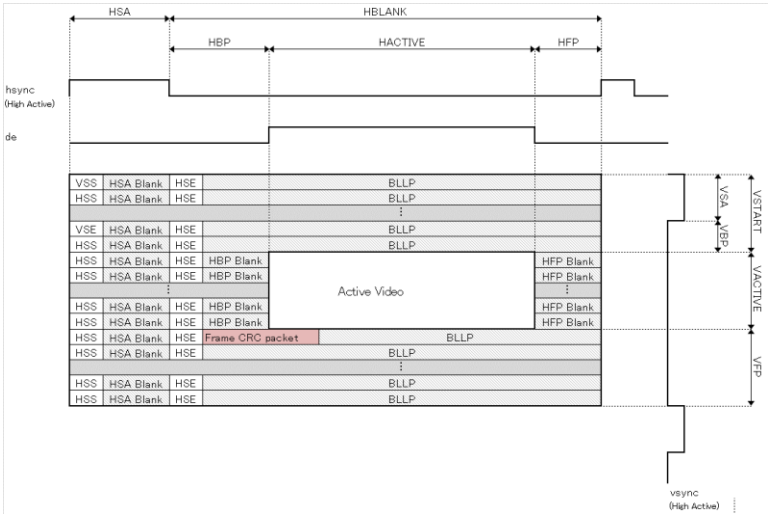



**Figure 67.48 Frame CRC transmission operation timing during video mode operation (sync pulse mode)**

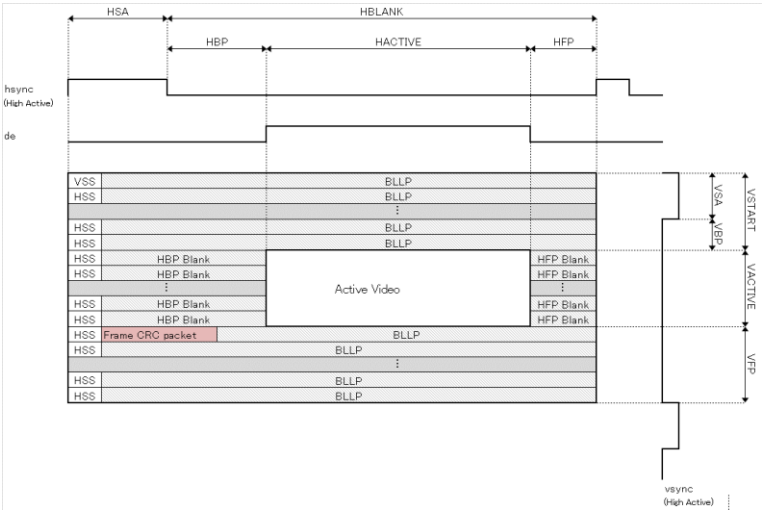



**Figure 67.49 Frame CRC transmission operation timing during video mode operation (sync event mode)**

**Table 67.23 Generic Long Write packet parameters**

| Field | Number of bits | Setting value | |
|---|---|---|---|
| VC | 2 | TXVMPSPHSET(F(i))R.VC | |
| Data Type | 6 | Generic Long Write Code (H'29) | |
| Word Count | 16 | H'10 (16 Byte) | |
| ECC | 8 | Calculated from VC, Data Type, Word Count values and automatically granted. | |
| Data | Word Count ×8 | Byte0 : Ch.0 Frame CRC[31:24]<br>Byte1 : Ch.0 Frame CRC[23:16]<br>Byte2 : Ch.0 Frame CRC[15:8]<br>Byte3 : Ch.0 Frame CRC[7:0]<br>Byte4 : Ch.0 {ERR flag, Frame ID[2:0], Frame number value[11:8]}<br>Byte5 : Ch.0 Frame number value[7:0]<br>Byte6 : Reserved (0x00)<br>Byte7 : Reserved (0x00) | Byte8 : Ch.1 Frame CRC[31:24]<br>Byte9 : Ch.1 Frame CRC[23:16]<br>Byte10 : Ch.1 Frame CRC[15:8]<br>Byte11 : Ch.1 Frame CRC[7:0]<br>Byte12 : Ch.1 {ERRflag, Frame ID[2:0], Frame number value[11:8]}<br>Byte13 : Ch.1 Frame number value[7:0]<br>Byte14 : Reserved (0x00)<br>Byte15 : Reserved (0x00) |
| CRC | 16 | Calculated from data value and automatically granted. | |

CSI-2 mode (IPSETR.CSI2M=B'1) :

Frame CRC can be transmit during video mode operation. When the video mode is initially set as shown in 67.3.2.4.1, the Frame CRC transmission setting is enabled (FDSETR.VFCEN(F(i))≠00b) to send embedded 8-bit non-image Data packets after the last line of video packets are sent.

Figure 67.50 Embedded 8-bit non-image Data packet format shows the format of the packet, and Table 67.24 shows the parameters of the packet.

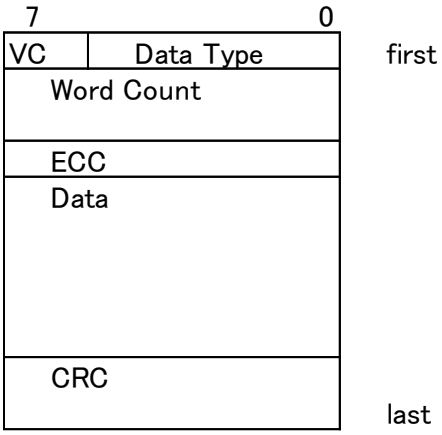



**Figure 67.50 Embedded 8-bit non-image Data packet format**

**Table 67.24 Embedded 8-bit non-image Data packet parameters**

| Field | Number of bits | Setting value | |
|---|---|---|---|
| VC | 2 | TXVMPSPHSET(F(i))R.VC | |
| Data Type | 6 | Embedded 8-bit non image Data Code (H'12)。 | |
| Word Count | 16 | H'10 (16 Byte) | |
| ECC | 8 | Calculated from VC, Data Type, Word Count values and automatically granted. | |
| Data (*1) | Word Count ×8 | Byte0 : Ch.0 Frame CRC[31:24]<br>Byte1 : Ch.0 Frame CRC[23:16]<br>Byte2 : Ch.0 Frame CRC[15:8]<br>Byte3 : Ch.0 Frame CRC[7:0]<br>Byte4 : Ch.0 {ERR flag, Frame ID[2:0], | Byte8 : Ch.1 Frame CRC[31:24]<br>Byte9 : Ch.1 Frame CRC[23:16]<br>Byte10 : Ch.1 Frame CRC[15:8]<br>Byte11 : Ch.1 Frame CRC[7:0]<br>Byte12 : Ch.1 {ERRflag, Frame ID[2:0], |

| | | | |
|---|---|---|---|
| | | Frame number value[11:8]}<br>Byte5：Ch.0 Frame number value [7:0]<br>Byte6：Reserved (0x00)<br>Byte7：Reserved (0x00) | Frame number value[11:8]}<br>Byte13：Ch.1 Frame number value [7:0]<br>Byte14：Reserved (0x00)<br>Byte15：Reserved (0x00) |
| CRC | 16 | Calculated from data value and automatically granted. | |

#### 67.5.1.3.1.2 Video input Check (Vsync, Hsync, DE)

Compares the video input (vsync, hsync, de, nvb) with the parameters set to TXVMVPRMSET1,2,3,4(F(i))R, and sets FDSR.VINERR=1 when there is an unexpected input. When transmitting different types of video data by switching frames (Chapter 67.3.2.4.3), the parameters are switched for each frame type for comparison.

Check for the following periods:

・Check start:

Detects VSYNC start edge after TXVMCR.EN=0->1 set.

After DU setting completion, set TXVMCR.EN=1.

・Check end:
TXVMSETR.VSTPM=0: TXVMCR.EN=1->0 clear.

TXVMSETR.VSTPM=1: Detects VSYNC start edge after clearing TXVMCR.EN=1->0.

#### 67.5.1.3.1.3 Tx Video Packet Inspection

When set to FDCR.DTTEN=1, a timer that monitors outbound packets inside the Transmitter module (between the PIF block and the LINK block) is started.

After startup, the timer counts up every cycle (hs2clk for HS, lpclk for LP). When it detects the transmission of a monitored packet, it resets the timer and resumes counting up after the monitored packet transmission is complete.

1 is set to FDDSR.DTTO when the target packet is not sent at the transfer speed set by FDDTTMSETR.SPDTYP for the period set by FDDTTMSETR.DTOVL, DTTOSCL".

The data type of the packet to be monitored should be set to FDDTEN0,1R.

It is prohibited to change the following registers between FDCR-DTTEN=1.

FDDTTMSETR、FDDTEN0R、FDDTEN1R

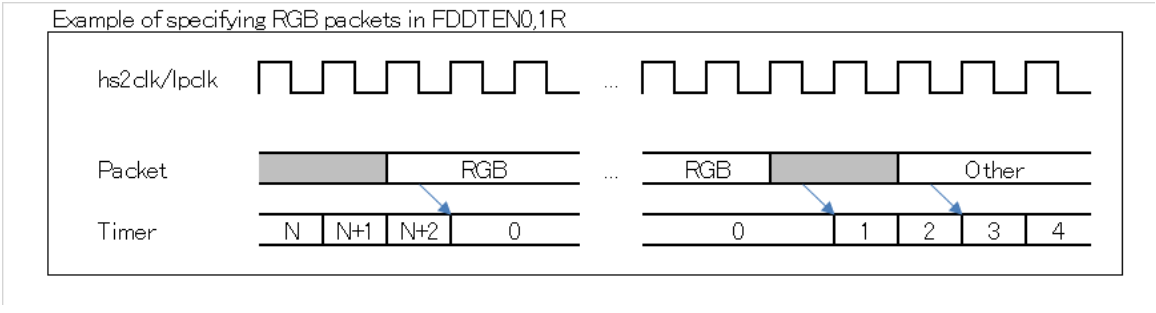



**Figure 67.51 Example of timer operation of Tx Video Packet Inspection**

#### 67.5.1.3.1.4 Tx PPI Inspection

When set to FDCR.PPITEN=1, a timer that monitors transmissions at the PPI end of the Transmitter module is activated.

The behavior of the timer varies depending on the transfer speed.

- HS (FDPPITMSETR.SPDTYP="0")
  After startup, the timer counts up every cycle (hsclk). When data transmission of the number of cycles set in the register is detected (considered as a single data transmission even if multiple packets are connected), the timer is reset, and the timer count-up is resumed after the data transmission is completed.
  The timer will not be reset unless data is transferred in all effective lanes set in TXSETR.LANECNT. So, for example, if one of the active lanes fails and no data is transferred, the timer will not reset and will eventually time out.

- LP (FDPPITMSETR.SPDTYP="1")
  After startup, the timer counts up every cycle (lplk). When it detects a data transmission of the number of cycles set in the register, it resets the timer and resumes counting up the timer immediately afterwards (without waiting for the data to complete).

During the period set to FDPPITMSETR.PPITOVL, PPITOSCL, FDSR.PPITO sets 1 when no more than the number of cycles set to FDPPITMSETR.PPITCVL, PPCCSCL is performed at the transfer speed set by FDPPITMSETR.SPDTYP.
During FDCR.PPITEN=1, it is prohibited to change the following registers:

FDPPITMSETR

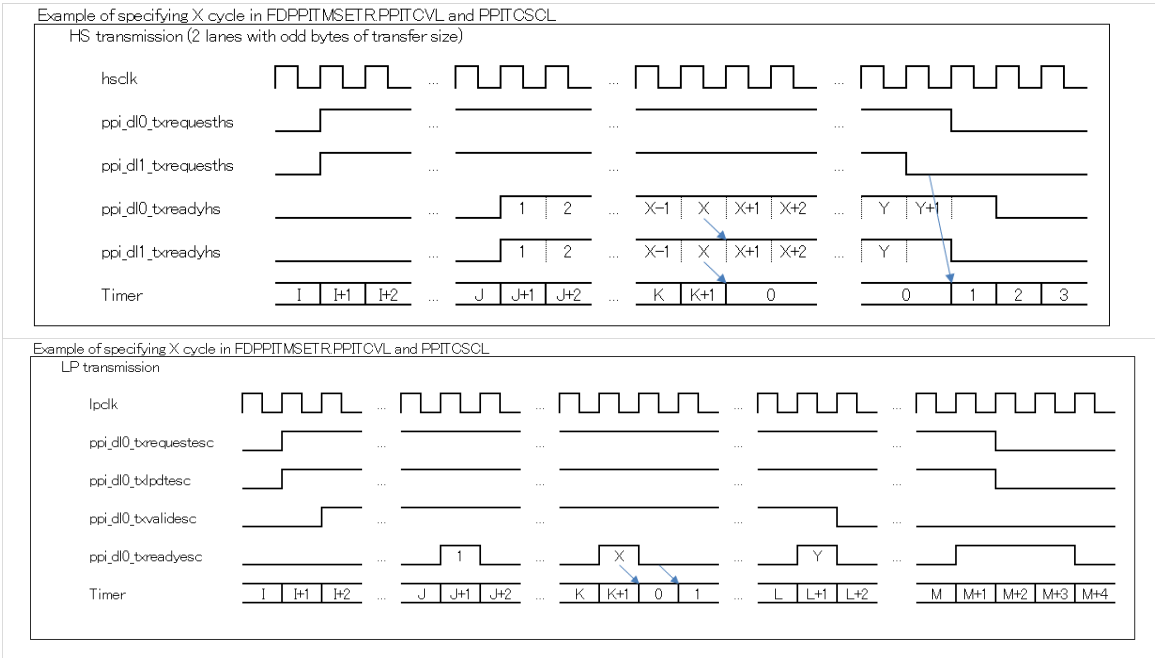



**Figure 67.52 Example of PPI monitoring timer operation**

# 81. Serial Sound Interface Unit (SSIU)

## 81.1 Overview

The SSIU, incorporating a SSI module, allows independent operation of SSI modules, operation of multiple SSI modules sharing the same serial clock, and connection or split of multichannel data. Refer to Section 82 to understand single SSI module in detail.

Note: Configure the system so that connected external devices can operate on the same clock source.

### 81.1.1 Features

- Incorporates the SSI module.
- Supports 6 channels/1 sound source with the SSI module.
- Supports 8 channels/1 sound source with the SSI module.
- TDM format (Basic Configuration) corresponds to 4, 6 or 8-channel data.
- TDM format corresponds to 16-channel data (TDM-16ch mode).
- Handles 8-channel data on the serial bus and 6-channel data in the SoC (TDM extend mode).
- Handles 6-channel data on the serial bus and 8-channel data in the SoC (TDM extend mode).
- Connects monaural or stereo data to output TDM format data (TDM split mode).
- Splits TDM format input data into monaural or stereo data (TDM split mode).
- Connects stereo or multi data to output TDM format data (TDM ex-split mode).
- Splits TDM format input data into stereo or multi data (TDM ex-split mode).
- The frequency range of SCK signal is from 297.3 kHz to 12.5MHz at master mode, and from 297.3 kHz to 15.1 MHz at slave mode.

### 81.1.2 Block Diagram

Figure 81.1 show the SSIU block diagrams.

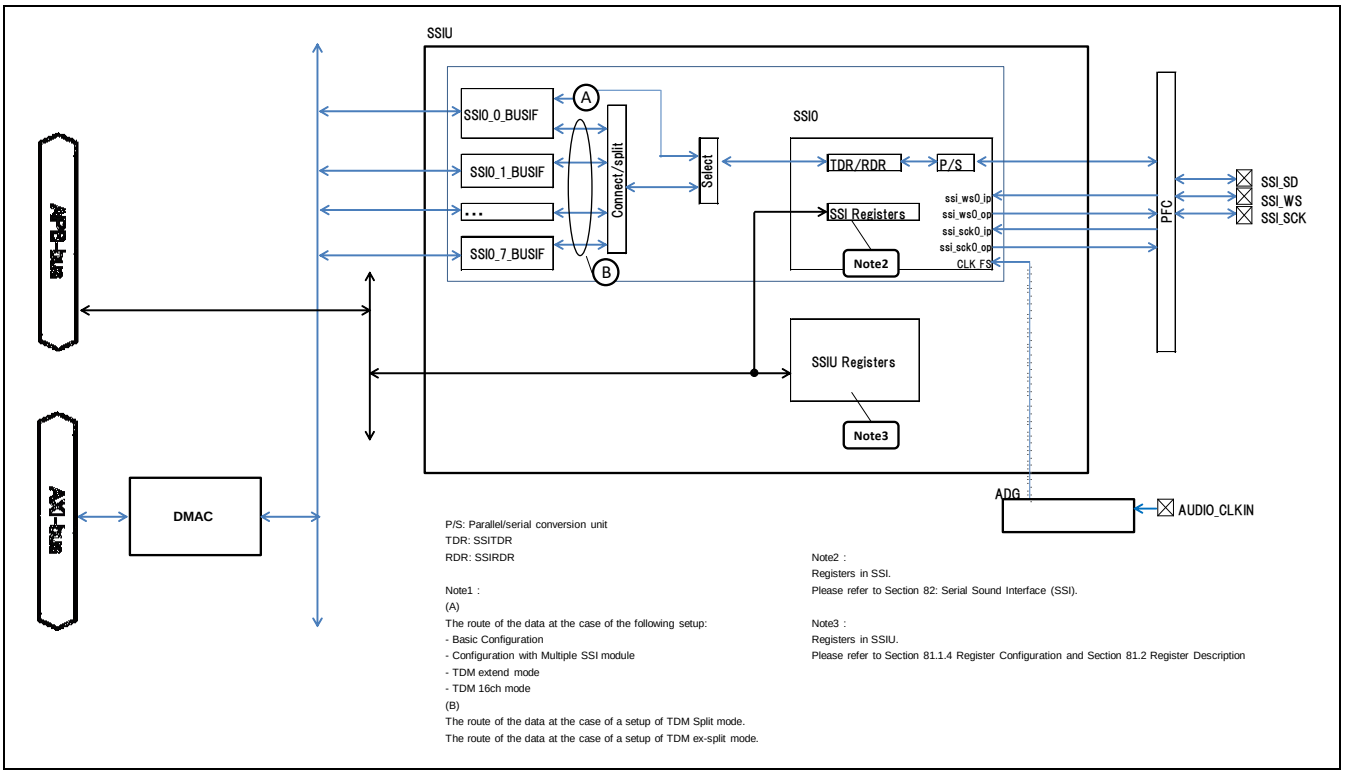



**Figure 81.1 Block Diagram of SSIU**

### 81.1.3 External Pins

Table 81.1 shows the pin configuration.

**Table 81.1 Pin Configuration**

| Name | Pin Name | I/O | Function |
|---|---|---|---|
| Serial clock pin | SSI_SCK | I/O | Serial clock |
| Word select pin | SSI_WS | I/O | Word select |
| Serial data pin | SSI_SD | I/O | Serial data |

### 81.1.4 Register Configuration

Refer to R-CarV4M_UM_081_SSIU_r0p81.xlsx

## 81.2 Register Description

Refer to R-CarV4M_UM_081_SSIU_r0p81.xlsx

### 81.2.1 Tables and Figures

**Table 81.2 CHNUM Setting**

| SSIm_x (m = 0) | Format: Basic Configuration (ex_func = 0, tdm_ext = 0, tdm_split = 0) | Format: TDM Extend Mode (ex_func = 0, tdm_ext = 1, tdm_split = 0) | Format: TDM Split Mode (ex_func = 0, tdm_ext = 0, tdm_split = 1) | Format: TDM 16Ch Mode (ex_func = 1, tdm_ext = 0, tdm_split = 0) | Format: TDM Ex-Split Mode (ex_func = 1, tdm_ext = 0, tdm_split = 1) |
|---|---|---|---|---|---|
| x = 0 | 00000: 0 (none)<br>00001: 1 channel<br>00010: 2 channels<br>00011: Reserved<br>00100: 4 channels<br>00101: Reserved<br>00110: 6 channels<br>00111: Reserved<br>01000: 8 channels<br>01001 to 11111: Reserved | 00000: 0 (none)<br>00001 to 00101: Reserved<br>00110: 6 channels<br>00111: Reserved<br>01000: 8 channels<br>01001 to 11111: Reserved | 00000: 0 (none)<br>00001: 1 channel<br>00010: 2 channels<br>00011 to 11111: Reserved | 00000: 0 (none)<br>00001 to 01111: Reserved<br>10000: 16 channels<br>10001 to 11111: Reserved | 00000: 0 (none)<br>00001: Reserved<br>00010: 2 channels<br>00011: Reserved<br>00100: 4 channels<br>00101: Reserved<br>00110: 6 channels<br>00111: Reserved<br>01000: 8 channels<br>01001: Reserved<br>01010: 10 channels<br>01011 to 11111: Reserved<br>*2<br>*3 |
| x = 1 | *1 | *1 | 00000: 0 (none)<br>00001: 1 channel<br>00010: 2 channels<br>00011 to 11111: Reserved | *1 | 00000: 0 (none)<br>00001: Reserved<br>00010: 2 channels<br>00011 to 11111: Reserved<br>*3 |
| x = 2 | *1 | *1 | 00000: 0 (none)<br>00001: 1 channel<br>00010: 2 channels<br>00011 to 11111: Reserved | *1 | 00000: 0 (none)<br>00001: Reserved<br>00010: 2 channels<br>00011: Reserved<br>00100: 4 channels<br>00101 to 11111: Reserved<br>*3 |
| x = 3 | *1 | *1 | 00000: 0 (none)<br>00001: 1 channel<br>00010: 2 channels<br>00011 to 11111: Reserved | *1 | 00000: 0 (none)<br>00001: Reserved<br>00010: 2 channels<br>00011 to 11111: Reserved<br>*3 |
| x = 4 | *1 | *1 | *1 | *1 | 00000: 0 (none)<br>00001: Reserved<br>00010: 2 channels<br>00011: Reserved<br>00100: 4 channels<br>00101: Reserved |

| SSIm_x (m = 0) | Format | | | | |
|---|---|---|---|---|---|
| | **Basic Configuration (ex_func = 0, tdm_ext = 0, tdm_split = 0)** | **TDM Extend Mode (ex_func = 0, tdm_ext = 1, tdm_split = 0)** | **TDM Split Mode (ex_func = 0, tdm_ext = 0, tdm_split = 1)** | **TDM 16Ch Mode (ex_func = 1, tdm_ext = 0, tdm_split = 0)** | **TDM Ex-Split Mode (ex_func = 1, tdm_ext = 0, tdm_split = 1)** |
| | | | | | 00110: 6 channels<br>00111: Reserved<br>01000: 8 channels<br>01001 to 11111: Reserved<br>*3<br>*4 |
| x = 5 | *1 | *1 | *1 | *1 | 00000: 0 (none)<br>00001: Reserved<br>00010: 2 channels<br>00011 to 11111: Reserved<br>*4 |
| x = 6 | *1 | *1 | *1 | *1 | 00000: 0 (none)<br>00001: Reserved<br>00010: 2 channels<br>00011: Reserved<br>00100: 4 channels<br>00101 to 11111: Reserved<br>*4 |
| x = 7 | *1 | *1 | *1 | *1 | 00000: 0 (none)<br>00001: Reserved<br>00010: 2 channels<br>00011 to 11111: Reserved<br>*4 |

Notes: 1. The write value should always be 0.
2. When use TDM-8ch, B'00111 to B'11111 is reserved.
3. Setting to share place in the stream data with multiple BUSIF channels is prohibited. For example, when x = 0 is 8channel, x = 1 to 3 cannot be used and write value should always be 0.
4. When use TDM-8ch, the write value should always be 0.

## 81.3 Operation

### 81.3.1 Module Specifications

Table 81.3 show the correspondences between the modules and functions.

**Table 81.3 Function Correspondences**

| | | | Format | | |
|---|---|---|---|---|---|
| | | **External Interface** | **Basic Configuration** | **TDM Extend Mode** | **TDM Split Mode/ TDM 16Ch Mode/ TDM Ex-Split Mode** |
| 1 | SSI0 | I2S | Supported | Not supported | Not supported |
| 2 | SSI0 | TDM | Supported | Supported | Supported |

### 81.3.2 Basic Configuration

Data transfer is performed between the SSIU and other audio modules or external memories via BUSIF connected to the audio local bus and the audio DMA bus.

Figure 81.2 shows the data transfer of SSI0. As shown in Figure 81.2, SSI0_1_BUSIF to SSI0_7_BUSIF are not used.

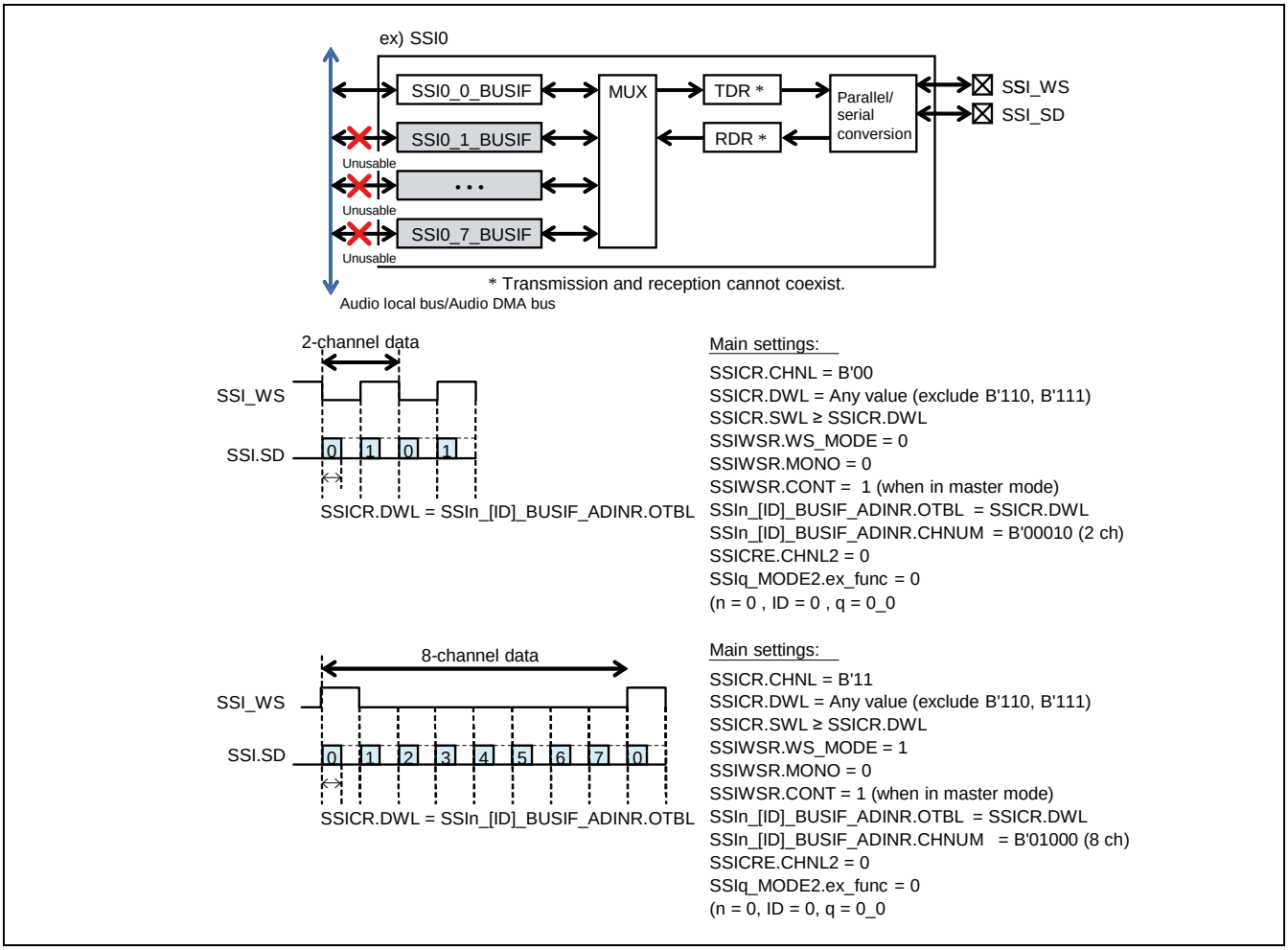



**Figure 81.2 Basic Configuration (SSI0)**

### 81.3.3 TDM Format Extension Function (TDM Extend Mode)

The TDM format extension function allows operation with 8-channel data in the serial bus and 6-channel data inside the SoC, or operation with 6-channel data in the serial bus and 8-channel data inside the SoC.

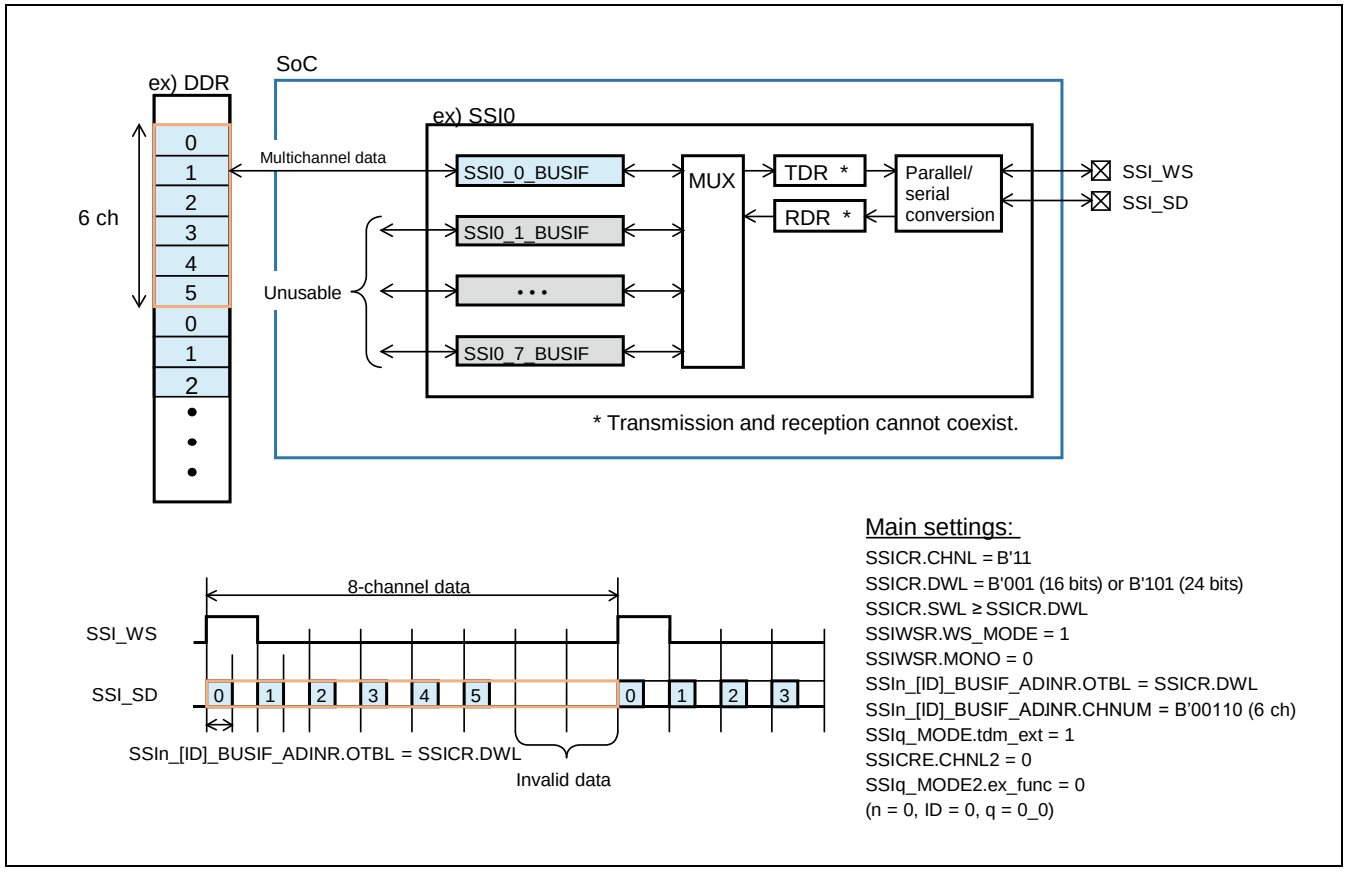



**Figure 81.3 TDM Extend Mode Operation with 8-channel Data in Serial Bus and 6-channel Data Inside the SoC (SSI0)**

Note: When the above configuration is used, SSI0_1_BUSIF to and SSI0_7_BUSIF cannot be used.

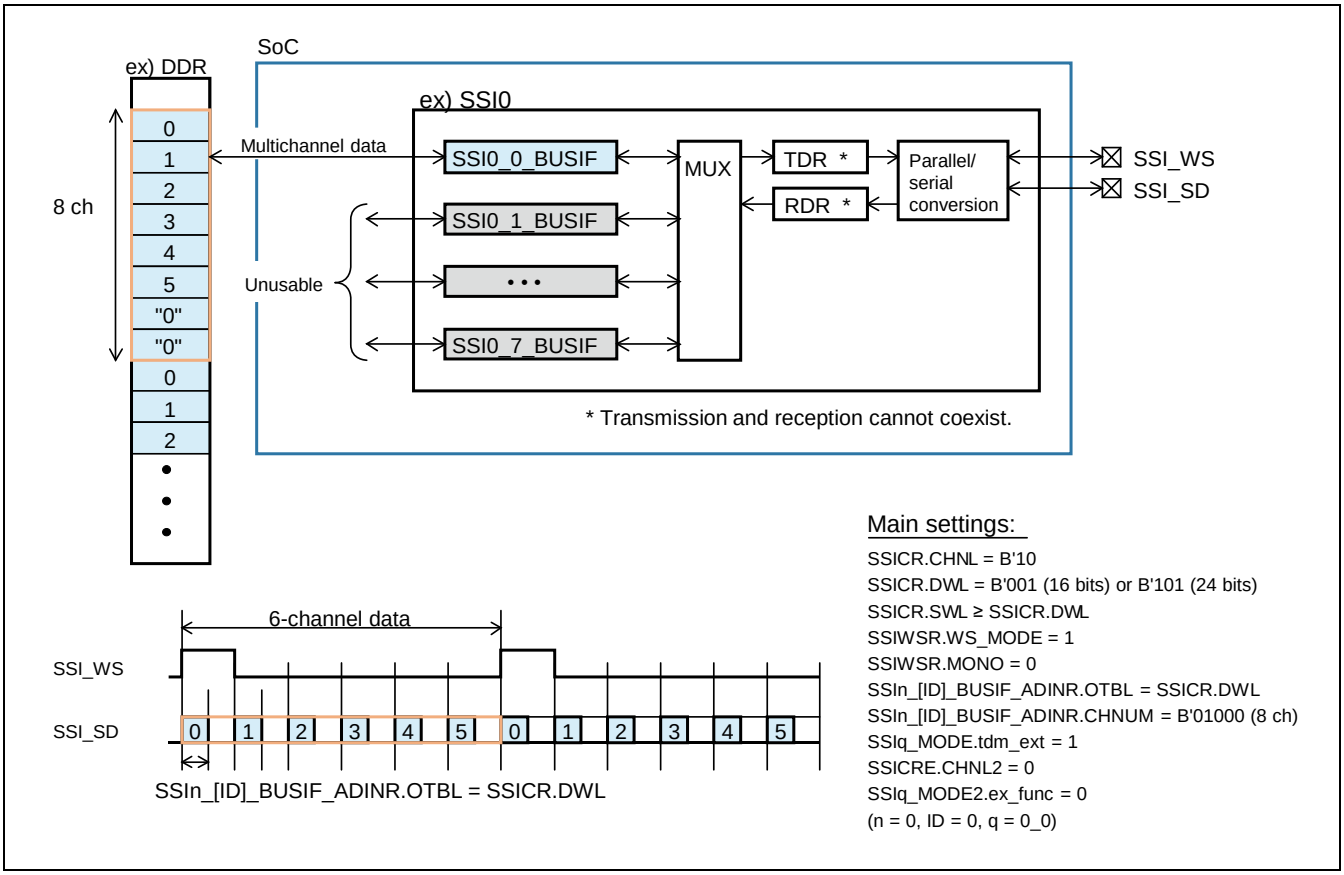



**Figure 81.4 TDM Extend Mode Operation with 6-channel Data in Serial Bus and 8-channel Data Inside the SoC (SSI0)**

Note: When the above configuration is used, SSI0_1_BUSIF to SSI0_7_BUSIF cannot be used.

### 81.3.4 TDM Format Split Function (TDM Split Mode)

The TDM format split function (TDM split mode) allows connecting four independent monaural or stereo data inside the SoC and outputting the data in the TDM format, and splitting TDM format input data into four independent monaural or stereo data and distributing the data inside the SoC. This function allows TDM-8ch or TDM-4ch.

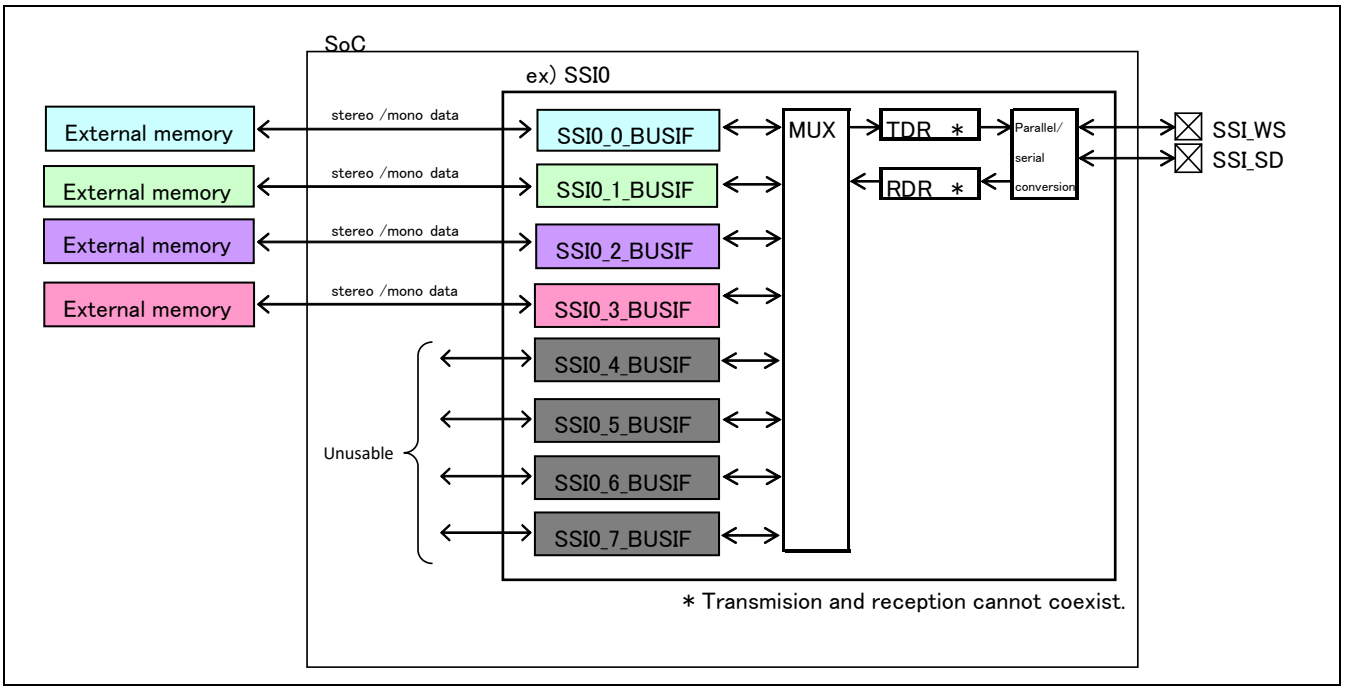



**Figure 81.5 TDM Split Mode**

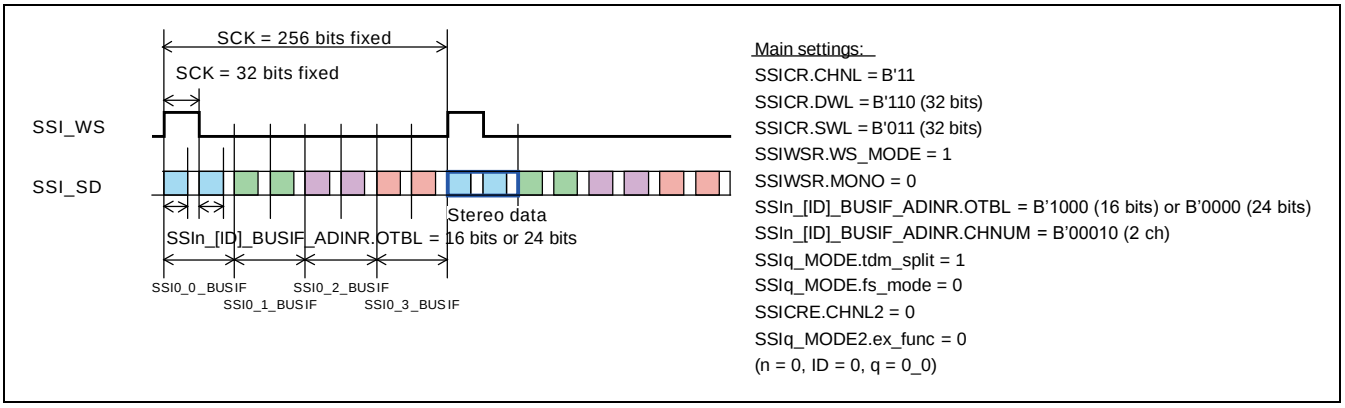
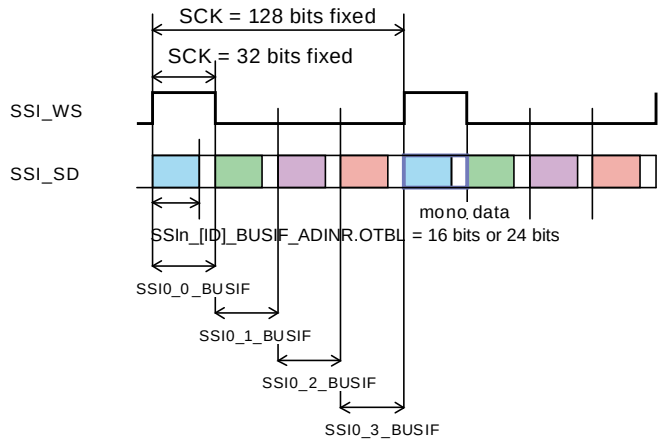



**Figure 81.6 TDM Split Mode (Stereo × 4)**




Main settings:

SSICR.CHNL = B'01
SSICR.DWL = B'110 (32 bits)
SSICR.SWL = B'011 (32 bits)
SSIWSR.WS_MODE = 1
SSIWSR.MONO = 0
SSIn_[ID]_BUSIF_ADINR.OTBL = B'1000 (16 bits) or B'0000 (24 bits)
SSIn_[ID]_BUSIF_ADINR.CHNUM = B'00001 (1 ch)
SSIq_MODE.tdm_split = 1
SSIq_MODE.fs_mode = 1
SSICRE.CHNL2 = 0
SSIq_MODE2.ex_func = 0
(n = 0, ID = 0-3, q = 0_0)

**Figure 81.7 TDM Split Mode (Monaural × 4)**

Notes: 1. The SWL and DWL bits in SSICR should be fixed at 32-bit setting.
2. Transmission and reception cannot coexist.
3. Data input to SSIn_[ID]_BUSIF (n = 0, ID = 0-3) should be operated synchronously with the SSI sampling frequency (WS signal cycle).
4. Setting of SSIn_[ID]_BUSIF_ADINR.OTBL(n = 0, ID = 0-3) = B'1000 (16bit) and SSIn_[ID]_BUSIF_ADINR.OTBL(n = 0, ID = 0-3) = B'0000 (24bit) cannot coexist within the same SSI.

### 81.3.5 TDM Format 16ch Function (TDM 16ch Mode)

The TDM format 16ch function (TDM 16ch mode) is similar to basic function. It is correspondence to TDM-16channel.

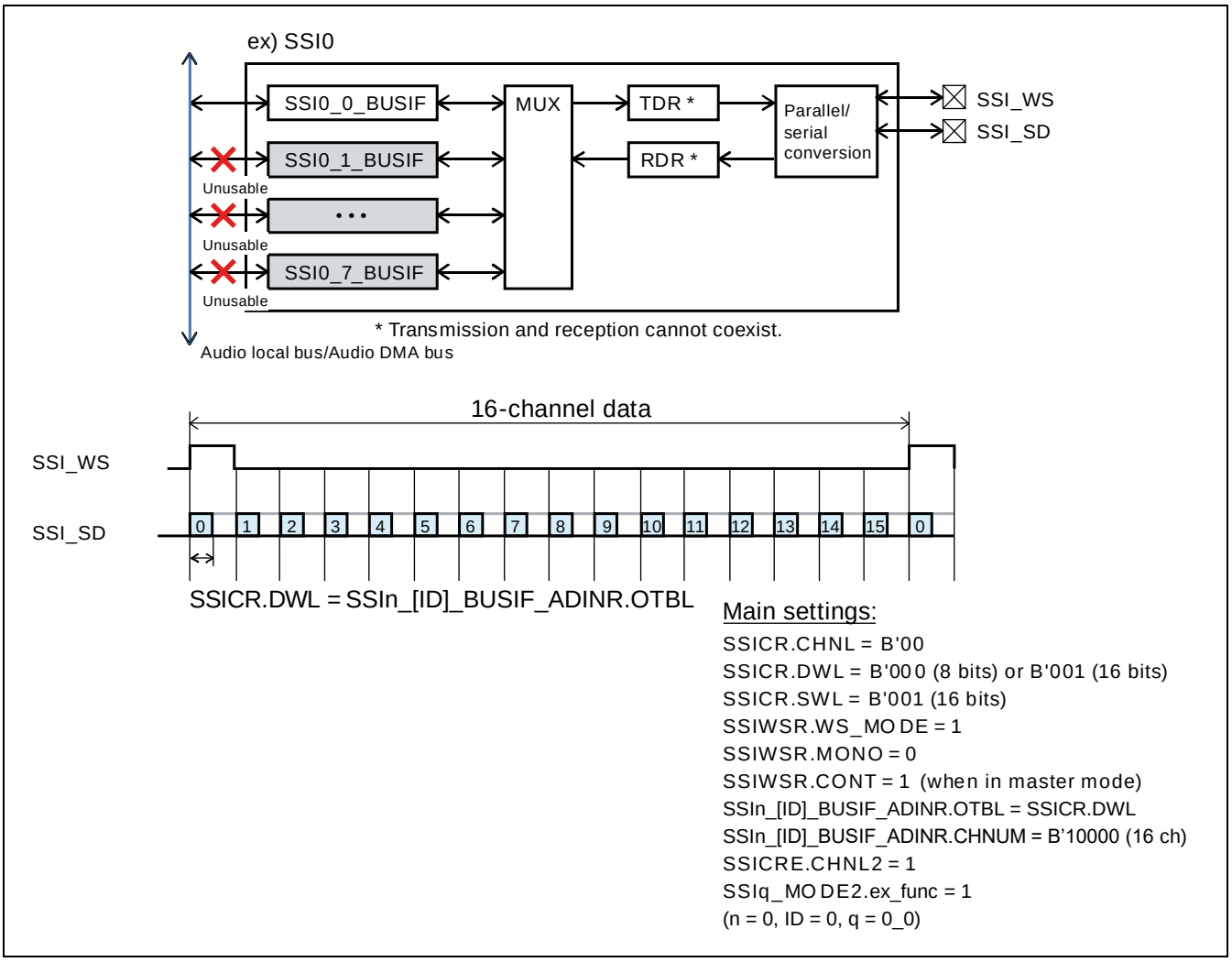



**Figure 81.8 TDM 16ch Mode**

### 81.3.6 TDM Format Extend Split Function (TDM Ex-Split Mode)

The TDM format extend split function (TDM ex-split mode) allows connecting eight independent stereo or multi data inside the SoC and outputting the data in the TDM format, and splitting TDM format input data into eight independent stereo or multi data and distributing the data inside the SoC. This function allows TDM-16ch or TDM-8ch (TDM-6ch, 4ch are not supported). Setting to share place in the stream data with multiple BUSIF channels is prohibited. For example, when SSI0_0_BUSIF used as 6channel, SSI0_1_BUSIF and SSI0_2_BUSIF cannot be used.

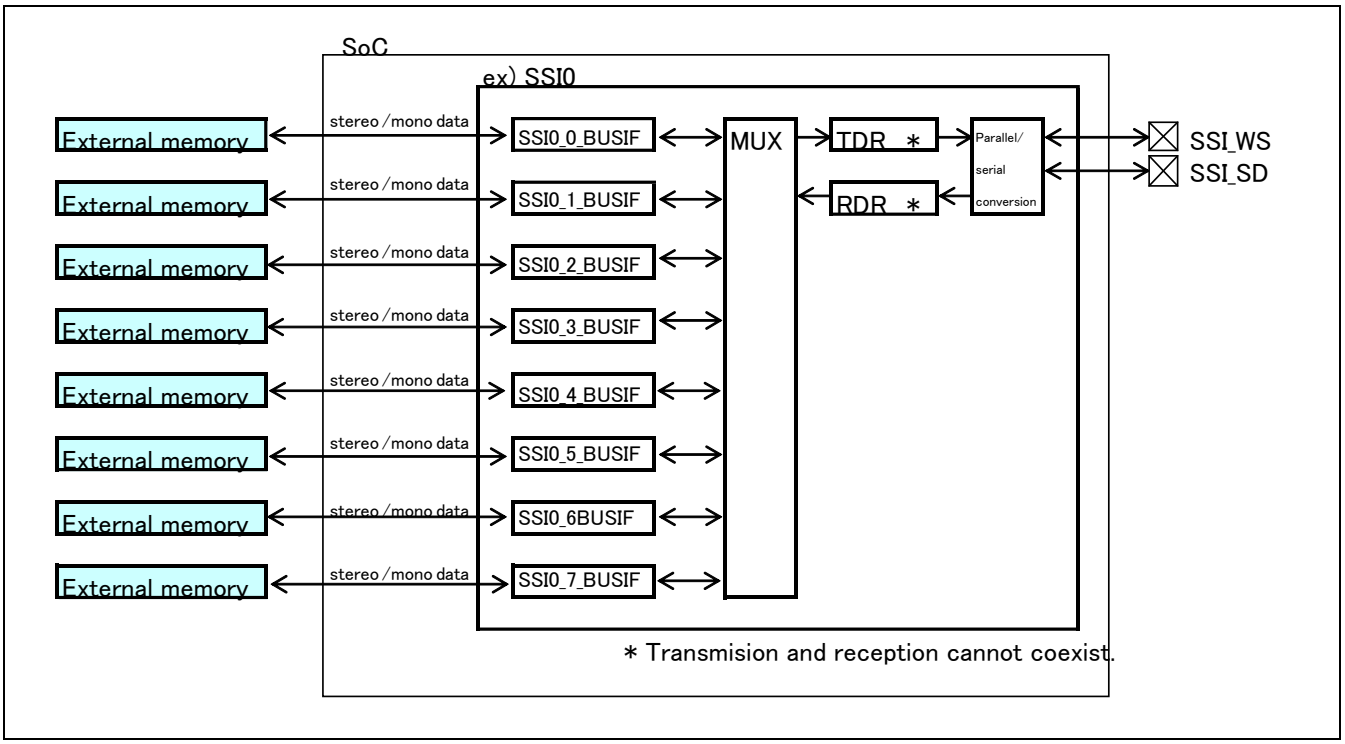
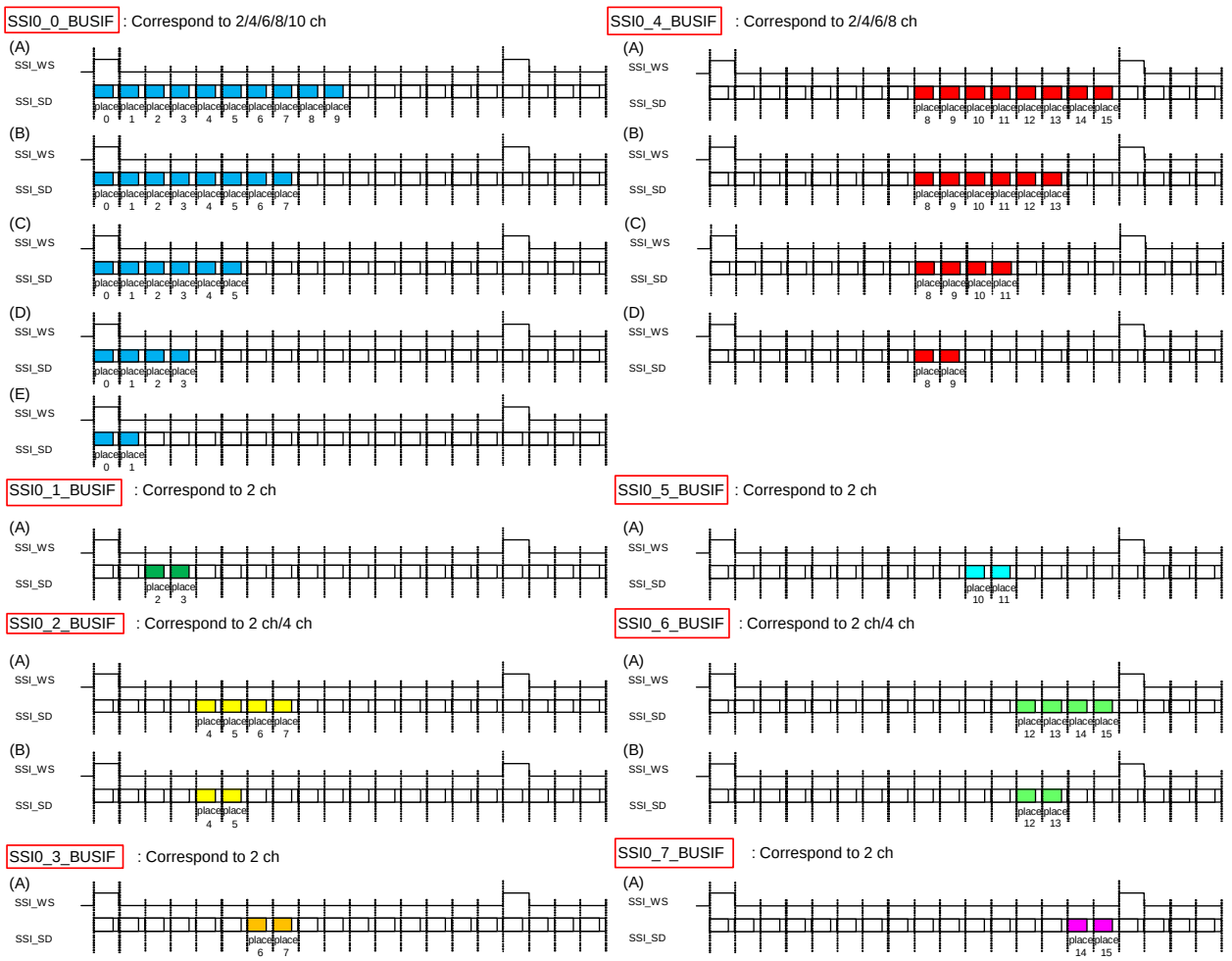



**Figure 81.9 TDM Ex-Split Mode**

TDM 16 ch ex-split mode (sound I/F format)

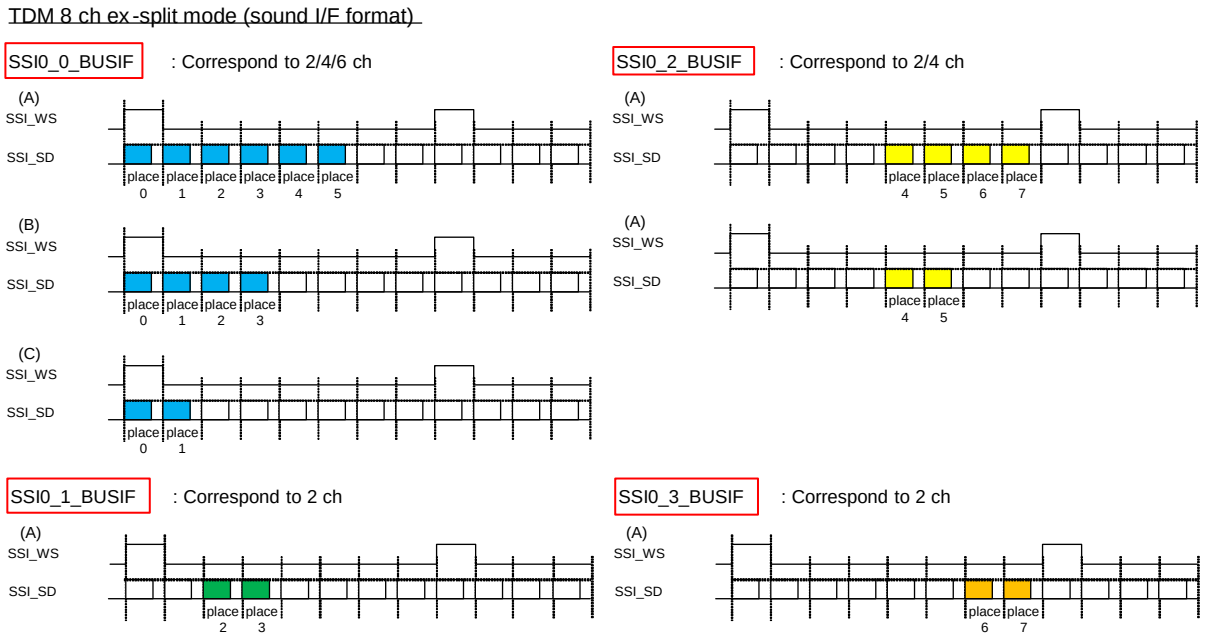



Main Setting

SSICR.CHNL = B'00
SSICR.DWL = B'000 (8-bit) or B'001 (16-bit)
SSICR.SWL = B'001 (16-bit)
SSIWSR.WS_MODE = 1
SSIWSR.MONO = 0
SSIn_[ID]_BUSIF_ADINR.OTB L= SSICR.DWL
SSIn_[ID]_BUSIF_ADINR.CHNUM = *
SSIq_MODE.tdm_split = 1
SSIq_MODE.fs_mode = 0
SSICRE.CHNL2 = 1
SSIq_MODE2.ex_func = 1 (n = 0, ID = 0-7, q = 0_0)

*: SSI0_0_BUSIF = B'00010/B'00100/B'00110/B'01000/B'01010
SSI0_1_BUSIF = B'00010
SSI0_2_BUSIF = B'00010/B'00100
SSI0_3_BUSIF = B'00010
SSI0_4_BUSIF = B'00010/B'00100/B'00110/B'01000
SSI0_5_BUSIF = B'00010
SSI0_6_BUSIF = B'00010/B'00100
SSI0_7_BUSIF = B'00010

**Figure 81.10 TDM 16ch ex-split mode (TDM 16ch ex-split)**




Main Setting
SSICR.CHNL = B'11
SSICR.DWL = B'000 (8-bit) or B'001 (16-bit) or B'101 (24-bit)
SSICR.SWL = B'001 (16-bit) or B'010 (24-bit) or B'011 (32-bit) (SWL?DWL)
SSIWSR.WS_MODE = 1
SSIWSR.MONO = 0
SSIn_[ID]_BUSIF_ADINR.OTB L= SSICR.DWL
SSIn_[ID]_BUSIF_ADINR.CHNUM = *
SSIq_MODE.tdm_split = 1
SSIq_MODE.fs_mode = 0
SSICRE.CHNL2 = 0
SSIq_MODE2.ex_func = 1 (n = 0, ID = 0-3, q = 0_0)
*: SSI0_0_BUSIF = B'00010/B'00100/B'00110
SSI0_1_BUSIF = B'00010
SSI0_2_BUSIF = B'00010/B'00100
SSI0_3_BUSIF = B'00010
SSI0_4_BUSIF to SSI0_7_BUSIF = B'00000 (Fix)

**Figure 81.11 TDM 16ch ex-split mode (TDM 8ch split)**

Notes: 1. Transmission and reception cannot coexist.
2. Data input to SSIn_[ID]_BUSIF (n = 0, ID = 0-7) should be operated synchronously with the SSI sampling frequency (WS signal cycle).
3. Another setting of SSIn_[ID]_BUSIF_ADINR.OTBL(n = 0, ID = 0-7) cannot coexist within the same SSI. (ex. Setting of SSIn_[ID]_BUSIF_ADINR.OTBL(n = 0, ID = 0-7) = B'1000 (16bit) and SSIn_[ID]_BUSIF_ADINR.OTBL(n = 0, ID = 0-7) = B'0000 (24bit) cannot coexist within the same SSI.)

### 81.3.7 Procedure of SSIU Transfer

Figure 81.12 to Figure 81.15 show the operation flow.

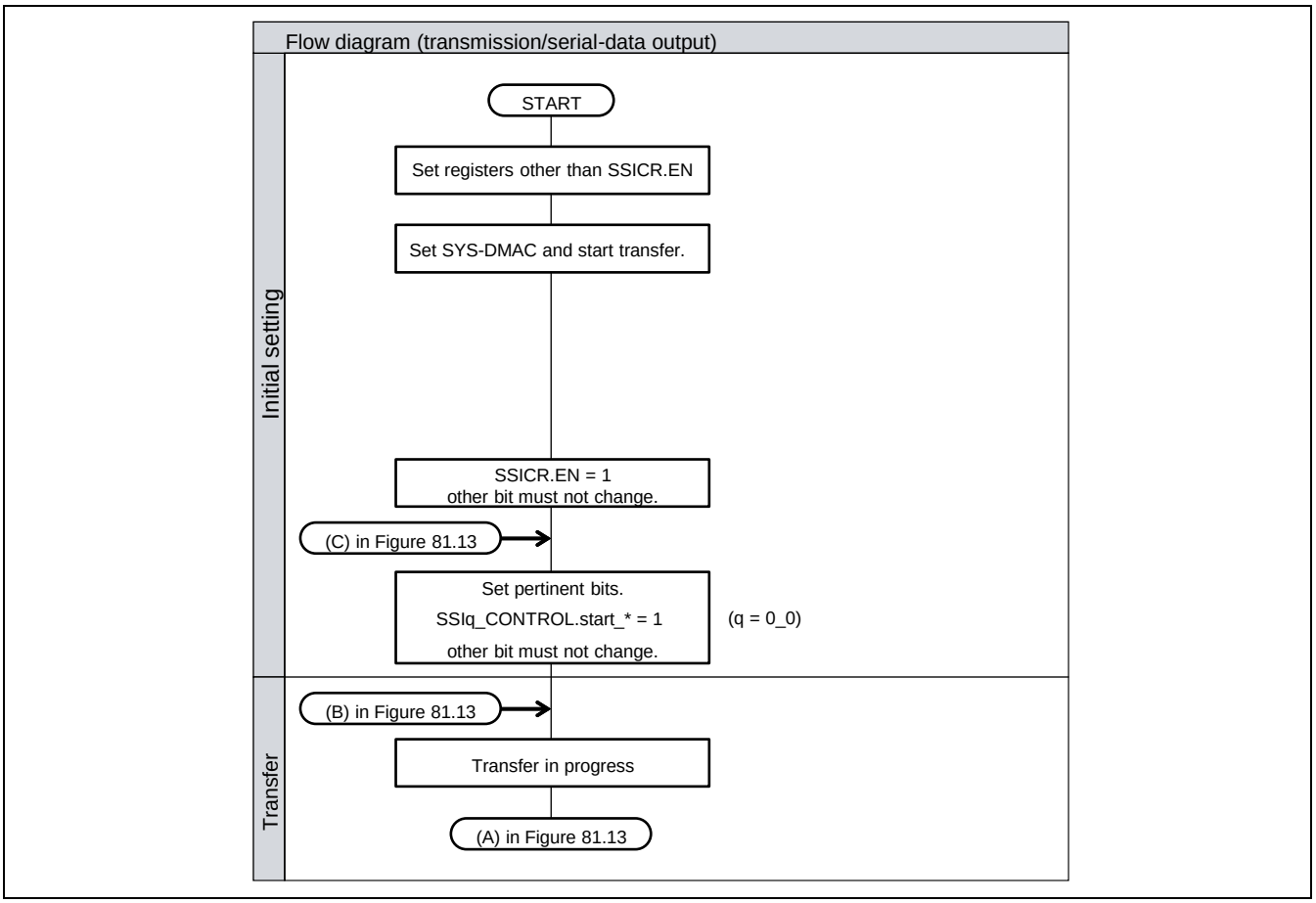



**Figure 81.12 Transmission (1)**

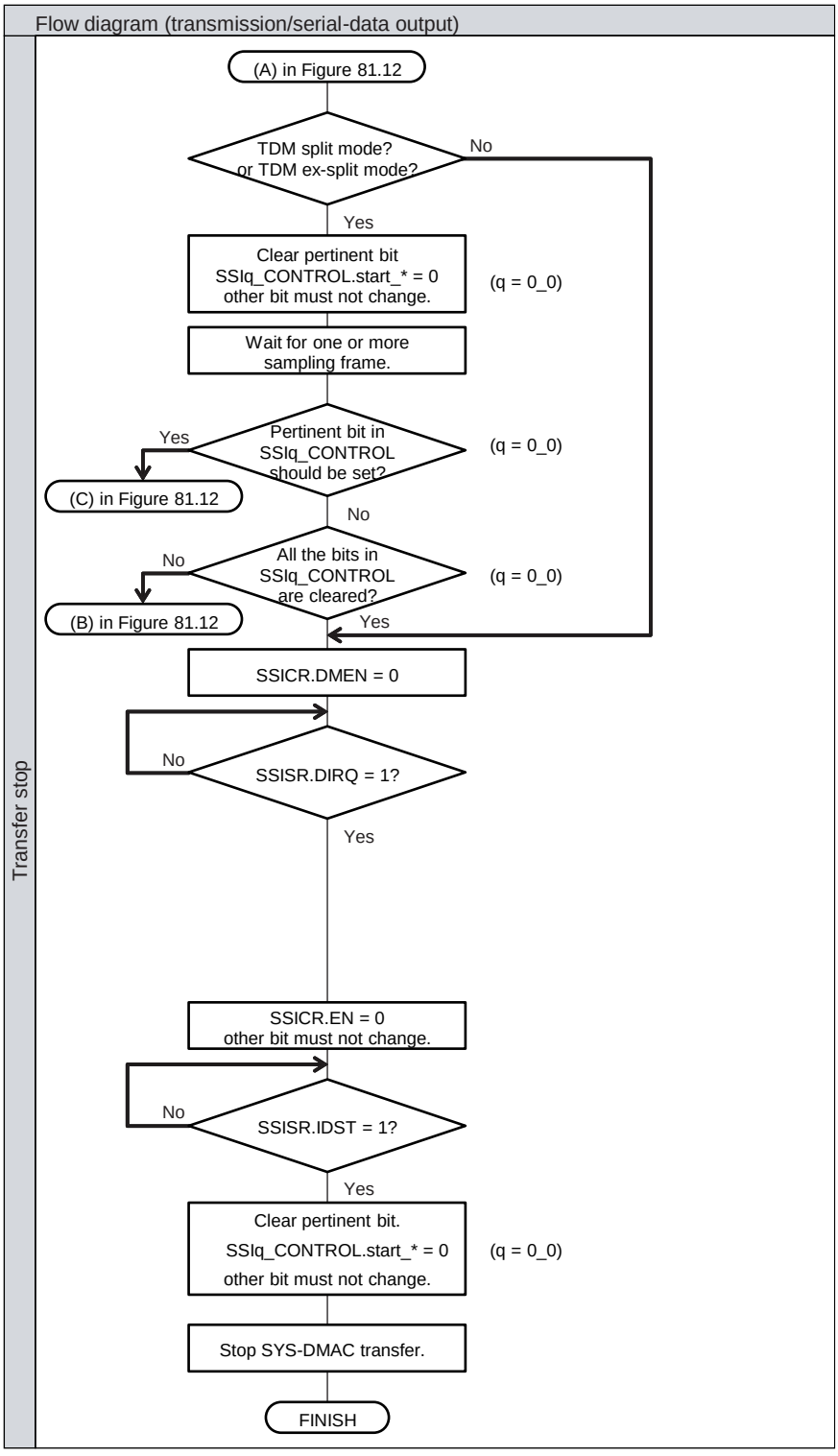



**Figure 81.13 Transmission (2)**

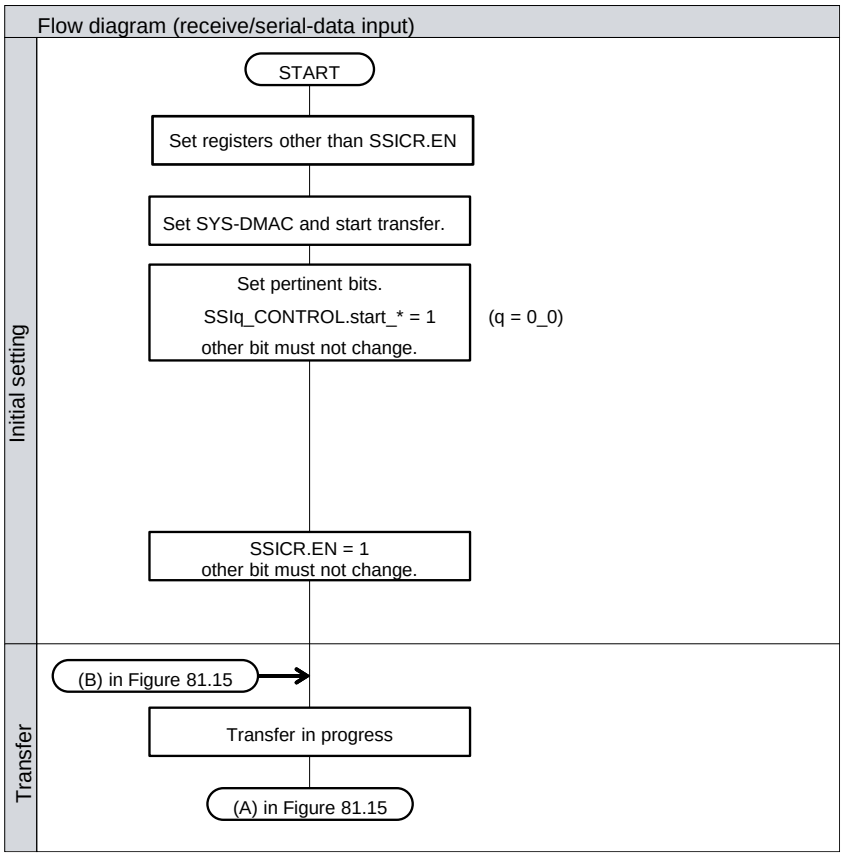



**Figure 81.14 Reception (1)**

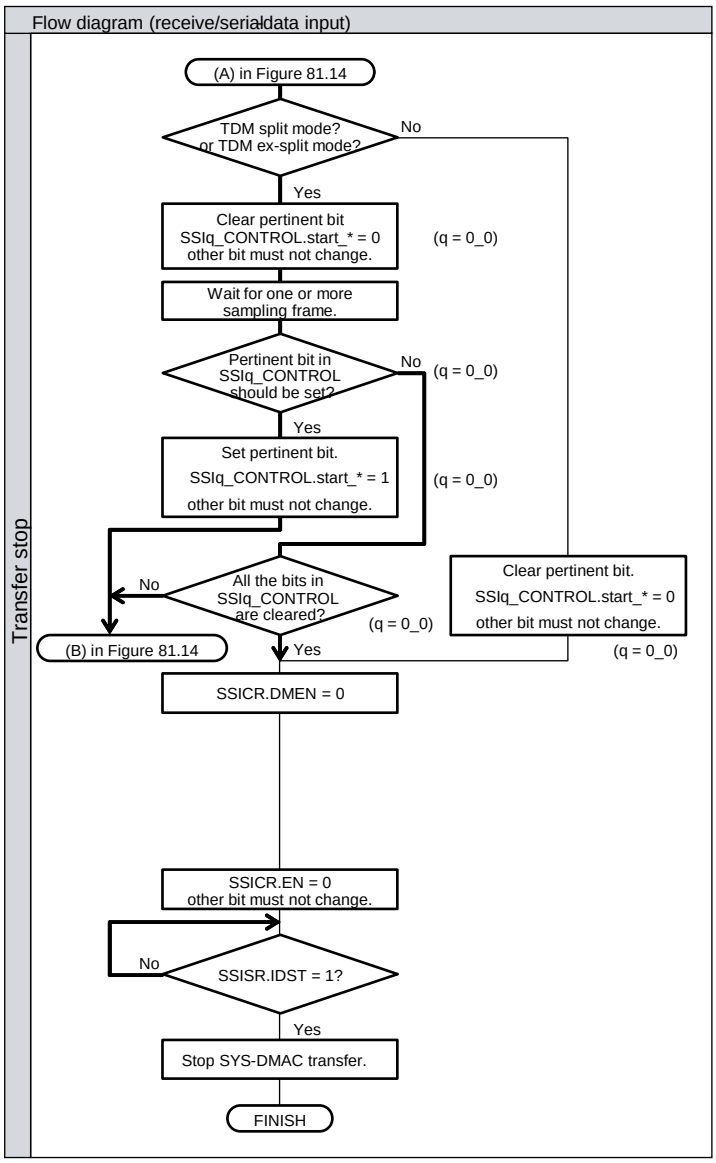



**Figure 81.15 Reception (2)**

### 81.3.8 Interrupts of SSI Setting

SSI can output interrupt signals when one of several events happen. Figure 81.16 illustrates how SSI generates interrupt signals and Table 81.4 and Table 81.5 indicate combinations of relevant registers for each SSI to generate interrupt signals (refer to explanations of relevant registers (Section 81.2) and SSI (Section 82))

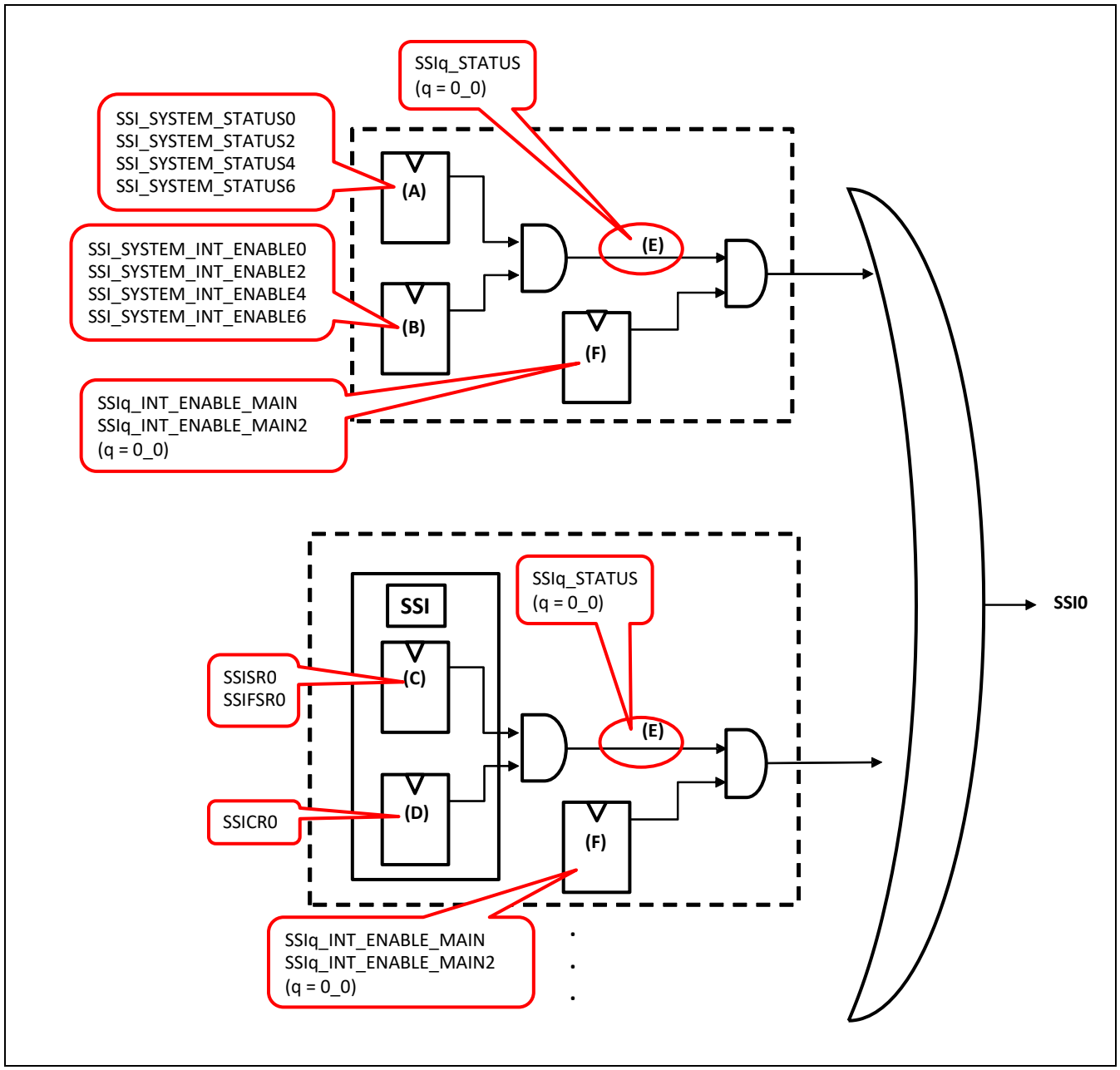



**Figure 81.16 Mechanism of generating interrupt signals**

**Table 81.4 Combinations of relevant registers for each SSI (1)**

| | | (A) | (B) | (E) | (F) |
|---|---|---|---|---|---|
| SSI | BUSIF | Register Name (Interrupt Factor) | Register Name (Interrupt Enable) | Register Name (Interrupt Factor) | Register Name (Interrupt Enable) |
| SSI0 | SSI0_0_BUSIF … SSI0_3_BUSIF | SSI_SYSTEM_STATUS0 | SSI_SYSTEM_INT_ENABLE0 | SSI0_0_STATUS | SSI0_0_INT_ENABLE_MAIN |
| | | SSI_SYSTEM_STATUS2 | SSI_SYSTEM_INT_ENABLE2 | SSI0_0_STATUS | SSI0_0_INT_ENABLE_MAIN |
| | SSI0_4_BUSIF … SSI0_7_BUSIF | SSI_SYSTEM_STATUS4 | SSI_SYSTEM_INT_ENABLE4 | SSI0_0_STATUS2 | SSI0_0_INT_ENABLE_MAIN2 |
| | | SSI_SYSTEM_STATUS6 | SSI_SYSTEM_INT_ENABLE6 | SSI0_0_STATUS2 | SSI0_0_INT_ENABLE_MAIN2 |

**Table 81.5 Combinations of relevant registers for each SSI (2)**

| | (C) | (D) | (E) | (F) |
|---|---|---|---|---|
| SSI | Register Name (Interrupt Factor) | Register Name (Interrupt Enable) | Register Name (Interrupt Factor) | Register Name (Interrupt Enable) |
| SSI0 | SSISR0 | SSICR0 | SSI0_0_STATUS | SSI0_0_INT_ENABLE_MAIN |
| | SSIFSR0 | SSICR0 | SSI0_0_STATUS | SSI0_0_INT_ENABLE_MAIN |

### 81.3.9 SYS-DMAC

The SYS-DMAC is used by data transfer between the external memory and SSIU module. The setting values of DMASAR and DMADAR of the SYS-DMAC are shown in Table 81.6.

**Table 81.6 Setting Values of DMASAR and DMADAR of SYS-DMAC**

| Name of Register | DMASAR/DMADAR | |
|---|---|---|
| SSI0_0_BUSIF | H'EC40 | 0000 |
| SSI0_1_BUSIF | | 8000 |
| SSI0_2_BUSIF | H'EC41 | 0000 |
| SSI0_3_BUSIF | | 8000 |
| SSI0_4_BUSIF | H'EC42 | 0000 |
| SSI0_5_BUSIF | | 8000 |
| SSI0_6_BUSIF | H'EC43 | 0000 |
| SSI0_7_BUSIF | | 8000 |

### 81.3.10 Rearranging the Order of Data

Rearranging the order of data is possible in the channel unit. Places of data in each data format are defined as in Table 81.7 to Table 81.10.

**Table 81.7 Definition of Data Places in Each Data Format (1)**

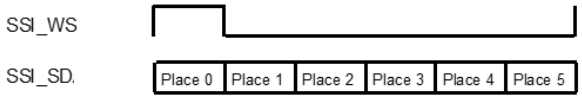
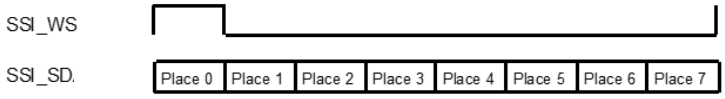
| | |
|---|---|
| SSI<br>· Stereo (2 channels) | |
| SSI<br>· TDM (6 channels) |  |
| SSI<br>· TDM (8 channels) |  |

**Table 81.8 Definition of Data Places in Each Data Format (2)**

| | |
|---|---|
| BUSIF<br>· Stereo (2 channels)<br>· 24 bits | External memory image<br>31 0<br>H'00 Place 0 *<br>H'04 Place 1 *<br>H'08 Place 0 *<br>H'0C Place 1 *<br>... ... * |
| BUSIF<br>· Stereo (2 channels)<br>· 16 bits | External memory image<br>31 0<br>H'00 Place 0 Place 1<br>H'04 Place 0 Place 1<br>H'08 Place 0 Place 1<br>H'0C Place 0 Place 1<br>... ... ... |
| BUSIF<br>· Stereo (2 channels)<br>· 8 bits | External memory image<br>31 0<br>H'00 Place 1 Place 0 Place 1 Place 0<br>H'04 Place 1 Place 0 Place 1 Place 0<br>... ... ... ... ... |
| BUSIF<br>· Monaural (1 channel)<br>· 8/16 bits | External memory image<br>31 0<br>H'00 Place 0 *<br>H'04 Place 0 *<br>H'08 Place 0 *<br>H'0C Place 0 *<br>... ... * |
| BUSIF<br>· Multichannel (4 channels)<br>· 24 bits | External memory image<br>31 0<br>H'00 Place 0 *<br>H'04 Place 1 *<br>H'08 Place 2 *<br>H'0C Place 3 *<br>H'10 Place 0 *<br>H'14 Place 1 *<br>H'18 Place 2 *<br>H'1C Place 3 *<br>... ... * |
| BUSIF<br>· Multichannel (4 channels)<br>· 16 bits | External memory image<br>31 0<br>H'00 Place 0 Place 1<br>H'04 Place 2 Place 3<br>H'08 Place 0 Place 1<br>H'0C Place 2 Place 3<br>... ... ... |

**Table 81.9 Definition of Data Places in Each Data Format (3)**

<table>
<tr>
<td>BUSIF<br>· Multichannel (6 channels)<br>· 24 bits</td>
<td>External memory image<br>
<table>
<tr><td></td><td>31</td><td>0</td></tr>
<tr><td>H'00</td><td>Place 0</td><td>*</td></tr>
<tr><td>H'04</td><td>Place 1</td><td>*</td></tr>
<tr><td>H'08</td><td>Place 2</td><td>*</td></tr>
<tr><td>H'0C</td><td>Place 3</td><td>*</td></tr>
<tr><td>H'10</td><td>Place 4</td><td>*</td></tr>
<tr><td>H'14</td><td>Place 5</td><td>*</td></tr>
<tr><td>H'18</td><td>Place 0</td><td>*</td></tr>
<tr><td>H'1C</td><td>Place 1</td><td>*</td></tr>
<tr><td>...</td><td>...</td><td>*</td></tr>
</table></td>
</tr>
<tr>
<td>BUSIF<br>· Multichannel (6 channels)<br>· 16 bits</td>
<td>External memory image<br>
<table>
<tr><td></td><td>31</td><td>0</td></tr>
<tr><td>H'00</td><td>Place 0</td><td>Place 1</td></tr>
<tr><td>H'04</td><td>Place 2</td><td>Place 3</td></tr>
<tr><td>H'08</td><td>Place 4</td><td>Place 5</td></tr>
<tr><td>H'0C</td><td>Place 0</td><td>Place 1</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td>
</tr>
<tr>
<td>BUSIF<br>· Multichannel (8 channels)<br>· 24 bits</td>
<td>External memory image<br>
<table>
<tr><td></td><td>31</td><td>0</td></tr>
<tr><td>H'00</td><td>Place 0</td><td>*</td></tr>
<tr><td>H'04</td><td>Place 1</td><td>*</td></tr>
<tr><td>H'08</td><td>Place 2</td><td>*</td></tr>
<tr><td>H'0C</td><td>Place 3</td><td>*</td></tr>
<tr><td>H'10</td><td>Place 4</td><td>*</td></tr>
<tr><td>H'14</td><td>Place 5</td><td>*</td></tr>
<tr><td>H'18</td><td>Place 6</td><td>*</td></tr>
<tr><td>H'1C</td><td>Place 7</td><td>*</td></tr>
<tr><td>H'20</td><td>Place 0</td><td>*</td></tr>
<tr><td>...</td><td>...</td><td>*</td></tr>
</table></td>
</tr>
<tr>
<td>BUSIF<br>· Multichannel (8 channels)<br>· 16 bits</td>
<td>External memory image<br>
<table>
<tr><td></td><td>31</td><td>0</td></tr>
<tr><td>H'00</td><td>Place 0</td><td>Place 1</td></tr>
<tr><td>H'04</td><td>Place 2</td><td>Place 3</td></tr>
<tr><td>H'08</td><td>Place 4</td><td>Place 5</td></tr>
<tr><td>H'0C</td><td>Place 6</td><td>Place 7</td></tr>
<tr><td>H'10</td><td>Place 0</td><td>Place 1</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td>
</tr>
</table>

**Table 81.10 Definition of Data Places in Each Data Format (4)**

| | |
|---|---|
| SSI<br>• TDM<br>(16 channels) | |
| BUSIF<br>• Multichannel<br>(16 channels)<br>• 16 bits | External memory image<br>31 0<br>H'00 Place0 Place1<br>H'04 Place2 Place3<br>H'08 Place4 Place5<br>H'0C Place6 Place7<br>H'10 Place8 Place9<br>H'14 Place10 Place11<br>H'18 Place12 Place13<br>H'1C Place14 Place15<br>H'20 Place0 Place1<br>••• ••• ••• |

### 81.3.11 Module Standby Function

For the register settings to make a transition to or from module standby mode, refer to section 9. Module Standby, Software Reset.

#### (1) Transition to Module Standby Mode

To make a transition to module standby mode by the module standby function, refer to the following transition procedure for each module.

**(a) SSIU/SSI**

1. Check the following registers' settings.
   [SSIU]
   — All bits in the SSIq control register (SSIq_CONTROL) (q = 0_0) are cleared to 0 (data transfer stopped).
   — All bits in the SSIq interrupt enable register (SSIq_INT_ENABLE_MAIN) (q = 0_0) are cleared to 0 (interrupt disabled).
   [SSI]
   — The DMEN and EN bits in the control register (SSICRn) (n = 0) are cleared to 0 (disabled).
   — The IDST bit in the status register (SSISRn) (n = 0) is set to 1.
   — The CONT bit in the WS mode register (SSIWSRn) (n = 0) is cleared to 0 (disabled).
2. Set the MSTP29.26 to MSTP29.27 bits in the module stop control register (MSTP29)

Note: The above bits should be set while the module operation has been completed and is placed in the idle state in which the module cannot be activated by external pins or other modules.

**(b) ADG**

1. Set the MSTP1.22 bit in the module stop control register (MSTP1) to 1.

Note: The above bit is for clock divider(brgclk).
The above bit should be set while the module cannot be activated by other modules

**(2) Releasing and Restarting from Module Standby Mode**

After the transition to module standby mode, modules can be released from module standby mode by a power-on reset or the appropriate procedure from the list below.

**(a) SSIU/SSI**

1. Supply the clock signal to SSIU/SSI modules.
2. Clear the MSTP29.26 to MSTP29.27 bits in the module stop control register (MSTP29).

**(b) ADG**

1. Supply the clock signal to ADG modules.
2. Clear the MSTP1.22 bit in the module stop control register (MSTP1).

## 81.4 Usage Notes

### 81.4.1 Note on Transfer

If an underflow or overflow occurs, stop the transfer and restart it.

Do not stop serial clock (SCK) and word select (WS) signals during data transfer. Confirm that SSISR.IDST = 1 before stopping SCK and WS signals.

In case of transmission at slave mode, it is necessary that WS signal is stable. Data transmission at slave mode should be started after more than two frames period since WS signal is entered and is stable.

In case of multiple SSI modules and case that one SSI works at master mode and the others do at slave mode, data transfer should be started after more than two frames period since registers of SSI at master mode are set.

## 81.5 Safety Mechanisms

This module has no Mechanism of FuSa.

# 82. Serial Sound Interface (SSI)

## 82.1 Overview

The serial sound interface (hereinafter referred to as the "SSI") is a transceiver module designed to send or receive audio data interfacing with a variety of devices offering I2S format. It also supports multi-channel mode in addition to other common formats.

### 82.1.1 Features

The SSI has the following features:

- Number of channels: Maximum of four (when a multichannel format is specified).
- The SSI module can serve as both a transmitter and a receiver.
- Asynchronous transfer takes place between the data buffer and the shift register.
- Only the MSB first data alignment is supported.
- A value as the dividing ratio for the clock used by the serial bus interface is selectable.
- Controlling of data transmission or reception with DMAC or interrupt requests is possible.
  - SSI does not work on its own. Please use via SSIU.
- TDM format is supported.
- The frequency range of SCK signal is from 297.3 kHz to 12.5MHz at master mode, and from 297.3 kHz to 15.1 MHz at slave mode.
- The WS continue function by which operation can be performed without stopping WS signal is supported.
- Monaural mode (8 bits or 16 bits) is supported.
- In monaural mode, WS signal pulse width can be changed (short or long frame).
- If the sampling clock frequency is switched during transfer, it will be notified of the CPU with interrupts (a function to detect switching frequency).
- Operating mode: non-compressed mode (Not support compressed mode).
- Supports versatile serial audio formats (I2S/left justified/right justified).
- Supports master/slave functions.
- Programmable word clock, bit clock generation functions.

### 82.1.2 Block Diagram

Figure 82.1 shows a block diagram of a single SSI module.

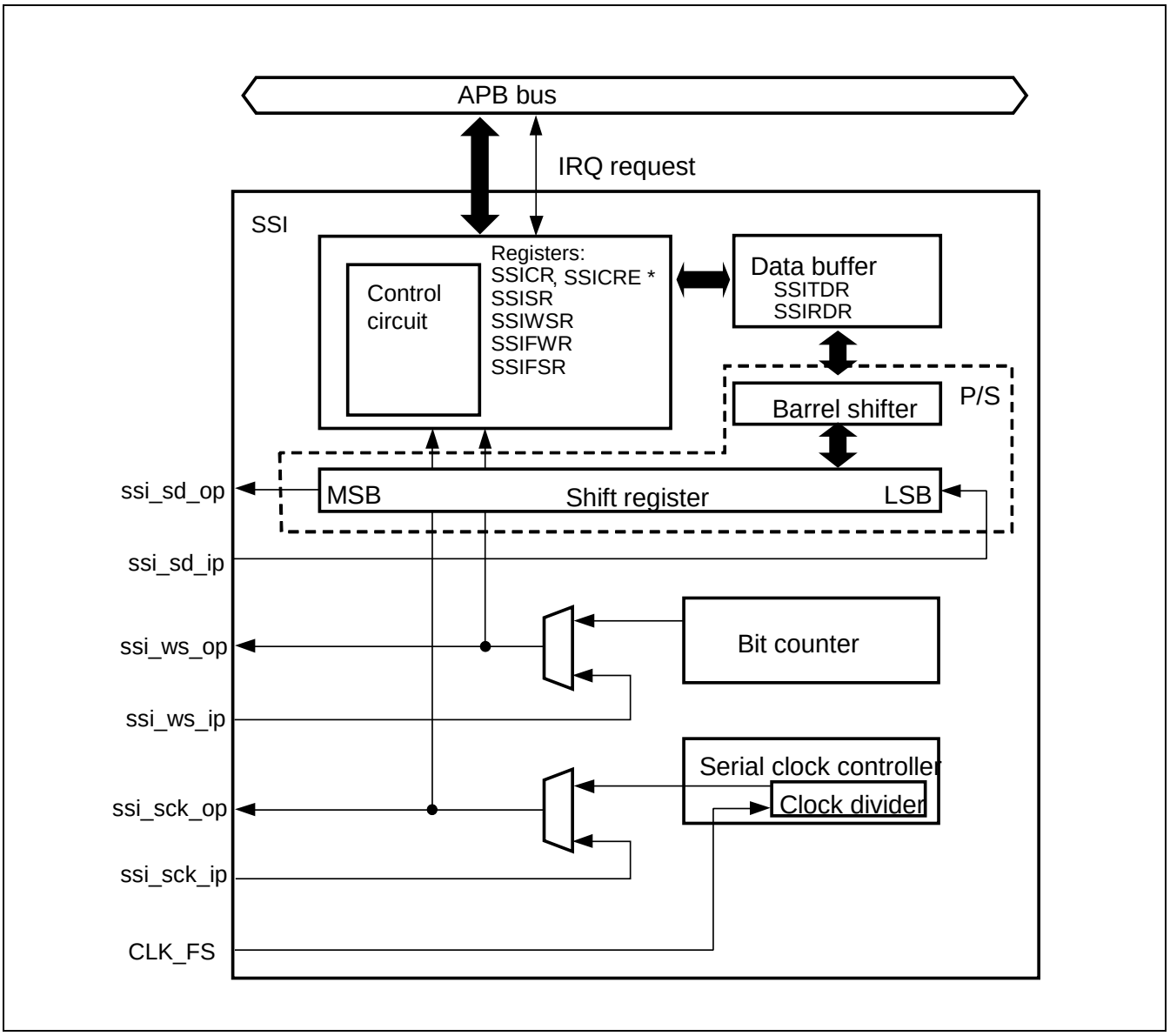



**Figure 82.1 Block Diagram of SSI**

### 82.1.3 Register Configuration

The SSI has the following registers. Note that the module numbers are basically omitted from the register names in the text.

**Table 82.1 Register Configurations**

| Module | Register Name | Abbreviation | R/W | Address | Access Size |
|---|---|---|---|---|---|
| 0 | Control Register 0 | SSICR0 | R/W | H'EC54 1000 | 32 |
| | Status Register 0 | SSISR0 | R/W*1 | H'EC54 1004 | 32 |
| | Transmit Data Register 0 | SSITDR0 | R/W | H'EC54 1008/ | 32 |
| | Receive Data Register 0 | SSIRDR0 | R | H'EC54 100C | 32 |
| | WS Mode Register 0 | SSIWSR0 | R/W | H'EC54 1020 | 32 |
| | FS Mode Register 0 | SSIFMR0 | R/W | H'EC54 1024 | 32 |
| | FS Status Register 0 | SSIFSR0 | R/W*2 | H'EC54 1028 | 32 |
| | Control register extend 0 | SSICRE0 | R/W | H'EC54 1030 | 32 |

Notes: Do not write to any addresses other than listed above. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed above are undefined.

1. For this register, bits 26 and 27 are readable/writable bits, although the others are read-only bits. For details, refer to Section 82.2.2 Status Register (SSISRn).
2. For this register, bits 14 and 15 are readable/writable bits, although the others are read-only bits. For details, refer to Section 82.2.7 FS Status Register (SSIFSRn).

## 82.2 Register Description

[Legend for Register Descriptions]

Initial value: Register value after a reset

—: Undefined value

R/W: Readable/writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

R: Read-only. The write value should always be 0.

### 82.2.1 Control Register (SSICRn)

Note: n = 0

SSICR is a readable/writable 32-bit register that controls the IRQ, selects the polarity status, and sets operating mode.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FORCE | — | FIEN | DMEN | UIEN | OIEN | IIEN | DIEN | CHNL[1:0] | | DWL[2:0] | | | SWL[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SCKD | SWSD | SCKP | SWSP | SPDP | SDTA | PDTA | DEL | — | CKDV[2:0] | | | MUEN | — | TRMD | EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | FORCE | 0 | R/W | Fixed<br>The bit should always be set to 1. |
| 30 | — | 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 29 | FIEN | 0 | R/W | Frequency Switching Detection Interrupt Enable<br>0: Frequency switching detection interrupt is disabled.<br>1: Frequency switching detection interrupt is enabled. |
| 28 | DMEN | 0 | R/W | DMA Enable<br>Enables or disables the DMA request.<br>0: DMA request is disabled.<br>1: DMA request is enabled. |
| 27 | UIEN | 0 | R/W | Underflow Interrupt Enable<br>0: Underflow interrupt is disabled.<br>1: Underflow Interrupt is enabled. |
| 26 | OIEN | 0 | R/W | Overflow Interrupt Enable<br>0: Overflow interrupt is disabled.<br>1: Overflow interrupt is enabled. |
| 25 | IIEN | 0 | R/W | Idle Mode Interrupt Enable<br>0: Idle mode interrupt is disabled.<br>1: Idle mode interrupt is enabled. |
| 24 | DIEN | 0 | R/W | Data Interrupt Enable<br>0: Data interrupt is disabled.<br>1: Data interrupt is enabled. |
| 23, 22 | CHNL[1:0] | 00 | R/W | Channels<br>These bits set the number of channels in each system word.<br>When the stereo format or multi-channel format is used (WS_MODE = 0, MONO = 0, and WIDTH = B'00000 in the WS Mode Register):<br>00: A system word has one channel.<br>01: A system word has two channels.<br>10: A system word has three channels.<br>11: A system word has four channels.<br>When the TDM format is used (WS_MODE = 1, MONO = 0, and WIDTH = B'00000 in the WS Mode Register):<br>00: Setting prohibited *<br>01: A TDM frame consists of four system words.<br>10: A TDM frame consists of six system words.<br>11: A TDM frame consists of eight system words.<br>* In case of CHNL = B'00 and SSICREm.CHNL2 = 1, a TDM frame consists of sixteen system words.<br>When the monaural format is used (WS_MODE = 1, MONO = 1, and WIDTH = B'00001 to B'11111 in the WS Mode Register):<br>00: A monaural frame consists of one system word.<br>01: Setting prohibited<br>10: Setting prohibited<br>11: Setting prohibited |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 21 to 19 | DWL[2:0] | 000 | R/W | Data Word Length<br>These bits set the number of bits in a data word.<br>When the stereo format or multi-channel format (WS_MODE = 0, MONO = 0, and WIDTH = B'00000 in the WS Mode Register) or TDM format (WS_MODE = 1, MONO = 0, and WIDTH = B'00000 in the WS Mode Register) is used:<br>000: 8 bits<br>001: 16 bits<br>010: 18 bits<br>011: 20 bits<br>100: 22 bits<br>101: 24 bits<br>110: 32 bits<br>111: Setting prohibited<br>When the monaural format is used (WS_MODE = 1, MONO = 1, and WIDTH = B'00001 to B'11111 in the WS Mode Register):<br>000: 8 bits<br>001: 16 bits<br>Other than above: Setting prohibited |
| 18 to 16 | SWL[2:0] | 000 | R/W | System Word Length<br>These bits set the number of bits in a system word.<br>When the stereo format or multi-channel format (WS_MODE = 0, MONO = 0, and WIDTH = B'00000 in the WS Mode Register) or TDM format (WS_MODE = 1, MONO = 0, and WIDTH = B'00000 in the WS Mode Register) is used:<br>000: 8 bits<br>001: 16 bits<br>010: 24 bits<br>011: 32 bits<br>100: 48 bits<br>101: 64 bits<br>110: 128 bits<br>111: 256 bits<br>When the monaural format is used (WS_MODE = 1, MONO = 1, and WIDTH = B'00001 to B'11111 in the WS Mode Register):<br>000: 16 bits<br>001: 32 bits<br>010: 48 bits<br>011: 64 bits<br>100: 96 bits<br>101: 128 bits<br>110: 256 bits<br>111: 512 bits |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | SCKD | 0 | R/W | Serial Bit Clock Direction<br>0: Serial bit clock is input, slave mode.<br>1: Serial bit clock is output, master mode.<br>Note: (SCKD, SWSD) = (0, 0) or (1, 1) can be set. Other settings are prohibited. |
| 14 | SWSD | 0 | R/W | Serial WS Direction<br>0: Serial word select is input, slave mode.<br>1: Serial word select is output, master mode.<br>Note: (SCKD, SWSD) = (0, 0) or (1, 1) can be set. Other settings are prohibited. |
| 13 | SCKP | 0 | R/W | Serial Bit Clock Polarity<br>0: SSI_WS and SSI_SDATA change at the SSI_SCK falling edge (sampled at the SCK rising edge).<br>1: SSI_WS and SSI_SDATA change at the SSI_SCK rising edge (sampled at the SCK falling edge). |

| | SCKP = 0 | SCKP = 1 |
|---|---|---|
| SSI_SDATA input sampling timing at the time of reception (TRMD = 0) | SSI_SCK rising edge | SSI_SCK falling edge |
| SSI_SDATA output change timing at the time of transmission (TRMD = 1) | SSI_SCK falling edge | SSI_SCK rising edge |
| SSI_WS input sampling timing at the time of slave mode (SWSD = 0) | SSI_SCK rising edge | SSI_SCK falling edge |
| SSI_WS output change timing at the time of master mode (SWSD = 1) | SSI_SCK falling edge | SSI_SCK rising edge |

| | | | | |
|---|---|---|---|---|
| 12 | SWSP | 0 | R/W | Serial WS Polarity<br>The value of this bit must not be changed when the SSI module enable (EN) bit in this register is set to 1.<br>• When the stereo format or multi-channel format is used (WS_MODE = 0, MONO = 0, and WIDTH = B'00000 in the WS Mode Register):<br>0: SSI_WS is low for 1st channel, high for 2nd channel.<br>1: SSI_WS is high for 1st channel, low for 2nd channel.<br>• When the TDM format is used (WS_MODE = 1, MONO = 0, and WIDTH = B'00000 in the WS Mode Register):<br>0: The SYNC pulse is high over the period of system word 1, and low otherwise.<br>1: The SYNC pulse is low for over the period of system word 1, and high otherwise.<br>• When the monaural format is used (WS_MODE = 1, MONO = 1, and WIDTH = B'00001 to B'11111 in the WS Mode Register):<br>0: WS pulse is high over the period specified with the WIDTH bits in the WS Mode Register, and low otherwise.<br>1: WS pulse is low over the period specified with the WIDTH bits in the WS Mode Register, and high otherwise. |
| 11 | SPDP | 0 | R/W | Serial Padding Polarity<br>0: Padding bits are low.<br>1: Padding bits are high.<br>Padding bits are low when MUEN = 1. (The mute function takes priority.) |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | SDTA | 0 | R/W | Serial Data Alignment<br>This bit should be set to 0 when the monaural format is used (WS_MODE = 1, MONO = 1, and WIDTH = B'00001 to B'11111 in the WS Mode Register).<br>0: Transmitting and receiving in the order of serial data and padding bits.<br>1: Transmitting and receiving in the order of padding bits and serial data. |
| 9 | PDTA | 0 | R/W | Parallel Data Alignment<br>When the length of data word is 32, 16 or 8 bits, this configuration field has no meaning. This bit should be set to 0 when the monaural format is used (WS_MODE = 1, MONO = 1, and WIDTH = B'00001 to B'11111 in the WS Mode Register).<br>This bit applies to SSIRDR in receive mode and SSITDR in transmit mode.<br>0: Parallel data (SSITDR, SSIRDR) is left-aligned.<br>1: Parallel data (SSITDR, SSIRDR) is right-aligned.<br>DWL = 000 (with a data word length of 8 bits), the PDTA setting is ignored.<br>All data bits in SSIRDR or SSITDR are used on the audio serial bus. Four data words are transmitted or received at each 32-bit access. The first data word is derived from bits 7 to 0, the second from bits 15 to 8, the third from bits 23 to 16 and the last data word is derived from bits 31 to 24.<br>DWL = 001 (with a data word length of 16 bits), the PDTA setting is ignored.<br>All data bits in SSIRDR or SSITDR are used on the audio serial bus. Two data words are transmitted or received at each 32-bit access. The first data word is derived from bits 15 to 0 and the second data word is derived from bits 31 to 16.<br>DWL = 010, 011, 100, 101 (with a data word length of 18, 20, 22 or 24 bits), PDTA = 0 (left-aligned)<br>The data bits used in SSIRDR or SSITDR are the following:<br>Bits 31 down to (32 minus the number of bits in the data word length specified by DWL).<br>That is, if DWL = 011, the data word length is 20 bits; therefore, bits 31 to 12 in either SSIRDR or SSITDR are used. All other bits are ignored or reserved.<br>DWL = 010, 011, 100, 101 (with a data word length of 18, 20, 22 or 24 bits), PDTA = 1 (right-aligned)<br>The data bits used in SSIRDR or SSITDR are the following:<br>Bits (the number of bits in the data word length specified by DWL minus 1) to 0<br>i.e. if DWL = 011, then DWL = 20 and bits 19 to 0 are used in either SSIRDR or SSITDR. All other bits are ignored or reserved.<br>DWL = 110 (with a data word length of 32 bits), the PDTA setting is ignored.<br>All data bits in SSIRDR or SSITDR are used on the audio serial bus. |
| 8 | DEL | 0 | R/W | Serial Data Delay<br>0: One clock cycle delay between SSI_WS and SSI_SDATA<br>1: No delay between SSI_WS and SSI_SDATA |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 6 to 4 | CKDV[2:0] | 000 | R/W | Serial Oversampling Clock Division Ratio<br>Set the ratio between oversampling clock, CLK_FS, and the serial bit clock.<br>These bits are ignored if SCKD = 0.<br>The serial bit clock is used in the shift register and is provided on the SSI_SCK module pin.<br>000: Serial bit clock frequency = oversampling clock frequency/1<br>001: Serial bit clock frequency = oversampling clock frequency/2<br>010: Serial bit clock frequency = oversampling clock frequency/4<br>011: Serial bit clock frequency = oversampling clock frequency/8<br>100: Serial bit clock frequency = oversampling clock frequency/16<br>101: Serial bit clock frequency = oversampling clock frequency/6<br>110: Serial bit clock frequency = oversampling clock frequency/12<br>111: Setting prohibited<br>CKDV = 000 is invalid when WS_MODE = 1 or CONT = 1 in the WS Mode Register. |
| 3 | MUEN | 0 | R/W | Serial Data Output Disable<br>0: Module is not muted.<br>1: Module is muted.<br>Note: This bit can be used to stop output (low output) or to enable output. However, the operation is not synchronized with the change of SSI_WS. |
| 2 | — | 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 1 | TRMD | 0 | R/W | Transmit/Receive Mode Select<br>0: Module is in receive mode.<br>1: Module is in transmit mode. |
| 0 | EN | 0 | R/W | SSI Module Enable<br>0: Module is disabled.<br>1: Module is enabled.<br>When changing EN bit, do not change other bits. |

### 82.2.2 Status Register (SSISRn)

Note: n = 0

SSISR consists of status flags indicating the operational status of the SSI module and bits indicating the current channel numbers and word numbers.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DMRQ | UIRQ | OIRQ | IIRQ | DIRQ | — | — | — | — | — | — | — | — |
| Initial value: | — | — | — | 0 | 0 | 0 | — | 0 | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R/WC0 | R/WC0 | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CHNO[1:0] | | SWNO | IDST |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | 0 | 0 | 1 | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | Undefined | R | Reserved<br>The read value is not guaranteed. The write value should always be 0. |
| 28 | DMRQ | 0 | R | DMA Request Status Flag<br>This status flag allows the CPU to recognize the value of the DMA request pin on the SSI module.<br>• TRMD = 0 (receive mode)<br>If DMRQ = 1, the SSIRDR has unread data.<br>If SSIRDR is read, DMRQ = 0 until there is new unread data.<br>• TRMD = 1 (transmit mode)<br>If DMRQ = 1, SSITDR requires data to be written to continue the transmission to the audio serial bus.<br>Once data is written to SSITDR, DMRQ = 0 until it requires further transmit data. |
| 27 | UIRQ | 0 | R/WC0 | Underflow Error Interrupt Status Flag<br>This status flag indicates that data was supplied at a lower rate than was required.<br>In either case, this bit is set to 1 regardless of the value of the UIEN bit and can be cleared by writing 0 to this bit.<br>If UIRQ = 1 and UIEN = 1, an interrupt occurs.<br>• TRMD = 0 (receive mode)<br>If UIRQ = 1, SSIRDR was read before there was new unread data indicated by the DMRQ or DIRQ bit. This can lead to the same received data being stored twice by the host leading to potential corruption of multi-channel data.<br>• TRMD = 1 (transmit mode)<br>If UIRQ = 1, SSITDR did not have data written to it before it was required for transmission. This will lead to the same data being transmitted once more and a potential corruption of multi-channel data. This is more serious error than a receive mode underflow as the output SSI data results in error.<br>Note: When underflow error occurs, the current data in the data buffer of this module is transmitted until the next data is filled. |
| 26 | OIRQ | 0 | R/WC0 | Overflow Error Interrupt Status Flag<br>This status flag indicates that data was supplied at a higher rate than was required.<br>In either case this bit is set to 1 regardless of the value of the OIEN bit and can be cleared by writing 0 to this bit.<br>If OIRQ = 1 and OIEN = 1, an interrupt occurs.<br>• TRMD = 0 (receive mode)<br>If OIRQ = 1, SSIRDR was not read before there was new unread data written to it. This will lead to the loss of a data and a potential corruption of multi-channel data.<br>Note: When overflow error occurs, the current data in the data buffer of this module is overwritten by the next incoming data from the SSI interface.<br>• TRMD = 1 (transmit mode)<br>If OIRQ = 1, SSITDR had data written to it before it was transferred to the shift register. This will lead to the loss of a data and a potential corruption of multi-channel data. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 25 | IIRQ | Undefined | R | Idle Mode Interrupt Status Flag<br>This interrupt status flag indicates whether the SSI module is in idle state.<br>This bit is set regardless of the value of the IIEN bit to allow polling.<br>“Idle state” refers to the state where the serial bus has been stopped after activation of the SSI.<br>The interrupt can be masked by clearing IIEN, but cannot be cleared by writing to this bit.<br>If IIRQ = 1 and IIEN = 1, an interrupt occurs.<br>0: The SSI module is not in idle state.<br>1: The SSI module is in idle state. |
| 24 | DIRQ | 0 | R | Data Interrupt Status Flag<br>This status flag indicates that the module has data to be read or requires data to be written.<br>In either case this bit is set to 1 regardless of the value of the DIEN bit to allow polling.<br>The interrupt can be masked by clearing DIEN, but cannot be cleared by writing to this bit.<br>If DIRQ = 1 and DIEN = 1, an interrupt occurs.<br>• TRMD = 0 (receive mode)<br>0: No unread data in SSIRDR<br>1: Unread data in SSIRDR<br>• TRMD = 1 (transmit mode)<br>0: Transmit buffer is full.<br>1: Transmit buffer is empty and requires data to be written to SSITDR. |
| 23 to 4 | — | Undefined | R | Reserved<br>The read value is not guaranteed. The write value should always be 0. |
| 3, 2 | CHNO[1:0] | 00 | R | Channel Number<br>These bits indicate the current channel number. However, if the length of data words is 8 or 16 bits, the value of these bits is meaningless.<br>00: 1st channel<br>01: 2nd channel<br>10: 3rd channel<br>11: 4th channel<br>• TRMD = 0 (receive mode)<br>CHNO indicates which channel the data in SSIRDR currently represents. This value will change as the data in SSIRDR is updated from the shift register.<br>• TRMD = 1 (transmit mode)<br>CHNO indicates which channel is required to be written to SSITDR. This value will change as the data is copied to the shift register, regardless of whether the data is written to SSITDR.<br>These bits cannot be used when WS_MODE = 1 or CONT = 1 in the WS Mode Register. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | SWNO | 1 | R | System Word Number<br>This status bit indicates the current word number. However, if the length of data words is 8 or 16 bits, the value of this bit is meaningless.<br>• TRMD = 0 (receive mode)<br>SWNO indicates which system word the data in SSIRDR currently represents. This value will change as the data in SSIRDR is updated from the shift register, regardless of whether SSIRDR has been read.<br>• TRMD = 1 (transmit mode)<br>SWNO indicates which system word is required to be written to SSITDR. This value will change as the data is copied to the shift register, regardless of whether the data is written to SSITDR.<br>This bit cannot be used when WS_MODE = 1 or CONT = 1 in the WS Mode Register. |
| 0 | IDST | Undefined | R | Idle Mode Status Flag<br>This status flag indicates that the serial bus activity has stopped.<br>This bit is cleared if EN = 1 and the serial bus are currently active.<br>This bit is automatically set to 1 under the following conditions.<br>• SSI = Master transmitter (SWSD = 1 and TRMD = 1)<br>This bit is set to 1 when the EN bit is cleared and data written in SSITDR has been output from the serial data input/output pin (SSI_SDATA).<br>• SSI = Master receiver (SWSD = 1 and TRMD = 0)<br>This bit is set to 1 when the EN bit is cleared and transfer of the current system word is completed.<br>• SSI = Slave transmitter/receiver (SWSD = 0)<br>This bit is set to 1 when the EN bit is cleared and transfer of the current system word is completed.<br>Note: If an external device stops the serial bus clock before transfer of the current system word is completed, this bit is not set. |

### 82.2.3 Transmit Data Register (SSITDRn)

Note: n = 0

SSITDR is a 32-bit register that holds data to be transmitted.

Data written to this register is transferred to the shift register upon transmission request. If the data word length is less than 32 bits, the alignment is determined by the setting of the PDTA control bit in SSICR. The data in the buffer can be accessed by reading this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSITDR[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSITDR[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SSITDR[31:0] | All 0 | R/W | Transmit Data<br>Data transmitted from SSI. |

### 82.2.4 Receive Data Register (SSIRDRn)

Note: n = 0

SSIRDR is a 32-bit register that stores receive messages.

Data in this register is transferred from the shift register each time data word is received. If the data word length is less than 32 bits, the alignment is determined by the setting of the PDTA control bit in SSICR.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSIRDR[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSRDR[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SSIRDR[31:0] | All 0 | R | Receive Data<br>Data received from external pins. |

### 82.2.5 WS Mode Register (SSIWSRn)

Note: n = 0

SSIWSR is a 32-bit readable/writable register that sets the TDM format, monaural format, and WS continue function.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | WIDTH[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CONT | — | — | — | — | — | — | MONO | WS_ MODE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 20 to 16 | WIDTH[4:0] | 00000 | R/W | SYNC Pulse Width Change<br>• When the TDM format is used (WS_MODE = 1 and MONO = 0 in this register):<br>00000: TDM format<br>Other than 00000: Setting prohibited<br>• When the monaural format is used (WS_MODE = 1 and MONO = 1 in this register):<br>00000: Setting prohibited<br>00001: Pulse width is equivalent to one cycle of SCK.<br>00010: Pulse width is equivalent to two cycles of SCK.<br>...<br>11111: Pulse width is equivalent to 31 cycles of SCK.<br>Note that the set value of the system word length bits (SWL) in SSICR should be greater than that of the SYNC pulse width bits (WIDTH) when the monaural format is used. |
| 15 to 9 | — | All 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 8 | CONT | 0 | R/W | WS Continue Function<br>0: WS continue function is disabled.<br>1: WS continue function is enabled.<br>Note: This bit can only be set in master mode (SSICR.SCKD = 1, SSICR.SWSD = 1). |
| 7 to 2 | — | All 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 1 | MONO | 0 | R/W | TDM Format/Monaural Format<br>Selects the TDM format or monaural format. When the normal stereo format, multi-channel format, this bit should be 0.<br>0: TDM format<br>1: Monaural format |
| 0 | WS_MODE | 0 | R/W | WS Mode<br>0: Stereo format, multi-channel format<br>1: TDM format, monaural format |

### 82.2.6 FS Mode Register (SSIFMRn)

Note: n = 0

SSIFMR is used to set the frequency switching detection function.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | DTCT[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CTDV[1:0] | | — | — | — | FSEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 22 | — | All 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 21 to 16 | DTCT[5:0] | 000000 | R/W | Frequency Switching Detection Range Set<br>000100: Set the value in the range as 5.<br>Other than above: Setting prohibited. |
| 15 to 6 | — | All 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 5, 4 | CTDV[1:0] | 00 | R/W | Bus Clock Division Ratio<br>Sets the bus clock division ratio used for the frequency switching detection function.<br>00: Setting prohibited.<br>01: Setting prohibited.<br>10: Setting prohibited.<br>11: Set the switching detection clock frequency = bus clock frequency/8. |
| 3 to 1 | — | All 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 0 | FSEN | 0 | R/W | Frequency Switching Detection Function Enable<br>0: The frequency switching detection function is disabled.<br>1: The frequency switching detection function is enabled.<br>Note: This bit must be enabled after the desired settings have been made for SWSP in SSICR and WS_MODE in SSIWSR. |

### 82.2.7 FS Status Register (SSIFSRn)

Note: n = 0

SSIFSR is used to read the frequency switching detection status and information on the frequency counter.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FCST | DTST | — | — | FCNT[11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/WC0 | R/WC0 | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 15 | FCST | 0 | R/WC0 | WS Stopped Status Flag<br>In the frequency switching detection function, this status flag reflects the state that the WS signal is stopped.<br>When FCST = 1, this bit indicates that the WS signal has stopped (the state is detected only when FSEN in SSIFMR = 1).<br>Write 0 if this bit is cleared to 0. |
| 14 | DTST | 0 | R/WC0 | Frequency Switching Detection Status Flag<br>In the frequency switching detection function, this status flag reflects the state that frequency switching has been detected.<br>When DTST = 1, this bit indicates that the switching has been detected (the state is detected only when FSEN in SSIFMR = 1).<br>Write 0 if this bit is cleared to 0. |
| 13, 12 | — | 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 11 to 0 | FCNT[11:0] | H'000 | R | Frequency Count Monitor<br>When the frequency switching detection function is enabled, the count value between WS signal edges is reflected according to the setting of the operating mode and the serial WS polarity (SWSP in SSICR).<br>This value is updated when the frequency is switched and the WS signal has been detected. |

### 82.2.8 Control Register Extend (SSICREm) (m = 0)

CHNL2 is an extension bit of SSICRn.CHNL[1:0]. This bit should be set 0 except SSIq_MODE2.ex_func (q = 0_0) in SSIU is 1 and TDM format is not selected (SSIWSRn.WS_MODE = 1 and SSIWSRn.MONO = 0).

This register should be set up before SSICRn is set up.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CHNL2 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | 0 | R | Reserved<br>The read value is 0. The write value should always be 0. |
| 0 | CHNL2 | 0 | R/W | Extension bit of SSICRn.CHNL[1:0]<br>0: Default value<br>1: A TDM frame consists of sixteen system words. In this case, SSICRn.CHNL should be set B'00. |

## 82.3 Operation

### 82.3.1 Bus Format

The SSI module can operate as a transmitter or a receiver and can be configured into many serial bus formats in either mode.

The bus format can be selected from one of the four major modes shown in Table 82.2.

**Table 82.2 Bus Format for SSI Module**

| | SSICR | | | | | | | | | | | | | | | | | | SSIWSR | | | SSICRE * |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRMD | SCKD | SWSD | EN | MUEN | DIEN | IIEN | OIEN | UIEN | DEL | PDTA | SDTA | SPDP | SWSP | SCKP | SWL[2:0] | DWL[2:0] | CHNL[1:0] | WS_MODE | MONO | CONT | CHNL2 |
| Slave receiver | 0 | 0 | 0 | Control bits | | | | | | | Configuration bits | | | | | | | | WS mode bits | | | Control bits |
| Slave transmitter | 1 | 0 | 0 | | | | | | | | | | | | | | | | | | | |
| Master receiver | 0 | 1 | 1 | | | | | | | | | | | | | | | | | | | |
| Master transmitter | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | |

**(1) Slave receiver**

This mode allows the module to receive serial data from another device. The clock and word select signal used for the serial data stream is also supplied from an external device. If these signals do not conform to the format specified in the configuration fields of the SSI module, operation is not guaranteed.

**(2) Slave transmitter**

This mode allows the module to transmit serial data to another device. The clock and word select signal used for the serial data stream is also supplied from an external device. If these signals do not conform to the format specified in the configuration fields of the SSI module, operation is not guaranteed.

**(3) Master receiver**

This mode allows the module to receive serial data from another device. The clock and word select signals are internally derived from the CLK_FS input clock. The format of these signals is defined in the configuration fields of the SSI module. If the incoming data does not follow the configured format, operation is not guaranteed.

**(4) Master transmitter**

This mode allows the module to transmit serial data to another device. The clock and word select signals are internally derived from the CLK_FS input clock. The format of these signals is defined in the configuration fields of the SSI module.

**(5) Setting for each format**

Table 82.3 shows the setting for each format:

**Table 82.3 Setting for Each Format**

| | SSICR | SSIWSR | | | SSICRE |
|---|---|---|---|---|---|
| **Format** | **CHNL[1:0]** | **WS_MODE** | **MONO** | **WIDTH[4:0]** | **CHNL2** |
| Stereo format | B'00 | 0 | 0 | B'00000 | B'0 |
| Multi-channel format | B'01/B'10/B'11 | 0 | 0 | B'00000 | B'0 |
| Monaural format | B'00 | 1 | 1 | B'00001 to B'11111 | B'0 |
| TDM format | B'00/B'01/B'10/B'11 * | 1 | 0 | B'00000 | B'0/1 * |

Note: * SSICR.CHNL[1:0] should be set B'00 to when SSICRE.CHNL2 = 1.

### (6) Configuration bits (related to word length)

There are many configurations the SSI module supports, but only some of the combinations are shown below for the I2S, left-aligned, and right-aligned formats.

Figure 82.2 and Figure 82.3 demonstrate the supported I2S format both with and without padding. Padding occurs when the data word length is smaller than the system word length.

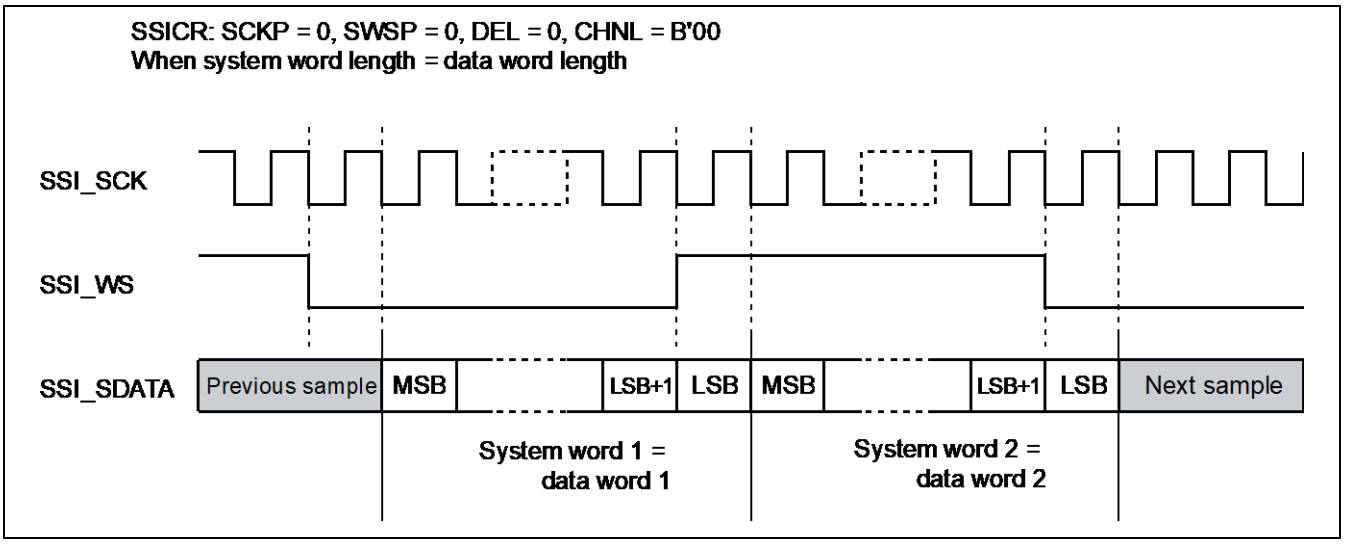



**Figure 82.2 I2S Format (without Padding)**

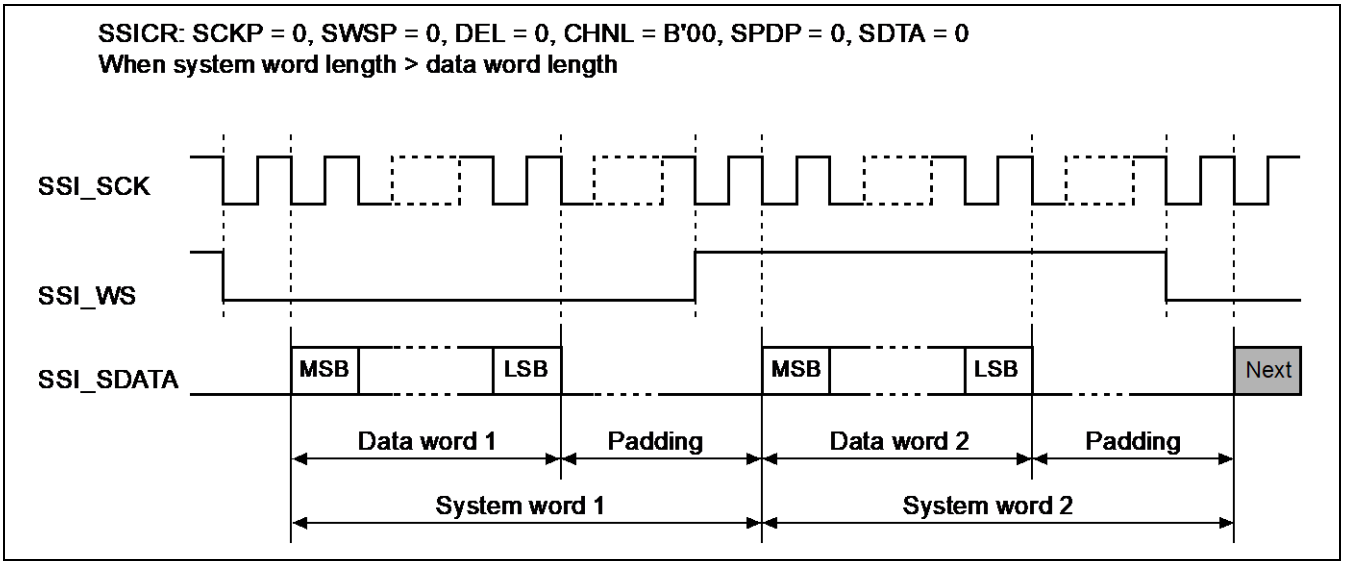



**Figure 82.3 I2S Format (with Padding)**

Figure 82.4 shows the left-aligned format and Figure 82.5 shows the right-aligned format. Padding is assumed in both cases, but may not be present in a final implementation if the system word length equals the data word length.

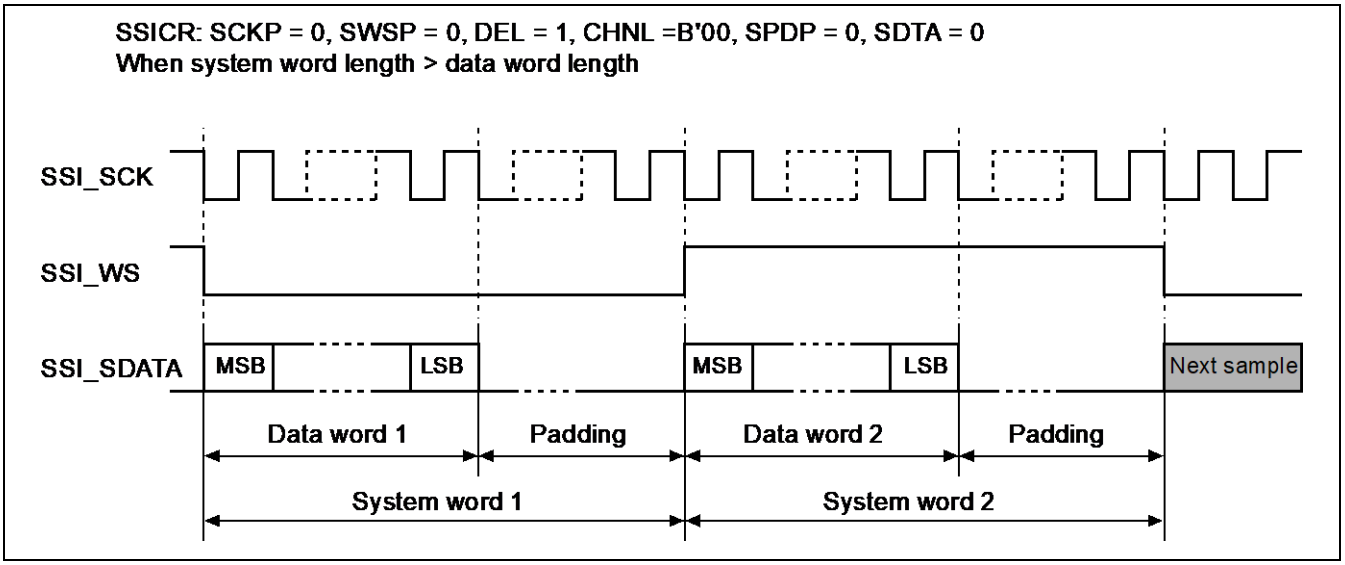



**Figure 82.4 Left-Aligned Format**
**(Transmitting and Receiving in the Order of Serial Data and Padding Bits)**

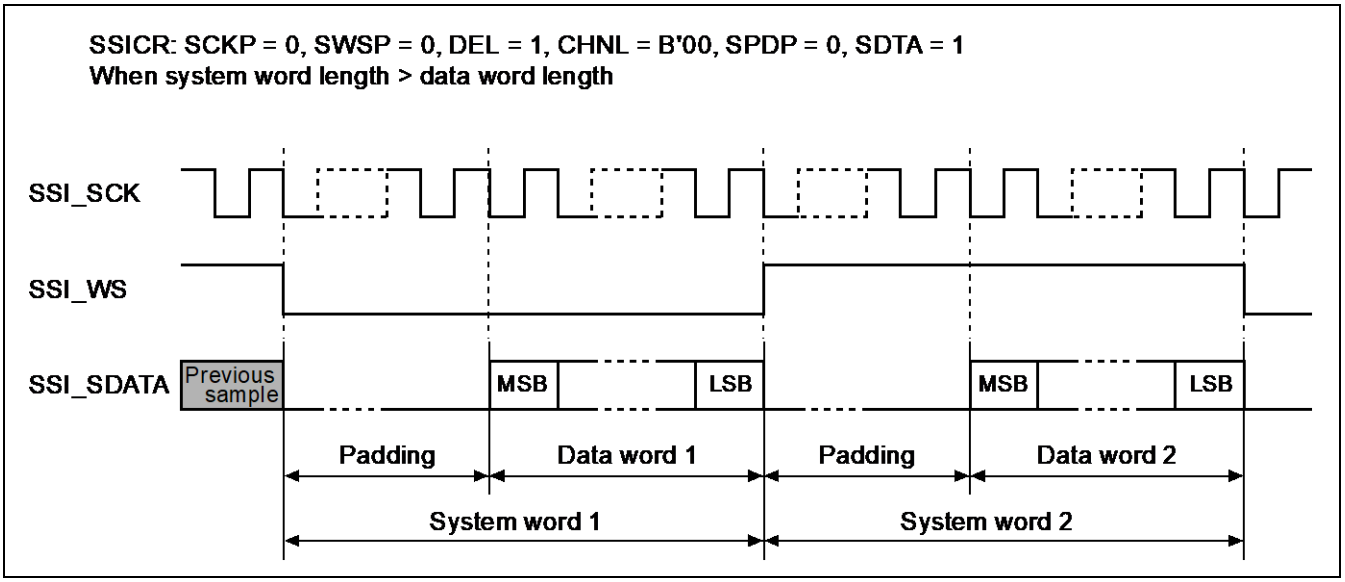



**Figure 82.5 Right-Aligned Format**
**(Transmitting and Receiving in the Order of Padding Bits and Serial Data)**

**(7) Multi-channel formats**

Some devices extend the definition of the I2S format and allow more than two channels to be transferred within two system words.

The SSI module supports the transfer of two, three, and four channels by using the CHNL, SWL and DWL bits only when the system word length (SWL) is greater than or equal to the data word length (DWL) multiplied by channels (CHNL).

Table 82.4 shows the number of padding bits for each of the valid setting. If setting is not valid, "—" is indicated instead of a number.

**Table 82.4 The Number of Padding Bits for Each Valid Setting**

| Padding Bits Per System Word | | DWL[2:0] | B'000 | B'001 | B'010 | B'011 | B'100 | B'101 | B'110 |
|---|---|---|---|---|---|---|---|---|---|
| CHNL [1:0] | Decoded Channels per System Word | SWL [2:0] | Decoded Word Length | 8 | 16 | 18 | 20 | 22 | 24 | 32 |
| B'00 | 1 | B'000 | 8 | 0 | — | — | — | — | — | — |
| | | B'001 | 16 | 8 | 0 | — | — | — | — | — |
| | | B'010 | 24 | 16 | 8 | 6 | 4 | 2 | 0 | — |
| | | B'011 | 32 | 24 | 16 | 14 | 12 | 10 | 8 | 0 |
| | | B'100 | 48 | 40 | 32 | 30 | 28 | 26 | 24 | 16 |
| | | B'101 | 64 | 56 | 48 | 46 | 44 | 42 | 40 | 32 |
| | | B'110 | 128 | 120 | 112 | 110 | 108 | 106 | 104 | 96 |
| | | B'111 | 256 | 248 | 240 | 238 | 236 | 234 | 232 | 224 |
| B'01 | 2 | B'000 | 8 | — | — | — | — | — | — | — |
| | | B'001 | 16 | 0 | — | — | — | — | — | — |
| | | B'010 | 24 | 8 | — | — | — | — | — | — |
| | | B'011 | 32 | 16 | 0 | — | — | — | — | — |
| | | B'100 | 48 | 32 | 16 | 12 | 8 | 4 | 0 | — |
| | | B'101 | 64 | 48 | 32 | 28 | 24 | 20 | 16 | 0 |
| | | B'110 | 128 | 112 | 96 | 92 | 88 | 84 | 80 | 64 |
| | | B'111 | 256 | 240 | 224 | 220 | 216 | 212 | 208 | 192 |
| B'10 | 3 | B'000 | 8 | — | — | — | — | — | — | — |
| | | B'001 | 16 | — | — | — | — | — | — | — |
| | | B'010 | 24 | 0 | — | — | — | — | — | — |
| | | B'011 | 32 | 8 | — | — | — | — | — | — |
| | | B'100 | 48 | 24 | 0 | — | — | — | — | — |
| | | B'101 | 64 | 40 | 16 | 10 | 4 | — | — | — |
| | | B'110 | 128 | 104 | 80 | 74 | 68 | 62 | 56 | 32 |
| | | B'111 | 256 | 232 | 208 | 202 | 196 | 190 | 184 | 160 |
| B'11 | 4 | B'000 | 8 | — | — | — | — | — | — | — |
| | | B'001 | 16 | — | — | — | — | — | — | — |
| | | B'010 | 24 | — | — | — | — | — | — | — |
| | | B'011 | 32 | 0 | — | — | — | — | — | — |
| | | B'100 | 48 | 16 | — | — | — | — | — | — |
| | | B'101 | 64 | 32 | 0 | — | — | — | — | — |
| | | B'110 | 128 | 96 | 64 | 56 | 48 | 40 | 32 | 0 |
| | | B'111 | 256 | 224 | 192 | 184 | 176 | 168 | 160 | 128 |

When the SSI module acts as a transmitter, each word written to SSITDR is transmitted to the serial audio bus in the order they are written. When the SSI module acts as a receiver, each word received by the serial audio bus is read in the order received from SSIRDR.

Figure 82.6 to Figure 82.8 show how two, three and four channels are transferred to the serial audio bus. Note that there are no padding bits in the first example, the second example is left-aligned and the third is right-aligned. This selection is arbitrary and is just for demonstration purposes only.

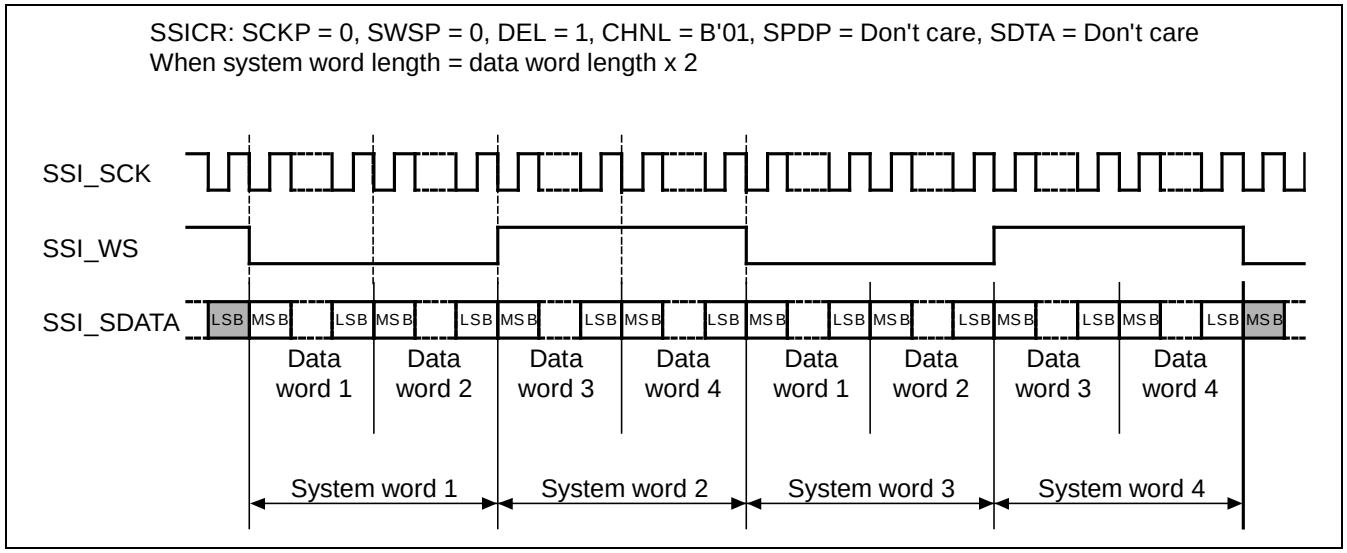



**Figure 82.6 Multi-Channel Format (Two Channels without Padding)**

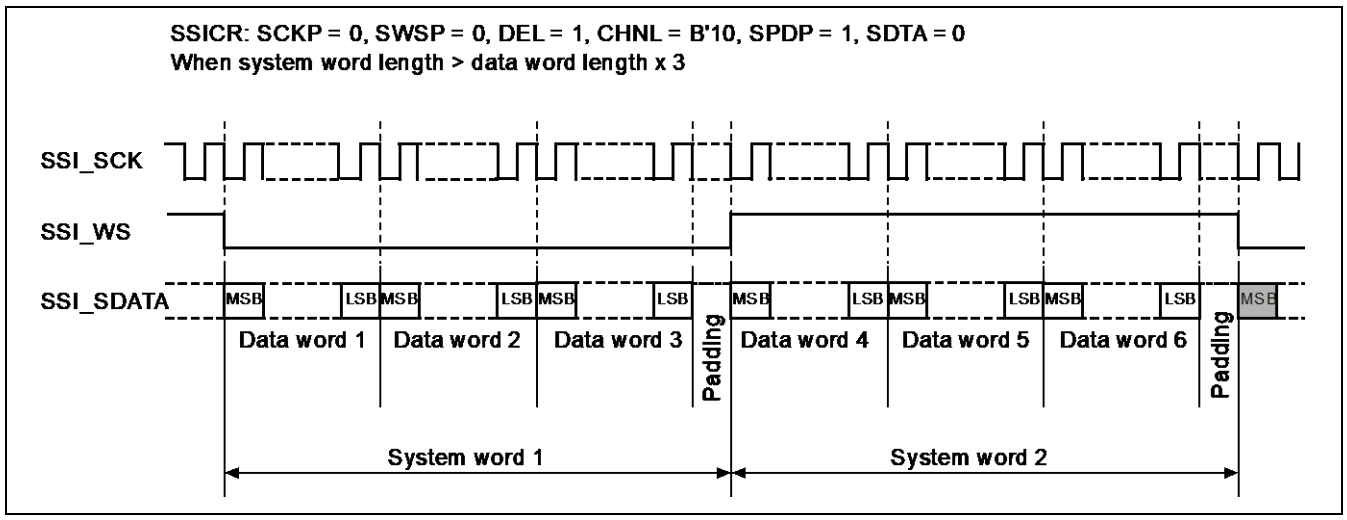



**Figure 82.7 Multi-Channel Format (Three Channels with High Padding)**

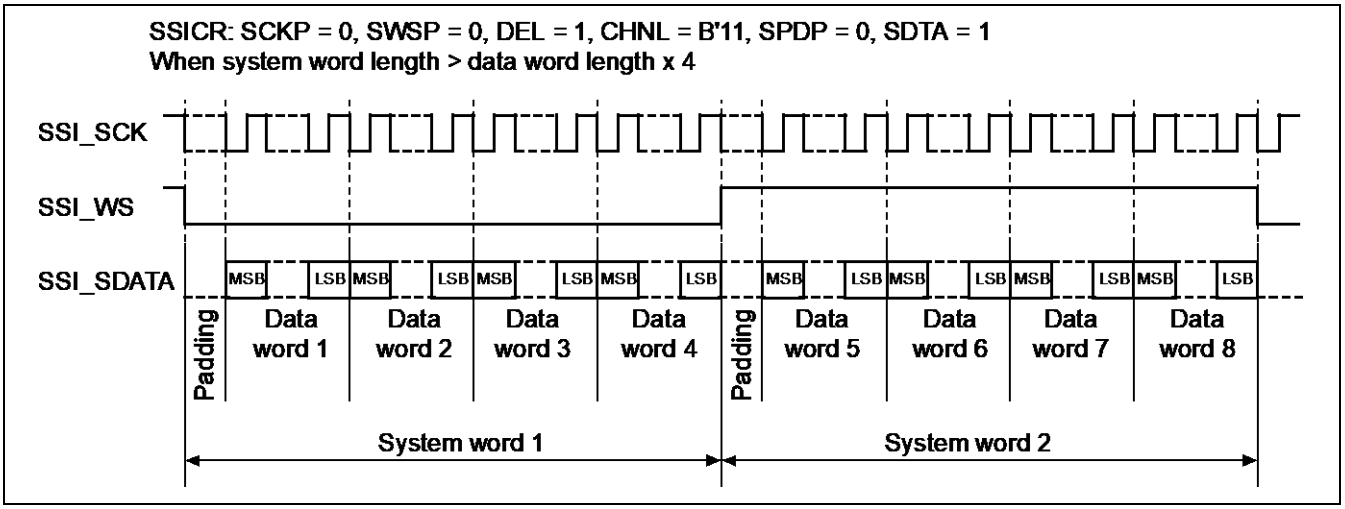



**Figure 82.8 Multi-Channel Format**
**(Four Channels; Transmitting and Receiving in the Order of Padding Bits and Serial Data; with Padding)**

### (8) TDM formats

The TDM format is used for connecting the SSI with a multi-channel device which supports TDM. The TDM format is set by using the WS_MODE and MONO bits in SSIWSR. When the SWSP bit in SSICR is 0 with the TDM format, SSI_WS is driven high over the period of system word 1, and pulled low otherwise. When the SWSP bit is 1, SSI_WS is pulled low over the period of system word 1, and driven high otherwise. The pulses generated on the SSI_WS signal are referred to as SYNC pulses.

Figure 82.9 and Figure 82.10 show the TDM formats without padding and with padding.

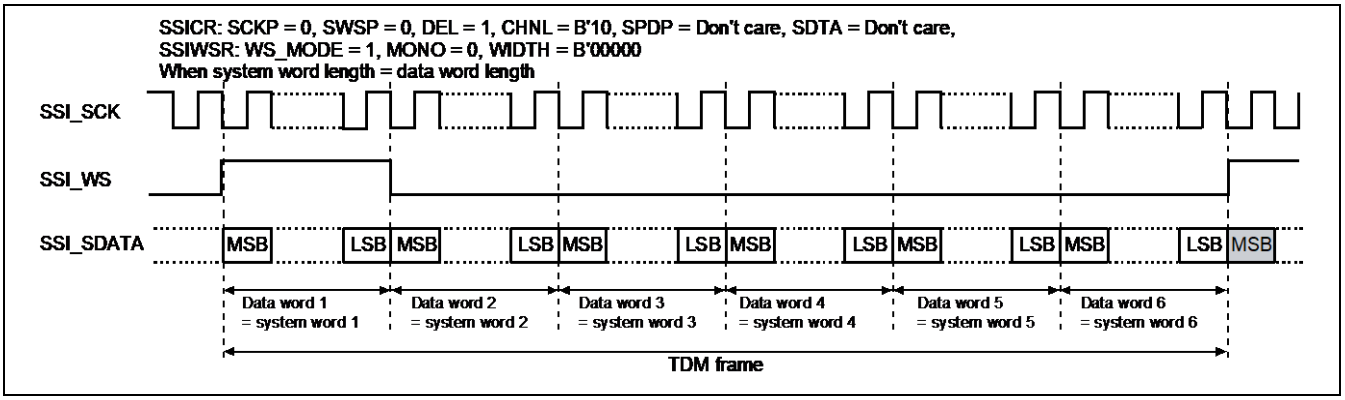



**Figure 82.9 TDM Format (Six System Words, without Padding)**

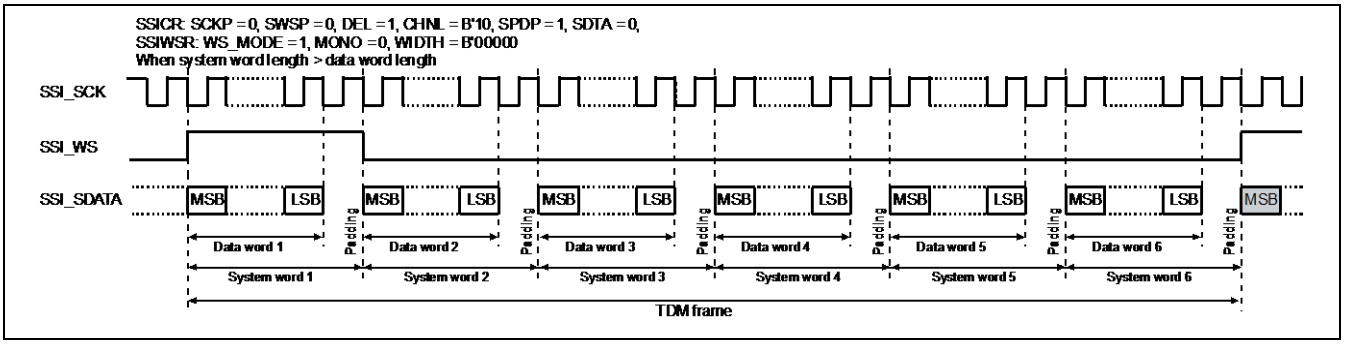



**Figure 82.10 TDM Format (Six System Words, with Padding)**

The following describes operation in each mode:

#### (a) Master transmitter

By a transfer start trigger (setting the EN bit in the control register to 1), a transfer of system word 1 begins synchronously with the SYNC pulses.

By a transfer stop trigger (setting the EN bit in the control register to 0), a transfer is stopped at the end of the currently-transferred system word and the SDATA signal is output according to the SPDP setting in SSICR (when SPDP = 0, a low-level signal is output).

If transmit data is not ready in the SSI module during transmission, an underflow will occur. When an underflow occurs, data to be output for the SYNC pulses cannot be determined. Therefore, stop and reconfigure the transfer.

#### (b) Master receiver

By a transfer start trigger (setting the EN bit in the control register to 1), a reception of system word 1 data begins at the point where the SSI module recognizes a SYNC pulse.

By a transfer stop trigger (setting the EN bit in the control register to 0), a reception is stopped at the end of the currently-transferred system word.

The receive data register should not be read when it does not have data. If read, an underflow will occur. When an underflow occurs, stop and reconfigure the transfer.

The receive data should not be written to before read. If written to, an overflow will occur. When an overflow occurs, stop and reconfigure the transfer.

**(c) Slave transmitter**

By a transfer start trigger (setting the EN bit in the control register to 1), a transfer of system word 1 begins synchronously with the SYNC pulses.

By a transfer stop trigger (setting the EN bit in the control register to 0), a transfer is stopped at the end of the currently-transferred system word and the SDATA signal is output according to the SPDP setting in SSICR (when SPDP = 0, a low-level signal is output).

If transmit data is not ready in the SSI module during transmission, an underflow will occur. When an underflow occurs, data to be output for SYNC pulses cannot be determined. Therefore, stop and re-set the transfer.

Transfers cannot be performed if the SCK signal and SYNC pulses are not provided to the SSI module during transfer.

**(d) Slave receiver**

By a transfer start trigger (setting the EN bit in the control register to 1), a reception of system word 1 data begins at the point where the SSI module recognizes a SYNC pulse.

By a transfer stop trigger (setting the EN bit in the control register to 0), a reception is stopped at the end of the currently-transferred system word.

The receive data should not be read when it does not have data. If read, an underflow will occur. When an underflow occurs, stop and reconfigure the transfer.

The receive data register should not be written to before read. If written to, an overflow will occur. When an overflow occurs, stop and reconfigure the transfer.

Transfer cannot proceed if the SCK signal and SYNC pulses are not being provided to the SSI module during transfer.

### (9) Monaural format

The monaural format is set by using the WS_MODE and MONO bits in SSIWSR. The available data lengths are 8 bits and 16 bits. When the SWSP bit in SSICR is 0 and the monaural format is selected, SSI_WS is driven high over the period from bit 1 to bit 31, and pulled low otherwise. When the SWSP bit is 1, SSI_WS is pulled low over the period from bit 1 to bit 31, and driven high otherwise. The pulses generated as the SSI_WS signal are referred to as SYNC pulses. The width of the SYNC pulses can be set to a value in bits between 1 and 31 by using the WIDTH bits in SSIWSR. The setting must be smaller than the system word length (SSICR.SWL). In the monaural format, the system word length is equal to one monaural frame period.

Figure 82.11 shows the left-aligned monaural format and Figure 82.12 shows the monaural format with a delay of one bit-period.

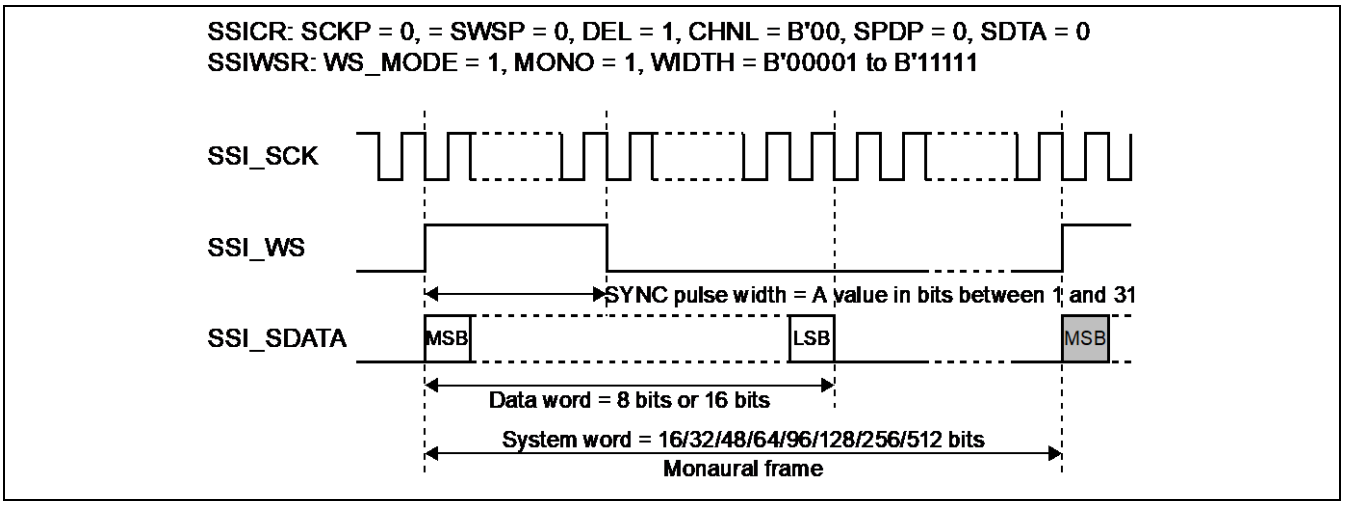



**Figure 82.11 Monaural Format (Left-Aligned)**

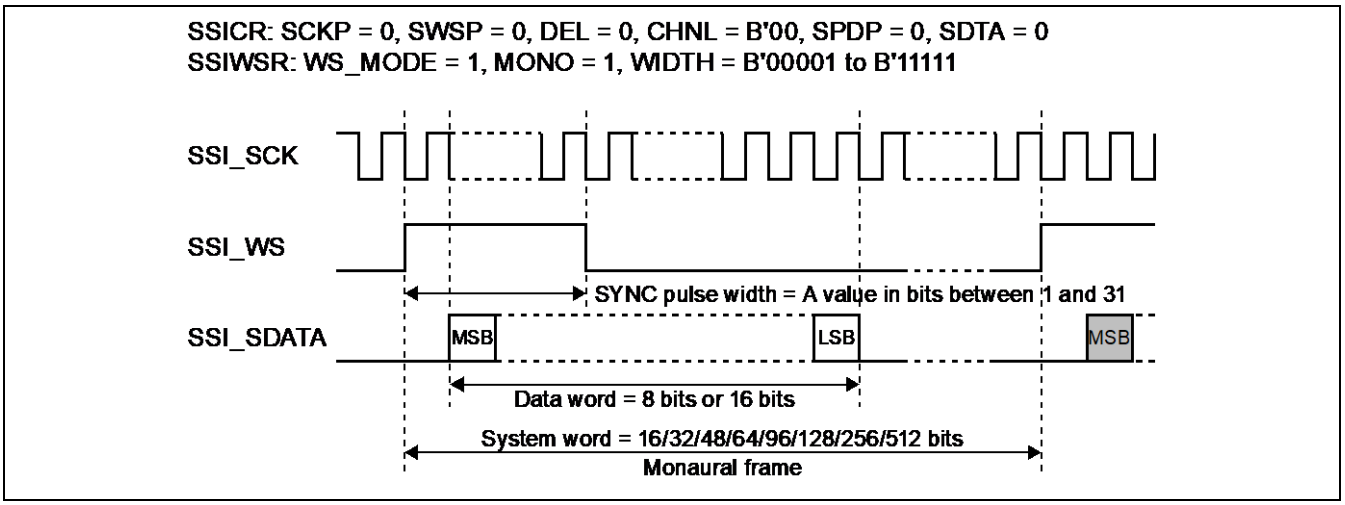



**Figure 82.12 Monaural Format (with 1-bit Delay)**

**(10) Configuration bits**

Several more configuration bits are shown below. These bits are not mutually exclusive, but some combinations may not be useful for any other device.

These configuration bits are described below with reference to Figure 82.13, Basic Sample Format.

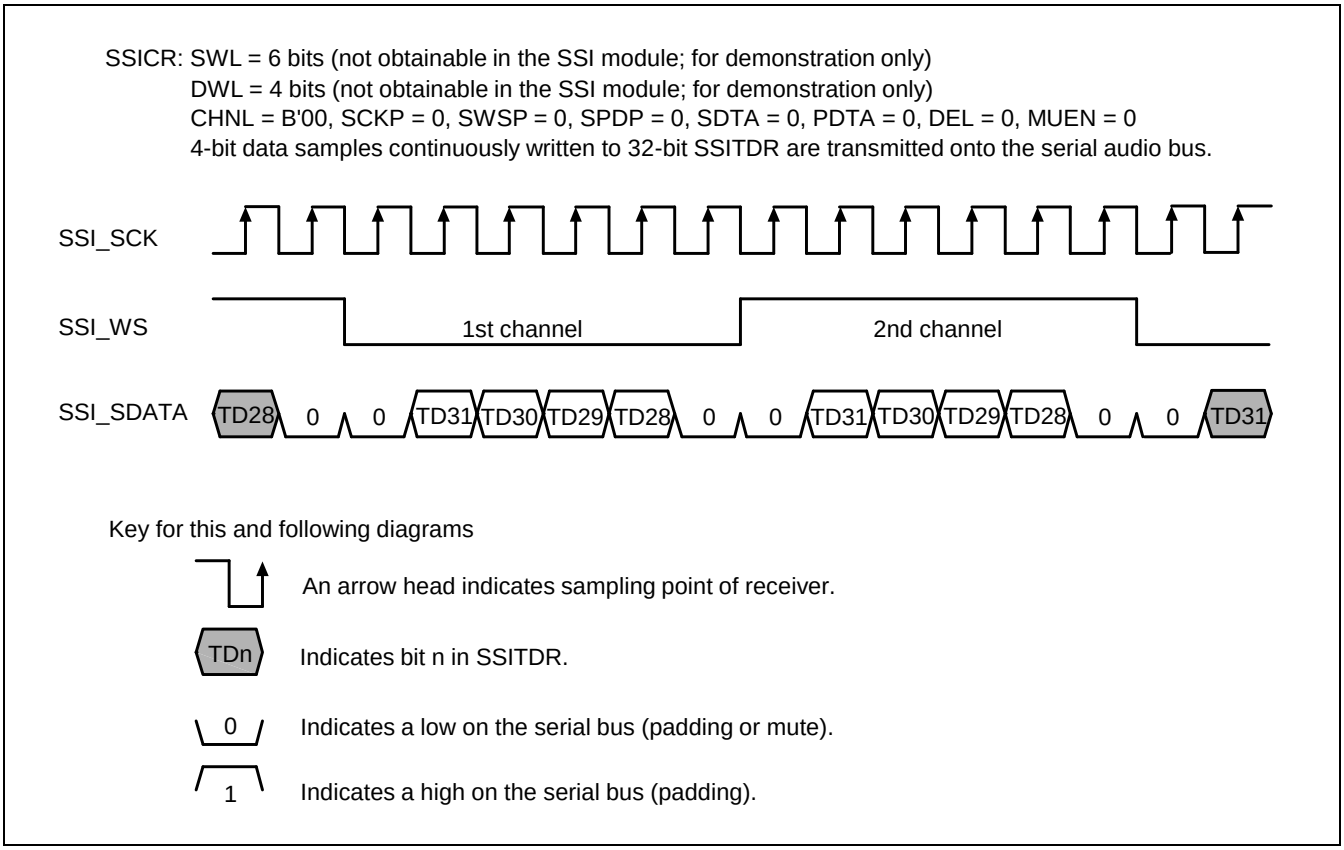



**Figure 82.13 Basic Sample Format**
**(Transmit Mode with Example System/Data Word Length)**

Figure 82.13 uses a system word length of 6 bits and a data word length of 4 bits. These settings are not possible with the SSI module but are used only for clarification of the other configuration bits.

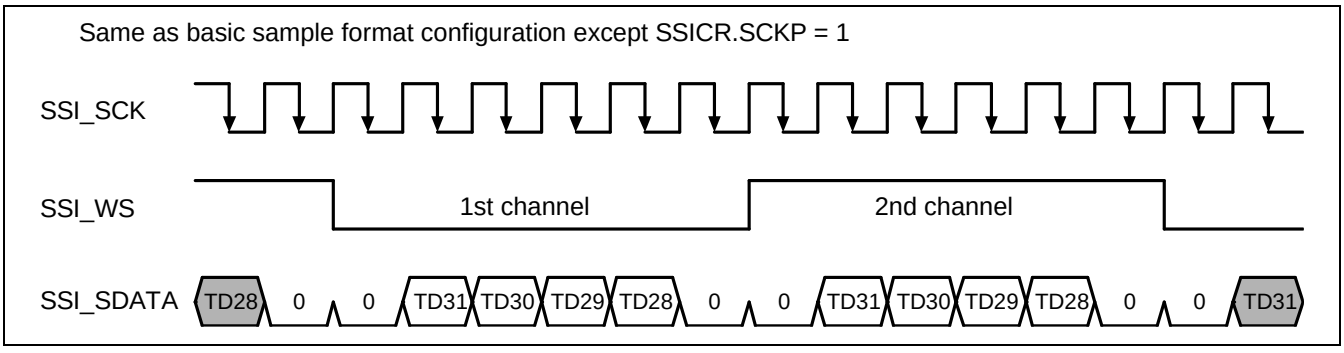



**Figure 82.14 Inverted Clock**

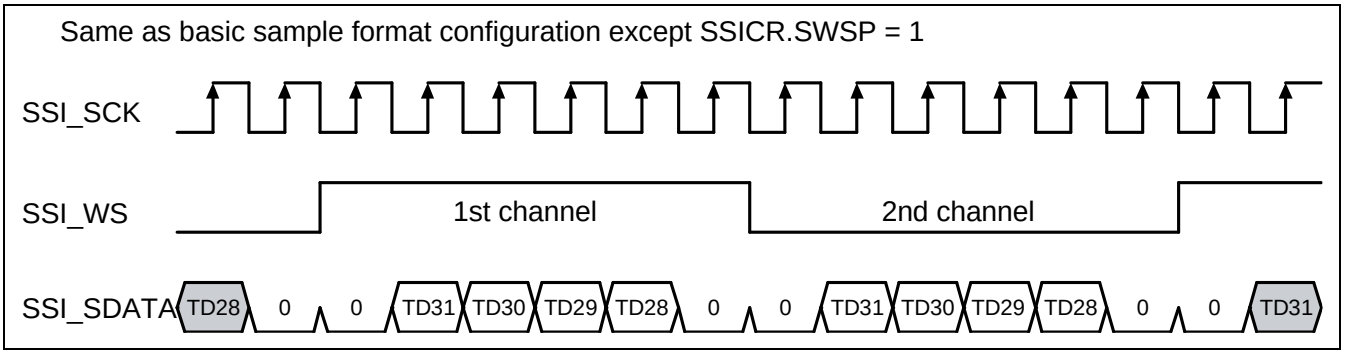



**Figure 82.15 Inverted Word Select**

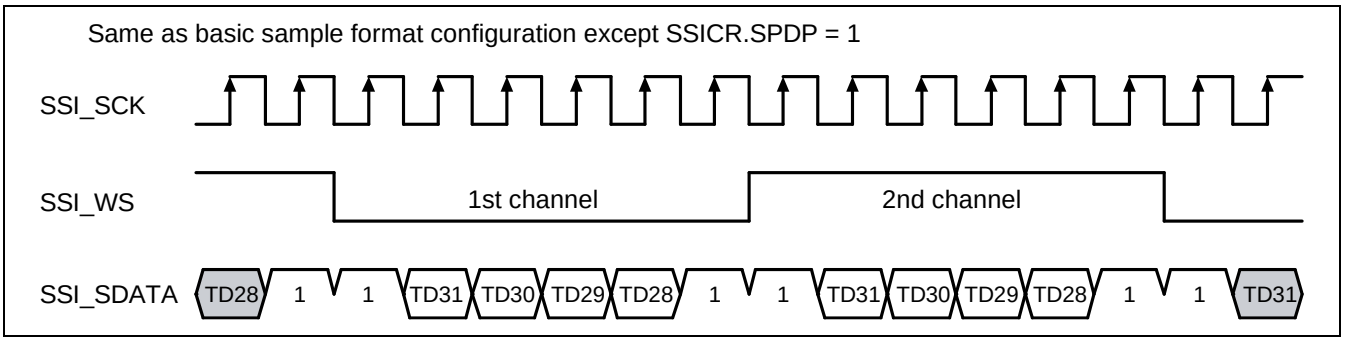



**Figure 82.16 Inverted Padding Polarity**

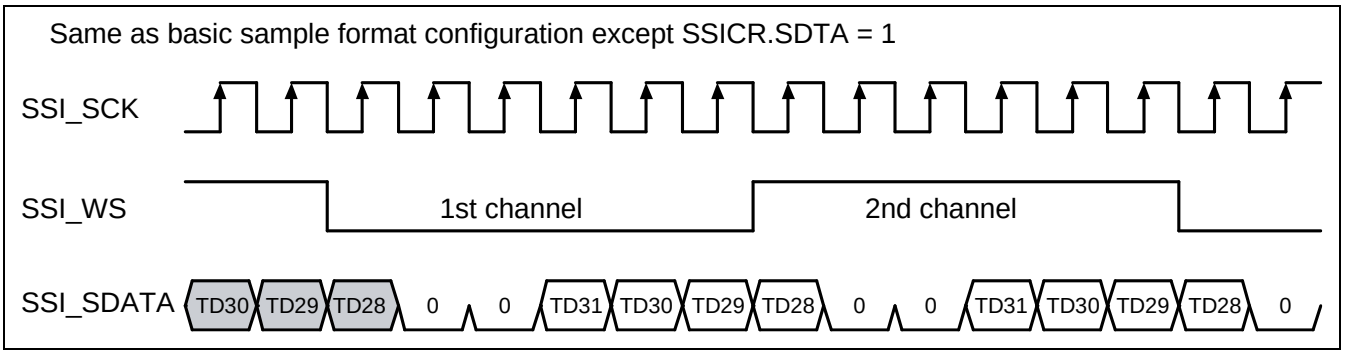



**Figure 82.17 Transmitting and Receiving in the Order of Padding Bits and Serial Data; with Delay**

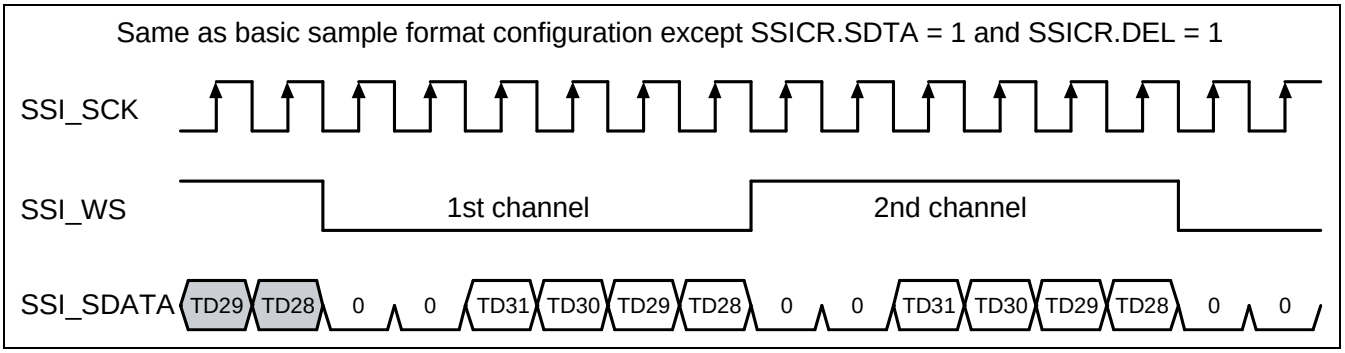



**Figure 82.18 Transmitting and Receiving in the Order of Padding Bits and Serial Data; without Delay**

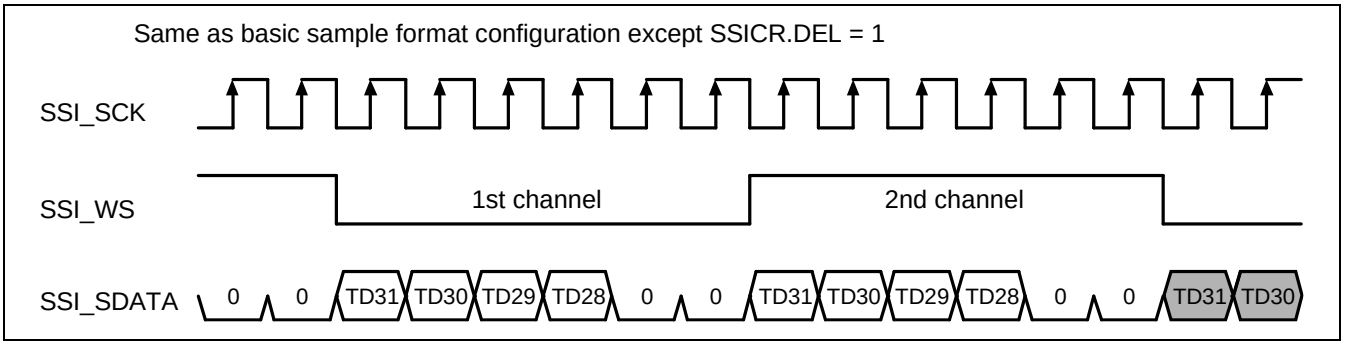



**Figure 82.19 Transmitting and Receiving in the Order of Serial Data and Padding Bits; without Delay**

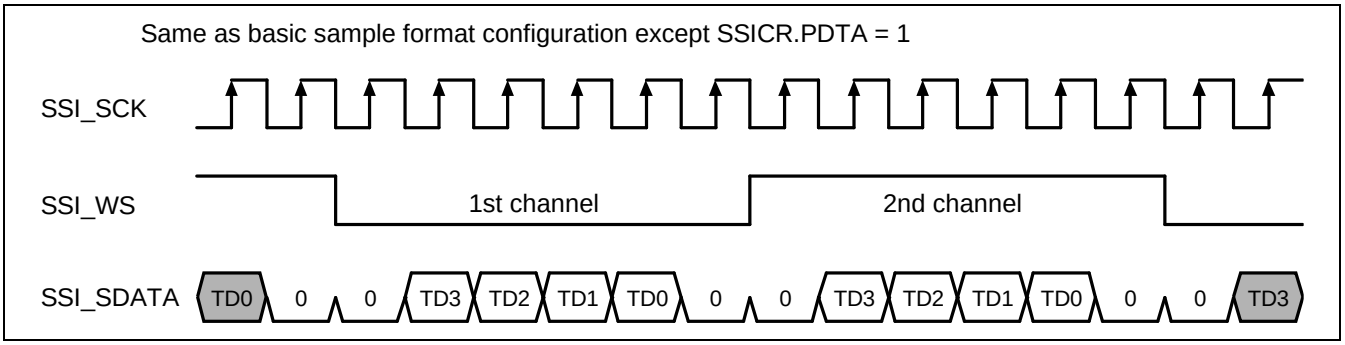



**Figure 82.20 Parallel Right-Aligned with Delay**

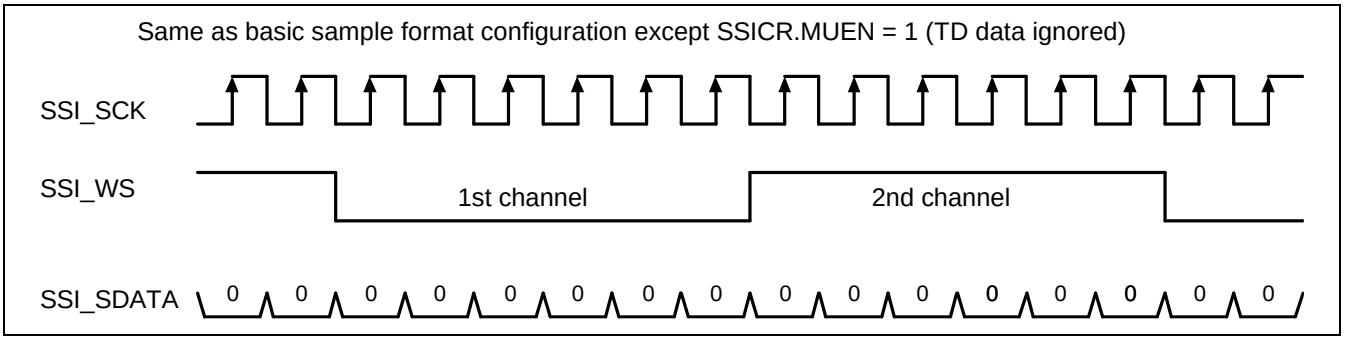



**Figure 82.21 Mute Enabled**

### (11) WS continue function

The WS continue function is used to output SSI_WS signal continuously regardless of whether data transfer is enabled or disabled. The WS continue function can be set by using the CONT bit in SSIWSR. This function can be combined with master mode only. When this function is enabled, the SSI_WS signal continues to operate even if the EN bit in SSICR is cleared to 0 (stopping a transfer). When this function is disabled, the SSI_WS signal operates synchronously with the EN bit.

Figure 82.22 and Figure 82.23 show operations when the WS continue function is enabled and disabled, respectively.

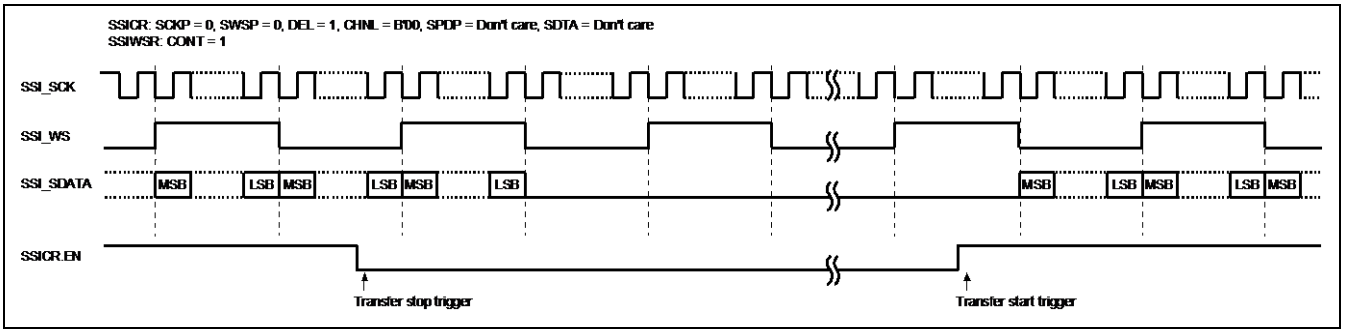



**Figure 82.22 WS Continue Function Operation (When the Function is Enabled)**

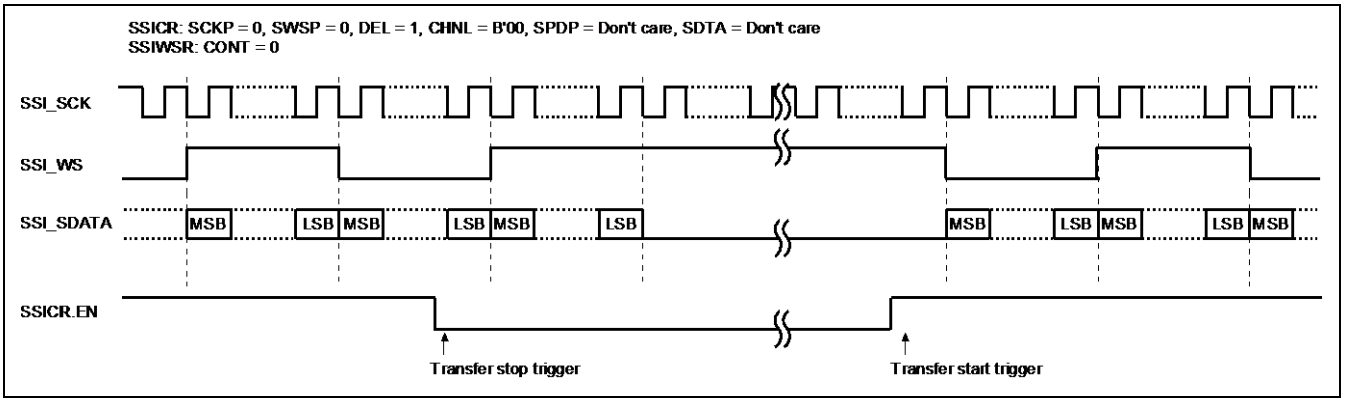



**Figure 82.23 WS Continue Function Operation (When the Function is Disabled)**

### (12) Bit alignment in transferred data

Figure 82.24 to Figure 82.36 show the bit alignment in transmit or receive data. Only the MSB first alignment is available. The waveforms in Figure 82.24 to Figure 82.36 are setting examples without delay (SSICR.DEL is set to 1, the number of bits specified with SSICR.DWL is the same as that specified with SSICR.SWL).

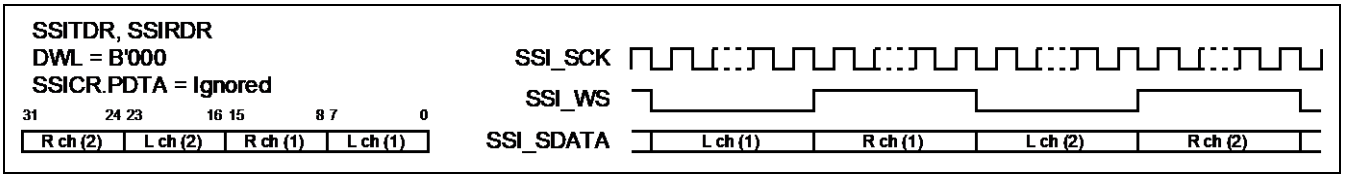



**Figure 82.24 Bit Alignment in 8-Bit Stereo Format**

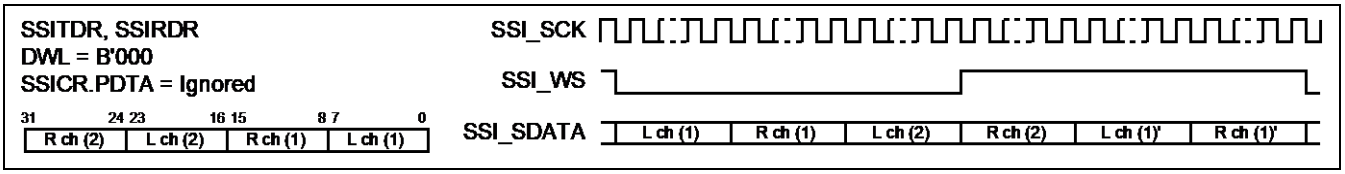



**Figure 82.25 Bit Alignment in 8-Bit Multi-Channel Format (Three Channels)**

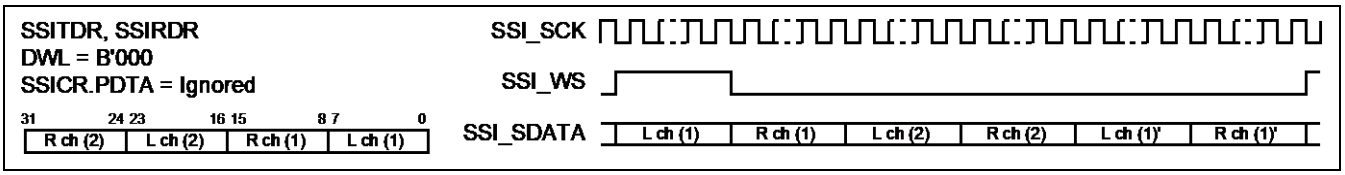



**Figure 82.26 Bit Alignment in 8-Bit TDM Format (Six System Words)**

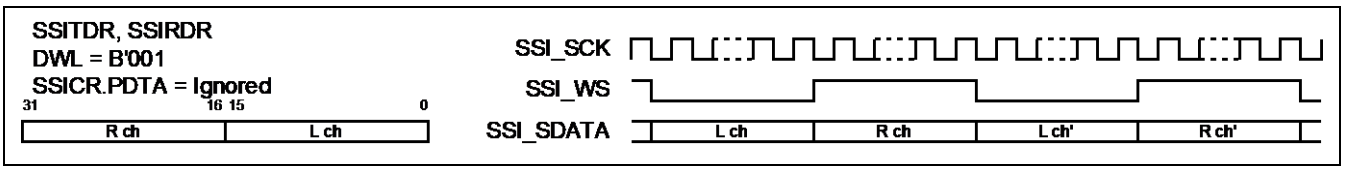



**Figure 82.27 Bit Alignment in 16-Bit Stereo Format**

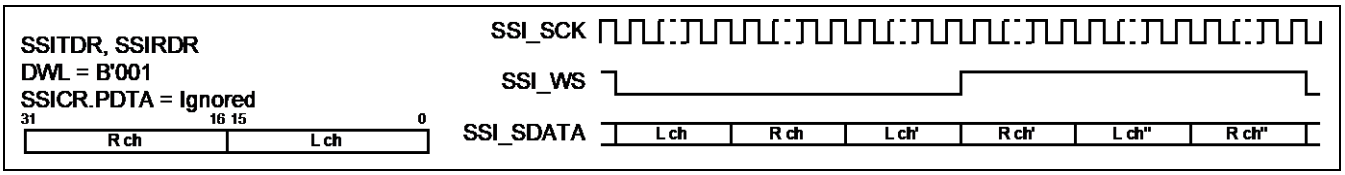



**Figure 82.28 Bit Alignment in 16-Bit Multi-Channel Format (Three Channels)**

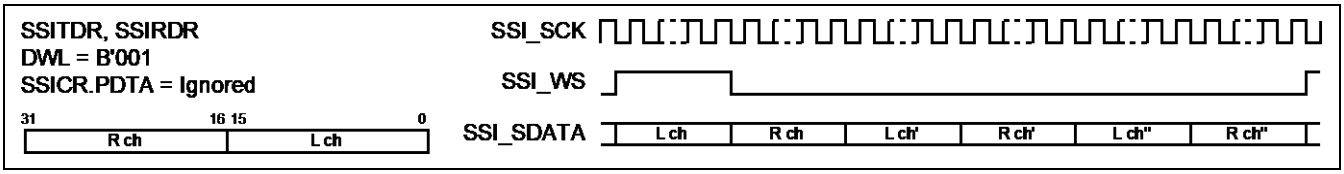



**Figure 82.29 Bit Alignment in 16-Bit TDM Format (Six System Words)**

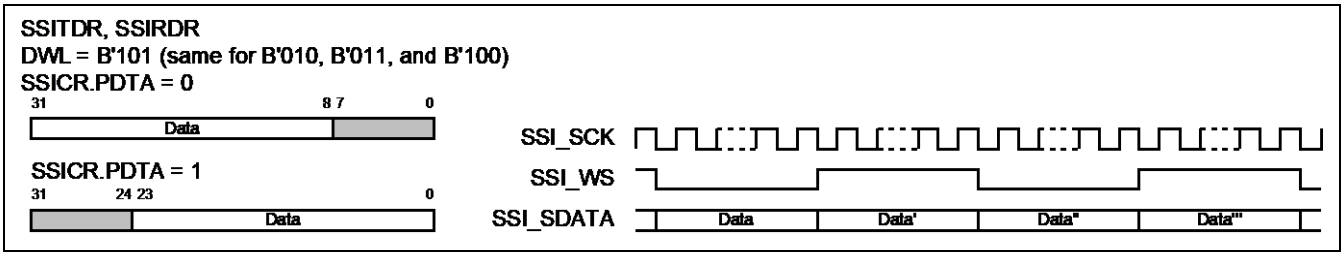



**Figure 82.30 Bit Alignment in 18-/20-/22-/24-Bit Stereo Format**

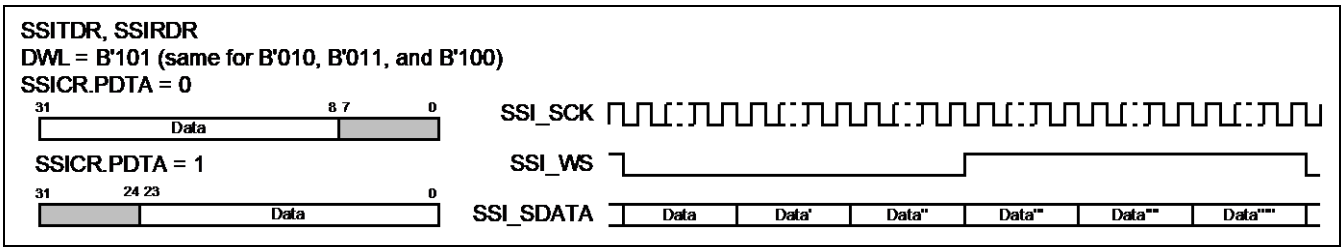



**Figure 82.31 Bit Alignment in 18-/20-/22-/24-Bit Multi-Channel Format (Three Channels)**

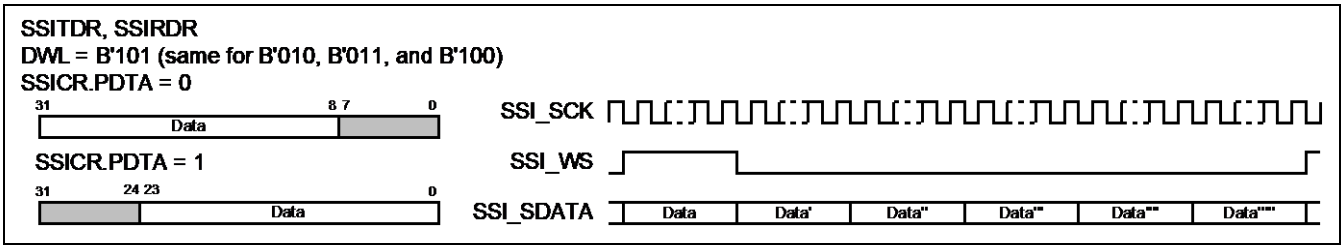



**Figure 82.32 Bit Alignment in 18-/20-/22-/24-Bit TDM Format (Six System Words)**

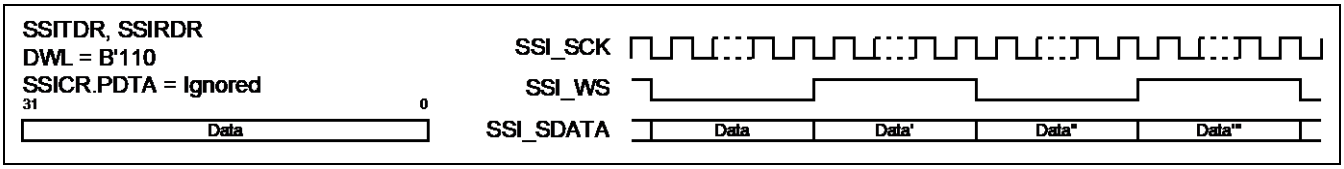



**Figure 82.33 Bit Alignment in 32-Bit Stereo Format**

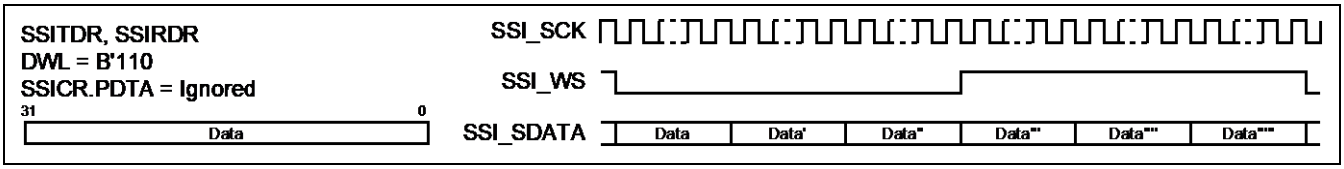



**Figure 82.34 Bit Alignment in 32-Bit Multi-Channel Format (Three Channels)**

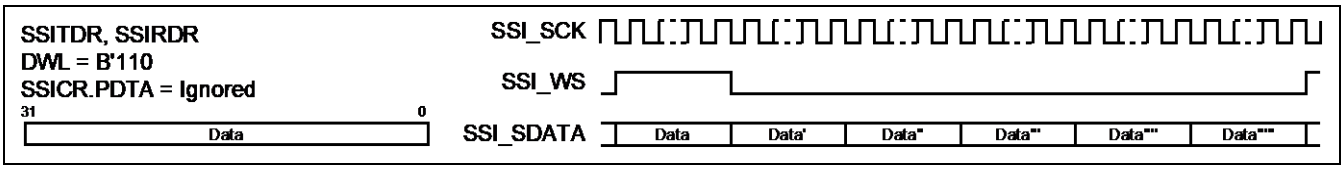



**Figure 82.35 Bit Alignment in 32-Bit TDM Format (Six System Words)**

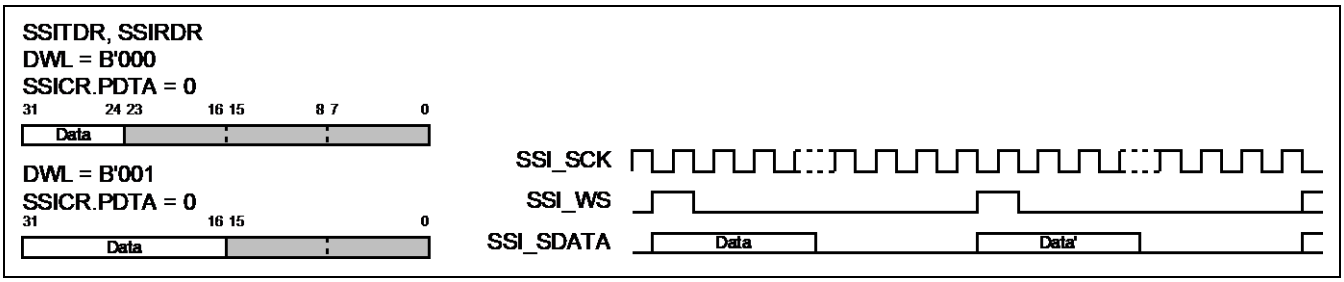



**Figure 82.36 Bit Alignment in 8-/16-Bit Monaural Format**

### 82.3.2 Operating Modes

There are three modes of operation: configuration, module enabled and module disabled. Figure 82.37 shows how the module enters each of these modes.

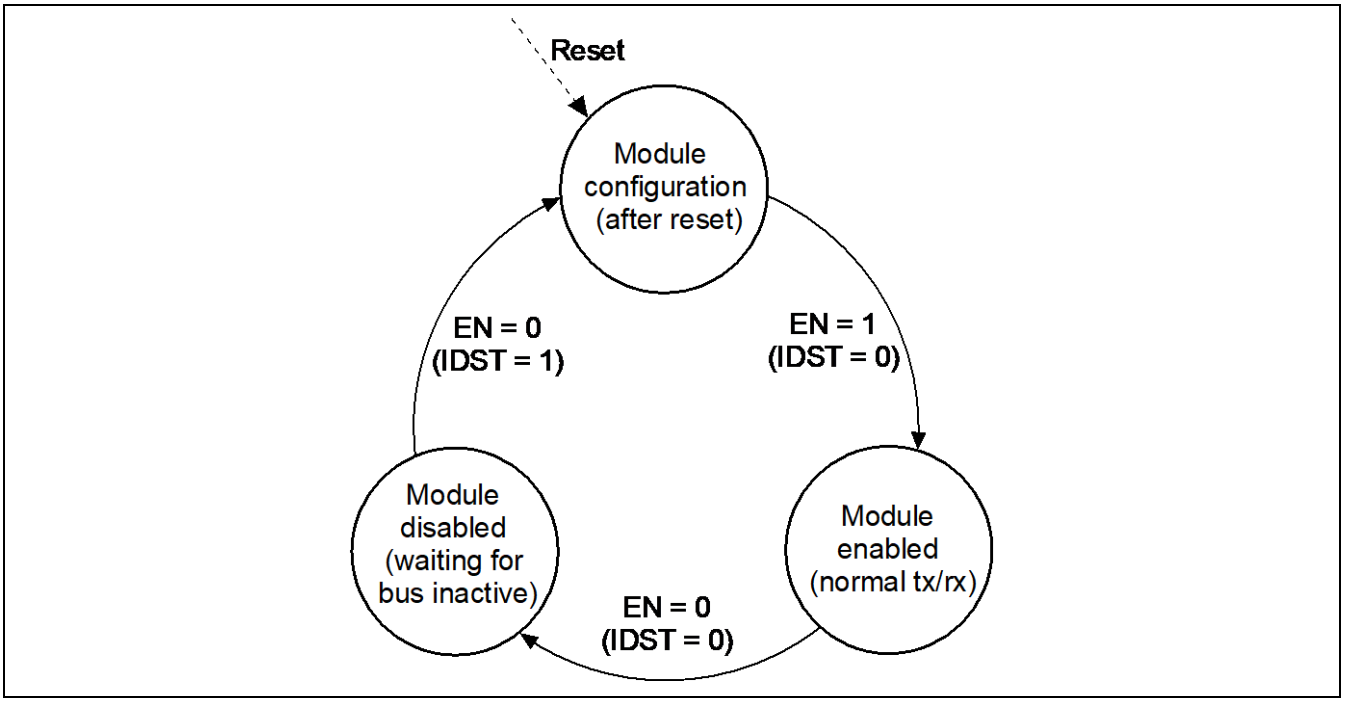



**Figure 82.37 Operating Modes**

#### (1) Configuration mode

This mode is entered after the module is released from reset. All required configuration fields in the control register should be defined in this mode, before the SSI module is enabled by setting the EN bit.

Setting the EN bit causes the module to enter the module enabled mode.

#### (2) Module enabled mode

Operation of the module in this mode is dependent on the operating mode selected. For details, refer to Section 82.3.3 Transmit Operation and Section 82.3.4 Receive Operation.

### 82.3.3 Transmit Operation

Transmission can be controlled either by DMA or interrupt.

DMA control is preferred to reduce the processor load. In DMA control mode, the processor will only receive interrupts if there is an underflow or overflow of data or the DMAC has finished its transfer.

The alternative method is using the interrupts that the SSI module generates to supply data as required. This mode has a higher interrupt load as the module is only double buffered and will require data to be written at least every system word period.

When disabling the module, the SSI clock* must remain present until the SSI module is in idle state, indicated by the SSISR.IIRQ bit.

Figure 82.38 shows the transmit operation in DMA control mode, and Figure 82.39 shows the transmit operation in interrupt control mode.

Note: * Input clock from the SSI_SCK pin when SCKD = 0.
Input clock from the CLK_FS pin when SCKD = 1.

**(1) Transmission using DMA controller**

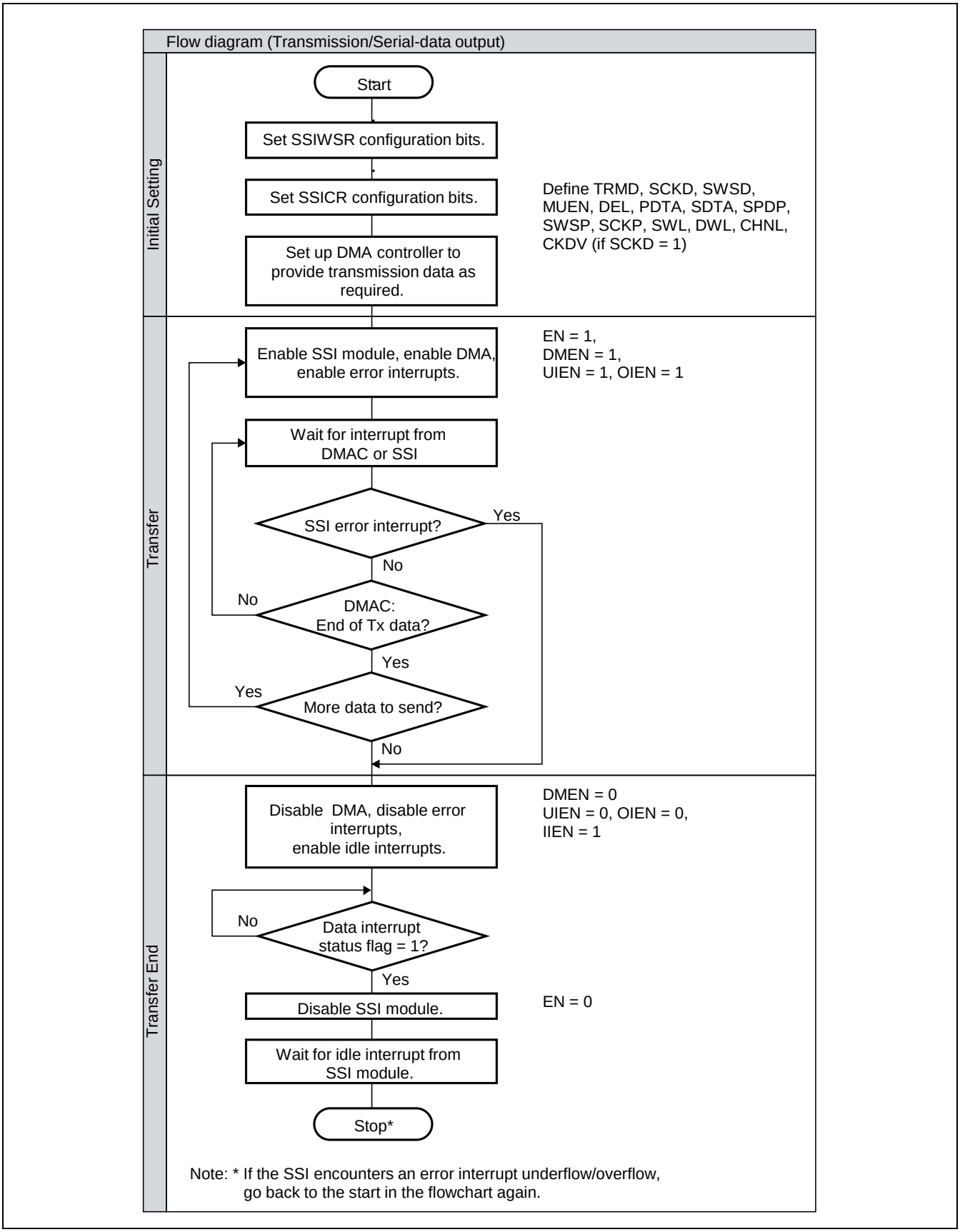



**Figure 82.38 Transmission Using DMA Controller**

**(2) Transmission using interrupt data flow control**

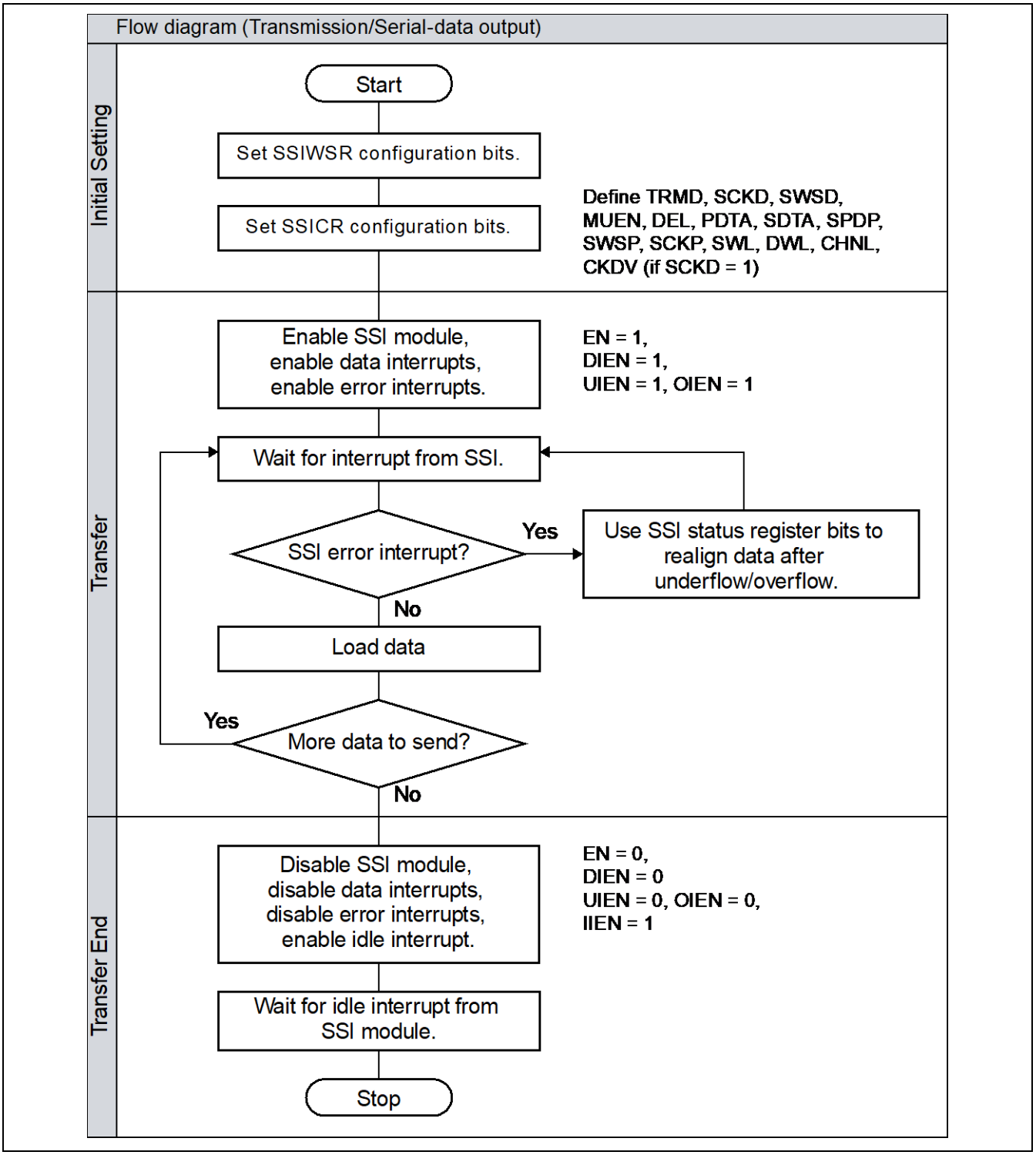



**Figure 82.39 Transmission Using Interrupt Data Flow Control**

### 82.3.4 Receive Operation

Like transmission, reception can be controlled either by DMA or interrupt.

Figure 82.40 and Figure 82.41 show the flow of operation.

When disabling the SSI module, the SSI clock* must be kept supplied until the SSISR.IIRQ bit is in idle state.

Note: * Input clock from the SSI_SCK pin when SCKD = 0.
Input clock from the CLK_FS pin when SCKD = 1.

**(1) Reception using DMA controller**

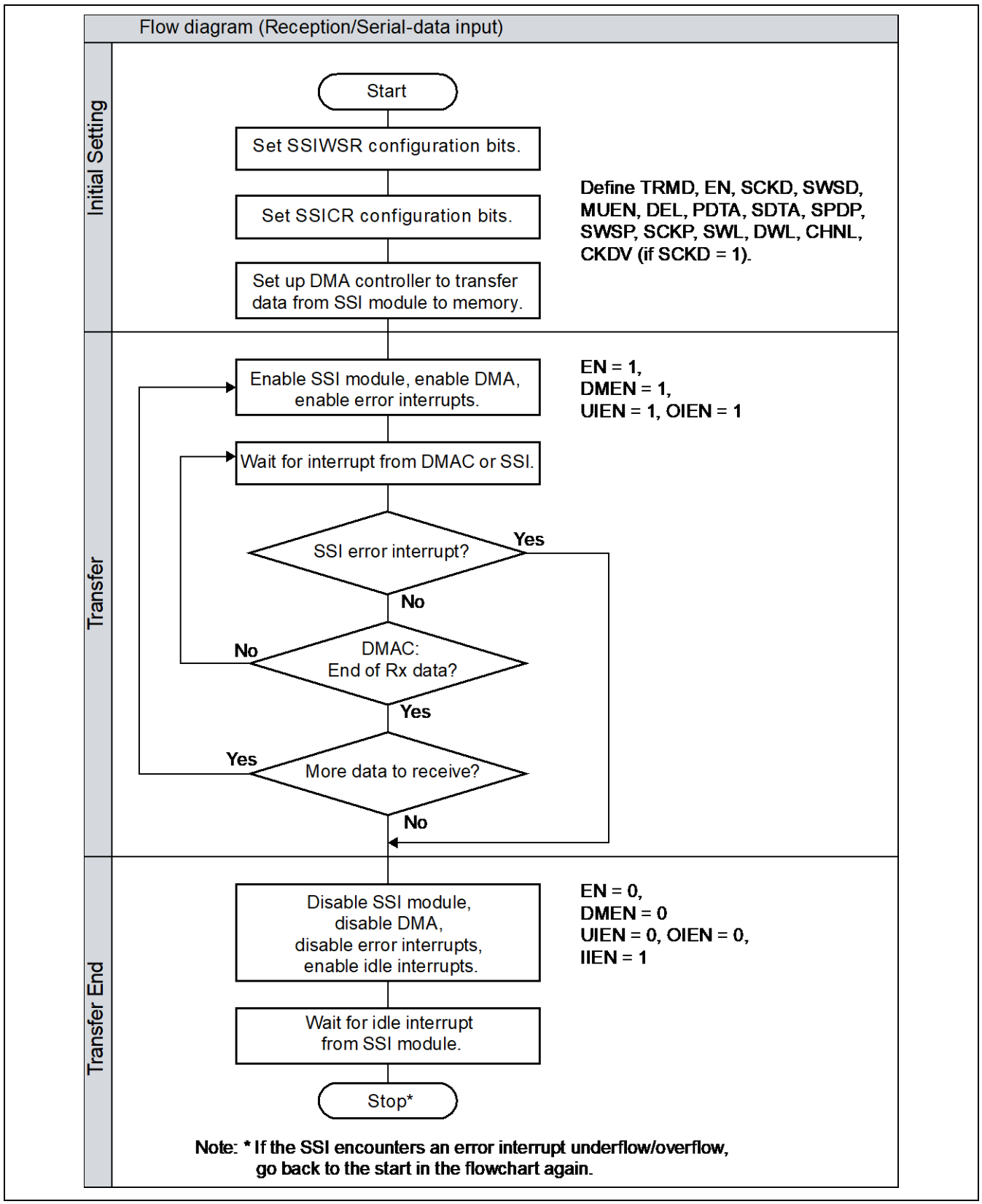



**Figure 82.40 Reception Using DMA Controller**

**(2) Reception using interrupt data flow control**

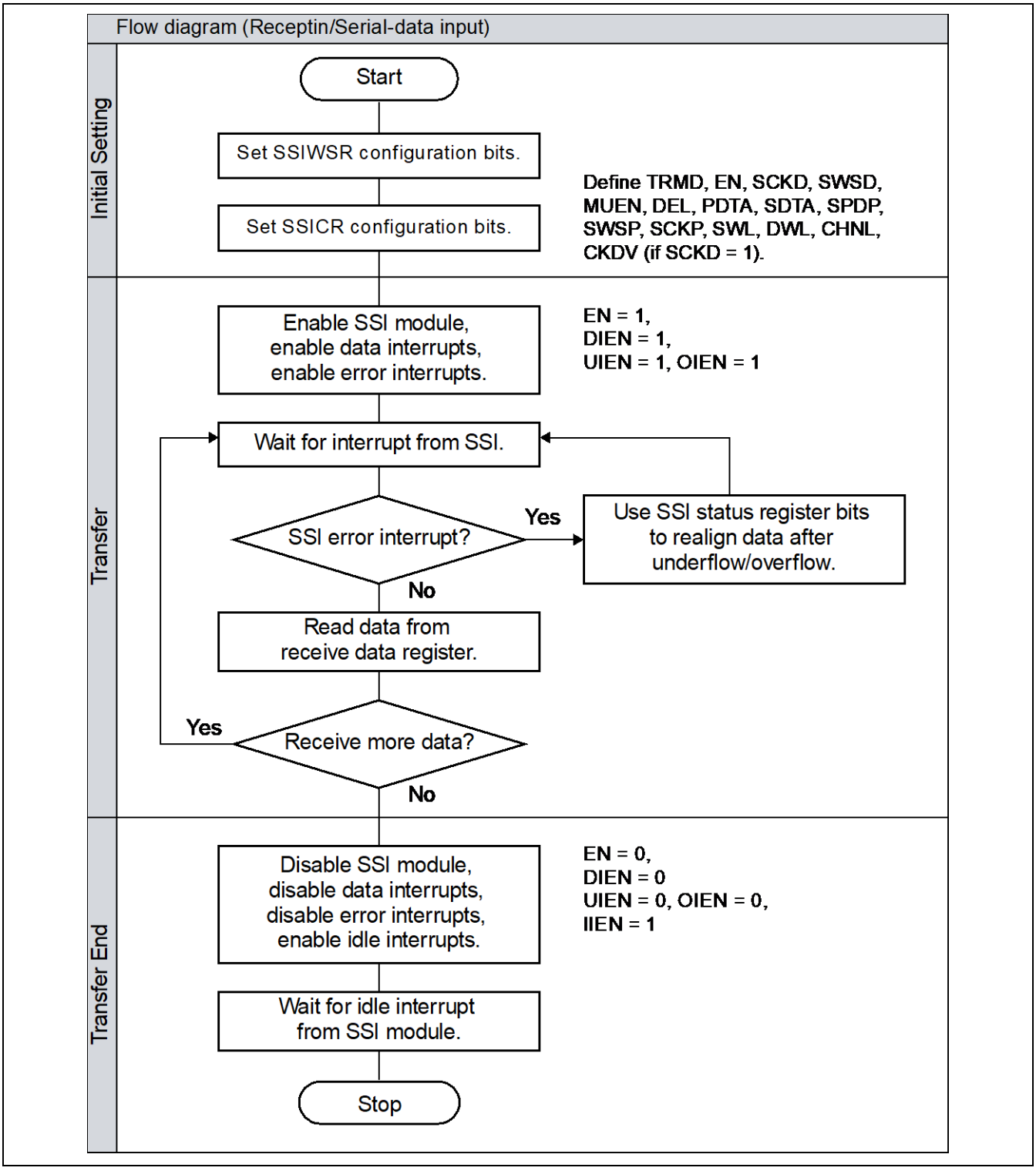



**Figure 82.41 Reception Using Interrupt Data Flow Control**

### 82.3.5 Serial Bit Clock Control

This function is used to control and select which clock is used for the serial bus interface.

If the serial clock direction is set to input (SSICR.SCKD = 0), the SSI module is in clock slave mode and the shift register uses the bit clock that was input to the SSI_SCK pin.

If the serial clock direction is set to output (SSICR.SCKD = 1), the SSI module is in clock master mode, and the shift register uses the bit clock that is input from the CLK_FS pin or the bit clock that is obtained by dividing the input from the CLK_FS pin. In the latter case, the clock input from the CLK_FS pin is divided by the ratio in the serial oversampling clock divide ratio (CKDV) in SSICR and used as the bit clock in the shift register.

In either case, the SSI_SCK pin output is the same as the bit clock.

## 82.4 Usage Notes

### 82.4.1 Limitations from Overflow during Receive DMA Operation

If an overflow occurs while the receive DMA is in operation, the module should be restarted. The receive buffer in the SSI consists of 32-bit registers that share the L and R channels. Therefore, data to be received at the L channel may sometimes be received at the R channel if an overflow occurs, for example, under the following condition: the control register (SSICR) has a 32-bit setting for both data word length (DWL2 to DWL0) and system word length (SWL2 to SWL).

If an overflow is confirmed with the overflow error interrupt or overflow error status flag (SSISR. OIRQ), write 0 to the EN and DMEN bits in SSICR to disable DMA in the SSI module, thus stopping the operation (In this case, the controller setting should also be stopped). After this, write 0 to the OIRQ bit in SSISR to clear the overflow status, set DMA again and restart the transfer.

### 82.4.2 Limitations on Slave Mode Operation

When this LSI is used in slave mode, in ending a data transfer process, data transfer on this LSI should be stopped (SSICR.EN = 0) before the input word selection signal (SSI_WS) is stopped.

In slave mode, data transfer is stopped by clearing the EN bit in SSICR (setting to stop transfer) and detecting the falling edge of the word selection signal (SSI_WS). If the input word selection signal is stopped first, the falling edge of the word selection signal cannot be detected and data transfer cannot be ended successfully.

In case of transmission at slave mode, it is necessary that WS signal is stable. Data transmission at slave mode should be started after more than two frames period since WS signal is entered and is stable.

In case of multiple SSI modules and case that one SSI works at master mode and the others do at slave mode, data transfer should be started after more than two frames period since registers of SSI at master mode are set.

### 82.4.3 Limitations on Changes to Settings

The SSI_SCK and SSI_WS signals are not guaranteed immediately after changes of the WS mode bit in the WS Mode Register (SSIWSR) and configuration bits in the control register (SSICR). Settings must not be changed dynamically if this would affect any connected device.

### 82.4.4 Stopping or Resuming Transmission

Follow the procedure below to stop or resume transmission.

**(1) Stopping transmission:**

1. Set the DMEN bit in SSICR to 0 for stopping transmission.
2. Wait for SSISR.DIRQ = 1 by using polling or interrupts.
3. Set the EN bit in SSICR to 0 to stop transmission.
4. Check that SSISR.IDST is 1.

**(2) Resuming transmission:**

Set DMEN and EN bits in SSICR to 1 for resuming transmission (The DMEN bit should be set to 1 at the same time of or before setting the EN bit).

### 82.4.5 Stable period of SCK and WS

The period during which SCK signal and WS signal stabilizes is shown below. It is assumed that the clock supplied from ADG is stable.

SCK signal (when SSI is used in the master mode)

1. Set the SSICR (exclude EN bit) register.
2. Wait count end to setting value of SSICR.CKDV (counted by CLK_FS).
3. SCK signal is stable.

WS signal (when SSI is used in the master mode)

1. Set the SSIWSR and SSICR (exclude EN bit) register.
2. Wait 2 frames.
3. WS signal is stable.

### 82.4.6 About CLK_FS input clock

For details on generation and selection of master clock signal, see section 83. Audio Clock Generator (ADG).

## 82.5 Safety Mechanisms

This module has no Mechanism of FuSa.

# 83. Audio Clock Generator (ADG)

## 83.1 Overview

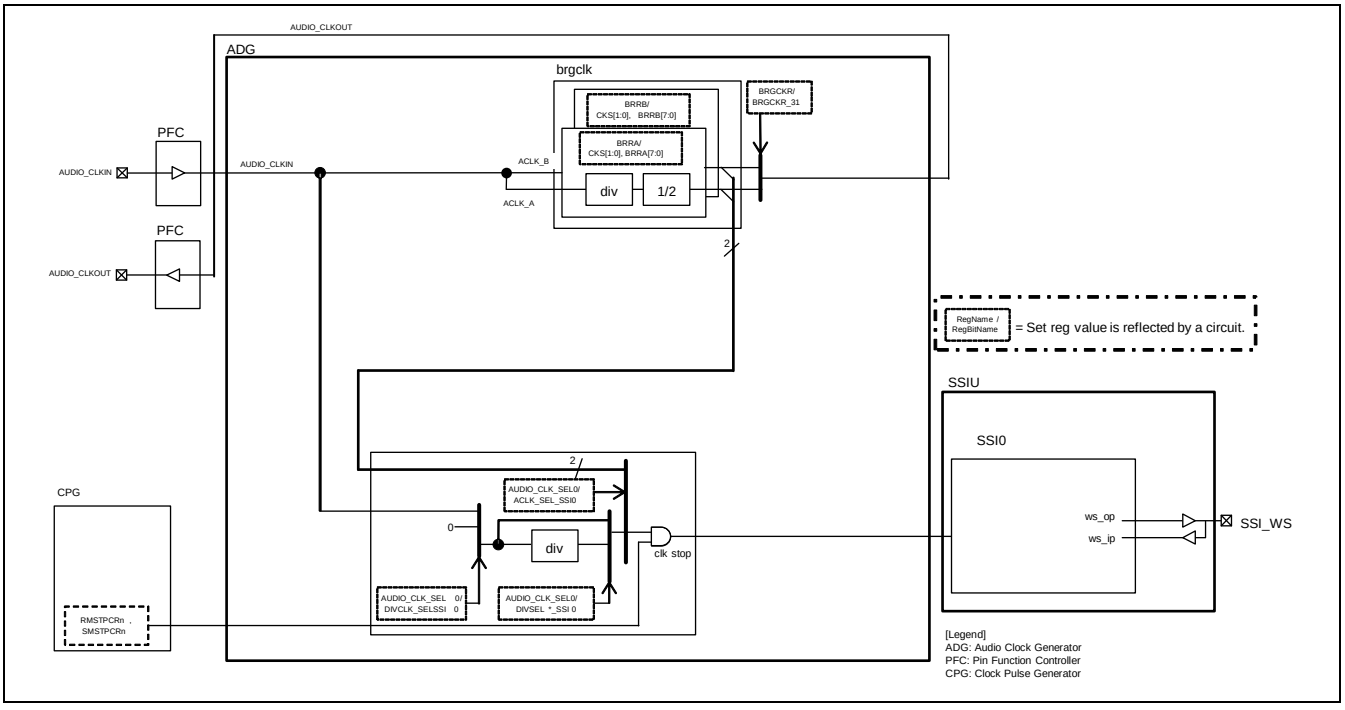
The audio clock generator (ADG) selects and supplies the necessary clock for the SSIU module. It also divides the frequency of the selected clock and sends it outside the chip.

### 83.1.1 Features

- The clock signal from the AUDIO_CLKIN pin supplies it to the SSIU module.
- The frequency of the clock signals from the AUDIO_CLKIN pins can be divided before use.
- The divided clock can be output through the AUDIO_CLKOUT pin.

Note: The presence or absence of pins depends on the product. For details, see Table 83.1.

### 83.1.2 Block Diagram




n = 29

**Figure 83.1** Block Diagram

### 83.1.3 External Pins

**Table 83.1 Pin Configuration**

| Pin Name | I/O | Function |
|---|---|---|
| AUDIO_CLKIN | Input | AUDIO CLOCK A |
| AUDIO_CLKOUT | Output | AUDIO CLOCK OUT |

### 83.1.4 Register Configuration

Table 83.2 shows the register configuration. The base address is H'EC5A 0000. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined.

**Table 83.2 Register Configuration**

| Name | Abbreviation | R/W | Address | Initial Value | Access size |
|---|---|---|---|---|---|
| BRGA Baud Rate Setting Register | BRRA | R/W | H'000 | H'0000 00FF | 32 |
| BRGB Baud Rate Setting Register | BRRB | R/W | H'004 | H'0000 00FF | 32 |
| BRG Clock Select Register | BRGCKR | R/W | H'008 | H'2300 0000 | 32 |
| Audio Clock Select Register 0 | AUDIO_CLK_SEL0 | R/W | H'00C | H'0000 0000 | 32 |

Note: * Some restrictions apply.

## 83.2 Register Description

[Legend for Register Description]

Initial value: Register value after a reset.

—: Undefined value

R/W: Bit or field is readable and writable. The written value can be read.

R/WC1: The bit or field is readable and writable. Writing 1 to the bit initializes the bit.

All registers are accessed in longword units.

### 83.2.1 BRGA Baud Rate Setting Register (BRRA)

Function: BRRA is a 32-bit readable/writable register that specifies the baud rate for ACLK_A (see Table 83.3).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CKS[1:0] | | BRRA[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | — | All 0 | R | Reserved<br>The initial value is always read from these bits. The write value should always be the initial value. |
| 9, 8 | CKS[1:0] | H'0 | R/W | These bits specify the clock source for the on-chip baud rate generator.<br>B'00: ACLK_A<br>B'01: ACLK_A/4<br>B'10: ACLK_A/16<br>B'11: ACLK_A/64 |
| 7 to 0 | BRRA[7:0] | H'FF | R/W | These bits specify the division ratio.<br>For details, see Table 83.3. |

**Table 83.3 Division Ratio for BRGA**

| BRGA Operating Clock (CKS[1], CKS[0]) | Division Ratio BRRA (N = 0 to 255) | Calculating Formula |
|---|---|---|
| ACLK_A | 1/2, 1/4, 1/6, ..., 1/512 | 1 / (2(N + 1)) |
| ACLK_A/4 | 1/8, 1/16, 1/24, ..., 1/2048 | 1 / (8(N + 1)) |
| ACLK_A/16 | 1/32, 1/64, 1/96, ..., 1/8192 | 1 / (32(N + 1)) |
| ACLK_A/64 | 1/128, 1/256, 1/384, ..., 1/32768 | 1 / (128(N + 1)) |

Note: Use the output of BRGA in less than 25 MHz.

### 83.2.2 BRGB Baud Rate Setting Register (BRRB)

Function: BRRB is a 32-bit readable/writable register that specifies the baud rate for ACLK_B (see Table 83.4).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CKS[1:0] | | BRRB[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | — | All 0 | R | Reserved<br>The initial value is always read from these bits. The write value should always be the initial value. |
| 9, 8 | CKS[1:0] | H'0 | R/W | These bits specify the clock source for the on-chip baud rate generator.<br>B'00: ACLK_B<br>B'01: ACLK_B/4<br>B'10: ACLK_B/16<br>B'11: ACLK_B/64 |
| 7 to 0 | BRRB[7:0] | H'FF | R/W | These bits specify the division ratio.<br>For details, see Table 83.4. |

**Table 83.4 Division Ratio for BRGB**

| BRGB Operating Clock (CKS[1], CKS[0]) | Division Ratio BRRB (N = 0 to 255) | Calculating Formula |
|---|---|---|
| ACLK_B | 1/2, 1/4, 1/6, ..., 1/512 | 1 / (2(N + 1)) |
| ACLK_B/4 | 1/8, 1/16, 1/24, ..., 1/2048 | 1 / (8(N + 1)) |
| ACLK_B/16 | 1/32, 1/64, 1/96, ..., 1/8192 | 1 / (32(N + 1)) |
| ACLK_B/64 | 1/128, 1/256, 1/384, ..., 1/32768 | 1 / (128(N + 1)) |

Note: Use the output of BRGB in less than 25 MHz.

### 83.2.3 BRG Clock Select Register (BRGCKR)

Function: BRGCKR selects the clocks input to and output from the ADG.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BRGCKR_31 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | BRGCKR_31 | B'0 | R/W | This bit selects the clock signal output to the AUDIO_CLKOUT or external pin.<br><br>0: BRGA output clock<br>1: BRGB output clock |
| 30 to 23 | — | H'46 | R | Reserved<br>The initial value is always read from these bits. The write value should always be the initial value. |
| 22 to 0 | — | All 0 | R | Reserved<br>The initial value is always read from these bits. The write value should always be the initial value. |

### 83.2.4 Audio Clock Select Register 0 (AUDIO_CLK_SEL0)

Function: AUDIO_CLK_SEL0 selects the clocks for the SSIU (for SSI0).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | — | | — | | — | | — | | — | | — | | — | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | — | | — | | — | | DIVSEL_SSI0 [1:0] | | ACLK_SEL_ SSI0[1:0] | | DIVSEL2 _SSI0 | DIVCLK_SEL_SSI0 [2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>The initial value is always read from these bits. The write value should always be the initial value. |
| 7, 6 | DIVSEL_SSI0[1:0] | H'0 | R/W | SSI0 Frequency Divider Select<br>[3], [7:6]:<br>ACLK_SEL_SSI0[1:0] = B'00<br>B'000: Not divided<br>B'001: Divided by 2<br>B'010: Divided by 4<br>B'011: Divided by 8<br>B'100: Divided by 16<br>B'101: Divided by 32<br>B'110, B'111: Fixed at 0 |
| 5, 4 | ACLK_SEL_SSI0[1:0] | H'0 | R/W | SSI0 Clock Select<br>B'00: DIVCLK<br>B'01: BRGA output clock<br>B'10: BRGB output clock<br>Others: Setting Prohibited |
| 3 | DIVSEL2_SSI0 | B’0 | R/W | See Bit Description of DIVSEL_SSI0[1:0] |
| 2 to 0 | DIVCLK_SEL_SSI0 [2:0] | H'0 | R/W | SSI0 Clock Select<br>B'000: Fixed at 0<br>B'001: AUDIO_CLKIN<br>Others: Setting Prohibited |

## 83.3 Operation

The following sections give examples of setting the ADG registers .

### 83.3.1 Supplying SSI with Audio Clock

The following procedure can be used to divide the external clock (AUDIO_CLKIN: 25 MHz) by d'2 and supply the resulting BRGB clock (12.500 MHz) to SSI0.

If the procedure is applied before starting SSI0, steps 1 to 4 can be executed in any order.

1. Set BRRB so that the frequency division ratio should be d'2.
2. Select external clock (AUDIO_CLKIN) as the clock to be input to BRGB using BRGCKR.
3. Select the BRGB output clock as SSI0 clock using AUDIO_CLK_SEL0.
4. Release SSI0 from the module standby state using MSTPCR29 of the CPG module.

**Table 83.5 Setting Example 1**

| Register Name | Address | Setting Value | Description |
|---|---|---|---|
| BRRB | H'EC5A 0004 | H'0000 0000 | Divided by d'2 |
| BRGCKR | H'EC5A 0008 | H'2300 0000 | Selects AUDIO_CLKIN as the clock to be input to BRGB. |
| AUDIO_CLK_SEL0 | H'EC5A 000C | H'0000 0020 | Selects the BRGB output clock as the clock to be output to SSI0. |
| MSTPCR29 | H'E615 2D74 | Set bits 27, 26. | Releases the SSI0 from the module standby state. |

Note: * CPG register is enabled by writing the inverse of the value to CPGWPR register.

## 83.4 Usage Notes

### 83.4.1 Note on using AUDIO_CLK and Input/Output timing signal in stable condition

AUDIO_CLK and Input/Output timing signal should be in a stable state before starting transfer of related audio modules. When Input/Output timing signal counts the number of divider register setting, it becomes stable.

## 83.5 Safety Mechanisms

This module has no Mechanism of FuSa.

# 100. Ethernet（EtherAVB-IF, gPTP, RGMII）

Timer Unit module is a module composed of EtherAVB-IF, gPTP, and RGMII.

Refer to following section for the detailed function.

- EtherAVB-IF :100.1 EtherAVB-IF
- gPTP :100.2 gPTP
- RGMII:100.3 RGMII

# 100.1 EthernetAVB-IF

## 100.1.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The EthernetAVB-IF includes an Ethernet controller (E-MAC) that conforms to the definition of the MAC (Media Access Control) layer for Ethernet in the IEEE 802.3 standard.

When connected with a physical-layer LSI chip (PHY-LSI) that complies with the standard, the E-MAC is able to transmit and receive Ethernet (IEEE 802.3) frames. The E-MAC has a single MAC layer interface.

The EthernetAVB-IF has a dedicated direct memory access controller (AVB-DMAC) for transferring transmitted Ethernet frames to and received Ethernet frames from respective storage areas in the User RAM(URAM) at high speed.

The AVB-DMAC is compliant with the following three standards formulated for IEEE 802.1BA: the IEEE 802.1AS timing and synchronization protocol, the IEEE 802.1Qav real-time transfer, and the IEEE 802.1Qat stream reservation protocol.

In this section, URAM refers to the local RAM and external memory for the SoC.

### 100.1.1.1 Features

Table 100.1 lists the specifications of the EthernetAVB-IF module.

**Table 100.1 Specifications (Common Functions)**

| Item | Description |
|---|---|
| Protocol | Flow control conforming with the IEEE 802.3x standard |
| Data transmission and reception | Transmission and reception of Ethernet (IEEE 802.3) frames |
| Transfer speed | Supports transfer at 100 and 1000 Mbps |
| Mode | Full-duplex mode |
| Interface | Supports the RGMII (Reduced Gigabit Media Independent Interface), MII Interface |
| Summary of the EthernetAVB-IF function | An intelligent frame separation DMAC (AVB-DMAC) conforming with the following standards stipulated for IEEE 802.1BA:<br>IEEE 802.1AS (time synchronization protocol)<br>IEEE 802.1Qav (real-time transfer)<br>IEEE 1722 (AVTP presentation timestamp)<br>IEEE 802.1Qat is supported by software.<br>Descriptor management system<br>Identification and sorting of frame data, and extraction and gathering of valid data<br>Controllable interrupt frequency (reducing the load on the CPU) |
| Transmit/Receive FIFO | For transmission : 16Kbytes<br>For reception : 8Kbytes |
| Max frame size | For transmission : 2047bytes<br>For reception : 4092bytes |

| Item | Description |
|---|---|
| Magic Packet™ | Detection of Magic Packets™* and output of a detected signal |

Note: * Magic Packet™ is a trademark of Advanced Micro Devices, Inc.

**Table 100.2 Specifications (Functions by Product)**

| Product name | R-Car V4M |
|---|---|
| Number of Units | 3 |
| Name | AVBn<br>(n = 0,1,2) |
| UDP/IP support | Support |

### 100.1.1.2 Block Diagram

Figure 100.1 are block diagram of the EthernetAVB-IF.

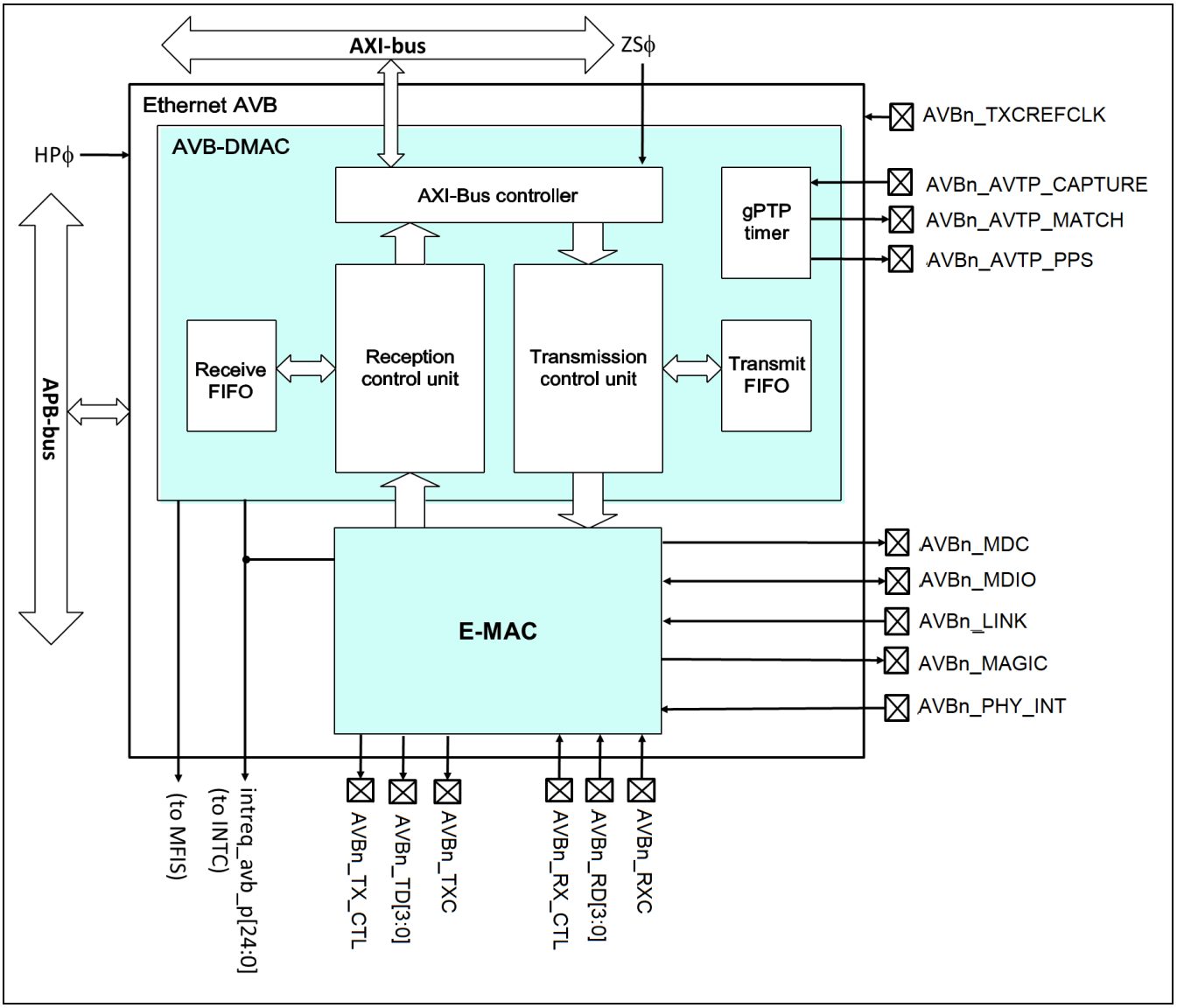



**Figure 100.1 Block Diagram of EthernetAVB-IF**

**Table 100.3 Clock Specification**

| Name | Function | Clock |
|---|---|---|
| ZSΦ | AXI bus clock | S0D4Φ |
| HPΦ | High Speed Peripheral clock | S0D8_HSCΦ |

### 100.1.1.3 External Pins

This lists the pins for use by EthernetAVB-IF.

**Table 100.4 Pin Configuration3**

| Name | I/O | Function |
|---|---|---|
| AVBn_TD[3:0] | O | Transmit data signal |
| AVBn_TX_CTL | O | Transmit control signal |
| AVBn_RD[3:0] | I | Receive data signal |
| AVBn_RX_CTL | I | Receive control signal |
| AVBn_RXC | I | Receive clock signal |
| AVBn_TXC | O | transmit clock signal |
| AVBn_TXCREFCLK | I | reference clock signal |
| AVBn_MDC | O | Management information transfer clock signal |
| AVBn_MDIO | I/O | Management information transmit/receive data |
| AVBn_LINK | I | Link status signal |
| AVBn_MAGIC | O | Magic packet signal |
| AVBn_PHY_INT | I | PHY interrupt signal |
| AVBn_AVTP_CAPTURE | I | AVTP capture signal |
| AVBn_AVTP_MATCH | O | AVTP match signal |
| AVBn_AVTP_PPS | O | AVTP Pulse-per-second signal |
| AVBn_MII_TXC | I | 25MHz transmit clock |
| AVBn_MII_RXC | I | 25MHz receive clock |
| AVBn_MII_TX_EN | O | Transmit enable |
| AVBn_MII_TD[3:0] | O | Transmit data |
| AVBn_MII_TX_ER | O | Transmit coding error |
| AVBn_MII_RX_DV | I | Receive data valid |
| AVBn_MII_RD[3:0] | I | Receive data |
| AVBn_MII_RX_ER | I | Receive error |
| AVBn_MII_CRS | I | Carrier sense |
| AVBn_MII_COL | I | Collision detected |

### 100.1.1.4 Register Configuration

Table 100.6 and Table 100.7 lists the EthernetAVB-IF related registers and their configurations.

**Table 100.5 Base address of EthernetAVB-IF- Registers**

| Ch | Base Address |
|---|---|
| AVB0 | H'E680 0000 |
| AVB1 | H'E681 0000 |
| AVB2 | H'E682 0000 |

**Table 100.6 Configurations of AVB-DMAC-related Registers**

| Name | Abbreviation*1 | R/W | Offset Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| AVB-DMAC mode register | AVBnCCC | R/W | H'0000 | H'0000 0000 | 32 |
| Descriptor base address table register | AVBnDBAT | R/W | H'0004 | H'0000 0000 | 32 |
| Descriptor base address load request register | AVBnDLR | R/W | H'0008 | H'003F FFFF | 32 |
| AVB-DMAC status register | AVBnCSR | R | H'000C | H'0000 0001 | 32 |
| Current descriptor address register q (q = 0 to 21) | AVBnCDARq | R | H'0010 + q × 4 | H'0000 0000 | 32 |
| Error status register | AVBnESR | R | H'0088 | H'0000 0000 | 32 |
| AVB-DMAC Product Specific Register | AVBnAPSR | R/W | H'008C | H'0000 0000 | 32 |
| Receive configuration register | AVBnRCR | R/W | H'0090 | H'1800 0000 | 32 |
| Receive queue configuration register i (i = 0 to 4) | AVBnRQCi | R/W | H'0094 + i × 4 | H'0000 0000 | 32 |
| Receive padding configuration register | AVBnRPC | R/W | H'00B0 | H'0000 0100 | 32 |
| Reception Truncation Configuration register | AVBnRTC | R/W | H'00B4 | H'0FFC 0FFC | 32 |
| Unread frame counter warning level register | AVBnUFCW | R/W | H'00BC | H'0000 0000 | 32 |
| Unread frame counter stop level register | AVBnUFCS | R/W | H'00C0 | H'0000 0000 | 32 |
| Unread frame counter register i (i = 0 to 4) | AVBnUFCVi | R | H'00C4 + i × 4 | H'0000 0000 | 32 |
| Unread frame counter decrement register i (i = 0 to 4) | AVBnUFCDi | R/W | H'00E0 + i × 4 | H'0000 0000 | 32 |
| Separation filter offset register | AVBnSFO | R/W | H'00FC | H'0000 0000 | 32 |
| Separation filter pattern register i (i = 0 to 31) | AVBnSFPi | R/W | H'0100 + i × 4 | H'0000 0000 | 32 |
| Separation Filter Value register i (i = 0 to 1) | AVBnSFVi | R/W | H'01B8 + i × 4 | H'0000 0000 | 32 |

| Name | Abbreviation*1 | R/W | Offset Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| Separation filter mask register i (i = 0, 1) | AVBnSFMi | R/W | H'01C0 + i × 4 | H'0000 0000 | 32 |
| Separation Filter Load register | AVBnSFL | R/W | H'01C8 | H'0000 001F | 32 |
| Payload CRC register | AVBnPCRC | R/W | H'01CC | H'0000 0000 | 32 |
| Current Incremental Address Register r (r = 0 to 17) | AVBnCIARr | R | H'0200 + r × 4 | H'0000 0000 | 32 |
| Last Incremental Address Register r (r=0 to 17) | AVBnLIARr | R | H'0280 + r × 4 | H'0000 0000 | 32 |
| Transmit configuration register | AVBnTGC | R/W | H'0300 | H'0022 2200 | 32 |
| Transmit configuration control register | AVBnTCCR | R/W | H'0304 | H'0000 0000 | 32 |
| Transmit status register | AVBnTSR | R | H'0308 | H'0000 0000 | 32 |
| E-MAC status FIFO Access register | AVBnMFA | R | H'030C | H'0000 0000 | 32 |
| Time stamp FIFO access register 0 | AVBnTFA0 | R | H'0310 | H'0000 0000 | 32 |
| Time stamp FIFO access register 1 | AVBnTFA1 | R | H'0314 | H'0000 0000 | 32 |
| Time stamp FIFO access register 2 | AVBnTFA2 | R | H'0318 | H'0000 0000 | 32 |
| Version and Release Register | AVBnVRR | R | H'031C | H'0000 E300 | 32 |
| CBS increment value register c (c= 0, 1) | AVBnCIVRc | R/W | H'0320 + c × 4 | H'0000 0001 | 32 |
| CBS decrement value register c (c = 0, 1) | AVBnCDVRc | R/W | H'0328 + c × 4 | H'FFFF FFFF | 32 |
| CBS upper limit register c (c = 0, 1) | AVBnCULc | R/W | H'0330 + c × 4 | H'7FFF FFFF | 32 |
| CBS lower limit register c (c = 0, 1) | AVBnCLLc | R/W | H'0338 + c × 4 | H'8000 0001 | 32 |
| Descriptor interrupt control register | AVBnDIC | R/W | H'0350 | H'0000 0000 | 32 |
| Descriptor interrupt status register | AVBnDIS | R/W | H'0354 | H'0000 0000 | 32 |
| Error interrupt control register | AVBnEIC | R/W | H'0358 | H'0000 0000 | 32 |
| Error interrupt status register | AVBnEIS | R/W | H'035C | H'0000 0000 | 32 |
| Receive interrupt control register 0 | AVBnRIC0 | R/W | H'0360 | H'0000 0000 | 32 |
| Receive interrupt status register 0 | AVBnRIS0 | R/W | H'0364 | H'0000 0000 | 32 |
| Receive interrupt control register 1 | AVBnRIC1 | R/W | H'0368 | H'0000 0000 | 32 |
| Receive interrupt status register 1 | AVBnRIS1 | R/W | H'036C | H'0000 0000 | 32 |
| Receive interrupt control register 2 | AVBnRIC2 | R/W | H'0370 | H'0000 0000 | 32 |
| Receive interrupt status register 2 | AVBnRIS2 | R/W | H'0374 | H'0000 0000 | 32 |
| Transmit interrupt control register | AVBnTIC | R/W | H'0378 | H'0000 0000 | 32 |
| Transmit interrupt status register | AVBnTIS | R/W | H'037C | H'0000 0000 | 32 |
| Interrupt summary status register | AVBnISS | R | H'0380 | H'0000 0000 | 32 |
| Common Interrupt Enable register | AVBnCIE | R/W | H'0384 | H'0000 0000 | 32 |
| Reception Interrupt Control register 3 | AVBnRIC3 | R/W | H'0388 | H'0000 0000 | 32 |
| Reception Interrupt Status register 3 | AVBnRIS3 | R/W | H'038C | H'0000 0000 | 32 |

| Name | Abbreviation*1 | R/W | Offset Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| Descriptor Interrupt Line selection register | AVBnDIL | R/W | H'0440 | H'0000 0000 | 32 |
| Error Interrupt Line selection register | AVBnEIL | R/W | H'0444 | H'0000 0000 | 32 |
| Transmission Interrupt Line selection register | AVBnTIL | R/W | H'0448 | H'0000 0000 | 32 |
| Descriptor Interrupt Enable register | AVBnDIE | R/W | H'0450 | H'0000 0000 | 32 |
| Descriptor Interrupt Disable register | AVBnDID | R/W | H'0454 | H'0000 0000 | 32 |
| Error Interrupt Enable register | AVBnEIE | R/W | H'0458 | H'0000 0000 | 32 |
| Error Interrupt Disable register | AVBnEID | R/W | H'045C | H'0000 0000 | 32 |
| Reception Interrupt Enable register 0 | AVBnRIE0 | R/W | H'0460 | H'0000 0000 | 32 |
| Reception Interrupt Disable register 0 | AVBnRID0 | R/W | H'0464 | H'0000 0000 | 32 |
| Reception Interrupt Enable register 1 | AVBnRIE1 | R/W | H'0468 | H'0000 0000 | 32 |
| Reception Interrupt Disable register 1 | AVBnRID1 | R/W | H'046C | H'0000 0000 | 32 |
| Reception Interrupt Enable register 2 | AVBnRIE2 | R/W | H'0470 | H'0000 0000 | 32 |
| Reception Interrupt Disable register 2 | AVBnRID2 | R/W | H'0474 | H'0000 0000 | 32 |
| Transmission Interrupt Enable register | AVBnTIE | R/W | H'0478 | H'0000 0000 | 32 |
| Transmission Interrupt Disable register | AVBnTID | R/W | H'047C | H'0000 0000 | 32 |
| Reception Interrupt Enable register 3 | AVBnRIE3 | R/W | H'0488 | H'0000 0000 | 32 |
| Reception Interrupt Disable register 3 | AVBnRID3 | R/W | H'048C | H'0000 0000 | 32 |

**Note: *1 In other section, Abbreviation is described without AVBn.**

**Table 100.7 Configuration of E-MAC-related Registers**

| Name | Abbreviation *1 | R/W | Offset Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| E-MAC mode register | AVBnECMR | R/W | H'0500 | H'0000 0000 | 32 |
| Receive frame length register | AVBnRFLR | R/W | H'0508 | H'0000 0000 | 32 |
| E-MAC status register | AVBnECSR | R/W | H'0510 | H'0000 0000 | 32 |
| E-MAC interrupt permission register | AVBnECSIPR | R/W | H'0518 | H'0000 0000 | 32 |
| PHY interface register | AVBnPIR | R/W | H'0520 | H'0000 000X | 32 |
| PHY Status Register | AVBnPSR | R | H'0528 | H'0000 0000 | 32 |
| PHY_INT Polarity Register | AVBnPIPR | R/W | H'052C | H'0000 0000 | 32 |
| Automatic PAUSE frame register | AVBnAPR | R/W | H'0554 | H'0000 0000 | 32 |
| Manual PAUSE frame register | AVBnMPR | R/W | H'0558 | H'0000 0000 | 32 |
| PAUSE frame transmit counter | AVBnPFTCR | R | H'055C | H'0000 0000 | 32 |
| PAUSE frame receive counter | AVBnPFRCR | R | H'0560 | H'0000 0000 | 32 |
| Automatic PAUSE frame retransmit count register | AVBnTPAUSER | R/W | H'0564 | H'0000 0000 | 32 |
| PAUSE frame transmit times counter | AVBnPFTTCR | R | H'0568 | H'0000 0000 | 32 |
| E-MAC Mode Register 2 | AVBnGECMR | R/W | H'05B0 | H'0000 0000 | 32 |
| E-MAC address high register | AVBnMAHR | R/W | H'05C0 | H'0000 0000 | 32 |
| E-MAC address low register | AVBnMALR | R/W | H'05C8 | H'0000 0000 | 32 |
| Transmit retry over counter register | AVBnTROCR | R/W | H'0700 | H'0000 0000 | 32 |
| CRC error frame receive counter register | AVBnCEFCR | R/W | H'0740 | H'0000 0000 | 32 |
| Frame receive error counter register | AVBnFRECR | R/W | H'0748 | H'0000 0000 | 32 |
| Too-short frame receive counter register | AVBnTSFRCR | R/W | H'0750 | H'0000 0000 | 32 |
| Too-long frame receive counter register | AVBnTLFRCR | R/W | H'0758 | H'0000 0000 | 32 |
| Residual-bit frame receive counter register | AVBnRFCR | R/W | H'0760 | H'0000 0000 | 32 |
| Multicast address frame receive counter register | AVBnMAFCR | R/W | H'0778 | H'0000 0000 | 32 |

**Note: *1 In other section, Abbreviation is described without AVBn.**

**Table 100.8 Configuration of UDP/IP support-related Registers**

| Name | Abbreviation *1 | R/W | Offset Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| Checksum Operating Mode Register | AVBnCSR0 | R/W | H'0800 | H'0000 0200 | 32 |
| Rx Frame Checksum Enable Register | AVBnCSR2 | R/W | H'0808 | H'0000 0000 | 32 |
| Unsupported Rx Frame Counter Register | AVBCSR30 | R | H'0824 | H'0000 0000 | 32 |
| Rx IPv4 Header checksum Error Counter Register | AVBnCSR31 | R | H'0828 | H'0000 0000 | 32 |
| Rx TCP/UDP/ICMP checksum Error Counter Register | AVBnCSR32 | R | H'082C | H'0000 0000 | 32 |
| UDP/IP Support Mode Register | AVBnUSMR | R/W | H'0A00 | H'0000 0000 | 32 |
| Separation Filter Offset Register 2 | AVBnSFO2 | R/W | H'0AE8 | H'0000 002A | 32 |
| IPv4 Source Address Filter Offset Register | AVBnI4SFO | R/W | H'0AEC | H'0000 001A | 32 |
| IPv4 Destination Address Filter Offset Register | AVBnI4DFO | R/W | H'0AF0 | H'0000 001E | 32 |
| IPv6 Source Address Filter Offset Register | AVBnI6SFO | R/W | H'0AF4 | H'0000 0016 | 32 |
| IPv6 Destination Address Filter Offset Register | AVBnI6DFO | R/W | H'0AF8 | H'0000 0026 | 32 |
| UDP Port Number Filter Offset Register | AVBnUPFO | R/W | H'0AFC | H'0000 0022 | 32 |
| IPv4 Source Address Filter Pattern Register i (i = 0 to 15) | AVBnI4SFPRi | R/W | H'0B00 + i x 4 | H'0000 0000 | 32 |
| IPv4 Destination Address Filter Pattern Register i (i = 0 to 15) | AVBnI4DFPRi | R/W | H'0C00 + i x 4 | H'0000 0000 | 32 |
| IPv6 Source Address Filter Pattern Register i (i = 0 to 63) | AVBnI6SFPRi | R/W | H'0D00 + i x 4 | H'0000 0000 | 32 |
| IPv6 Destination Address Filter Pattern Register i (i = 0 to 63) | AVBnI6DFPRi | R/W | H'0E00 + i x 4 | H'0000 0000 | 32 |
| UDP Port Number Filter Pattern Register I (i = 0 to 15) | AVBnUPFPRi | R/W | H'0F00 + i x 4 | H'0000 0000 | 32 |
| Separation Filter Value Register 2 | AVBnSFV2 | R/W | H'0F40 | H'0000 0000 | 32 |
| IPv4 Source Address Filter Value Register | AVBnI4SFVR | R/W | H'0F48 | H'0000 0000 | 32 |
| IPv4 Destination Address Filter Value Register | AVBnI4DFVR | R/W | H'0F4C | H'0000 0000 | 32 |
| IPv6 Source Address Filter Value Register i (i = 0 to 3) | AVBnI6SFVRi | R/W | H'0F50 + i x 4 | H'0000 0000 | 32 |

| Name | Abbreviation *1 | R/W | Offset Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| IPv6 Destination Address Filter Value Register i (i = 0 to 3) | AVBnI6DFVRi | R/W | H'0F60 + i x 4 | H'0000 0000 | 32 |
| UDP Port Number Filter Value Register | AVBnUPFVR | R/W | H'0F70 | H'0000 0000 | 32 |
| IPv4 Source Address Filter Mask Register | AVBnI4SFMR | R/W | H'0F7C | H'0000 0000 | 32 |
| IPv4 Destination Address Filter Mask Register | AVBnI4DFMR | R/W | H'0F80 | H'0000 0000 | 32 |
| IPv6 Source Address Filter Mask Register i (i = 0 or 3) | AVBnI6SFMRi | R/W | H'0F84 + i x 4 | H'0000 0000 | 32 |
| IPv6 Destination Address Filter Mask Register i (i = 0 or 3) | AVBnI6DFMRi | R/W | H'0F94 + i x 4 | H'0000 0000 | 32 |
| UDP Port Number Filter Mask Register | AVBnUPFMR | R/W | H'0FA4 | H'0000 0000 | 32 |
| Separation Filter Load register 2 | AVBnSFL2 | R/W | H'0FA8 | H'0000 001F | 32 |
| IPv4 Source Address Filter Load Register | AVBnI4SFLR | R/W | H'0FAC | H'0000 001F | 32 |
| IPv4 Destination Address Filter Load Register | AVBnI4DFLR | R/W | H'0FB0 | H'0000 001F | 32 |
| IPv6 Source Address Filter Load Register | AVBnI6SFLR | R/W | H'0FB4 | H'0000 001F | 32 |
| IPv6 Destination Address Filter Load Register | AVBnI6DFLR | R/W | H'0FB8 | H'0000 001F | 32 |
| UDP Port Number Filter Load Register | AVBnUPFLR | R/W | H'0FBC | H'0000 001F | 32 |

**Note: *1 In other section, Abbreviation is described without AVBn.**

#### 100.1.1.5 Connected Module

**Table 100.9 Connected module**

| Module name | Connected module name | Function of connected module |
|---|---|---|
| EthernetAVB-IF | AP-System Core | Access the Registers |
| | CPG | Output clocks |
| | PFC | Select the External pins |
| | Module Standby | Control to stop clocks |
| | Software Reset | Execute software reset |
| | INTC-SYS, INTC-RT | Control to interrupt |
| | DU | Output capture signal |

## 100.1.2 Register Description

### 100.1.2.1 AVB-DMAC Mode Register (CCC)

The CCC register specifies the operating mode of the AVB-DMAC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | FCE | LBME | — | — | — | — | — | — | CSEL[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

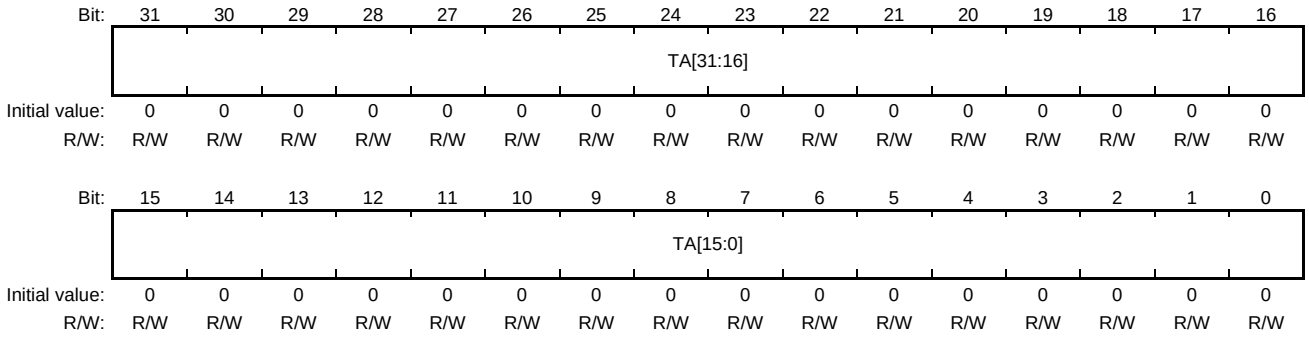
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DTSR | GAC | — | — | — | — | — | OPC[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 25 | FCE | 0 | R/W | Flow Control Enable<br>This bit enables the flow control support of E-MAC.<br>When flow control is enabled, the E-MAC gets informed about the Rx-FIFO level (Rx-FIFO fill level reached **RCR.RFCL**).<br>0: Flow control disabled<br>1: Flow control enabled |
| 24 | LBME | 0 | R/W | Loopback Mode Enable<br>This bit enables loopback mode.<br>In loopback mode, the transmission lines are internally connected to the reception lines. When loopback mode is to be used, the Ethernet transmission clock must be supplied to the RGMII interface. A received clock signal is not required. Writing to this bit is only possible when the current operating mode is configuration mode.<br>0: Normal operation<br>1: Loopback mode is enabled.<br>Note: Data for transmission are still output normally. To eliminate effects on external modules, pin control should be applied to block the output of data. |
| 23 to 18 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17, 16 | CSEL[1:0] | 00 | R/W | gPTP Clock Select<br>These bits select the clock source for the gPTP timer.<br>Writing to these bits is only possible when the current operating mode is configuration mode and CCC.GAC is 0b.<br>Writing to these bits is only possible when the current operating mode is Reset mode when writing CCC.GAC to 1b and CCC.OPC to 01b by same write access.<br>00: gPTP is disabled<br>01: High-speed peripheral bus clock<br>10 and 11: Setting prohibited |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 9 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 8 | DTSR | 0 | R/W | Data Transmission Suspend Request<br>This bit can suspend access to the URAM.<br>The access is suspended on completion of the transfer of the AXI transaction currently being transferred.<br>This function disables access to the URAM without affecting normal operation of the AVB-DMAC. Use this bit when exclusive control over the contents of the URAM is necessary, for example, in checking its integrity.<br>Note that the transmission and reception queues are not processed while access is suspended.<br>Change neither the AVB-DMAC settings nor the mode while access is suspended.<br>0: Normal operation<br>1: Requests suspension |
| 7 | GAC | 0 | R/W | Gptp Active in Config<br>This bit enables the gPTP support of EthernetAVB-IF in CONFIG mode. Function of gPTP support in OPERATION and STANDBY is not influenced by this bit. When gPTP support is active in CONFIG, **CCC.CSEL** defines the timer clock source.<br>The CPU can only write 1b to this bit if **CSR.OPS** is RESET when writing **CCC.OPC** to 01b by same write access.<br>0: Normal operation<br>1: gPTP support active in CONFIG mode |
| 6 to 2 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 1, 0 | OPC[1:0] | 00 | R/W | Operating Mode Configuration<br>These bits specify the operating mode.<br>For the operating modes, see section 100.1.3.1(1) Operating Modes.<br>Writing to these bits is possible in any of the operating modes, but should not be done after the application system has issued a Power Off request.<br>When **CCC.GAC** is 1b CPU should not write 11b to these bits.<br>[Changing condition]<br>These bits are set to 00b when RESET mode is entered due to 'enter power off' request.<br>00: Reset mode<br>01: Configuration mode<br>10: Operation mode<br>11: standby mode |

### 100.1.2.2 Descriptor Base Address Table Register (DBAT)

The DBAT register specifies the base address of the descriptor table in the URAM. For the structure of this table, see section 100.1.3.3 Descriptors. Writing to this bit is only possible when the current operating mode is configuration mode.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TA[31:0] | H'0000 0000 | R/W | Descriptor Base Table Address<br>Base address of the descriptor table in the URAM<br>Note: The setting of this bit must be a multiple of four (i.e. bit 0 and bit 1 must be set to 0). |

### 100.1.2.3 Descriptor Base Address Load Request Register (DLR)

The DLR register is used to issue a request to load the values from the current descriptor address register q (CDARq) for each queue to the descriptor base address table register (DBAT).

Setting a bit to 1 issues a request for loading the descriptor base address for the queue q. If transfer is currently in progress, loading is executed on completion of transfer for the current frame. Completion of loading leads to automatic setting of the corresponding bit to 0.

For the transmission queues, base address load requests are executed even while fetching is in progress (the transmit start request bit in the transmit configuration control register (TCCR.TSRQt) is 1). Therefore, be sure to check that fetching is not in progress before issuing a request.

Writing to a bit of this register is only possible when the current operating mode is operation mode. Only 1 can be written to this bit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | LBA21 | LBA20 | LBA19 | LBA18 | LBA17 | LBA16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LBA15 | LBA14 | LBA13 | LBA12 | LBA11 | LBA10 | LBA9 | LBA8 | LBA7 | LBA6 | LBA5 | LBA4 | LBA3 | LBA2 | LBA1 | LBA0 |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 22 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 21 | LBA21 | 1 | R/W | Base Address Load Request (Rx17: Stream 15)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 20 | LBA20 | 1 | R/W | Base Address Load Request (Rx16: Stream 14)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 19 | LBA19 | 1 | R/W | Base Address Load Request (Rx15: Stream 13)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 18 | LBA18 | 1 | R/W | Base Address Load Request (Rx14: Stream 12)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | LBA17 | 1 | R/W | Base Address Load Request (Rx13: Stream 11)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 16 | LBA16 | 1 | R/W | Base Address Load Request (Rx12: Stream 10)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 15 | LBA15 | 1 | R/W | Base Address Load Request (Rx11: Stream 9)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 14 | LBA14 | 1 | R/W | Base Address Load Request (Rx10: Stream 8)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 13 | LBA13 | 1 | R/W | Base Address Load Request (Rx9: Stream 7)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 12 | LBA12 | 1 | R/W | Base Address Load Request (Rx8: Stream 6)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 11 | LBA11 | 1 | R/W | Base Address Load Request (Rx7: Stream 5)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 10 | LBA10 | 1 | R/W | Base Address Load Request (Rx6: Stream 4)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 9 | LBA9 | 1 | R/W | Base Address Load Request (Rx5: Stream 3)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 8 | LBA8 | 1 | R/W | Base Address Load Request (Rx4: Stream 2)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | LBA7 | 1 | R/W | Base Address Load Request (Rx3: Stream 1)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 6 | LBA6 | 1 | R/W | Base Address Load Request (Rx2: Stream 0)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 5 | LBA5 | 1 | R/W | Base Address Load Request (Rx1: Network Control)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 4 | LBA4 | 1 | R/W | Base Address Load Request (Rx0: Best Effort)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 3 | LBA3 | 1 | R/W | Base Address Load Request (Tx3: Stream Class A)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 2 | LBA2 | 1 | R/W | Base Address Load Request (Tx2: Stream Class B)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 1 | LBA1 | 1 | R/W | Base Address Load Request (Tx1: Network Control)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |
| 0 | LBA0 | 1 | R/W | Base Address Load Request (Tx0: Best Effort)<br>0: No load request is issued.<br>1: When written: A request for loading the corresponding base address is issued.<br>When read: The given base address is being loaded. |

### 100.1.2.4 AVB-DMAC Status Register (CSR)

The CSR register is used to indicate the operating mode in which the AVB-DMAC is running and the individual communications states.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TDUO | RPO | TPO3 | TPO2 | TPO1 | TPO0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DTS | — | — | — | — | OPS[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 22 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 21 | TDUO | 0 | R | Transmission Descriptor Update On-going<br>This bit indicates that there is pending descriptor update for one or more transmit queues.<br>When this bit is 1b, there are descriptors of already or currently processed transmit storage elements not updated in URAM.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when all transmit queue related storage elements processed.<br>This bit is set to 1b when descriptor related status update procedure starts (descriptor update and register flagging).<br>0: No pending descriptor update<br>1: Pending descriptor update in URAM |
| 20 | RPO | 0 | R | Receive Process Status<br>This bit indicates whether a reception queue contains an unread received frame.<br>This bit being set to 1 indicates that a received frame is yet to be stored in the URAM.<br>0: Normal operation<br>[Clearing conditions]<br>— The current operating mode is not operation mode.<br>— Received frames in the reception FIFO all being stored in the URAM.<br>1: Reception is in progress.<br>[Setting condition]<br>— A received frame being stored in the reception FIFO (but not yet in the URAM). |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 19 | TPO3 | 0 | R | Transmit Process Status 3 (Stream Class A)<br>This bit indicates whether a class A stream is being transmitted.<br>This bit being set to 1 indicates that the AVB-DMAC is fetching data for transmission from the URAM, or the E-MAC is transmitting data.<br>0: Normal operation<br>[Clearing conditions]<br>— The current operating mode is not operation mode.<br>— Completion of transfer of all frames for transmission from the transmission FIFO and TCCR.TSRQ3 is 0.<br>1: Transmission is in progress.<br>[Setting condition]<br>— Transmission being started (by writing 1 to the transmit start request bit in the transmit configuration control register (TCCR.TSRQ3)). |
| 18 | TPO2 | 0 | R | Transmit Process Status 2 (Stream Class B)<br>This bit indicates whether a class B stream is being transmitted.<br>This bit being set to 1 indicates that the AVB-DMAC is fetching data for transmission from the URAM, or the E-MAC is transmitting data.<br>0: Normal operation<br>[Clearing conditions]<br>— The current operating mode is not operation mode.<br>— Completion of transfer of all frames for transmission from the transmission FIFO and TCCR.TSRQ2 is 0.<br>1: Transmission is in progress.<br>[Setting condition]<br>— Transmission being started (by writing 1 to the transmit start request bit in the transmit configuration control register (TCCR.TSRQ2)). |
| 17 | TPO1 | 0 | R | Transmit Process Status 1 (Network Control)<br>This bit indicates whether a network control is being transmitted.<br>This bit being set to 1 indicates that the AVB-DMAC is fetching data for transmission from the URAM, or the E-MAC is transmitting data.<br>0: Normal operation<br>[Clearing conditions]<br>— The current operating mode is not operation mode.<br>— Completion of transfer of all frames for transmission from the transmission FIFO and TCCR.TSRQ1 is 0.<br>1: Transmission is in progress.<br>[Setting condition]<br>— Transmission being started (by writing 1 to the transmit start request bit in the transmit configuration control register (TCCR.TSRQ1)). |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | TPO0 | 0 | R | Transmit Process Status 0 (Best Effort)<br>This bit indicates whether a best effort is being transmitted.<br>This bit being set to 1 indicates that the AVB-DMAC is fetching data for transmission from the URAM, or the E-MAC is transmitting data.<br>0: Normal operation<br>[Clearing conditions]<br>— The current operating mode is not operation mode.<br>— Completion of transfer of all frames for transmission from the transmission FIFO and TCCR.TSRQ0 is 0.<br>1: Transmission is in progress.<br>[Setting condition]<br>— Transmission being started (by writing 1 to the transmit start request bit in the transmit configuration control register (TCCR.TSRQ0)). |
| 15 to 9 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 8 | DTS | 0 | R | Data Transmission Suspended Status<br>This bit indicates whether access to the URAM is enabled.<br>0: Normal operation<br>[Clearing conditions]<br>— The data transmission suspend request bit in the AVB-DMAC mode register (CCC.DTSR) being 0.<br>1: Transmission is in progress.<br>[Setting condition]<br>— Access to the URAM not proceeding while the data transmission suspend request bit (CCC.DTSR) in the AVB-DMAC mode register (CCC) is 1 (if the URAM is being accessed, this bit is set to 1 on completion of access). |
| 7 to 4 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 3 to 0 | OPS[3:0] | 0001 | R | Operating Mode Status<br>These bits indicate the current operating mode.<br>For the operating modes, see section 100.1.3.1(1) Operating Modes.<br>0001: Reset mode<br>0010: Configuration mode<br>0100: Operation mode<br>1000: standby mode<br>Other settings are reserved. |

### 100.1.2.5 Current Descriptor Address Register q (CDARq) (q = 0 to 21)

The CDARq register indicates the current descriptor address.

CDAR0 to CDAR3 indicate the addresses of the current descriptors for the corresponding transmission queues while CDAR4 to CDAR21 indicate the addresses of the current descriptors for the corresponding reception queues.

If the operating mode is changed to operation mode, the contents of the register for the queue to be used are set in the descriptor base address table register (DBAT).

Also, when the descriptor base address load request register (DLR) issues a load request, the contents of the descriptor base address table register (DBAT) are set in to the corresponding CDAR registers.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CDA[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CDA[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CDA[31:0] | H'0000 0000 | R | Current Descriptor Address<br>The address of the current descriptors for the transmission queues.<br>Conditions for updating:<br>• These bits are set to 0 when the operating mode is not operation mode. This register is updated in response to processing of the descriptor for a queue. |

### 100.1.2.6 Error Status Register (ESR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | EIL | ET[3:0] | | | | — | — | — | EQN[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 12 | EIL | 0 | R | Error Information Lost<br>This bit indicates that error information detect by EthernetAVB-IF is lost, because the previous reported error has not been processed by CPU.<br>[Changing conditions]<br>— This bit is set to 0 when leaving operation mode.<br>— This bit is set to 0 when CPU writes 0 to EIS.QEF.<br>— This bit is set to 1 when the set condition of EIS.QEF is fulfilled while EIS.QEF is 1.<br>0: No loss of error information<br>1: Lost of error information detected |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 to 8 | ET[3:0] | 0000 | R | Error Type<br>These bits indicate details about the transfer stage which was handled when EthernetAVB-IF has detected an error.<br>When the fault is related to the read descriptor (ESR.ET = B'0000 or B'0010), CPU needs to correct the faulty descriptor before the related queue can continue processing. Because the queue halts at the faulty descriptor CDARq.CDA (q = ESR.EQN) is identifying the faulty descriptor directly.<br>When the fault is related to descriptor writing (ESR.ET = B'0001), CPU needs to recognize the not-updated or incorrectly updated descriptor in queue ESR.EQN. The write problem is not influencing how EthernetAVB-IF processes the descriptor chain.<br>When the fault is related to the Tx-buffer (ESR.ET = B'0011), CPU needs to clean-up the Tx-buffer to correct the buffer control structures.<br>All other errors are transient in nature and may be corrected by continuation of hardware or software operation; so there is no strong demand on CPU interaction.<br>Refer to section 100.1.3.2(2) Checking Integrity for details of error handling.<br>The CPU should only evaluate these bits when the EIS.QEF bit is 1.<br>[Changing condition]<br>— These bits are updated when the set condition of the EIS.QEF bit is fulfilled and the EIS.QEF bit is 0.<br>0000: Read descriptor from URAM<br>0001: Write descriptor to URAM<br>0010: Interpret data descriptor<br>0011: Tx-buffer is corrupted<br>0100: Read data from URAM<br>0101: Write data or timestamp to URAM<br>0110: Reading from Rx-FIFO<br>0111: Rx-FIFO is corrupted<br>1000: Frame size error during reception detected<br>1001: Frame size error during transmission detected<br>1010: Tx-buffer overflow<br>1011: AVTP FIFO error |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are read as 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 to 0 | EQN[4:0] | 00000 | R | Error Queue Number<br>These bits indicate the queue number which was handled when EthernetAVB-IF has detected an error.<br>A fault reported for ESR.EQN = 0 to 3 is related to transmit queue t (t = 0 to 3).<br>From ESR.EQN = 4, the fault is related to receive queue r (r = ESR.EQN – 4).<br>The CPU should only evaluate these bits when the EIS.QEF bit is 1. The CPU should not evaluate these bits when the ESR.ET bit is B'0011, B'0111 or B'1011.<br>[Changing condition]<br>— These bits are updated when the set condition of the EIS.QEF bit is fulfilled and the EIS.QEF bit is 0. |

### 100.1.2.7 AVB-DMAC Product Specific Register (APSR)

Writing to this bit is only possible when the current operating mode is configuration mode.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | RSEL | GPTPCLOCK | — | — | — | GPTPTIMERSOURCE | MIISELECT | — | — | — | — | ERREN[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | TDM | RDM | — | — | — | — | — | — | — | — | CMSW | — | ERRMOD | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | 0 | R/W | Reserved<br>These bits are read as 0. |
| 30 | RSEL | 0 | R/W | Reference Clock Selection for Transmission<br>0: Use AVBn_TXCREFCLK (n = own channel number)<br>1: Use AVB0_TXCREFCLK |
| 29 | GPTPCLOCK | 0 | R/W | Select clock source for GPTP synchronization<br>0: Original clock source<br>1: S0D4_HSC<br>This bit should be set to 1 when APSR.GPTPTIMERSOURCE is 1 |
| 28 to 26 | — | All 0 | R/W | Reserved<br>These bits are read as 0. |
| 25 | GPTPTIMERSOURCE | 0 | R/W | Select gptp_timer[77:0] source for GPTP synchronization<br>0: Original gptp_timer source<br>1: gptp_timer source from GPTP module<br>This bit should be set 1. |
| 24 | MIISELECT | 0 | R/W | MII clock selection<br>0: RGMII clock<br>1: MII clock<br>*This functional is used AVB0 and AVB1 only. Not support AVB2. |
| 23 to 20 | — | All 0 | R/W | Reserved<br>These bits are read as 0. |
| 19 to 16 | ERREN[3:0] | All 0 | R/W | Dummy Error Setting<br>This bit is functional safety use only. Keep initial value. |
| 15 | — | 0 | R/W | Reserved<br>These bits are read as 0. |
| 14 | TDM | 0 | R/W | Tx clock internal Delay Mode<br>This bit can add internal Tx clock delay typ 2.0ns*.<br>0: normal mode<br>1: delayed mode<br>*Refer to Electrical Characteristics for details. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 13 | RDM | 0 | R/W | Rx clock internal Delay Mode<br>This bit can add internal Rx clock delay typ 1.8ns*.<br>0: normal mode<br>1: delayed mode<br>*Refer to Electrical Characteristics for details. |
| 12 | | 0 | R/W | Reserved<br>These bits are read as 0. |
| 11 to 6 | — | All 0 | R/W | Reserved<br>These bits are read as 0. |
| 4 | CMSW | 0 | R/W | select width of internal compare match signal<br>0: 1clock period pulse<br>1: 4clock period pulse |
| 3 | — | 0 | R/W | Reserved<br>These bits are read as 0. |
| 2 | ERRMOD | 0 | R/W | Error injection mode<br>0: ERREN[3:0] invalid<br>1: ERREN[3:0] valid |
| 1 | — | 0 | R/W | Reserved<br>These bits are read as 0. |
| 0 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.8 Receive Configuration Register (RCR)

The RCR register is used to make settings related to reception for the AVB-DMAC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | RFCL[12:0] | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ETS2 | ETS0 | ESF[1:0] | | ENCF | EFFS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 28 to 16 | RFCL[12:0] | H'1800 | R/W | Receive FIFO Caution Level<br>These bits set the caution level for the reception FIFO and are used to maintain the priority order of the storage of received data and the fetching of data for transmission.<br>If the reception FIFO contains less data than this level, processing of both transmission and reception queues becomes pending.<br>If the reception FIFO contains more data than this level, only data in the reception queue are transferred, and processing of the transmission queue becomes pending.<br>Writing to this bit is only possible when the current operating mode is configuration mode.<br>Recommended value: H'1800<br>Notes:<br>• The setting of this bit must be a multiple of four (i.e. set RFCL[1:0] = B'00).<br>• In the case of this LSI chip, set these bits to H'1800. |
| 15 to 6 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 5 | ETS2 | 0 | R/W | Time Stamp Enable (Stream)<br>Enables the inclusion of time-stamp information in reception queues 2 to 17.<br>Writing to this bit is only possible when the current operating mode is configuration mode.<br>0: Time stamping is disabled.<br>1: Time stamping is enabled.<br>Recommended value: 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | ETS0 | 0 | R/W | Time Stamp Enable (Best Effort)<br>Enables the inclusion of time-stamp information in reception queue 0.<br>Writing to this bit is only possible when the current operating mode is configuration mode.<br>0: Time stamping is disabled.<br>1: Time stamping is enabled.<br>Recommended value: 0 |
| 3, 2 | ESF[1:0] | 00 | R/W | Stream Filtering Select<br>Settings for reception queues 2 to 17.<br>These bits select separation filtering for reception queues 2 to 17.<br>The queue-dependent separation filter can be used in combination with the identification of AVB stream frames.<br>When the value is B'00, filtering is disabled and frames from streams are processed in reception queue 0 (best effort).<br>When the value is B'01, the separation filter is enabled for both AVB stream frames and non-AVB stream frames; frames from non-matching streams are processed in reception queue 0 (best effort).<br>When the value is B'10, the separation filter is enabled for AVB stream frames; frames from non-matching streams are discarded.<br>When the value is B'11, the separation filter is enabled for AVB stream frames; frames from non-matching streams are processed in reception queue 0 (best effort).<br>For separation filtering, see section 100.1.3.4(1)(a) Separation Filtering.<br>Writing to this bit is only possible when the current operating mode is configuration mode.<br>00: Filtering is disabled. Frames are processed in queue 0 (best effort).<br>01: The filter for both AVB stream frames and non-AVB stream frames is enabled; non-matching frames from the stream are processed in queue 0 (best effort).<br>10: The filter for separating AVB stream frames from non-AVB stream frames is enabled; non-matching frames are discarded.<br>11: The filter for separating AVB stream frames from non-AVB stream frames is enabled; non-matching frames from a stream are processed in queue 0 (best effort).<br>Recommended value: B'10 or B'11 |
| 1 | ENCF | 0 | R/W | Network Control Filtering Enable<br>Setting for reception queue 1 (network control)<br>Enables the AVB network control frame for reception queue 1.<br>When reception queue 1 is disabled, a received frame is stored in reception queue 0 (best effort).<br>Writing to this bit is only possible when the current operating mode is configuration mode.<br>0: Network control is disabled.<br>1: Network control is enabled. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | EFFS | 0 | R/W | Error Frame Enable<br>Enables or disables the reception of frames that have been classified as error frames by the E-MAC.<br>Received error frames are stored in reception queue 0 (best effort).<br>An indicator of error detection by the E-MAC during reception is stored in the descriptor (DESCR.MS).<br>Writing to this bit is only possible when the current operating mode is configuration mode.<br>0: Error frames are disabled.<br>1: Error frames are enabled.<br>Recommended value: 0 |

### 100.1.2.9 Receive Queue Configuration Register i (RQCi) (i = 0 to 4)

The RQC0 register is used to set up reception queues 0 to 3.

The RQC1 register is used to set up reception queues 4 to 7.

The RQC2 register is used to set up reception queues 8 to 11.

The RQC3 register is used to set up reception queues 12 to 15.

The RQC4 register is used to set up reception queues 16 to 17.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PIA3 | UFCC3[1:0] | | TSEL3[1:0] | | RSM3[1:0] | | — | PIA2 | UFCC2[1:0] | | TSEL2[1:0] | | RSM2[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | PIA2 | UFCC1[1:0] | | TSEL1[1:0] | | RSM1[1:0] | | — | PIA0 | UFCC0[1:0] | | TSEL0[1:0] | | RSM0[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 30 | PIA3 | 0 | R/W | Packed Incremental data Area (Receive Queue 3+i×4)<br>This bit defines how addresses inside the incremental data area of queue 3+i×4 are handled.<br>When this bit is 0b, frame data in incremental data area starts always at a 32 bit address. Depending on received frame length there are up to 3 undefined bytes between two frames in URAM.<br>When this bit is 1b, there will no gaps generated between two frames in incremental data area.<br>The CPU can only write to this bit if **CSR.OPS** is CONFIG.<br>0: Frame data starts always 32 bit aligned<br>1: No gaps between frame data in incremental data area |
| 29, 28 | UFCC3[1:0] | 00 | R/W | Unread Frame Counter Configuration (Receive Queue 3+i×4)<br>These bits set the unread frame counter used in reception queue 3+i×4.<br>With the AVB-DMAC, four patterns of settings are available for the unread frame counter. Use the unread frame counter warning level configuration register (UFCW) and the unread frame counter stop level configuration register (UFCS) to set the warning level and stop level of the unread frame counter.<br>Set the pattern number 3 set in the unread frame counter warning level configuration register (UFCW) and unread frame counter stop level configuration register (UFCS) in this bit.<br>When the value is B'00, the stop function is disabled.<br>Writing to the bits is only possible when the current operating mode is configuration mode. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27, 26 | TSEL3 | 00 | R/W | Truncation SELection (Receive Queue 3+i×4)<br>This bit defines a maximum number of bytes for storing frames in receive queue 3+i×4. Frames longer than the selected maximum are truncated.<br>EthernetAVB-IF provides 2 common configurations for the maximum frame length. This bit selects one of the configurations j defined by **RTCi.MFLj**.<br>00: Maximum frame length defined by **RTC0.MFL0**<br>01: Maximum frame length defined by **RTC0.MFL1**<br>Other configurations are Invalid<br>The CPU can only write to these bits if **CSR.OPS** is CONFIG. |
| 25, 24 | RSM3[1:0] | 00 | R/W | Receive Synchronous Mode (Receive Queue 3+i×4)<br>These bits set receive synchronous mode.<br>Set B'00 in this bit.<br>For receive synchronous mode, see section 100.1.3.4(3)(c) Mode with Write-Back. Writing to the bits is only possible when the current operating mode is configuration mode.<br>00: Mode with write-back<br>01: Keep DT mode (no update of DT field at descriptor write back)<br>10: No write-back (no descriptor write back)<br>11: Invalid |
| 23 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 22 | PIA2 | 0 | R/W | Packed Incremental data Area (Receive Queue 2+i×4)<br>This bit defines how addresses inside the incremental data area of queue 2+i×4 are handled.<br>When this bit is 0b, frame data in incremental data area starts always at a 32 bit address. Depending on received frame length there are up to 3 undefined bytes between two frames in URAM.<br>When this bit is 1b, there will no gaps generated between two frames in incremental data area.<br>The CPU can only write to this bit if **CSR.OPS** is CONFIG.<br>0: Frame data starts always 32 bit aligned<br>1: No gaps between frame data in incremental data area |
| 21, 20 | UFCC2[1:0] | 00 | R/W | Unread Frame Counter Configuration (Receive Queue 2+i×4)<br>These bits set the unread frame counter used in reception queue 2+i×4.<br>With the AVB-DMAC, four patterns of settings are available for the unread frame counter. Use the unread frame counter warning level configuration register (UFCW) and the unread frame counter stop level configuration register (UFCS) to set the warning level and stop level of the unread frame counter.<br>Set the pattern number 2 set in the unread frame counter warning level configuration register (UFCW) and unread frame counter stop level configuration register (UFCS) in this bit.<br>When the value is B'00, the stop function is disabled.<br>Writing to the bits is only possible when the current operating mode is configuration mode. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 19, 18 | TSEL2 | 00 | R/W | Truncation SELection (Receive Queue 2+i×4)<br>This bit defines a maximum number of bytes for storing frames in receive queue 2+i×4. Frames longer than the selected maximum are truncated.<br>EthernetAVB provides 2 common configurations for the maximum frame length. This bit selects one of the configurations j defined by **RTCi.MFLj**.<br>00: Maximum frame length defined by **RTC0.MFL0**<br>01: Maximum frame length defined by **RTC0.MFL1**<br>Other configurations are Invalid<br>The CPU can only write to these bits if **CSR.OPS** is CONFIG. |
| 17, 16 | RSM2[1:0] | 00 | R/W | Receive Synchronous Mode (Receive Queue 2+i×4)<br>These bits set receive synchronous mode.<br>Set B'00 in this bit.<br>For receive synchronous mode, see section 100.1.3.4(3)(c) Mode with Write-Back. Writing to the bits is only possible when the current operating mode is configuration mode.<br>00: Mode with write-back<br>01: Keep DT mode (no update of DT field at descriptor write back)<br>10: No write-back (no descriptor write back)<br>11: Invalid |
| 15 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 14 | PIA1 | 0 | R/W | Packed Incremental data Area (Receive Queue 1+i×4)<br>This bit defines how addresses inside the incremental data area of queue 1+i×4 are handled.<br>When this bit is 0b, frame data in incremental data area starts always at a 32 bit address. Depending on received frame length there are up to 3 undefined bytes between two frames in URAM.<br>When this bit is 1b, there will no gaps generated between two frames in incremental data area.<br>The CPU can only write to this bit if **CSR.OPS** is CONFIG.<br>0: Frame data starts always 32 bit aligned<br>1: No gaps between frame data in incremental data area |
| 13, 12 | UFCC1[1:0] | 00 | R/W | Unread Frame Counter Configuration (Receive Queue 1+i×4)<br>These bits set the unread frame counter used in reception queue 1+i×4.<br>With the AVB-DMAC, four patterns of settings are available for the unread frame counter. Use the unread frame counter warning level configuration register (UFCW) and the unread frame counter stop level configuration register (UFCS) to set the warning level and stop level of the unread frame counter.<br>Set the pattern number 1 set in the unread frame counter warning level configuration register (UFCW) and unread frame counter stop level configuration register (UFCS) in this bit.<br>When the value is B'00, the stop function is disabled.<br>Writing to the bits is only possible when the current operating mode is configuration mode. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11, 10 | TSEL1 | 00 | R/W | Truncation SELection (Receive Queue 1+i×4)<br>This bit defines a maximum number of bytes for storing frames in receive queue 1+i×4. Frames longer than the selected maximum are truncated.<br>EthernetAVB-IF provides 2 common configurations for the maximum frame length. This bit selects one of the configurations j defined by **RTCi.MFLj**.<br>00: Maximum frame length defined by **RTC0.MFL0**<br>01: Maximum frame length defined by **RTC0.MFL1**<br>Other configurations are Invalid<br>The CPU can only write to these bits if **CSR.OPS** is CONFIG. |
| 9, 8 | RSM1[1:0] | 00 | R/W | Receive Synchronous Mode (Receive Queue 1+i×4)<br>These bits set receive synchronous mode.<br>Set B'00 in this bit.<br>For receive synchronous mode, see section 100.1.3.4(3)(c) Mode with Write-Back. Writing to the bits is only possible when the current operating mode is configuration mode.<br>00: Mode with write-back<br>01: Keep DT mode (no update of DT field at descriptor write back)<br>10: No write-back (no descriptor write back)<br>11: Invalid |
| 7 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 6 | PIA0 | 0 | R/W | Packed Incremental data Area (Receive Queue 0+i×4)<br>This bit defines how addresses inside the incremental data area of queue 0+i×4 are handled.<br>When this bit is 0b, frame data in incremental data area starts always at a 32 bit address. Depending on received frame length there are up to 3 undefined bytes between two frames in URAM.<br>When this bit is 1b, there will no gaps generated between two frames in incremental data area.<br>The CPU can only write to this bit if **CSR.OPS** is CONFIG.<br>0: Frame data starts always 32 bit aligned<br>1: No gaps between frame data in incremental data area |
| 5, 4 | UFCC0[1:0] | 00 | R/W | Unread Frame Counter Configuration (Receive Queue 0+i×4)<br>These bits set the unread frame counter used in reception queue 0+i×4.<br>With the AVB-DMAC, four patterns of settings are available for the unread frame counter. Use the unread frame counter warning level configuration register (UFCW) and the unread frame counter stop level configuration register (UFCS) to set the warning level and stop level of the unread frame counter.<br>Set the pattern number 0 set in the unread frame counter warning level configuration register (UFCW) and unread frame counter stop level configuration register (UFCS) in this bit.<br>When the value is B'00, the stop function is disabled.<br>Writing to the bits is only possible when the current operating mode is configuration mode. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3, 2 | TSEL0 | 00 | R/W | Truncation SELection (Receive Queue 0+i×4)<br>This bit defines a maximum number of bytes for storing frames in receive queue 0+i×4. Frames longer than the selected maximum are truncated.<br>EthernetAVB-IF provides 2 common configurations for the maximum frame length. This bit selects one of the configurations j defined by **RTCi.MFLj**.<br>00: Maximum frame length defined by **RTC0.MFL0**<br>01: Maximum frame length defined by **RTC0.MFL1**<br>Other configurations are Invalid<br>The CPU can only write to these bits if **CSR.OPS** is CONFIG. |
| 1, 0 | RSM0[1:0] | 00 | R/W | Receive Synchronous Mode (Receive Queue 0+i×4)<br>These bits set receive synchronous mode.<br>Set B'00 in this bit.<br>For receive synchronous mode, see section 100.1.3.4(3)(c) Mode with Write-Back. Writing to the bits is only possible when the current operating mode is configuration mode.<br>00: Mode with write-back<br>01: Keep DT mode (no update of DT field at descriptor write back)<br>10: No write-back (no descriptor write back)<br>11: Invalid |

### 100.1.2.10 Receive Padding Configuration Register (RPC)

The RPC register is used to set padding for received frames.

Note: Padding can be used to extend frame lengths, but frame lengths should not exceed 4 Kbytes.

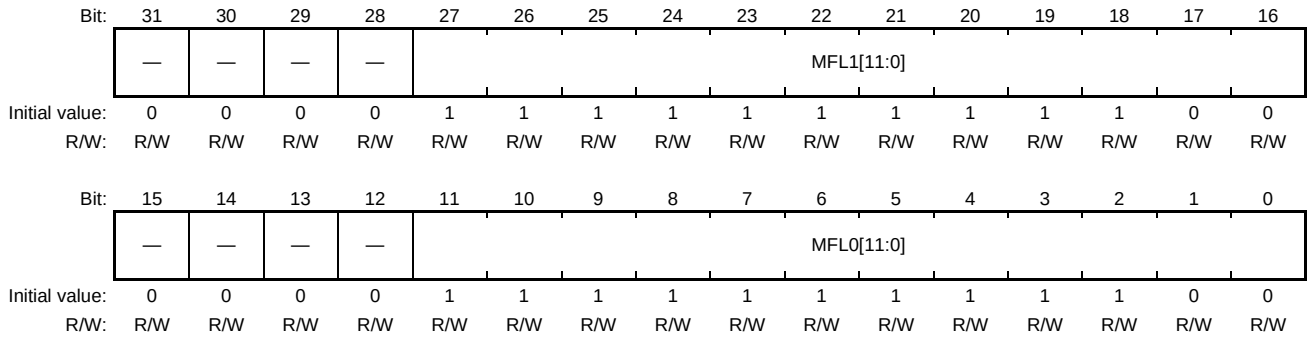
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DCNT[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | PCNT[2:0] | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 23 to 16 | DCNT[7:0] | H'00 | R/W | Stored Data Counter<br>These bits specify the amount of the frame data (1 to 255) to be stored following the padding. Counting by one indicates one word (4 bytes). For example, when these bits are set to 47, the amount of data is 47 words (= 188 bytes).<br>When these bits are 0, all received data have been stored following the initial padding.<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>For details on padding, see section 100.1.3.4(3)(b) Incremental Data Areas. |
| 15 to 11 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 to 8 | PCNT[2:0] | 001 | R/W | Stored Padding Counter<br>These bits specify the amount of padding to be appended to the URAM. Counting by one indicates one word (4 bytes). For example, when these bits are set to 1, the amount of padding is one word (= 4 bytes).<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>For details on padding, see section 100.1.3.4(3)(b) Incremental Data Areas.<br>The CPU should not write 0 to these bits. |
| 7 to 0 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.11 Reception Truncation Configuration register (RTC)

The RTC register is used to set Maximum Number of bytes stored per received frame.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 27to 16 | MFL1[11:0] | H'FFC | R/W | Maximum Frame Length 1<br>These bits define the maximum frame length (64 to 4092 bytes) stored to URAM. For each receive queue one parameter 1 used for truncation is selected.<br>When a frame received by E-MAC is truncated, CPU gets informed by **EIS.QEF** and frame size error (**ESR.ET** is 1000b). Additionally the **DESCR.TI** in the updated descriptor is set to 1b.<br>The CPU can only write to these bits if **CSR.OPS** is CONFIG.<br>The CPU should only write values which are a multiple of 4 to these bits.<br>The CPU should not write values less than 64 to these bits.<br>The CPU should not write values less then **SFO.FBP** + 8 to these bits when **RCR.ESF** is not equal to 00b. |
| 15 to 12 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 11to 0 | MFL0[11:0] | H'FFC | R/W | Maximum Frame Length 0<br>These bits define the maximum frame length (64 to 4092 bytes) stored to URAM. For each receive queue one parameter 0 used for truncation is selected.<br>When a frame received by E-MAC is truncated, CPU gets informed by **EIS.QEF** and frame size error (**ESR.ET** is 1000b). Additionally the **DESCR.TI** in the updated descriptor is set to 1b.<br>The CPU can only write to these bits if **CSR.OPS** is CONFIG.<br>The CPU should only write values which are a multiple of 4 to these bits.<br>The CPU should not write values less than 64 to these bits.<br>The CPU should not write values less then **SFO.FBP** + 8 to these bits when **RCR.ESF** is not equal to 00b. |

### 100.1.2.12 Unread frame counter warning level register (UFCW)

The UFCW register sets the warning levels for the number of unread frames.

One of the four warning levels from 0 to 3 can be set for each reception queue. When these bits are set to 0, the stop function is disabled. The level to be used is specified by the receive queue configuration register i (RQCi) (i = 0 to 4).

Writing to the bits is only possible when the current operating mode is configuration mode.

The CPU should not write 63 to these bits.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | WL3[5:0] | | | | | | — | — | WL2[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | WL1[5:0] | | | | | | — | — | WL0[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 29 to 24 | WL3[5:0] | 000000 | R/W | Warning Level 3<br>Unread frame count warning level 3 |
| 23, 22 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 21 to 16 | WL2[5:0] | 000000 | R/W | Warning Level 2<br>Unread frame count warning level 2 |
| 15, 14 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 13 to 8 | WL1[5:0] | 000000 | R/W | Warning Level 1<br>Unread frame count warning level 1 |
| 7, 6 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 5 to 0 | WL0[5:0] | 000000 | R/W | Warning Level 0<br>Unread frame count warning level 0 |

### 100.1.2.13 Unread frame counter stop level register (UFCS)

The UFCS register sets the stop levels for unread frames.

One of the four stop levels from 0 to 3 can be set for each reception queue. When these bits are set to 0, the stop function is disabled. The level to be used is specified by the receive queue configuration register i (RQCi) (i = 0 to 4).

Writing to the bits is only possible when the current operating mode is configuration mode.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | SL3[5:0] | | | | | | — | — | SL2[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | SL1[5:0] | | | | | | — | — | SL0[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 29 to 24 | SL3[5:0] | 000000 | R/W | Stop Level 3<br>Unread frame count stop level 3 |
| 23, 22 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 21 to 16 | SL2[5:0] | 000000 | R/W | Stop Level 2<br>Unread frame count stop level 2 |
| 15, 14 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 13 to 8 | SL1[5:0] | 000000 | R/W | Stop Level 1<br>Unread frame count stop level 1 |
| 7, 6 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 5 to 0 | SL0[5:0] | 000000 | R/W | Stop Level 0<br>Unread frame count stop level 0<br>The CPU can only write 0 |

### 100.1.2.14 Unread Frame Counter Register i (UFCVi) (i = 0 to 4)

The UFCV0 register indicates the number of unread frames in reception queues 0 to 3.

The UFCV1 register indicates the number of unread frames in reception queues 4 to 7.

The UFCV2 register indicates the number of unread frames in reception queues 8 to 11.

The UFCV3 register indicates the number of unread frames in reception queues 12 to 15.

The UFCV4 register indicates the number of unread frames in reception queues 16 and 17.

For a description of how to use unread frames, refer to section 100.1.3.4(4) Unread Frame Counters.

**Conditions for updating:**

The bits are set to 0 when the operating mode is not operation mode and when the descriptor base address load request register (DLR) issues a base address load request.

The number is incremented when data received in reception queue r are stored normally. The maximum increment is H'3F. If the value exceeds H'3F, incrementation will not proceed.)

The number is decremented by the value written to the unread frame counter decrement register i (UFCDi).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | CV3[5:0] | | | | | | — | — | CV2[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | CV1[5:0] | | | | | | — | — | CV0[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 29 to 24 | CV3[5:0] | 000000 | R | Unread Frame Count 3+4×i<br>Number of unread frames in reception queue 3+4×i |
| 23, 22 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 21 to 16 | CV2[5:0] | 000000 | R | Unread Frame Count 2+4×i<br>Number of unread frames in reception queue 2+4×i |
| 15, 14 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 13 to 8 | CV1[5:0] | 000000 | R | Unread Frame Count 1+4×i<br>Number of unread frames in reception queue 1+4×i |
| 7, 6 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 5 to 0 | CV0[5:0] | 000000 | R | Unread Frame Count 0+4×i<br>Number of unread frames in reception queue 0+4×i |

### 100.1.2.15 Unread Frame Counter Decrement Register i (UFCDi) (i = 0 to 4)

The UFCD0 register is used to decrement unread counters in reception queues 0 to 3.

The UFCD1 register is used to decrement unread counters in reception queues 4 to 7.

The UFCD2 register is used to decrement unread counters in reception queues 8 to 11.

The UFCD3 register is used to decrement unread counters in reception queues 12 to 15.

The UFCD4 register is used to decrement unread counters in reception queues 16 and 17.

Write H'3F to these bits to reset the unread counters in reception queue r (r = 0 to 17). These bits are always read as 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | DV3[5:0] | | | | | | — | — | DV2[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | DV1[5:0] | | | | | | — | — | DV0[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 29 to 24 | DV3[5:0] | 000000 | R/W | Unread Frame Decrement Value 3+4×i<br>Unread frame decrement value for reception queue 3+4×i |
| 23, 22 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 21 to 16 | DV2[5:0] | 000000 | R/W | Unread Frame Decrement Value 2+4×i<br>Unread frame decrement value for reception queue 2+4×i |
| 15, 14 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 13 to 8 | DV1[5:0] | 000000 | R/W | Unread Frame Decrement Value 1+4×i<br>Unread frame decrement value for reception queue 1+4×i |
| 7, 6 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 5 to 0 | DV0[5:0] | 000000 | R/W | Unread Frame Decrement Value 0+4×i<br>Unread frame decrement value for reception queue 0+4×i |

### 100.1.2.16 Separation Filter Offset Register (SFO)

The SFO register sets an offset into frames for use by the separation filter.

Note: Received frames having fewer bytes than the setting of these bits + 8 bytes are judged to be non-matching by the separation filter. In this case, the data will either be sorted into a reception queue or discarded in accord with the setting of the separation filtering select bits in the receive configuration register (RCR.ESF).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | FBP[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 5 to 0 | FBP[5:0] | 000000 | R/W | First Byte Position<br>These bits set the position in Ethernet frames of the first byte of the bytes to be used by the separation filter.<br>When these bits are 0, the separation filter starts from the start of each Ethernet frame (first byte of the destination address). For bytes in Ethernet frames, see Figure 100.1.3.1.1 Operating Mode of AVB-DMAC, in section 100.1.3.1(1) Operating Modes.<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>[Changing condition]<br>These bits are updated to the value of **SFVi.LV**[5:0] when **SFL.LC** changes from 30 (Load SFO.FBP) to 31 while **CSR.OPS** is OPERATION.<br>For separation filtering, see section 100.1.3.4(1)(a) Separation Filtering. |

### 100.1.2.17 Separation Filter Pattern Register i (SFPi) (i = 0 to 31)

A pair of SFPi registers set the pattern for the separation filters to be used by the corresponding reception queues 2 to 17 (for streams 0 to 15).

Each queue shares a 64-bit setting; reception queue 2 (for stream 0) uses SFP0 and SFP1, reception queue 17 (for stream 15) uses SFP30 and SFP31, and so on.

The separation filter passes a frame when, after masking by the mask value set in the separation filter mask register (SFMi), data from received frames match the value defined in these bits.

SFPi.FP[7:0] (where i is an even number) are used for the byte of Ethernet frame data specified by the separation filter offset register, while SFPi.FP[63:56] (where i is the corresponding odd number) are used for the byte at the address specified by the separation filter offset register (SFO) + 7.

Writing to the bits is only possible when the current operating mode is configuration mode.

[Changing condition]

These bits are updated to the value of **SFVi.LV**[63:0] when **SFL.LC** changes from s (Load SFPi.FPs) to 31 while **CSR.OPS** is OPERATION.

For separation filtering, see section 100.1.3.4(1)(a) Separation Filtering.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FP[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FP[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | FP[31:0] | H'0000 0000 | R/W | Separation Filter Pattern<br>These bits set the pattern of the separation filter.<br>The 64-bit filter pattern is set for each queue. |

### 100.1.2.18 Separation Filter Value register i (SFVi) (i = 0 to 1)

These bits define the 64 bit value to be loaded for separation filtering.

When loading the first by position (**SFO.FBP**), **SFVi.LV**[63:6] are ignored.

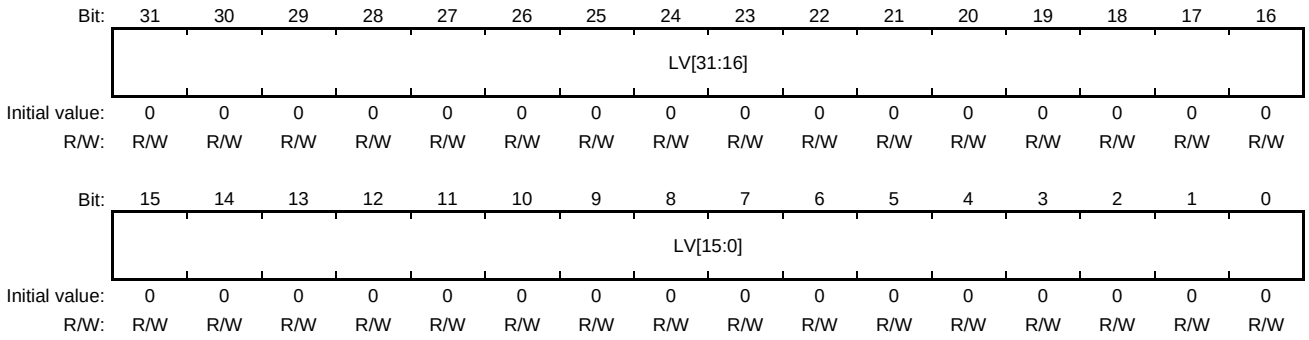
The CPU can only write to these bits if **CSR.OPS** is OPERATION.

The CPU can only write to these bits if **SFL.LC** is 31.

[Changing condition]

These bits are set to 0 when leaving OPERATION mode.

For separation filtering, see section 100.1.3.4(1)(a) Separation Filtering.



| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LV[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LV[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LV[31:0] | H'0000 0000 | R/W | Separation Filter Value<br>These bits set the Load value of the separation filter. |

### 100.1.2.19 Separation Filter Mask Register i (SFMi) (i = 0 or 1)

A pair of SFMi registers sets the mask value for the separation filter used by the corresponding reception queue 2 to 17 (stream 0 to 15).

SFM0.CFM[7:0] are used for bytes of Ethernet frame data specified by the separation filter offset register, while SFM1.CFM[63:56] are used for the separation filter offset register (SFO) + 7.

Frame data at the positions of mask bits that are set to 0 are masked; that is, they do not affect pattern-matching by the separation filter.

Writing to the bits is only possible when the current operating mode is configuration mode.

[Changing condition]

These bits are updated to the value of **SFVi.LV**[63:0] when **SFL.LC** changes from 29 (Load SFMi.CFM) to 31 while **CSR.OPS** is OPERATION.

For separation filtering, see section 100.1.3.4(1)(a) Separation Filtering.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFM[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFM[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CFM[31:0] | H'0000 0000 | R/W | Separation Filter Mask<br>These bits set the mask value for the separation filter. |

### 100.1.2.20 Separation Filter Load register (SFL)

The SFL register is used to separation filter load.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | LC[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 to 0 | LC[4:0] | H'1F | R/W | Load Command<br>These bits control update of separation filter configuration in OPERATION mode.<br>By writing 30 to these bits CPU triggers an update of First Byte Position (**SFO.FBP**) value.<br>By writing 29 to these bits CPU triggers an update of Common Filter Mask (**SFMi.CFM**) value.<br>By writing number between 0 and 15 to these bits CPU triggers an update of Filter Pattern s (**SFPi.FPs**) value. The number written to **SFL.LC** defines the index s of the updated Filter Pattern.<br>The CPU can only write to these bits if **CSR.OPS** is OPERATION.<br>The CPU can only write to these bits if **SFL.LC** is 31.<br>[Changing condition]<br>These bits are set to 31 when leaving OPERATION mode.<br>These bits are set to 31 when the requested update has been processed.<br>0-15: Load of SFPi.FPs with s = SFL.LC<br>16-28: Invalid<br>29: Load of SFMi.CFM<br>30: Load of SFO.FBP<br>31: No pending load request |

### 100.1.2.21 Payload CRC register (PCRC)

The PCRC register is used to set the number of the first data byte in received frame to check a potential payload CRC.

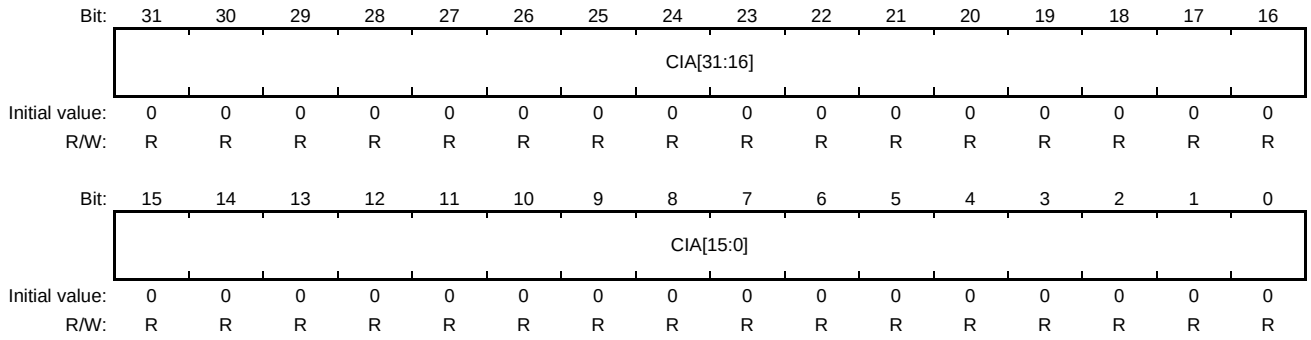
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CAS[9:0] | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

This register is functional safety use only. Keep initial value.

### 100.1.2.22 Current Incremental Address Register r (CIARr) (r = 0 to 17)

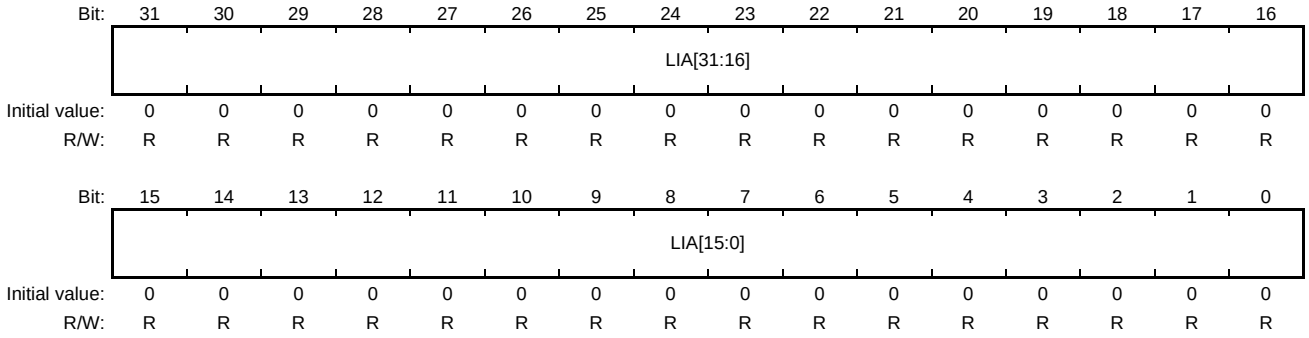
The CIARr register is used to check queue address.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CIA[31:0] | H'0000 0000 | R | Current incremental address used by receive queue<br>These bits indicate the incremental address of receive queue r, where currently frame data is stored or where frame data will be stored when queue r gets active.<br>Refer to section 100.1.3.4(3)(b) Incremental Data Areas.<br>[Changing condition]<br>These bits are set to 0 when leaving OPERATION mode.<br>These bits are updated to next address in incremental data area when processing a FEMPTY_IS or FEMPTY_IC descriptor. |

### 100.1.2.23 Last Incremental Address Register r (LIARr) (r = 0 to 17)

The LIARr register is used to check queue address.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LIA[31:0] | H'0000 0000 | R | Last incremental address used by receive queue<br>These bits indicate the current incremental address of receive queue r (**CIARr.CIA**) before its address was updated due to processing a FEMPTY_IS descriptor. So it describes the first not used address inside the incremental data area of receive queue r.<br>Refer to section 100.1.3.4(3)(b) Incremental Data Areas.<br>[Changing condition]<br>These bits are set to 0 when leaving OPERATION mode.<br>These bits are set to **CIARr.CIA** when processing a FEMPTY_IS descriptor. |

### 100.1.2.24 Transmit Configuration Register (TGC)

The TGC register is used to make settings related to transmission for the AVB-DMAC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TBD3[1:0] | | — | — | TBD2[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TBD1[1:0] | | — | — | TBD0[1:0] | | — | — | TQP | ECBS | TSM3 | TSM2 | TSM1 | TSM0 |
| Initial value: | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 22 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 21, 20 | TBD3[1:0] | 10 | R/W | Transmit FIFO Size (Stream Class A)<br>These bits set the sizes of the transmission FIFO buffers for use with each of the transmission queue 3 (for stream class A).<br>Writing to these bits is only possible when the current operating mode is configuration mode.<br>Set these bits to B'10. |
| 19, 18 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17, 16 | TBD2[1:0] | 10 | R/W | Transmit FIFO Size (Stream Class B)<br>These bits set the sizes of the transmission FIFO buffers for use with each of the transmission queue 2 (for stream class B).<br>Writing to these bits is only possible when the current operating mode is configuration mode.<br>Set these bits to B'10. |
| 15, 14 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 13, 12 | TBD1[1:0] | 10 | R/W | Transmit FIFO Size (Network Control)<br>These bits set the sizes of the transmission FIFO buffers for use with each of the transmission queue 1 (for network control).<br>Writing to these bits is only possible when the current operating mode is configuration mode. |
| 11, 10 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 9, 8 | TBD0[1:0] | 10 | R/W | Transmit FIFO Size (Best Effort)<br>These bits set the sizes of the transmission FIFO buffers for use with each of the transmission queue 0 (for best effort).<br>Writing to these bits is only possible when the current operating mode is configuration mode. |
| 7, 6 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | TQP | 0 | R/W | Transmission Queues Priority<br>This bit defines how EtherAVB-IF prioritises between transmit queues when scheduling transmission. For fetching always the queue with highest priority is selected, for transmission the queue with highest priority where data is available in Tx-Buffer.<br>Transmission priority of queue 2 and 3 can be influenced by the credit based shaping algorithm.<br>The priority of the transmit queues is given from high to low.<br>0: Default priority: Q3(CBS), Q2(CBS), Q1, Q0<br>1: Alternate priority: Q1, Q3(CBS), Q2(CBS), Q0<br>For the credit-based shaping (CBS) algorithm, see section 100.1.3.6 CBS (Credit-Based Shaping).<br>Writing to the bits is only possible when the current operating mode is configuration mode. |
| 4 | ECBS | 0 | R/W | Enable Credit Based Shaping<br>This bit defines if CBS algorithm is used for transmit queue 2 and 3 to prioritise between transmit queues when scheduling transmission. When CBS is enabled, the shaping bases on queue specific configuration in CIVRc, CDVRc, CULc and CLLc.<br>0: CBS globally disabled<br>1: CBS enabled based on queue specific configuration<br>For the credit-based shaping (CBS) algorithm, see section 100.1.3.6 CBS (Credit-Based Shaping).<br>Writing to the bits is only possible when the current operating mode is configuration mode. |
| 3 | TSM3 | 0 | R/W | Transmit Synchronous Mode (Stream Class A)<br>Set these bits to 0.<br>0: With write-back<br>1: Setting prohibited<br>Writing to the bits is only possible when the current operating mode is configuration mode. |
| 2 | TSM2 | 0 | R/W | Transmit Synchronous Mode (Stream Class B)<br>Set these bits to 0.<br>0: With write-back<br>1: Setting prohibited<br>Writing to the bits is only possible when the current operating mode is configuration mode. |
| 1 | TSM1 | 0 | R/W | Transmit Synchronous Mode (Network Control)<br>Set these bits to 0.<br>0: With write-back<br>1: Setting prohibited<br>Writing to the bits is only possible when the current operating mode is configuration mode. |
| 0 | TSM0 | 0 | R/W | Transmit Synchronous Mode (Best Effort)<br>Set these bits to 0.<br>0: With write-back<br>1: Setting prohibited<br>Writing to the bits is only possible when the current operating mode is configuration mode. |

### 100.1.2.25 Transmit Configuration Control Register (TCCR)

The TCCR register controls transmission by the AVB-DMAC and is used to make related settings.

**Conditions for updating:**

The bit is set to 0 when the operating mode is not operation mode, when a descriptor of type EEMPTY, FEMPTY or LEMPTY (no usable data) is processed, when an EOS descriptor is processed, and when a descriptor with defective data is processed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MFR | MFEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TFR | TFEN | — | — | — | — | TSRQ3 | TSRQ2 | TSRQ1 | TSRQ0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | MFR | 0 | R/W | E-MAC status FIFO Release<br>This bit releases the oldest entry of the E-MAC status FIFO.<br>When 1b is written to this bit, EthernetAVB-IF gets informed that CPU has processed the oldest E-MAC status FIFO entry as visible in **MFA** register. The oldest entry is removed from E-MAC status FIFO.<br>This bit is always read as 0b.<br>0: No request to E-MAC status FIFO<br>1: Release oldest entry of E-MAC status FIFO |
| 16 | MFEN | 0 | R/W | E-MAC status FIFO ENable<br>This bit enables the storage of transmission status information (the MAC flags for frame transmission) in the E-MAC status FIFO. Additionally CPU can control the status storage for each frame provided for transmission using **DESCR.MSR**.<br>When this bit is set to 0b all pending FIFO entries are removed.<br>0: Disabled<br>1: Enabled |
| 15 to 10 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 9 | TFR | 0 | R/W | Time Stamp FIFO Release<br>This bit releases the oldest entry in the time-stamp FIFO.<br>For a description of how to use the time-stamp FIFO, see section 100.1.3.5(4) Time Stamping in Transmission.<br>0: (Not operating)<br>1: Releases the oldest entry in the time-stamp FIFO. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8 | TFEN | 0 | R/W | Time Stamp FIFO Enable<br>This bit enables storage in the time-stamp FIFO.<br>When it is set, time-stamp information is stored for descriptors with DESCR.TSR set to 1 (for DESCR.TSR, see section 100.1.3.5(2)(b) Configuration of Transmission Frame Data Descriptors.<br>When 0 is set in this bit, no entries are made in the time-stamp FIFO.<br>For a description of how to use the time-stamp FIFO, see section 100.1.3.5(4) Time Stamping in Transmission.<br>0: Recording of transmission time stamps in the time-stamp FIFO is disabled.<br>1: Recording of transmission time stamps in the time-stamp FIFO is enabled. |
| 7 to 4 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 3 | TSRQ3 | 0 | R/W | Transmit Start Request (Queue 3 (Stream Class A))<br>This bit issues a request to start transmission for transmission queue 3.<br>When read, this bit being set to 1 indicates that transmission queue t has a frame that has not yet been fetched to the transmission FIFO.<br>Frame transmission by the E-MAC is processed independently from fetching to the transmission FIFO. The timing of transmission from a queue depends on the priority order of transmission.<br>For the scheduling of transmission queues, see section 100.1.3.5(1) Transmission Modes.<br>Writing to this bit is only possible when the current operating mode is operation mode.<br>Only 1 can be written to the bit. Writing 0 to the bit has no effect.<br>0: Transmission queue is empty or stopped.<br>1: When written: A transmission start request is issued.<br>When read: Fetching of data for transmission is pending. |
| 2 | TSRQ2 | 0 | R/W | Transmit Start Request (Queue 2 (Stream Class B))<br>This bit issues a request to start transmission for transmission queue 2.<br>When read, this bit being set to 1 indicates that transmission queue t has a frame that has not yet been fetched to the transmission FIFO.<br>Frame transmission by the E-MAC is processed independently from fetching to the transmission FIFO. The timing of transmission from a queue depends on the priority order of transmission.<br>For the scheduling of transmission queues, see section 100.1.3.5(1) Transmission Modes.<br>Writing to this bit is only possible when the current operating mode is operation mode.<br>Only 1 can be written to the bit. Writing 0 to the bit has no effect.<br>0: Transmission queue is empty or stopped.<br>1: When written: A transmission start request is issued.<br>When read: Fetching of data for transmission is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | TSRQ1 | 0 | R/W | Transmit Start Request (Queue 1 (Network Control))<br>This bit issues a request to start transmission for transmission queue 1.<br>When read, this bit being set to 1 indicates that transmission queue t has a frame that has not yet been fetched to the transmission FIFO.<br>Frame transmission by the E-MAC is processed independently from fetching to the transmission FIFO. The timing of transmission from a queue depends on the priority order of transmission.<br>For the scheduling of transmission queues, see section 100.1.3.5(1) Transmission Modes.<br>Writing to this bit is only possible when the current operating mode is operation mode.<br>Only 1 can be written to the bit. Writing 0 to the bit has no effect.<br>0: Transmission queue is empty or stopped.<br>1: When written: A transmission start request is issued.<br>When read: Fetching of data for transmission is pending. |
| 0 | TSRQ0 | 0 | R/W | Transmit Start Request (Queue 0 (Best Effort))<br>This bit issues a request to start transmission for transmission queue 0.<br>When read, this bit being set to 1 indicates that transmission queue t has a frame that has not yet been fetched to the transmission FIFO.<br>Frame transmission by the E-MAC is processed independently from fetching to the transmission FIFO. The timing of transmission from a queue depends on the priority order of transmission.<br>For the scheduling of transmission queues, see section 100.1.3.5(1) Transmission Modes.<br>Writing to this bit is only possible when the current operating mode is operation mode.<br>Only 1 can be written to the bit. Writing 0 to the bit has no effect.<br>0: Transmission queue is empty or stopped.<br>1: When written: A transmission start request is issued.<br>When read: Fetching of data for transmission is pending. |

### 100.1.2.26 Transmit Status Register (TSR)

The TSR register indicates the state of transmission by the AVB-DMAC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | MFFL[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TFFL[2:0] | | | — | — | — | — | CCS1[1:0] | | CCS0[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 20 to 16 | MFFL[4:0] | H'00 | R | Number of entries stored in the E-MAC status FIFO<br>E-MAC status FIFO Fill Level<br>These bits indicate the number of entries available in the MAC status FIFO.<br>If this value is 0, the FIFO is empty. If this value is 16, the FIFO is full. The values 17 to 31 are reserved.<br>[Changing condition]<br>These bits are set to 0 when leaving OPERATION mode.<br>These bits are set to 0 when **TCCR.MFEN** is 0b.<br>This value is incremented when frame with **DESCR.MSR** has been transmitted by E-MAC, **TCCR.MFEN** is 1b and **TSR.MFFL** is not 16.<br>This value is incremented when E-MAC detects an error during transmission, **TCCR.MFEN** is 1b and **TSR.MFFL** is not 16.<br>This value is decremented when 1b is written to **TCCR.MFR** and **TSR.MFFL** is not 0. |
| 15 to 11 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 to 8 | TFFL[2:0] | 00 | R | Time Stamp FIFO Count<br>These bits indicate the number of time stamps in the time-stamp FIFO.<br>The values 0 and 3 indicate that the time-stamp FIFO is empty and full, respectively (values 4 to 7 are reserved).<br>Conditions for updating:<br>• The bits are set to 0 when the operating mode is not operation mode and when the time stamp FIFO enable bit in the transmit configuration control register (TCCR.TFEN) = 0.<br>• When the time stamp FIFO enable bit (TCCR.TFEN) is 1 and these bits are not 3, the value of these bits is incremented after a frame with DESCR.TSR set has been transmitted by the E-MAC (for DESCR.TSR, see section 100.1.3.5(2)(b) Configuration of Transmission Frame Data Descriptors.<br>• The value of these bits is decremented if it is not 0 when 1 is written to the time stamp FIFO release bit in the transmit configuration control register (TCCR.TFR). |
| 7 to 4 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 3, 2 | CCS1[1:0] | 00 | R | CBS Counter Status 1 (Class A)<br>These bits indicate the CBS (credit-based shaping) state of stream data transmission queue 1. If the calculated credit value is outside the range specified by CBS upper limit register c (CULc) and CBS lower limit register c (CLLc), it falls outside the range for CBS.<br>Conditions for updating:<br>• The bits are set to B'00 when the operating mode is not operation mode. The bits are set to B'00 when the CBS is inside the condition limits.<br>• The bits are set to B'01 if the credit value calculated by the CBS is lower than the value in CBS lower limit register c (CLLc).<br>• The bits are set to B'10 if the credit value calculated by the CBS is higher than the value in CBS upper limit register c (CULc).<br>00: The current credit value is within the limit.<br>01: The current credit value is less than or equal to the lower limit.<br>10: The current credit value is greater than or equal to the upper limit.<br>11: (Reserved) |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1, 0 | CCS0[1:0] | 00 | R | CBS Counter Status 0 (Class B)<br>These bits indicate the CBS (credit-based shaping) state of stream data transmission queue 0. If the calculated credit value is outside the range specified by CBS upper limit register c (CULc) and CBS lower limit register c (CLLc), it falls outside the range for CBS.<br>Conditions for updating:<br>• The bits are set to B'00 when the operating mode is not operation mode. The bits are set to B'00 when the CBS is inside the condition limits.<br>• The bits are set to B'01 if the credit value calculated by the CBS is lower than the value in CBS lower limit register c (CLLc).<br>• The bits are set to B'10 if the credit value calculated by the CBS is higher than the value in CBS upper limit register c (CULc).<br>00: The current credit value is within the limit.<br>01: The current credit value is less than or equal to the lower limit.<br>10: The current credit value is greater than or equal to the upper limit.<br>11: (Reserved) |

### 100.1.2.27 E-MAC status FIFO Access register (MFA)

The MFA register indicates the state of MAC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MST[9:0] | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

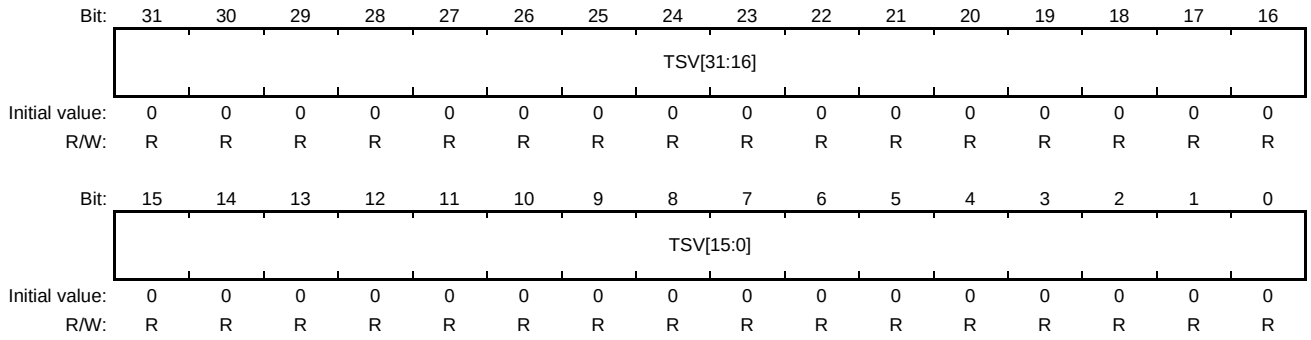
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MSV[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 25 to 16 | MST[9:0] | H'000 | R | Tag number from descriptor identifying the frame E-MAC status relation<br>E-MAC Status Tag<br>These bits represent the **DESCR.TAG** bits from the descriptor defining the data for frame transmission. The tag is used to get the relation between the frame inside a transmit queue and the E-MAC status available in the FIFO (**MFA.MSV**).<br>The CPU should not read these bits when **TSR.MFFL** is 0.<br>[Changing condition]<br>These bits are updated when the first entry is stored in FIFO (**TSR.MFFL** changes from 0 to 1).<br>These bits are updated when the oldest entry is released (writing 1b to **TCCR.MFR**).<br>These bits are updated when the FIFO is overwritten (a frame with **DESCR.MSR** in descriptor has been transmitted by MAC, **TCCR.MFEN** is 1b and **TSR.MFFL** is 16).<br>These bits are updated when the FIFO is overwritten (E-MAC detects an error during transmission and **TSR.MFFL** is 16).<br>***Note for verification: These bits are set to 0 when leaving OPERATION mode.*** |
| 15 to 4 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 to 0 | MSV[3:0] | H'0 | R | E-MAC Status Value<br>These bits represent the E-MAC status bits as stored in the oldest E-MAC status FIFO entry.<br>***MSV**[3] Reserved*<br>***MSV**[2] Carrier lost during frame transmission*<br>***MSV**[1] Delay collision*<br>***MSV**[0] Transmission time out*<br>The CPU should not read these bits when **TSR.MFFL** is 0.<br>[Changing condition]<br>These bits are updated when the first entry is stored in FIFO (**TSR.MFFL** changes from 0 to 1).<br>These bits are updated when the oldest entry is released (writing 1b to **TCCR.MFR**).<br>These bits are updated when the FIFO overwritten (a frame with **DESCR.MSR** in descriptor has been transmitted by E-MAC, **TCCR.MFEN** is 1b and **TSR.MFFL** is 16).<br>These bits are updated when the FIFO is overwritten (E-MAC detects an error during transmission and **TSR.MFFL** is 16).<br>***Note for verification: These bits are set to 0 when leaving OPERATION mode.*** |

### 100.1.2.28 Time Stamp FIFO Access Register 0 (TFA0)

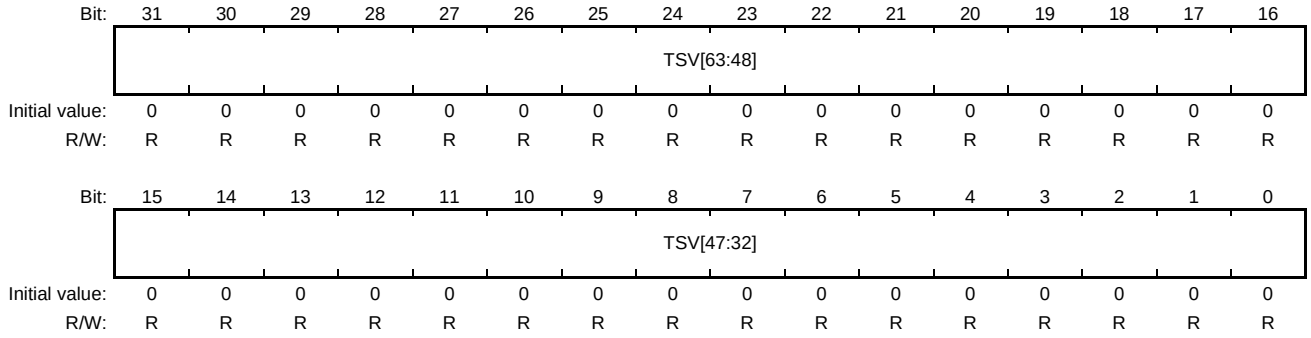
TFA0 indicates the nano seconds portion of the timestamp value.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TSV[31:0] | H'0000 0000 | R | Time Stamp Value<br>These 80 bits consist of TFA0.TSV[31:0], TFA1.TSV[63:32], and TFA2.TSV[79:64], which together indicate the oldest time stamp value stored in the time-stamp FIFO.<br>Once the time-stamp FIFO is full, no further time-stamp values are stored.<br>Conditions for updating:<br>• The bits are set to H'0000 0000 when the operating mode is not operation mode.<br>• The register is updated whenever a value is stored in the time-stamp FIFO (when the time-stamp FIFO count bit in the transmit status register (TSR.TFFL) changes from 0 to 1).<br>• The register is updated when the oldest entry is released (when the time stamp FIFO release bit in the transmit configuration control register (TCCR.TFR) is set to 1). |

### 100.1.2.29 Time Stamp FIFO Access Register 1 (TFA1)

The TFA1 register indicates the lower seconds portion of the timestamp value.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TSV[63:32] | H'0000 0000 | R | Time Stamp Value<br>For details, see section 100.1.2.28 Time Stamp FIFO Access Register 0 (TFA0). |

### 100.1.2.30 Time Stamp FIFO Access Register 2 (TFA2)

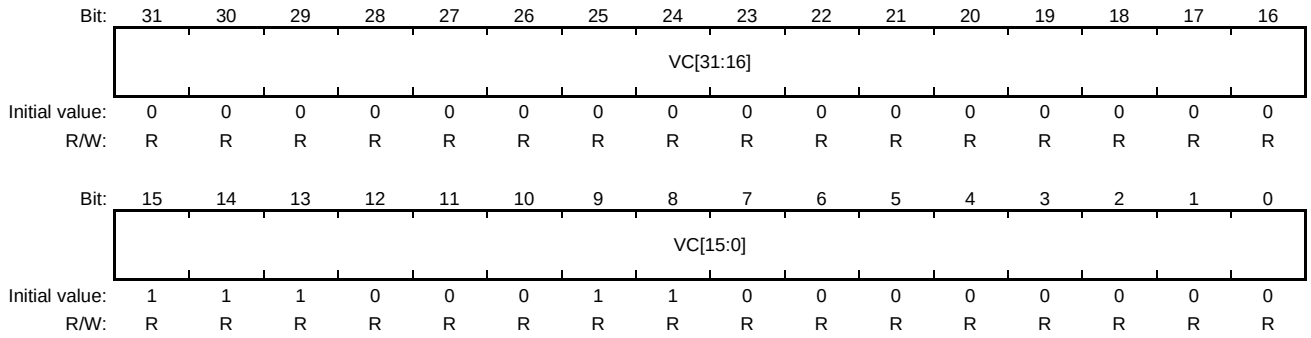
The TFA2 register indicates the timestamp tag and the higher seconds portion of the timestamp value.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TST[9:0] | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSV[79:64] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 25 to 16 | TST[9:0] | H'000 | R | Time Stamp Tag<br>These bits indicate the contents of the DESCR.TAG bit within the descriptor for frame transmission. These values are used to check the correlation between frames within the transmission queue and the time-stamp values (accessible through time stamp FIFO access register i (TFAi)) which can be placed in the FIFO.<br>For the tagging of frames in transmission, see section 100.1.3.5(4) Time Stamping in Transmission.<br>Conditions for updating:<br>• The bits are set to 000H when the operating mode is not operation mode.<br>• Updated when a value is stored in the time-stamp FIFO (when the value of the time stamp FIFO count bit in the transmit status register (TSR.TFFL) changes from 0 to 1).<br>• Updated when the oldest entry has been released (1 is set in the time stamp FIFO release bit in the transmit configuration control register (TCCR.TFR)). |
| 15 to 0 | TSV[79:64] | H'0000 | R | Time Stamp Value<br>For details, see section 100.1.2.28 Time Stamp FIFO Access Register 0 (TFA0). |

### 100.1.2.31 Version and Release Register (VRR)

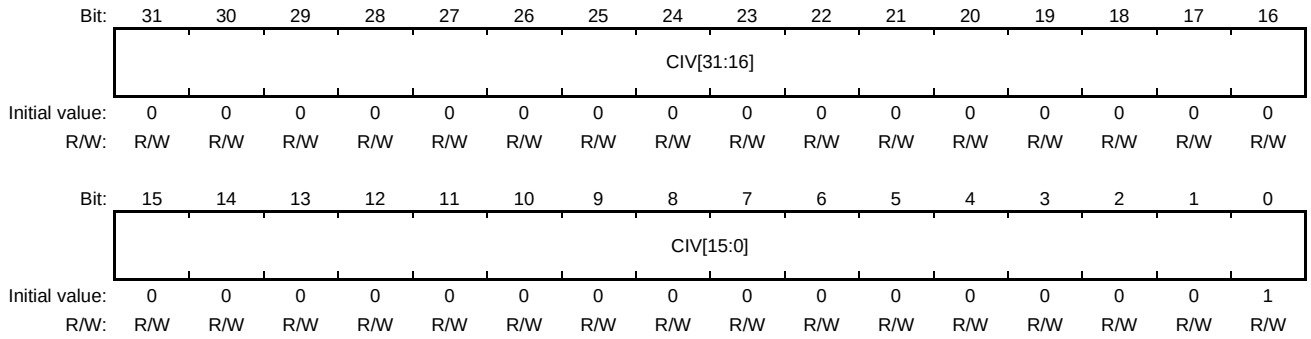



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | VC[31:0] | H'0000 E300 | R | Version Code<br>These bits identify the version of used EtherAVB-IF.<br>0000 E300 : RAVBES3 |

### 100.1.2.32 CBS Increment Value Register c (CIVRc) (c = 0 or 1)

The CIVR0 register sets the increment in the CBS algorithm for transmission queue 2 (for stream class B).

The CIVR1 register sets the increment in the CBS algorithm for transmission queue 3 (for stream class A).




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CIV[31:0] | H'0000 0001 | R/W | CBS Increment Value<br>CBS increment value (1 to H'FFFF)<br>These bits define the increment value related to idleSlope for the CBS algorithm.<br>Set a value in the range from H'0000 0001 to H'0000 FFFF.<br>The value to be written to these bits depends on the Ethernet bit rate and HPϕ (high-speed peripheral clock). For details, see section 100.1.3.6 CBS (Credit-Based Shaping). |

### 100.1.2.33 CBS Decrement Value Register c (CDVRc) (c = 0 or 1)

The CDVR0 register sets the decrement in the CBS algorithm for transmission queue 2 (for stream class B).

The CDVR1 register sets the decrement in the CBS algorithm for transmission queue 3 (for stream class A).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CDV[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

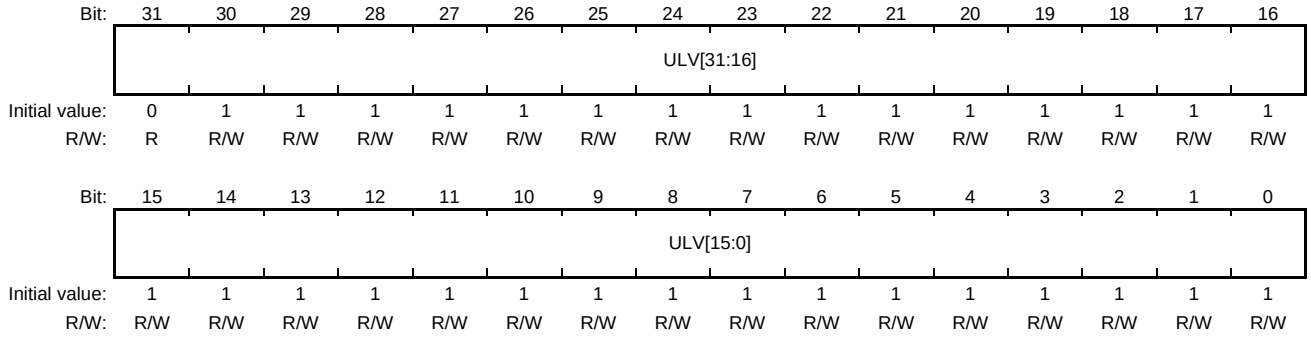
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CDV[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CDV[31:0] | H'FFFF FFFF | R/W | CBS Decrement Value<br>Setting value: –1 to –65536 (H'FFFF FFFF to H'FFFF 0000)<br>These bits set the decrement for the CBS algorithm.<br>These bits define the decrement value related to sendSlope for CBS algorithm.<br>The value to be written to these bits depends on the Ethernet bit rate and HPϕ (high-speed peripheral clock). For details, see section 100.1.3.6 CBS (Credit-Based Shaping). |

### 100.1.2.34 CBS Upper Limit Register c (CULc) (c = 0 or 1)

The CUL0 register sets the upper limit for credit values calculated by using the CBS algorithm for transmission queue 2 (for stream class B).

The CUL1 register sets the upper limit for credit values calculated by using the CBS algorithm for transmission queue 3 (for stream class A).



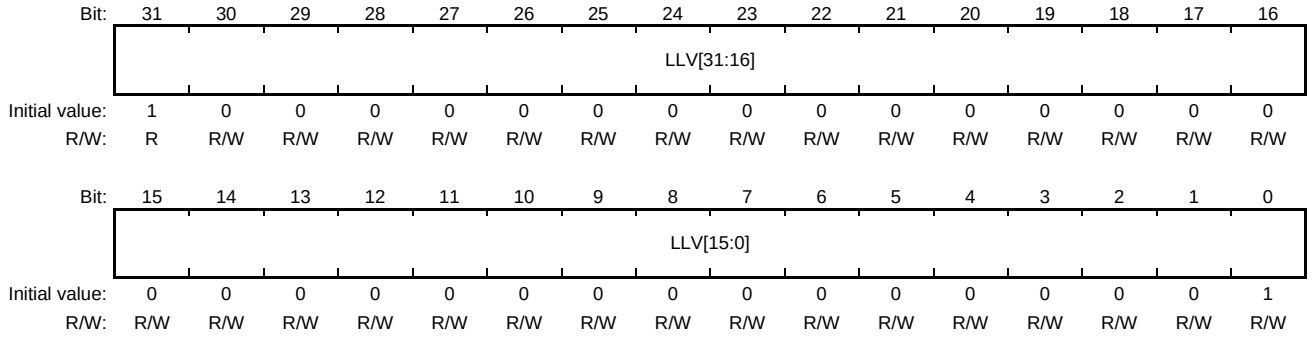

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ULV[31:0] | H'7FFF FFFF | R/W* | CBS Upper Limit<br>These bits set the upper limit for credit values calculated by using the CBS algorithm.<br>The setting is a limiting value for error detection and does not normally affect operation of the algorithm.<br>Write a positive value to these bits.<br>For details, see section 100.1.3.6 CBS (Credit-Based Shaping). |

Note* : Bit 31 is read only.

### 100.1.2.35 CBS Lower Limit Register c (CLLc) (c = 0 or 1)

The CUL0 register sets the lower limit for credit values calculated by using the CBS algorithm for transmission queue 2 (for stream class B).

The CUL1 register sets the lower limit for credit values calculated by using the CBS algorithm for transmission queue 3 (for stream class A).




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LLV[31:0] | H'8000 0001 | R/W* | CBS Lower Limit<br>These bits set the lower limit for credit values calculated by using the CBS algorithm.<br>The setting is a limiting value for error detection and does not normally affect operation of the algorithm.<br>Write a negative value to these bits.<br>For details, see section 100.1.3.6 CBS (Credit-Based Shaping). |

Note* : Bit 31 is read only.

### 100.1.2.36 Descriptor Interrupt Control Register (DIC)

The DIC register is used to control descriptor interrupts 1 to 15.

When an interrupt source flag is set (a bit from DPF1 to DPF15 bits in the descriptor interrupt status register (DIS) = 1) while the interrupt is enabled, the interrupt is issued.

[Changing condition]

This bit is set to 0b when writing 1b to **DID.DPDi**.

This bit is set to 1b when writing 1b to **DIE.DPSi**.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPE15 | DPE14 | DPE13 | DPE12 | DPE11 | DPE10 | DPE9 | DPE8 | DPE7 | DPE6 | DPE5 | DPE4 | DPE3 | DPE2 | DPE1 | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 | DPE15 | 0 | R/W | Descriptor Interrupt Enable 15<br>0: Disabled<br>1: Enabled |
| 14 | DPE14 | 0 | R/W | Descriptor Interrupt Enable 14<br>0: Disabled<br>1: Enabled |
| 13 | DPE13 | 0 | R/W | Descriptor Interrupt Enable 13<br>0: Disabled<br>1: Enabled |
| 12 | DPE12 | 0 | R/W | Descriptor Interrupt Enable 12<br>0: Disabled<br>1: Enabled |
| 11 | DPE11 | 0 | R/W | Descriptor Interrupt Enable 11<br>0: Disabled<br>1: Enabled |
| 10 | DPE10 | 0 | R/W | Descriptor Interrupt Enable 10<br>0: Disabled<br>1: Enabled |
| 9 | DPE9 | 0 | R/W | Descriptor Interrupt Enable 9<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8 | DPE8 | 0 | R/W | Descriptor Interrupt Enable 8<br>0: Disabled<br>1: Enabled |
| 7 | DPE7 | 0 | R/W | Descriptor Interrupt Enable 7<br>0: Disabled<br>1: Enabled |
| 6 | DPE6 | 0 | R/W | Descriptor Interrupt Enable 6<br>0: Disabled<br>1: Enabled |
| 5 | DPE5 | 0 | R/W | Descriptor Interrupt Enable 5<br>0: Disabled<br>1: Enabled |
| 4 | DPE4 | 0 | R/W | Descriptor Interrupt Enable 4<br>0: Disabled<br>1: Enabled |
| 3 | DPE3 | 0 | R/W | Descriptor Interrupt Enable 3<br>0: Disabled<br>1: Enabled |
| 2 | DPE2 | 0 | R/W | Descriptor Interrupt Enable 2<br>0: Disabled<br>1: Enabled |
| 1 | DPE1 | 0 | R/W | Descriptor Interrupt Enable 1<br>0: Disabled<br>1: Enabled |
| 0 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.37 Descriptor Interrupt Status Register (DIS)

The DIS register indicates the state of descriptor interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPF15 | DPF14 | DPF13 | DPF12 | DPF11 | DPF10 | DPF9 | DPF8 | DPF7 | DPF6 | DPF5 | DPF4 | DPF3 | DPF2 | DPF1 | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 | DPF15 | 0 | R/W | Descriptor Interrupt Status15<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 14 | DPF14 | 0 | R/W | Descriptor Interrupt Status14<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 13 | DPF13 | 0 | R/W | Descriptor Interrupt Status13<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 12 | DPF12 | 0 | R/W | Descriptor Interrupt Status12<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 11 | DPF11 | 0 | R/W | Descriptor Interrupt Status11<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 10 | DPF10 | 0 | R/W | Descriptor Interrupt Status10<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 9 | DPF9 | 0 | R/W | Descriptor Interrupt Status9<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 8 | DPF8 | 0 | R/W | Descriptor Interrupt Status8<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 7 | DPF7 | 0 | R/W | Descriptor Interrupt Status7<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 6 | DPF6 | 0 | R/W | Descriptor Interrupt Status6<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | DPF5 | 0 | R/W | Descriptor Interrupt Status5<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 4 | DPF4 | 0 | R/W | Descriptor Interrupt Status4<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 3 | DPF3 | 0 | R/W | Descriptor Interrupt Status3<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 2 | DPF2 | 0 | R/W | Descriptor Interrupt Status2<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 1 | DPF1 | 0 | R/W | Descriptor Interrupt Status1<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 0 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

**DPF1 to DPF15 Descriptor Interrupt Status Bits:**

When DESCR.DIE is 1 to 15, the corresponding bit indicates completion of the processing of a descriptor within the reception or transmission queue.

When DESCR.DIE is 0, the descriptor interrupt is not generated.

When DESCR.DIE is 0001b, in addition a queue specific descriptor interrupt is generated.

Only 0 can be written to these bits.

**[Conditions for Changing]**

- A bit is set to 0 when the operating mode is not operation mode.
- A bit is set to 1 when a descriptor with DESCR.DIE set to the corresponding number from 1 to 15 is processed.

### 100.1.2.38 Error Interrupt Control Register (EIC)

The EIC register controls the AVB-DMAC-related error interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TBFE | MFFE | TFFE | CULE1 | CULE0 | CLLE1 | CLLE0 | SEE | QEE | MTEE | MREE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 | TBFE | 0 | R/W | Tx-Buffer Full interrupt Enable<br>While this bit is 1b an interrupt will be generated when **EIS.TBFF** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.TBFD**.<br>This bit is set to 1b when writing 1b to **EIE.TBFS**.<br>0: Disabled<br>1: Enabled |
| 9 | MFFE | 0 | R/W | E-MAC status FIFO Full interrupt Enable<br>While this bit is 1b an interrupt will be generated when **EIS.MFFF** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.MFFD**.<br>This bit is set to 1b when writing 1b to **EIE.MFFS**.<br>0: Disabled<br>1: Enabled |
| 8 | TFFE | 0 | R/W | Time Stamp FIFO Full-Error Interrupt Enable<br>When the time stamp FIFO is full (TFFF in the error interrupt status register (EIS) = 1) and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.TFFD**.<br>This bit is set to 1b when writing 1b to **EIE.TFFS**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | CULE1 | 0 | R/W | CBS Upper Limit Error Interrupt Enable (Class A)<br>When the Class A CBS reaches its upper limit (CULF1 in the error interrupt status register (EIS) = 1), the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.CULD1**.<br>This bit is set to 1b when writing 1b to **EIE.CULS1**.<br>0: Disabled<br>1: Enabled |
| 6 | CULE0 | 0 | R/W | CBS Upper Limit Error Interrupt Enable (Class B)<br>When the Class B CBS reaches its upper limit (CULF0 in the error interrupt status register (EIS) = 1), the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.CULD0**.<br>This bit is set to 1b when writing 1b to **EIE.CULS0**.<br>0: Disabled<br>1: Enabled |
| 5 | CLLE1 | 0 | R/W | CBS Lower Limit Error Interrupt Enable (Class A)<br>When the Class A CBS reaches its lower limit (CLLF1 in the error interrupt status register (EIS) = 1), the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.CLLD1**.<br>This bit is set to 1b when writing 1b to **EIE.CLLS1**.<br>0: Disabled<br>1: Enabled |
| 4 | CLLE0 | 0 | R/W | CBS Lower Limit Error Interrupt Enable (Class B)<br>When the Class B CBS reaches its lower limit (CLLF0 in the error interrupt status register (EIS) = 1), the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.CLLD0**.<br>This bit is set to 1b when writing 1b to **EIE.CLLS0**.<br>0: Disabled<br>1: Enabled |
| 3 | SEE | 0 | R/W | Separation Error Interrupt Enable<br>While this bit is 1, an interrupt will be generated when the EIS.SEF bit is 1.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.SED**.<br>This bit is set to 1b when writing 1b to **EIE.SES**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | QEE | 0 | R/W | Queue Error Interrupt Enable<br>While this bit is 1, an interrupt will be generated when the EIS.QEF bit is 1.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.QED**.<br>This bit is set to 1b when writing 1b to **EIE.QES**.<br>0: Disabled<br>1: Enabled |
| 1 | MTEE | 0 | R/W | E-MAC Transmission Error Interrupt Enable<br>While this bit is 1, an interrupt will be generated when the EIS.MTEF bit is 1.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.MTED**.<br>This bit is set to 1b when writing 1b to **EIE.MTES**.<br>0: Disabled<br>1: Enabled |
| 0 | MREE | 0 | R/W | E-MAC Reception Error Interrupt Enable<br>While this bit is 1, an interrupt will be generated when the EIS.MREF bit is 1.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **EID.MRED**.<br>This bit is set to 1b when writing 1b to **EIE.MRES**.<br>0: Disabled<br>1: Enabled |

### 100.1.2.39 Error Interrupt Status Register (EIS)

The EIS register indicates the states of AVB-DMAC-related error interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | QFS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TBFF | MFFF | TFFF | CULF1 | CULF0 | CLLF1 | CLLF0 | SEF | QEF | MTEF | MREF |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 16 | QFS | 0 | R | Queue Full Error Interrupt Status<br>With the interrupts enabled, this bit indicates that a queue is full (the receive queue r full interrupt status bit (QFFr) or the receive FIFO full interrupt status bit (RFFF) in receive interrupt status register 2 (RIS2) = 1).<br>[Conditions for Changing]<br>— If the receive queue r full interrupt status bit (RIS2.QFFr) and/or the receive queue r full interrupt enable bit in the - receive interrupt control register 2 (RIC2.QFEr) are updated, this bit is also updated.<br>— If the receive FIFO full interrupt status bit (RIS2.RFFF) and/or the receive FIFO full interrupt enable bit (RIC2.RFFE) are updated, this bit is also updated.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 15 to 11 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 | TBFF | 0 | R/W | Tx-Buffer Full Flag<br>This bit indicates that fetching transmit data was delayed because the Tx-Buffer has no sufficient storage available to store fetched frame. It depends on configuration of **TGC.TBDt** (refer to section 100.1.3.5(1)) if a full Tx-Buffer is expected during normal operation.<br>No frame gets lost but transmission priority may be influenced.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when frame data has been fetched from URAM but there is no storage available in Tx-Buffer.<br>0: No interrupt pending<br>1: Tx-Buffer full condition detected |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | MFFF | 0 | R/W | E-MAC status FIFO Full Flag<br>This bit indicates that a transmission E-MAC status is overwritten because the E-MAC status FIFO is full (overwrite condition).<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a frame with **DESCR.MSR** has been transmitted by E-MAC, **TCCR.MFEN** is 1b and **TSR.MFFL** is 16.<br>This bit is set to 1b when E-MAC detects an error during transmission and **TSR.MFFL** is 16.<br>0: No interrupt pending<br>1: E-MAC status FIFO full, oldest E-MAC status lost |
| 8 | TFFF | 0 | R/W | Time Stamp FIFO Full Error Interrupt Status<br>This bit indicates that a new transmission time stamp has been discarded due to the time-stamp FIFO being full (i.e. has reached the overflow state).<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— The bit is set to 0 when the operating mode is not operation mode.<br>— The bit is set to 1 when a frame with DESCR.TSR set is transmitted while the time stamp FIFO enable bit in the transmit configuration control register (TCCR.TFEN) is set to 1 and the time stamp FIFO count bit in the transmit status register (TSR.TFFL) is set to 3.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 7 | CULF1 | 0 | R/W | CBS Upper Limit Error Interrupt Status (Class A)<br>This bit indicates that CBS counter 1 has exceeded the set upper limit (CUL1.ULV in the CBS upper limit register c (CULc)).<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— This bit is set to 0 when the operating mode is not operation mode.<br>— This bit is set to 1 when the value of the CBS counter status 1 (Class A) bits in the transmit status register (TSR.CCS1) change from B'00 (indicating a value within the range between the limits) to B'10 (indicating a value over the upper limit).<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | CULF0 | 0 | R/W | CBS Upper Limit Error Interrupt Status (Class B)<br>This bit indicates that CBS counter 0 has exceeded the set upper limit (CUL0.ULV in the CBS upper limit register c (CULc)).<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— The bit is set to 0 when the operating mode is not operation mode.<br>— The bit is set to 1 when the value of the CBS counter status 0 (Class B) bit in the transmit status register changes from B'00 (indicating a value within the range between the limits) to B'10 (indicating a value over the upper limit).<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 5 | CLLF1 | 0 | R/W | CBS Lower Limit Error Interrupt Status (Class A)<br>This bit indicates that CBS counter 1 has fallen below the set lower limit (CLL1.LLV in CBS lower limit register c (CLLc)).<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— The bit is set to 0 when the operating mode is not operation mode.<br>— The bit is set to 1 when the value of the CBS counter status 1 (Class A) bit in the transmit status register (TSR.CCS1) changes from B'00 (indicating a value within the range between the limits) to B'01 (indicating a value less than the lower limit).<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 4 | CLLF0 | 0 | R/W | CBS Lower Limit Error Interrupt Status (Class B)<br>This bit indicates that CBS counter 0 has fallen below the set lower limit (CLL0.LLV in the CBS lower limit register c (CLLc)).<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— The bit is set to 0 when the operating mode is not operation mode.<br>— The bit is set to 1 when the value of the CBS counter status 0 (Class B) bit in the transmit status register (TSR.CCS0) changes from B'00 (indicating a value within the range between the limits) to B'01 (indicating a value less than the lower limit).<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | SEF | 0 | R/W | Separation Error Flag<br>This bit indicates that a received frame was discarded because it has not matched any configurated separation filter for AVB stream data frames.<br>The CPU can only write 0 to this bit.<br>[Conditions for Changing]<br>— This bit is set to 0 when leaving operation mode.<br>— This bit is set to 1 when a valid AVB stream data frame was received by E-MAC but discarded because the RCR.ESF bits are B'10 and no separation filter has matched.<br>0: No interrupt pending.<br>1: AVB stream data frame has discarded. |
| 2 | QEF | 0 | R/W | Queue Error Flag<br>This bit indicates that an error has been detected while processing reception or transmit queue or while processing AVTP FIFO entries.<br>Details about the detected error is indicated by ESR.<br>Section 100.1.3.2(2) gives an overview of detail error conditions and the required interaction.<br>The CPU can only write 0 to this bit.<br>[Conditions for Changing]<br>— This bit is set to 0 when leaving operating mode.<br>— This bit is set to 1 when an error condition is detected.<br>0: No interrupt pending.<br>1: Interrupt pending. |
| 1 | MTEF | 0 | R/W | E-MAC Transmission Error Flag<br>This bit indicates that the E-MAC has detected a fault during transmission.<br>For detail, the E-MAC registers have to be checked.<br>The CPU can only write 0 to this bit.<br>[Conditions for Changing]<br>— This bit is set to 0 when leaving operating mode.<br>— This bit is set to 1 when E-MAC detects an error during frame transmission.<br>0: No interrupt pending.<br>1: E-MAC has reported an error during transmission. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | MREF | 0 | R/W | E-MAC Reception Error Flag<br>This bit indicates that the E-MAC has detected a fault during reception.<br>For detail, the E-MAC registers have to be checked.<br>Note: When the storage of faulty received frames (RCR.EFFS) is enabled, the E-MAC error code (DESCR.MSC) is stored in the descriptor. By evaluating this information, CPU can identify corrupted frames in URAM.<br>The CPU can only write 0 to this bit.<br>[Conditions for Changing]<br>— This bit is set to 0 when leaving operating mode.<br>— This bit is set to 1 when E-MAC detects an error during frame reception.<br>0: No interrupt pending.<br>1: E-MAC has reported an error during reception. |

### 100.1.2.40 Receive Interrupt Control Register 0 (RIC0)

The RIC0 register controls the AVB-DMAC receive interrupts.

When an interrupt source flag is set (a bit from among the receive interrupt status bits in the receive interrupt status register (RIS0.FRF0 to 17) = 1) while the interrupt is enabled, the interrupt is issued.

[Changing condition]

This bit is set to 0b when writing 1b to **RID0.FRDr**.

This bit is set to 1b when writing 1b to **RIE0.FRSr**.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FRE17 | FRE16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FRE15 | FRE14 | FRE13 | FRE12 | FRE11 | FRE10 | FRE9 | FRE8 | FRE7 | FRE6 | FRE5 | FRE4 | FRE3 | FRE2 | FRE1 | FRE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | FRE17 | 0 | R/W | Receive Frame Enable 17 (Stream)<br>0: Disabled<br>1: Enabled |
| 16 | FRE16 | 0 | R/W | Receive Frame Enable 16 (Stream)<br>0: Disabled<br>1: Enabled |
| 15 | FRE15 | 0 | R/W | Receive Frame Enable 15 (Stream)<br>0: Disabled<br>1: Enabled |
| 14 | FRE14 | 0 | R/W | Receive Frame Enable 14 (Stream)<br>0: Disabled<br>1: Enabled |
| 13 | FRE13 | 0 | R/W | Receive Frame Enable 13 (Stream)<br>0: Disabled<br>1: Enabled |
| 12 | FRE12 | 0 | R/W | Receive Frame Enable 12 (Stream)<br>0: Disabled<br>1: Enabled |
| 11 | FRE11 | 0 | R/W | Receive Frame Enable 11 (Stream)<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | FRE10 | 0 | R/W | Receive Frame Enable 10 (Stream)<br>0: Disabled<br>1: Enabled |
| 9 | FRE9 | 0 | R/W | Receive Frame Enable 9 (Stream)<br>0: Disabled<br>1: Enabled |
| 8 | FRE8 | 0 | R/W | Receive Frame Enable 8 (Stream)<br>0: Disabled<br>1: Enabled |
| 7 | FRE7 | 0 | R/W | Receive Frame Enable 7 (Stream)<br>0: Disabled<br>1: Enabled |
| 6 | FRE6 | 0 | R/W | Receive Frame Enable 6 (Stream)<br>0: Disabled<br>1: Enabled |
| 5 | FRE5 | 0 | R/W | Receive Frame Enable 5 (Stream)<br>0: Disabled<br>1: Enabled |
| 4 | FRE4 | 0 | R/W | Receive Frame Enable 4 (Stream)<br>0: Disabled<br>1: Enabled |
| 3 | FRE3 | 0 | R/W | Receive Frame Enable 3 (Stream)<br>0: Disabled<br>1: Enabled |
| 2 | FRE2 | 0 | R/W | Receive Frame Enable 2 (Stream)<br>0: Disabled<br>1: Enabled |
| 1 | FRE1 | 0 | R/W | Receive Frame Enable 1 (Network Control)<br>0: Disabled<br>1: Enabled |
| 0 | FRE0 | 0 | R/W | Receive Frame Enable 0 (Best Effort)<br>0: Disabled<br>1: Enabled |

### 100.1.2.41 Receive Interrupt Status Register 0 (RIS0)

The RIS0 register indicates the states of the AVB-DMAC receive interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FRF17 | FRF16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FRF15 | FRF14 | FRF13 | FRF12 | FRF11 | FRF10 | FRF9 | FRF8 | FRF7 | FRF6 | FRF5 | FRF4 | FRF3 | FRF2 | FRF1 | FRF0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | FRF17 | 0 | R/W | Receive Frame Interrupt Status 17 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 16 | FRF16 | 0 | R/W | Receive Frame Interrupt Status 16 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 15 | FRF15 | 0 | R/W | Receive Frame Interrupt Status 15 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 14 | FRF14 | 0 | R/W | Receive Frame Interrupt Status 14 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 13 | FRF13 | 0 | R/W | Receive Frame Interrupt Status 13 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 12 | FRF12 | 0 | R/W | Receive Frame Interrupt Status 12 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 11 | FRF11 | 0 | R/W | Receive Frame Interrupt Status 11 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 10 | FRF10 | 0 | R/W | Receive Frame Interrupt Status 10 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 9 | FRF9 | 0 | R/W | Receive Frame Interrupt Status 9 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 8 | FRF8 | 0 | R/W | Receive Frame Interrupt Status 8 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | FRF7 | 0 | R/W | Receive Frame Interrupt Status 7 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 6 | FRF6 | 0 | R/W | Receive Frame Interrupt Status 6 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 5 | FRF5 | 0 | R/W | Receive Frame Interrupt Status 5 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 4 | FRF4 | 0 | R/W | Receive Frame Interrupt Status 4 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 3 | FRF3 | 0 | R/W | Receive Frame Interrupt Status 3 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 2 | FRF2 | 0 | R/W | Receive Frame Interrupt Status 2 (Stream)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 1 | FRF1 | 0 | R/W | Receive Frame Interrupt Status 1 (Network Control)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 0 | FRF0 | 0 | R/W | Receive Frame Interrupt Status 0 (Best Effort)<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

**FRF0 to FRF17 Receive Frame Interrupt Status Bits 0 to 17:**

Each bit indicates that a corresponding frame has been stored normally in reception queues 0 to 17 and that data are ready for CPU processing.

Only 0 can be written to the bit.

**[Conditions for Changing]**

- A bit is set to 0 when the operating mode is not operation mode.
- A bit is set to 0 when a value is written to the unread frame counter decrement register i (UFCDi) (i = 0 to 4), and this decrements the value of unread frame counter register i (UFCVi) (i = 0 to 4) to 0.
- When a frame is stored normally in a reception queue, the corresponding bit is set to 1.

### 100.1.2.42 Receive Interrupt Control Register 1 (RIC1)

The RIC1 register controls the AVB-DMAC receive interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFWE | — | — | — | — | — | — | — | — | — | — | — | — | — | RWE17 | RWE16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RWE15 | RWE14 | RWE13 | RWE12 | RWE11 | RWE10 | RWE9 | RWE8 | RWE7 | RWE6 | RWE5 | RWE4 | RWE3 | RWE2 | RWE1 | RWE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RFWE | 0 | R/W | Receive FIFO Warning Interrupt Enable<br>If the reception FIFO reaches the caution level (the value set in the receive FIFO caution level bits in the receive configuration register (RCR.RFCL)) with the corresponding interrupt enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RFWD**.<br>This bit is set to 1b when writing 1b to **RIE1.RFWS**.<br>0: Disabled<br>1: Enabled |
| 30 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | RWE17 | 0 | R/W | Reception Warning interrupt Enable 17<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF17** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD17**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS17**.<br>0: Disabled<br>1: Enabled |
| 16 | RWE16 | 0 | R/W | Reception Warning interrupt Enable 16<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF16** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD16**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS16**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | RWE15 | 0 | R/W | Reception Warning interrupt Enable 15<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF15** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD15**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS15**.<br>0: Disabled<br>1: Enabled |
| 14 | RWE14 | 0 | R/W | Reception Warning interrupt Enable 14<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF14** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD14**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS14**.<br>0: Disabled<br>1: Enabled |
| 13 | RWE13 | 0 | R/W | Reception Warning interrupt Enable 13<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF13** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD13**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS13**.<br>0: Disabled<br>1: Enabled |
| 12 | RWE12 | 0 | R/W | Reception Warning interrupt Enable 12<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF12** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD12**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS12**.<br>0: Disabled<br>1: Enabled |
| 11 | RWE11 | 0 | R/W | Reception Warning interrupt Enable 11<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF11** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD11**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS11**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | RWE10 | 0 | R/W | Reception Warning interrupt Enable 10<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF10** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD10**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS10**.<br>0: Disabled<br>1: Enabled |
| 9 | RWE9 | 0 | R/W | Reception Warning interrupt Enable 9<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF9** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD9**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS9**.<br>0: Disabled<br>1: Enabled |
| 8 | RWE8 | 0 | R/W | Reception Warning interrupt Enable 8<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF8** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD18**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS18**.<br>0: Disabled<br>1: Enabled |
| 7 | RWE7 | 0 | R/W | Reception Warning interrupt Enable 7<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF7** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD7**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS7**.<br>0: Disabled<br>1: Enabled |
| 6 | RWE6 | 0 | R/W | Reception Warning interrupt Enable 6<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF6** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD6**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS6**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | RWE5 | 0 | R/W | Reception Warning interrupt Enable 5<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF5** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD5**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS5**.<br>0: Disabled<br>1: Enabled |
| 4 | RWE4 | 0 | R/W | Reception Warning interrupt Enable 4<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF4** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD4**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS4**.<br>0: Disabled<br>1: Enabled |
| 3 | RWE3 | 0 | R/W | Reception Warning interrupt Enable 3<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF3** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD3**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS3**.<br>0: Disabled<br>1: Enabled |
| 2 | RWE2 | 0 | R/W | Reception Warning interrupt Enable 2<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF2** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD2**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS2**.<br>0: Disabled<br>1: Enabled |
| 1 | RWE1 | 0 | R/W | Reception Warning interrupt Enable 1<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF1** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD1**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS1**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | RWE0 | 0 | R/W | Reception Warning interrupt Enable 0<br>While this bit is 1b an interrupt will be generated when **RIS1.RWF0** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID1.RWD0**.<br>This bit is set to 1b when writing 1b to **RIE1.RWS0**.<br>0: Disabled<br>1: Enabled |

### 100.1.2.43 Receive Interrupt Status Register 1 (RIS1)

The RIS1 register indicates the states of the AVB-DMAC receive interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFWF | — | — | — | — | — | — | — | — | — | — | — | — | — | RWF17 | RWF16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RWF15 | RWF14 | RWF13 | RWF12 | RWF11 | RWF10 | RWF9 | RWF8 | RWF7 | RWF6 | RWF5 | RWF4 | RWF3 | RWF2 | RWF1 | RWF0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RFWF | 0 | R/W | Receive FIFO Warning Interrupt Status<br>This bit indicates that the reception FIFO has exceeded the set caution level (the value set in the receive FIFO caution level bits in the receive configuration register (RCR.RFCL)).<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— The bit is set to 0 when the operating mode is not operation mode.<br>— The bit is set to 1 when the reception FIFO exceeded the set caution level (the value set in the receive FIFO caution level bits in the receive configuration register (RCR.RFCL)).<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 30 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | RWF17 | 0 | R/W | Reception Warning Flag 17<br>This bit indicates that the unread frame counter of receive queue 17 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA21** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD4.DV1** and **UFCV4.CV1** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV4.CV1** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | RWF16 | 0 | R/W | Reception Warning Flag 16<br>This bit indicates that the unread frame counter of receive queue 16 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA20** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD4.DV0** and **UFCV4.CV0** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV4.CV0** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 15 | RWF15 | 0 | R/W | Reception Warning Flag 15<br>This bit indicates that the unread frame counter of receive queue 15 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA19** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD3.DV3** and **UFCV3.CV3** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV3.CV3** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 14 | RWF14 | 0 | R/W | Reception Warning Flag 14<br>This bit indicates that the unread frame counter of receive queue 14 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA18** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD3.DV2** and **UFCV3.CV2** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV3.CV2** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 13 | RWF13 | 0 | R/W | Reception Warning Flag 13<br>This bit indicates that the unread frame counter of receive queue 13 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA17** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD3.DV1** and **UFCV3.CV1** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV3.CV1** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 12 | RWF12 | 0 | R/W | Reception Warning Flag 12<br>This bit indicates that the unread frame counter of receive queue 12 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA16** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD3.DV0** and **UFCV3.CV0** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV3.CV0** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 11 | RWF11 | 0 | R/W | Reception Warning Flag 11<br>This bit indicates that the unread frame counter of receive queue 11 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA15** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD2.DV3** and **UFCV2.CV3** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV2.CV3** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | RWF10 | 0 | R/W | Reception Warning Flag 10<br>This bit indicates that the unread frame counter of receive queue 10 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA14** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD2.DV2** and **UFCV2.CV2** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV2.CV2** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 9 | RWF9 | 0 | R/W | Reception Warning Flag 9<br>This bit indicates that the unread frame counter of receive queue 9 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA13** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD2.DV1** and **UFCV2.CV1** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV2.CV1** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 8 | RWF8 | 0 | R/W | Reception Warning Flag 8<br>This bit indicates that the unread frame counter of receive queue 8 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA12** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD2.DV0** and **UFCV2.CV0** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV2.CV0** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | RWF7 | 0 | R/W | Reception Warning Flag 7<br>This bit indicates that the unread frame counter of receive queue 7 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA11** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD1.DV3** and **UFCV1.CV3** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV1.CV3** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 6 | RWF6 | 0 | R/W | Reception Warning Flag 6<br>This bit indicates that the unread frame counter of receive queue 6 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA10** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD1.DV2** and **UFCV1.CV2** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV1.CV2** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 5 | RWF5 | 0 | R/W | Reception Warning Flag 5<br>This bit indicates that the unread frame counter of receive queue 5 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA9** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD1.DV1** and **UFCV1.CV1** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV1.CV1** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | RWF4 | 0 | R/W | Reception Warning Flag 4<br>This bit indicates that the unread frame counter of receive queue 4 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA8** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD1.DV0** and **UFCV1.CV0** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV1.CV0** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 3 | RWF3 | 0 | R/W | Reception Warning Flag 3<br>This bit indicates that the unread frame counter of receive queue 3 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA7** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD0.DV3** and **UFCV0.CV3** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV0.CV3** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 2 | RWF2 | 0 | R/W | Reception Warning Flag 2<br>This bit indicates that the unread frame counter of receive queue 2 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA6** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD0.DV2** and **UFCV0.CV2** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV0.CV2** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | RWF1 | 0 | R/W | Reception Warning Flag 1<br>This bit indicates that the unread frame counter of receive queue 1 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA5** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD0.DV1** and **UFCV0.CV1** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV0.CV1** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |
| 0 | RWF0 | 0 | R/W | Reception Warning Flag 0<br>This bit indicates that the unread frame counter of receive queue 0 has reached the configured warning level.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when CPU request a reload of the base address (**DLR.LBA4** is 1b).<br>This bit is set to 0b when CPU writes to **UFCD0.DV0** and **UFCV0.CV0** is decremented to a value less than the selected warning level **UFCW.WLj**.<br>This bit is set to 1b when a frame is stored in queue r and the unread frame counter value r is incremented to the selected warning level j (**UFCV0.CV0** + 1 == **UFCW.WLj**).<br>0: No interrupt pending<br>1: Unread frame counter warning level reached |

### 100.1.2.44 Receive Interrupt Control Register 2 (RIC2)

The RIC2 register controls the AVB-DMAC receive interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFFE | — | — | — | — | — | — | — | — | — | — | — | — | — | QFE17 | QFE16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QFE15 | QFE14 | QFE13 | QFE12 | QFE11 | QFE10 | QFE9 | QFE8 | QFE7 | QFE6 | QFE5 | QFE4 | QFE3 | QFE2 | QFE1 | QFE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RFFE | 0 | R/W | Receive FIFO Full Interrupt Enable<br>When the reception FIFO is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.RFFD**.<br>This bit is set to 1b when writing 1b to **RIE2.RFFS**.<br>0: Disabled<br>1: Enabled |
| 30 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | QFE17 | 0 | R/W | Receive Queue 17 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD17**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS17**.<br>0: Disabled<br>1: Enabled |
| 16 | QFE16 | 0 | R/W | Receive Queue 16 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD16**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS16**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | QFE15 | 0 | R/W | Receive Queue 15 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD15**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS15**.<br>0: Disabled<br>1: Enabled |
| 14 | QFE14 | 0 | R/W | Receive Queue 14 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD14**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS14**.<br>0: Disabled<br>1: Enabled |
| 13 | QFE13 | 0 | R/W | Receive Queue 13 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD13**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS13**.<br>0: Disabled<br>1: Enabled |
| 12 | QFE12 | 0 | R/W | Receive Queue 12 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD12**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS12**.<br>0: Disabled<br>1: Enabled |
| 11 | QFE11 | 0 | R/W | Receive Queue 11 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD11**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS11**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | QFE10 | 0 | R/W | Receive Queue 10 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD10**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS10**.<br>0: Disabled<br>1: Enabled |
| 9 | QFE9 | 0 | R/W | Receive Queue 9 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD9**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS9**.<br>0: Disabled<br>1: Enabled |
| 8 | QFE8 | 0 | R/W | Receive Queue 8 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD8**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS8**.<br>0: Disabled<br>1: Enabled |
| 7 | QFE7 | 0 | R/W | Receive Queue 7 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD7**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS7**.<br>0: Disabled<br>1: Enabled |
| 6 | QFE6 | 0 | R/W | Receive Queue 6 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD6**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS6**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | QFE5 | 0 | R/W | Receive Queue 5 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD5**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS5**.<br>0: Disabled<br>1: Enabled |
| 4 | QFE4 | 0 | R/W | Receive Queue 4 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD4**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS4**.<br>0: Disabled<br>1: Enabled |
| 3 | QFE3 | 0 | R/W | Receive Queue 3 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD3**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS3**.<br>0: Disabled<br>1: Enabled |
| 2 | QFE2 | 0 | R/W | Receive Queue 2 (Stream) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD2**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS2**.<br>0: Disabled<br>1: Enabled |
| 1 | QFE1 | 0 | R/W | Receive Queue 1 (Network Control) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD1**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS1**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | QFE0 | 0 | R/W | Receive Queue 0 (Best Effort) Full Interrupt Enable<br>When a reception queue is full and the interrupt is enabled, the interrupt is issued.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID2.QFD0**.<br>This bit is set to 1b when writing 1b to **RIE2.QFS0**.<br>0: Disabled<br>1: Enabled |

### 100.1.2.45 Receive Interrupt Status Register 2 (RIS2)

The RIS2 register indicates the states of the AVB-DMAC receive interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFFF | — | — | — | — | — | — | — | — | — | — | — | — | — | QFF17 | QFF16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QFF15 | QFF14 | QFF13 | QFF12 | QFF11 | QFF10 | QFF9 | QFF8 | QFF7 | QFF6 | QFF5 | QFF4 | QFF3 | QFF2 | QFF1 | QFF0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RFFF | 0 | R/W | Receive FIFO Full Interrupt Status<br>This bit indicates that a frame was received but storing it was not possible due to the reception FIFO being full.<br>When receiving a frame is not possible, the frame will be discarded.<br>Other information regarding discarded frames is not retained. Even if the frame is discarded, this bit may also be set to 1 if the E-MAC determines that the frame is an error frame<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— The bit is set to 0 when the operating mode is not operation mode.<br>— The bit is set to 1 when frame data provided by the E-MAC cannot be stored in the reception FIFO.<br>0: The interrupt is not pending.<br>1: The interrupt is pending |
| 30 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | QFF17 | 0 | R/W | Receive Queue 17 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 16 | QFF16 | 0 | R/W | Receive Queue 16 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 15 | QFF15 | 0 | R/W | Receive Queue 15 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 14 | QFF14 | 0 | R/W | Receive Queue 14 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 13 | QFF13 | 0 | R/W | Receive Queue 13 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 | QFF12 | 0 | R/W | Receive Queue 12 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 11 | QFF11 | 0 | R/W | Receive Queue 11 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 10 | QFF10 | 0 | R/W | Receive Queue 10 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 9 | QFF9 | 0 | R/W | Receive Queue 9 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 8 | QFF8 | 0 | R/W | Receive Queue 8 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 7 | QFF7 | 0 | R/W | Receive Queue 7 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 6 | QFF6 | 0 | R/W | Receive Queue 6 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 5 | QFF5 | 0 | R/W | Receive Queue 5 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 4 | QFF4 | 0 | R/W | Receive Queue 4 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 3 | QFF3 | 0 | R/W | Receive Queue 3 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 2 | QFF2 | 0 | R/W | Receive Queue 2 (Stream) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 1 | QFF1 | 0 | R/W | Receive Queue 1 (Network Control) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 0 | QFF0 | 0 | R/W | Receive Queue 0 (Best Effort) Full Interrupt Status<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

**QFF0 to 17 Receive 0 to 17 Full Interrupt Status Bits:**

These bits indicate that reception queue r did not have space for storing a received frame.

A reception queue is treated as full when it has no descriptors (descriptor type (DESCR.DT) = FEMPTY, FEMPTY_IS, FEMPTY_IC, or FEMPTY_ND) available or reaches the set level for stopping.

Note: If no FEMPTY descriptors or no empty space for descriptors remains in the queue during storing of a divided frame (see section 100.1.3.4(3)(a) Storing Frame Data in the Descriptor Data Area, for storing a frame as a divided frame), an error frame is stored in the queue. Such error frames are treated as descriptor sequence errors.

**[Conditions for Changing]**

- A bit is set to 0 when the operating mode is not operation mode.
- A bit is set to 1 when reception queue r has no space available for storage.
- A bit is set to 1 when a SW defined stop point (EOS descriptor) reached within a split frame.
- A bit is set to 1 when the unread frame counter (unread frame counter register i (UFCVi) (i = 0 to 4)) reaches the set level for stopping.

### 100.1.2.46 Transmit Interrupt Control Register (TIC)

The TIC register controls the AVB-DMAC transmit interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TDPE3 | TDPE2 | TDPE1 | TDPE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MFWE | MFUE | TFWE | TFUE | — | — | — | — | FTE3 | FTE2 | FTE1 | FTE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 19 | TDPE3 | 0 | R/W | Transmit Descriptor Processed interrupt Enable 3<br>While this bit is 1b an interrupt will be generated when **TIS.TDPF3** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.TDPD3**.<br>This bit is set to 1b when writing 1b to **TIE.TDPS3**.<br>0: Disabled<br>1: Enabled |
| 18 | TDPE2 | 0 | R/W | Transmit Descriptor Processed interrupt Enable 2<br>While this bit is 1b an interrupt will be generated when **TIS.TDPF2** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.TDPD2**.<br>This bit is set to 1b when writing 1b to **TIE.TDPS2**.<br>0: Disabled<br>1: Enabled |
| 17 | TDPE1 | 0 | R/W | Transmit Descriptor Processed interrupt Enable 1<br>While this bit is 1b an interrupt will be generated when **TIS.TDPF1** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.TDPD1**.<br>This bit is set to 1b when writing 1b to **TIE.TDPS1**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | TDPE0 | 0 | R/W | Transmit Descriptor Processed interrupt Enable 0<br>While this bit is 1b an interrupt will be generated when **TIS.TDPF0** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.TDPD0**.<br>This bit is set to 1b when writing 1b to **TIE.TDPS0**.<br>0: Disabled<br>1: Enabled |
| 15 to 12 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 11 | MFWE | 0 | R/W | MAC status FIFO Warning interrupt Enable<br>While this bit is 1b an interrupt will be generated when **TIS.MFWF** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.MFWD**.<br>This bit is set to 1b when writing 1b to **TIE.MFWS**.<br>0: Disabled<br>1: Enabled |
| 10 | MFUE | 0 | R/W | MAC status FIFO Updated interrupt Enable<br>While this bit is 1b an interrupt will be generated when **TIS.MFUF** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.MFUD**.<br>This bit is set to 1b when writing 1b to **TIE.MFUS**.<br>0: Disabled<br>1: Enabled |
| 9 | TFWE | 0 | R/W | Time Stamp FIFO Warning Interrupt Enable<br>When the time-stamp FIFO reaches the warning level while the interrupt is enabled, the interrupt is issued.<br>0: Disabled<br>1: Enabled |
| 8 | TFUE | 0 | R/W | Time Stamp FIFO Update Interrupt Enable<br>When the time-stamp FIFO is updated while the interrupt is enabled, the interrupt is issued.<br>0: Disabled<br>1: Enabled |
| 7 to 4 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 3 | FTE3 | 0 | R/W | Frame Transmitted interrupt Enable 3<br>While this bit is 1b an interrupt will be generated when **TIS.FTF3** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.FTD3**.<br>This bit is set to 1b when writing 1b to **TIE.FTS3**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | FTE2 | 0 | R/W | Frame Transmitted interrupt Enable 2<br>While this bit is 1b an interrupt will be generated when **TIS.FTF2** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.FTD2**.<br>This bit is set to 1b when writing 1b to **TIE.FTS2**.<br>0: Disabled<br>1: Enabled |
| 1 | FTE1 | 0 | R/W | Frame Transmitted interrupt Enable 1<br>While this bit is 1b an interrupt will be generated when **TIS.FTF1** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.FTD1**.<br>This bit is set to 1b when writing 1b to **TIE.FTS1**.<br>0: Disabled<br>1: Enabled |
| 0 | FTE0 | 0 | R/W | Frame Transmitted interrupt Enable 0<br>While this bit is 1b an interrupt will be generated when **TIS.FTF0** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **TID.FTD0**.<br>This bit is set to 1b when writing 1b to **TIE.FTS0**.<br>0: Disabled<br>1: Enabled |

### 100.1.2.47 Transmit Interrupt Status Register (TIS)

The TIS register indicates the states of the AVB-DMAC transmit interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TDPF3 | TDPF2 | TDPF1 | TDPF0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MFWF | MFUF | TFWF | TFUF | — | — | — | — | FTF3 | FTF2 | FTF1 | FTF0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 19 | TDPF3 | 0 | R/W | Transmit Descriptor Processed Flag 3<br>This bit indicates that a descriptor in transmit queue 3 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in transmit queue 3 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 18 | TDPF2 | 0 | R/W | Transmit Descriptor Processed Flag 2<br>This bit indicates that a descriptor in transmit queue 2 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in transmit queue 2 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 17 | TDPF1 | 0 | R/W | Transmit Descriptor Processed Flag 1<br>This bit indicates that a descriptor in transmit queue 1 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in transmit queue 1 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 16 | TDPF0 | 0 | R/W | Transmit Descriptor Processed Flag 0<br>This bit indicates that a descriptor in transmit queue 0 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in transmit queue 0 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 15 to 12 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 11 | MFWF | 0 | R/W | E-MAC status FIFO Warning Flag<br>This bit indicates that the warning level of the E-MAC status FIFO (12 out of 16 entries) has been reached.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when **TCCR.MFEN** is 0b.<br>This bit is set to 0b when writing 1b to **TCCR.MFR**.<br>This bit is set to 1b when a frame with **DESCR.MSR** has been transmitted by E-MAC and the E-MAC Status FIFO contains already 11 entries (**TSR.MFFL** is 11).<br>This bit is set to 1b when E-MAC detects an error during transmission and the E-MAC Status FIFO contains already 11 entries (**TSR.MFFL** is 11).<br>0: No interrupt pending<br>1: Tx Status FIFO warning level has been reached |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | MFUF | 0 | R/W | E-MAC status FIFO Updated Flag<br>This bit indicates that the E-MAC status FIFO has been updated after the E-MAC has transmitted a frame.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 0b when **TCCR.MFEN** is 0b.<br>This bit is set to 0b when writing 1b to **TCCR.MFR**.<br>This bit is set to 1b when a frame with **DESCR.MSR** has been transmitted by E-MAC and **TCCR.MFEN** is 1b.<br>This bit is set to 1b when E-MAC detects an error during transmission and **TCCR.MFEN** is 1b.<br>0: No interrupt pending<br>1: Tx Status FIFO has been updated |
| 9 | TFWF | 0 | R/W | Time Stamp FIFO Warning Interrupt Status<br>This bit indicates that the transmission time-stamp FIFO has reached the warning level.<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— The bit is set to 0 when the operating mode is not operation mode and when the time stamp FIFO enable bit in the transmit configuration control register (TCCR.TFEN) is 0.<br>— The bit is set to 1 after a frame including DESCR.TSR set has been transmitted and one entry has already been stored in the time-stamp FIFO.<br>0: The interrupt is not pending.<br>1: The time-stamp FIFO has reached the warning level. |
| 8 | TFUF | 0 | R/W | Time Stamp FIFO Update Interrupt Status<br>This bit indicates that the transmission time-stamp FIFO has been updated.<br>Only 0 can be written to the bit.<br>[Conditions for Changing]<br>— The bit is set to 0 when the operating mode is not operation mode, when the time stamp FIFO enable bit in the transmit configuration control register (TCCR.TFEN) is 0, and when 1 is written to the time stamp FIFO release bit in the transmit configuration control register (TCCR.TFR).<br>— The bit is set to 1 when the time stamp FIFO enable bit (TCCR.TFEN) is 1 after a frame including DESCR.TSR set has been transmitted.<br>0: The interrupt is not pending.<br>1: The time-stamp FIFO has been updated. |
| 7 to 4 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | FTF3 | 0 | R/W | Frame Transmitted Flag 3<br>This bit indicates that from transmit queue a frame is transmitted by E-MAC.<br>Note: This interrupt flag refers to the end of frame transmission by E-MAC whereas the descriptor interrupt (**DIS.DPFi**) refers to the end of processing storage element.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a frame from transmit queue 3 has been transmitted by the E-MAC.<br>0: No interrupt pending<br>1: Frame transmitted by E-MAC |
| 2 | FTF2 | 0 | R/W | Frame Transmitted Flag 2<br>This bit indicates that from transmit queue a frame is transmitted by E-MAC.<br>Note: This interrupt flag refers to the end of frame transmission by E-MAC whereas the descriptor interrupt (**DIS.DPFi**) refers to the end of processing storage element.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a frame from transmit queue 2 has been transmitted by the E-MAC.<br>0: No interrupt pending<br>1: Frame transmitted by E-MAC |
| 1 | FTF1 | 0 | R/W | Frame Transmitted Flag 1<br>This bit indicates that from transmit queue a frame is transmitted by E-MAC.<br>Note: This interrupt flag refers to the end of frame transmission by E-MAC whereas the descriptor interrupt (**DIS.DPFi**) refers to the end of processing storage element.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a frame from transmit queue 1 has been transmitted by the E-MAC.<br>0: No interrupt pending<br>1: Frame transmitted by E-MAC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | FTF0 | 0 | R/W | Frame Transmitted Flag 0<br>This bit indicates that from transmit queue a frame is transmitted by E-MAC.<br>Note: This interrupt flag refers to the end of frame transmission by E-MAC whereas the descriptor interrupt (**DIS.DPFi**) refers to the end of processing storage element.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a frame from transmit queue 0 has been transmitted by the E-MAC.<br>0: No interrupt pending<br>1: Frame transmitted by E-MAC |

### 100.1.2.48 Interrupt Summary Status Register (ISS)

The ISS register gives a summary of the states of AVB-DMAC-related interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPM15 | DPM14 | DPM13 | DPM12 | DPM11 | DPM10 | DPM9 | DPM8 | DPM7 | DPM6 | DPM5 | DPM4 | DPM3 | DPM2 | DPM1 | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | RFWM | MFWM | MFUM | TFWM | TFUM | MM | EM | — | — | — | FTM | RWM | FRM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | DPM15 | 0 | R | Descriptor Interrupt 15 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE15) and descriptor interrupt status flag (DIS.DPF15) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 30 | DPM14 | 0 | R | Descriptor Interrupt 14 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE14) and descriptor interrupt status flag (DIS.DPF14) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 29 | DPM13 | 0 | R | Descriptor Interrupt 13 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE13) and descriptor interrupt status flag (DIS.DPF13) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 28 | DPM12 | 0 | R | Descriptor Interrupt 12 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE12) and descriptor interrupt status flag (DIS.DPF12) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 27 | DPM11 | 0 | R | Descriptor Interrupt 11 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE11) and descriptor interrupt status flag (DIS.DPF11) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 26 | DPM10 | 0 | R | Descriptor Interrupt 10 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE10) and descriptor interrupt status flag (DIS.DPF10) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 25 | DPM9 | 0 | R | Descriptor Interrupt 9 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE9) and descriptor interrupt status flag (DIS.DPF9) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 24 | DPM8 | 0 | R | Descriptor Interrupt 8 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE8) and descriptor interrupt status flag (DIS.DPF8) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 23 | DPM7 | 0 | R | Descriptor Interrupt 7 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE7) and descriptor interrupt status flag (DIS.DPF7) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 22 | DPM6 | 0 | R | Descriptor Interrupt 6 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE6) and descriptor interrupt status flag (DIS.DPF6) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 21 | DPM5 | 0 | R | Descriptor Interrupt 5 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE5) and descriptor interrupt status flag (DIS.DPF5) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 20 | DPM4 | 0 | R | Descriptor Interrupt 4 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE4) and descriptor interrupt status flag (DIS.DPF4) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 19 | DPM3 | 0 | R | Descriptor Interrupt 3 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE3) and descriptor interrupt status flag (DIS.DPF3) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 18 | DPM2 | 0 | R | Descriptor Interrupt 2 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE2) and descriptor interrupt status flag (DIS.DPF2) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 17 | DPM1 | 0 | R | Descriptor Interrupt 1 Mirror<br>This bit is set to 1 when the given descriptor interrupt enable bit (DIC.DPE1) and descriptor interrupt status flag (DIS.DPF1) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 16 to 14 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 13 | — | 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 12 | RFWM | 0 | R | Receive FIFO Warning Interrupt Mirror<br>This bit is set to 1 when the receive FIFO warning interrupt enable bit (RIC1.RFWE) and receive FIFO warning interrupt status flag (RIS1.RFWF) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 11 | MFWM | 0 | R | E-MAC status FIFO Warning Mirror<br>This bit indicates that **TIS.MFWF** is 1b and **TIC.MFWE** is 1b.<br>[Changing condition]<br>This bit is updated when **TIS.MFWF** or **TIC.MFWE** changes.<br>0: No interrupt pending<br>1: E-MAC status FIFO warning interrupt pending |
| 10 | MFUM | 0 | R | E-MAC status FIFO Updated Mirror<br>This bit indicates that **TIS.MFUF** is 1b and **TIC.MFUE** is 1b.<br>[Changing condition]<br>This bit is updated when **TIS.MFUF** or **TIC.MFUE** changes.<br>0: No interrupt pending<br>1: E-MAC status FIFO updated interrupt pending |
| 9 | TFWM | 0 | R | Time Stamp FIFO Warning Interrupt Mirror<br>This bit is set to 1 when the time stamp FIFO warning interrupt enable bit (TIC.TFWE) and time stamp FIFO warning interrupt status flag (TIS.TFWF) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 8 | TFUM | 0 | R | Time Stamp FIFO Update Mirror<br>This bit is set to 1 when the time stamp FIFO update interrupt enable bit (TIC.TFUE) and time stamp FIFO update interrupt status flag (TIS.TFUF) are both 1.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | MM | 0 | R | E-MAC Interrupt Mirror<br>This bit is set to 1 when an E-MAC interrupt is issued.<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 6 | EM | 0 | R | Error Interrupt Mirror<br>This bit is set to 1 when an error interrupt is issued (both of a bit in EIS and corresponding to the bit in EIC are 1).<br>0: The interrupt is not pending.<br>1: The interrupt is pending. |
| 5 to 3 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 2 | FTM | 0 | R | Frame Transmitted Mirror<br>[Changing condition]<br>This bit is set when any matching pair of **TIC.FTEt** enable and **TIS.FTFt** flag are both 1b.<br>This bit is set when any matching pair of **TIC.TDPEt** enable and **TIS.TDPFt** flag are both 1b.<br>0: No interrupt pending<br>1: Frame transmitted interrupt pending |
| 1 | RWM | 0 | R | Reception Warning Mirror<br>[Changing condition]<br>This bit is set when any matching pair of **RIC1.RWEr** enable and **RIS1.RWFr** flag are both 1b.<br>0: No interrupt pending<br>1: Frame transmitted interrupt pending |
| 0 | FRM | 0 | R | Frame Received Mirror<br>[Changing condition]<br>This bit is set when any matching pair of **RIC0.FREr** enable and **RIS0.FRFr** flag are both 1b.<br>This bit is set when any matching pair of **RIC3.RDPEr** enable and **RIS3.RDPFr** flag are both 1b.<br>0: No interrupt pending<br>1: Frame received interrupt pending |

### 100.1.2.49 Common Interrupt Enable register (CIE)

The CIE register is used to control the Common Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | RFFL | RFWL | CL0M | RQFM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CTIE | — | — | — | — | — | — | — | CRIE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 19 | RFFL | 0 | R/W | Rx-FIFO Full interrupt Line select<br>This bit selects the interrupt line of notification flagged by **RIS2.RFFF**.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 18 | RFWL | 0 | R/W | Rx-FIFO Warning interrupt Line select<br>This bit selects the interrupt line of notification flagged by **RIS1.RFWF**.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 17 | CL0M | 0 | R/W | Common Line 0 Mode<br>This bit controls if all data related interrupts using a common interrupt line or if queue specific interrupt lines for data related interrupts are used. The selected mode has no influence to internal flagging in EthernetAVB-IF register (e.g. **ISS**).<br>To prevent interrupt notifications on different lines, it is recommended to set all **CIE.CTIE**, **CIE.CRIE**, and **DIL.DPLi** (i=2 to 15) to 0b when this bit is 1b.<br>0: Use common interrupt line 0<br>1: Use queue specific interrupt line 0 |
| 16 | RQFM | 0 | R/W | Reception Queue Full Mode<br>This bit controls if queue full notification is mapped to error interrupt line or to queue data interrupt line. The selected mode has no influence to internal flagging in EthernetAVB-IF register (e.g. **ISS**).<br>0: Use for error interrupt line<br>1: Use for queue specific interrupt line |
| 15 to 9 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8 | CTIE | 0 | R/W | Common Transmit Interrupt Enable<br>This bit controls transmit related interrupt line outputs of EthernetAVB-IF. It has no influence to internal flagging in EthernetAVB-IF register (e.g. **ISS**).<br>These lines are controlled by this bit:<br>**line0_Tx**[3:0] -- individual transmit queue interrupts<br>Transmission interrupts mapped on other lines are not influenced.<br>0: Disabled<br>1: Enabled |
| 7 to 1 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 0 | CRIE | 0 | R/W | Common Receive Interrupt enable<br>This bit controls receive related interrupt line outputs of EthernetAVB-IF. It has no influence to internal flagging in EthernetAVB-IF register (e.g. **ISS**).<br>These lines are controlled by this bit:<br>**line0_Rx**[17:0] -- individual receive queue interrupts<br>Receive interrupts mapped on other lines are not influenced.<br>0: Disabled<br>1: Enabled |

### 100.1.2.50 Reception Interrupt Control register 3 (RIC3)

The RIC3 register controls the AVB-DMAC receive descriptor interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RDPE17 | RDPE16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDPE15 | RDPE14 | RDPE13 | RDPE12 | RDPE11 | RDPE10 | RDPE9 | RDPE8 | RDPE7 | RDPE6 | RDPE5 | RDPE4 | RDPE3 | RDPE2 | RDPE1 | RDPE0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | RDPE17 | 0 | R/W | Receive Descriptor Processed interrupt Enable 17<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF17** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD17**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS17**.<br>0: Disabled<br>1: Enabled |
| 16 | RDPE16 | 0 | R/W | Receive Descriptor Processed interrupt Enable 16<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF16** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD16**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS16**.<br>0: Disabled<br>1: Enabled |
| 15 | RDPE15 | 0 | R/W | Receive Descriptor Processed interrupt Enable 15<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF15** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD15**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS15**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 14 | RDPE14 | 0 | R/W | Receive Descriptor Processed interrupt Enable 14<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF14** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD14**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS14**.<br>0: Disabled<br>1: Enabled |
| 13 | RDPE13 | 0 | R/W | Receive Descriptor Processed interrupt Enable 13<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF13** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD13**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS13**.<br>0: Disabled<br>1: Enabled |
| 12 | RDPE12 | 0 | R/W | Receive Descriptor Processed interrupt Enable 12<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF12** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD12**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS12**.<br>0: Disabled<br>1: Enabled |
| 11 | RDPE11 | 0 | R/W | Receive Descriptor Processed interrupt Enable 11<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF11** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD11**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS11**.<br>0: Disabled<br>1: Enabled |
| 10 | RDPE10 | 0 | R/W | Receive Descriptor Processed interrupt Enable 10<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF10** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD10**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS10**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | RDPE9 | 0 | R/W | Receive Descriptor Processed interrupt Enable 9<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF9** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD9**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS9**.<br>0: Disabled<br>1: Enabled |
| 8 | RDPE8 | 0 | R/W | Receive Descriptor Processed interrupt Enable 8<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF8** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD8**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS8**.<br>0: Disabled<br>1: Enabled |
| 7 | RDPE7 | 0 | R/W | Receive Descriptor Processed interrupt Enable 7<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF7** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD7**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS7**.<br>0: Disabled<br>1: Enabled |
| 6 | RDPE6 | 0 | R/W | Receive Descriptor Processed interrupt Enable 6<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF6** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD6**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS6**.<br>0b Disabled<br>1b Enabled |
| 5 | RDPE5 | 0 | R/W | Receive Descriptor Processed interrupt Enable 5<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF5** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD5**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS5**.<br>0: Disabled<br>1: Enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | RDPE4 | 0 | R/W | Receive Descriptor Processed interrupt Enable 4<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF4** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD4**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS4**.<br>0: Disabled<br>1: Enabled |
| 3 | RDPE3 | 0 | R/W | Receive Descriptor Processed interrupt Enable 3<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF3** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD3**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS3**.<br>0: Disabled<br>1: Enabled |
| 2 | RDPE2 | 0 | R/W | Receive Descriptor Processed interrupt Enable 2<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF2** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD2**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS2**.<br>0: Disabled<br>1: Enabled |
| 1 | RDPE1 | 0 | R/W | Receive Descriptor Processed interrupt Enable 1<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF1** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD1**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS1**.<br>0: Disabled<br>1: Enabled |
| 0 | RDPE0 | 0 | R/W | Receive Descriptor Processed interrupt Enable 0<br>While this bit is 1b an interrupt will be generated when **RIS3.RDPF0** is 1b.<br>[Changing condition]<br>This bit is set to 0b when writing 1b to **RID3.RDPD0**.<br>This bit is set to 1b when writing 1b to **RIE3.RDPS0**.<br>0: Disabled<br>1: Enabled |

### 100.1.2.51 Reception Interrupt Status register 3 (RIS3)

The RIS3 register indicates the states of the AVB-DMAC receive descriptor interrupts.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RDPF17 | RDPF16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDPF15 | RDPF14 | RDPF13 | RDPF12 | RDPF11 | RDPF10 | RDPF9 | RDPF8 | RDPF7 | RDPF6 | RDPF5 | RDPF4 | RDPF3 | RDPF2 | RDPF1 | RDPF0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | RDPF17 | 0 | R/W | Receive Descriptor Processed Flag 17<br>This bit indicates that a descriptor in reception queue 17 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 17 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 16 | RDPF16 | 0 | R/W | Receive Descriptor Processed Flag 16<br>This bit indicates that a descriptor in reception queue 16 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 16 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | RDPF15 | 0 | R/W | Receive Descriptor Processed Flag 15<br>This bit indicates that a descriptor in reception queue 15 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 15 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 14 | RDPF14 | 0 | R/W | Receive Descriptor Processed Flag 14<br>This bit indicates that a descriptor in reception queue 14 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 14 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 13 | RDPF13 | 0 | R/W | Receive Descriptor Processed Flag 13<br>This bit indicates that a descriptor in reception queue 13 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 13 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 12 | RDPF12 | 0 | R/W | Receive Descriptor Processed Flag 12<br>This bit indicates that a descriptor in reception queue 12 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 12 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 | RDPF11 | 0 | R/W | Receive Descriptor Processed Flag 11<br>This bit indicates that a descriptor in reception queue 11 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 11 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 10 | RDPF10 | 0 | R/W | Receive Descriptor Processed Flag 10<br>This bit indicates that a descriptor in reception queue 10 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 10 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 9 | RDPF9 | 0 | R/W | Receive Descriptor Processed Flag 9<br>This bit indicates that a descriptor in reception queue 9 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 9 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 8 | RDPF8 | 0 | R/W | Receive Descriptor Processed Flag 8<br>This bit indicates that a descriptor in reception queue 8 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 8 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | RDPF7 | 0 | R/W | Receive Descriptor Processed Flag 7<br>This bit indicates that a descriptor in reception queue 7 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 7 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 6 | RDPF6 | 0 | R/W | Receive Descriptor Processed Flag 6<br>This bit indicates that a descriptor in reception queue 6 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 6 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 5 | RDPF5 | 0 | R/W | Receive Descriptor Processed Flag 5<br>This bit indicates that a descriptor in reception queue 5 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 5 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 4 | RDPF4 | 0 | R/W | Receive Descriptor Processed Flag 4<br>This bit indicates that a descriptor in reception queue 4 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 4 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | RDPF3 | 0 | R/W | Receive Descriptor Processed Flag 3<br>This bit indicates that a descriptor in reception queue 3 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 3 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 2 | RDPF2 | 0 | R/W | Receive Descriptor Processed Flag 2<br>This bit indicates that a descriptor in reception queue 2 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 2 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 1 | RDPF1 | 0 | R/W | Receive Descriptor Processed Flag 1<br>This bit indicates that a descriptor in reception queue 1 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 1 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |
| 0 | RDPF0 | 0 | R/W | Receive Descriptor Processed Flag 0<br>This bit indicates that a descriptor in reception queue 0 has been processed where **DESCR.DIE** is 0001b.<br>Note: The descriptor with **DESCR.DIE** is 1, which sets this bit, sets in addition the universal descriptor interrupt **DIS.DPF1**.<br>The CPU can only write 0b to this bit.<br>[Changing condition]<br>This bit is set to 0b when leaving OPERATION mode.<br>This bit is set to 1b when a descriptor in reception queue 0 has been processed where **DESCR.DIE** is 1.<br>0: No interrupt pending<br>1: Descriptor interrupt pending |

### 100.1.2.52 Descriptor Interrupt Line selection register (DIL)

The DIL register is used to control the Descriptor Interrupt line.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPL15 | DPL14 | DPL13 | DPL12 | DPL11 | DPL10 | DPL9 | DPL8 | DPL7 | DPL6 | DPL5 | DPL4 | DPL3 | DPL2 | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 | DPL15 | 0 | R/W | Descriptor Processed interrupt Line select 15<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 14 | DPL14 | 0 | R/W | Descriptor Processed interrupt Line select 14<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 13 | DPL13 | 0 | R/W | Descriptor Processed interrupt Line select 13<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 12 | DPL12 | 0 | R/W | Descriptor Processed interrupt Line select 12<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 11 | DPL11 | 0 | R/W | Descriptor Processed interrupt Line select 11<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 10 | DPL10 | 0 | R/W | Descriptor Processed interrupt Line select 10<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | DPL9 | 0 | R/W | Descriptor Processed interrupt Line select 9<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 8 | DPL8 | 0 | R/W | Descriptor Processed interrupt Line select 8<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 7 | DPL7 | 0 | R/W | Descriptor Processed interrupt Line select 7<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 6 | DPL6 | 0 | R/W | Descriptor Processed interrupt Line select 6<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 5 | DPL5 | 0 | R/W | Descriptor Processed interrupt Line select 5<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 4 | DPL4 | 0 | R/W | Descriptor Processed interrupt Line select 4<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 3 | DPL3 | 0 | R/W | Descriptor Processed interrupt Line select 3<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 2 | DPL2 | 0 | R/W | Descriptor Processed interrupt Line select 2<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 1 to 0 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.53 Error Interrupt Line selection register (EIL)

The EIL register is used to control the Error Interrupt line.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TBFL | MFFL | TFFL | CULL1 | CULL0 | CLLL1 | CLLL0 | SEL | QEL | MTEL | MREL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 | TBFL | 0 | R/W | Tx-Buffer Full interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 9 | MFFL | 0 | R/W | E-MAC status FIFO Full interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 8 | TFFL | 0 | R/W | Timestamp FIFO Full interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 7 | CULL1 | 0 | R/W | CBS Upper Limit reached interrupt Line select 1<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 6 | CULL0 | 0 | R/W | CBS Upper Limit reached interrupt Line select 0<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 5 | CLLL1 | 0 | R/W | CBS Lower Limit reached interrupt Line select 1<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | CLLL0 | 0 | R/W | CBS Lower Limit reached interrupt Line select 0<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 3 | SEL | 0 | R/W | Separation Error interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 2 | QEL | 0 | R/W | Queue Error interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 1 | MTEL | 0 | R/W | E-MAC Transmission Error interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 0 | MREL | 0 | R/W | E-MAC Reception Error interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |

### 100.1.2.54 Transmission Interrupt Line selection register (TIL)

The TIL register is used to control the Transmission Interrupt line.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MFWL | MFUL | TFWL | TFUL | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 11 | MFWL | 0 | R/W | E-MAC status FIFO Warning interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 10 | MFUL | 0 | R/W | E-MAC status FIFO Updated interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 9 | TFWL | 0 | R/W | Timestamp FIFO Warning interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 8 | TFUL | 0 | R/W | Timestamp FIFO Updated interrupt Line select<br>This bit selects the interrupt line used for this notification.<br>It is recommended to disable interrupt before change the line.<br>0: Interrupt line A used for this notification<br>1: Interrupt line B used for this notification |
| 7 to 0 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.55 Descriptor Interrupt Enable register (DIE)

The DIE register is used to control the Descriptor Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPS15 | DPS14 | DPS13 | DPS12 | DPS11 | DPS10 | DPS9 | DPS8 | DPS7 | DPS6 | DPS5 | DPS4 | DPS3 | DPS2 | DPS1 | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 | DPS15 | 0 | R/W | Descriptor Processed interrupt Set 15<br>This bit supports interrupt enable. It controls set of **DIC.DPE15**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE15<br>1: Set DIC.DPE15 to 1b |
| 14 | DPS14 | 0 | R/W | Descriptor Processed interrupt Set 14<br>This bit supports interrupt enable. It controls set of **DIC.DPE14**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE14<br>1: Set DIC.DPE14 to 1b |
| 13 | DPS13 | 0 | R/W | Descriptor Processed interrupt Set 13<br>This bit supports interrupt enable. It controls set of **DIC.DPE13**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE13<br>1: Set DIC.DPE13 to 1b |
| 12 | DPS12 | 0 | R/W | Descriptor Processed interrupt Set 12<br>This bit supports interrupt enable. It controls set of **DIC.DPE12**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE12<br>1: Set DIC.DPE12 to 1b |
| 11 | DPS11 | 0 | R/W | Descriptor Processed interrupt Set 11<br>This bit supports interrupt enable. It controls set of **DIC.DPE11**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE11<br>1: Set DIC.DPE11 to 1b |
| 10 | DPS10 | 0 | R/W | Descriptor Processed interrupt Set 10<br>This bit supports interrupt enable. It controls set of **DIC.DPE10**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE10<br>1: Set DIC.DPE10 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | DPS9 | 0 | R/W | Descriptor Processed interrupt Set 9<br>This bit supports interrupt enable. It controls set of **DIC.DPE9**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE9<br>1: Set DIC.DPE9 to 1b |
| 8 | DPS8 | 0 | R/W | Descriptor Processed interrupt Set 8<br>This bit supports interrupt enable. It controls set of **DIC.DPE8**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE8<br>1: Set DIC.DPE8 to 1b |
| 7 | DPS7 | 0 | R/W | Descriptor Processed interrupt Set 7<br>This bit supports interrupt enable. It controls set of **DIC.DPE7**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE7<br>1: Set DIC.DPE7 to 1b |
| 6 | DPS6 | 0 | R/W | Descriptor Processed interrupt Set 6<br>This bit supports interrupt enable. It controls set of **DIC.DPE6**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE6<br>1: Set DIC.DPE6 to 1b |
| 5 | DPS5 | 0 | R/W | Descriptor Processed interrupt Set 5<br>This bit supports interrupt enable. It controls set of **DIC.DPE5**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE5<br>1: Set DIC.DPE5 to 1b |
| 4 | DPS4 | 0 | R/W | Descriptor Processed interrupt Set 4<br>This bit supports interrupt enable. It controls set of **DIC.DPE4**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE4<br>1: Set DIC.DPE4 to 1b |
| 3 | DPS3 | 0 | R/W | Descriptor Processed interrupt Set 3<br>This bit supports interrupt enable. It controls set of **DIC.DPE3**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE3<br>1: Set DIC.DPE3 to 1b |
| 2 | DPS2 | 0 | R/W | Descriptor Processed interrupt Set 2<br>This bit supports interrupt enable. It controls set of **DIC.DPE2**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE2<br>1:Set DIC.DPEi to 1b |
| 1 | DPS1 | 0 | R/W | Descriptor Processed interrupt Set 1<br>This bit supports interrupt enable. It controls set of **DIC.DPE1**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE1<br>1: Set DIC.DPE1 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.56 Descriptor Interrupt Disable register (DID)

The DID register is used to control the Descriptor Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPD15 | DPD14 | DPD13 | DPD12 | DPD11 | DPD10 | DPD9 | DPD8 | DPD7 | DPD6 | DPD5 | DPD4 | DPD3 | DPD2 | DPD1 | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 | DPD15 | 0 | R/W | Descriptor Processed interrupt Disable 15<br>This bit supports interrupt enable. It controls set of **DIC.DPE15**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE15<br>1: Set DIC.DPE15 to 0b |
| 14 | DPD14 | 0 | R/W | Descriptor Processed interrupt Disable 14<br>This bit supports interrupt enable. It controls set of **DIC.DPE14**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE14<br>1: Set DIC.DPE14 to 0b |
| 13 | DPD13 | 0 | R/W | Descriptor Processed interrupt Disable 13<br>This bit supports interrupt enable. It controls set of **DIC.DPE13**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE13<br>1: Set DIC.DPE13 to 0b |
| 12 | DPD12 | 0 | R/W | Descriptor Processed interrupt Disable 12<br>This bit supports interrupt enable. It controls set of **DIC.DPE12**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE12<br>1: Set DIC.DPE12 to 0b |
| 11 | DPD11 | 0 | R/W | Descriptor Processed interrupt Disable 11<br>This bit supports interrupt enable. It controls set of **DIC.DPE11**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE11<br>1: Set DIC.DPE11 to 0b |
| 10 | DPD10 | 0 | R/W | Descriptor Processed interrupt Disable 10<br>This bit supports interrupt enable. It controls set of **DIC.DPE10**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE10<br>1: Set DIC.DPE10 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | DPD9 | 0 | R/W | Descriptor Processed interrupt Disable 9<br>This bit supports interrupt enable. It controls set of **DIC.DPE9**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE9<br>1: Set DIC.DPE9 to 0b |
| 8 | DPD8 | 0 | R/W | Descriptor Processed interrupt Disable 8<br>This bit supports interrupt enable. It controls set of **DIC.DPE8**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE8<br>1: Set DIC.DPE8 to 0b |
| 7 | DPD7 | 0 | R/W | Descriptor Processed interrupt Disable 7<br>This bit supports interrupt enable. It controls set of **DIC.DPE7**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE7<br>1: Set DIC.DPE7 to 0b |
| 6 | DPD6 | 0 | R/W | Descriptor Processed interrupt Disable 6<br>This bit supports interrupt enable. It controls set of **DIC.DPE6**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE6<br>1: Set DIC.DPE6 to 0b |
| 5 | DPD5 | 0 | R/W | Descriptor Processed interrupt Disable 5<br>This bit supports interrupt enable. It controls set of **DIC.DPE5**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE5<br>1: Set DIC.DPE5 to 0b |
| 4 | DPD4 | 0 | R/W | Descriptor Processed interrupt Disable 4<br>This bit supports interrupt enable. It controls set of **DIC.DPE4**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE4<br>1: Set DIC.DPE4 to 0b |
| 3 | DPD3 | 0 | R/W | Descriptor Processed interrupt Disable 3<br>This bit supports interrupt enable. It controls set of **DIC.DPE3**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE3<br>1: Set DIC.DPE3 to 0b |
| 2 | DPD2 | 0 | R/W | Descriptor Processed interrupt Disable 2<br>This bit supports interrupt enable. It controls set of **DIC.DPE2**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE2<br>1: Set DIC.DPE2 to 0b |
| 1 | DPD1 | 0 | R/W | Descriptor Processed interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **DIC.DPE1**.<br>This bit is always read as 0b.<br>0: No change of DIC.DPE1<br>1: Set DIC.DPE1 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | — | 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.57 Error Interrupt Enable register (EIE)

The EIE register is used to control the Error Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TBFS | MFFS | TFFS | CULS1 | CULS0 | CLLS1 | CLLS0 | SES | QES | MTES | MRES |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 | TBFS | 0 | R/W | Tx-Buffer Full interrupt Set<br>This bit supports interrupt enable. It controls set of **EIC.TBFE**.<br>This bit is always read as 0b.<br>0: No change of EIC.TBFE<br>1: Set EIC.TBFE to 1b |
| 9 | MFFS | 0 | R/W | E-MAC status FIFO Full interrupt Set<br>This bit supports interrupt enable. It controls set of **EIC.MFFE**.<br>This bit is always read as 0b.<br>0: No change of EIC.MFFE<br>1: Set EIC.MFFE to 1b |
| 8 | TFFS | 0 | R/W | Timestamp FIFO Full interrupt Set<br>This bit supports interrupt enable. It controls set of **EIC.TFFE**.<br>This bit is always read as 0b.<br>0: No change of EIC.TFFE<br>1: Set EIC.TFFE to 1b |
| 7 | CULS1 | 0 | R/W | CBS Upper Limit reached interrupt Set 1<br>This bit supports interrupt enable. It controls set of **EIC.CULE1**.<br>This bit is always read as 0b.<br>0: No change of EIC.CULE1<br>1: Set EIC.CULE1 to 1b |
| 6 | CULS0 | 0 | R/W | CBS Upper Limit reached interrupt Set 0<br>This bit supports interrupt enable. It controls set of **EIC.CULE0**.<br>This bit is always read as 0b.<br>0: No change of EIC.CULE0<br>1: Set EIC.CULE0 to 1b |
| 5 | CLLS1 | 0 | R/W | CBS Lower Limit reached interrupt Set 1<br>This bit supports interrupt enable. It controls set of **EIC.CLLE1**.<br>This bit is always read as 0b.<br>0: No change of EIC.CLLE1<br>1: Set EIC.CLLE1 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | CLLS0 | 0 | R/W | CBS Lower Limit reached interrupt Set 0<br>This bit supports interrupt enable. It controls set of **EIC.CLLE0**.<br>This bit is always read as 0b.<br>0: No change of EIC.CLLE0<br>1: Set EIC.CLLE0 to 1b |
| 3 | SES | 0 | R/W | Separation Error interrupt Set<br>This bit supports interrupt enable. It controls set of **EIC.SEE**.<br>This bit is always read as 0b.<br>0: No change of EIC.SEE<br>1: Set EIC.QEE to 1b |
| 2 | QES | 0 | R/W | Queue Error interrupt Set<br>This bit supports interrupt enable. It controls set of **EIC.QEE**.<br>This bit is always read as 0b.<br>0: No change of EIC.QEE<br>1: Set EIC.QEE to 1b |
| 1 | MTES | 0 | R/W | E-MAC Transmission Error interrupt Set<br>This bit supports interrupt enable. It controls set of **EIC.MTEE**.<br>This bit is always read as 0b.<br>0: No change of EIC.MTEE<br>1: Set EIC.MTEE to 1b |
| 0 | MRES | 0 | R/W | E-MAC Reception Error interrupt Set<br>This bit supports interrupt enable. It controls set of **EIC.MREE**.<br>This bit is always read as 0b.<br>0: No change of EIC.MREE<br>1: Set EIC.MREE to 1b |

### 100.1.2.58 Error Interrupt Disable register (EID)

The EID register is used to control the Error Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TBFD | MFFD | TFFD | CULD1 | CULD0 | CLLD1 | CLLD0 | SED | QED | MTED | MRED |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 | TBFD | 0 | R/W | Tx-Buffer Full interrupt Disable<br>This bit supports interrupt enable. It controls set of **EIC.TBFE**.<br>This bit is always read as 0b.<br>0: No change of EIC.TBFE<br>1: Set EIC.TBFE to 0b |
| 9 | MFFD | 0 | R/W | E-MAC status FIFO Full interrupt Disable<br>This bit supports interrupt enable. It controls set of **EIC.MFFE**.<br>This bit is always read as 0b.<br>0: No change of EIC.MFFE<br>1: Set EIC.MFFE to 0b |
| 8 | TFFD | 0 | R/W | Timestamp FIFO Full interrupt Disable<br>This bit supports interrupt enable. It controls set of **EIC.TFFE**.<br>This bit is always read as 0b.<br>0: No change of EIC.TFFE<br>1: Set EIC.TFFE to 0b |
| 7 | CULD1 | 0 | R/W | CBS Upper Limit reached interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **EIC.CULE1**.<br>This bit is always read as 0b.<br>0: No change of EIC.CULE1<br>1: Set EIC.CULE1 to 0b |
| 6 | CULD0 | 0 | R/W | CBS Upper Limit reached interrupt Disable 0<br>This bit supports interrupt enable. It controls set of **EIC.CULE0**.<br>This bit is always read as 0b.<br>0: No change of EIC.CULE0<br>1: Set EIC.CULE0 to 0b |
| 5 | CLLD1 | 0 | R/W | CBS Lower Limit reached interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **EIC.CLLE1**.<br>This bit is always read as 0b.<br>0: No change of EIC.CLLE1<br>1: Set EIC.CLLE1 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | CLLD0 | 0 | R/W | CBS Lower Limit reached interrupt Disable 0<br>This bit supports interrupt enable. It controls set of **EIC.CLLE0**.<br>This bit is always read as 0b.<br>0: No change of EIC.CLLE0<br>1: Set EIC.CLLE0 to 0b |
| 3 | SED | 0 | R/W | Separation Error interrupt Disable<br>This bit supports interrupt enable. It controls set of **EIC.SEE**.<br>This bit is always read as 0b.<br>0: No change of EIC.SEE<br>1: Set EIC.SEE to 0b |
| 2 | QED | 0 | R/W | Queue Error interrupt Disable<br>This bit supports interrupt enable. It controls set of **EIC.QEE**.<br>This bit is always read as 0b.<br>0: No change of EIC.QEE<br>1: Set EIC.QEE to 0b |
| 1 | MTED | 0 | R/W | E-MAC Transmission Error interrupt Disable<br>This bit supports interrupt enable. It controls set of **EIC.MTEE**.<br>This bit is always read as 0b.<br>0: No change of EIC.MTEE<br>1: Set EIC.MTEE to 0b |
| 0 | MRED | 0 | R/W | E-MAC Reception Error interrupt Disable<br>This bit supports interrupt enable. It controls set of **EIC.MREE**.<br>This bit is always read as 0b.<br>0: No change of EIC.MREE<br>1: Set EIC.MREE to 0b |

### 100.1.2.59 Reception Interrupt Enable register 0 (RIE0)

The RIE0 register is used to control the Reception Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FRS17 | FRS16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FRS15 | FRS14 | FRS13 | FRS12 | FRS11 | FRS10 | FRS9 | FRS8 | FRS7 | FRS6 | FRS5 | FRS4 | FRS3 | FRS2 | FRS1 | FRS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | FRS17 | 0 | R/W | Frame Received interrupt Set 17<br>This bit supports interrupt enable. It controls set of **RIC0.FRE17**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE17<br>1: Set RIC0.FRE17 to 1b |
| 16 | FRS16 | 0 | R/W | Frame Received interrupt Set 16<br>This bit supports interrupt enable. It controls set of **RIC0.FRE16**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE16<br>1: Set RIC0.FRE16 to 1b |
| 15 | FRS15 | 0 | R/W | Frame Received interrupt Set 15<br>This bit supports interrupt enable. It controls set of **RIC0.FRE15**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE15<br>1: Set RIC0.FRE15 to 1b |
| 14 | FRS14 | 0 | R/W | Frame Received interrupt Set 14<br>This bit supports interrupt enable. It controls set of **RIC0.FRE14**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE14<br>1: Set RIC0.FRE14 to 1b |
| 13 | FRS13 | 0 | R/W | Frame Received interrupt Set 13<br>This bit supports interrupt enable. It controls set of **RIC0.FRE13**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE13<br>1: Set RIC0.FRE13 to 1b |
| 12 | FRS12 | 0 | R/W | Frame Received interrupt Set 12<br>This bit supports interrupt enable. It controls set of **RIC0.FRE12**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE12<br>1: Set RIC0.FRE12 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 | FRS11 | 0 | R/W | Frame Received interrupt Set 11<br>This bit supports interrupt enable. It controls set of **RIC0.FRE11**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE11<br>1: Set RIC0.FRE11 to 1b |
| 10 | FRS10 | 0 | R/W | Frame Received interrupt Set 10<br>This bit supports interrupt enable. It controls set of **RIC0.FRE10**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE10<br>1: Set RIC0.FRE10 to 1b |
| 9 | FRS9 | 0 | R/W | Frame Received interrupt Set 9<br>This bit supports interrupt enable. It controls set of **RIC0.FRE9**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE9<br>1: Set RIC0.FRE9 to 1b |
| 8 | FRS8 | 0 | R/W | Frame Received interrupt Set 8<br>This bit supports interrupt enable. It controls set of **RIC0.FRE8**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE8<br>1: Set RIC0.FRE8 to 1b |
| 7 | FRS7 | 0 | R/W | Frame Received interrupt Set 7<br>This bit supports interrupt enable. It controls set of **RIC0.FRE7**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE7<br>1: Set RIC0.FRE7 to 1b |
| 6 | FRS6 | 0 | R/W | Frame Received interrupt Set 6<br>This bit supports interrupt enable. It controls set of **RIC0.FRE6**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE6<br>1: Set RIC0.FRE6 to 1b |
| 5 | FRS5 | 0 | R/W | Frame Received interrupt Set 5<br>This bit supports interrupt enable. It controls set of **RIC0.FRE5**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE5<br>1: Set RIC0.FRE5 to 1b |
| 4 | FRS4 | 0 | R/W | Frame Received interrupt Set 4<br>This bit supports interrupt enable. It controls set of **RIC0.FRE4**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE4<br>1: Set RIC0.FRE4 to 1b |
| 3 | FRS3 | 0 | R/W | Frame Received interrupt Set 3<br>This bit supports interrupt enable. It controls set of **RIC0.FRE3**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE3<br>1: Set RIC0.FRE3 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | FRS2 | 0 | R/W | Frame Received interrupt Set 2<br>This bit supports interrupt enable. It controls set of **RIC0.FRE2**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE2<br>1: Set RIC0.FRE2 to 1b |
| 1 | FRS1 | 0 | R/W | Frame Received interrupt Set 1<br>This bit supports interrupt enable. It controls set of **RIC0.FRE1**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE1<br>1: Set RIC0.FRE1 to 1b |
| 0 | FRS0 | 0 | R/W | Frame Received interrupt Set 0<br>This bit supports interrupt enable. It controls set of **RIC0.FRE0**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE0<br>1: Set RIC0.FRE0 to 1b |

### 100.1.2.60 Reception Interrupt Disable register 0 (RID0)

The RID0 register is used to control the Reception Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FRD17 | FRD16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FRD15 | FRD14 | FRD13 | FRD12 | FRD11 | FRD10 | FRD9 | FRD8 | FRD7 | FRD6 | FRD5 | FRD4 | FRD3 | FRD2 | FRD1 | FRD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | FRD17 | 0 | R/W | Frame Received interrupt Disable 17<br>This bit supports interrupt enable. It controls set of **RIC0.FRE17**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE17<br>1: Set RIC0.FRE17 to 0b |
| 16 | FRD16 | 0 | R/W | Frame Received interrupt Disable 16<br>This bit supports interrupt enable. It controls set of **RIC0.FRE16**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE16<br>1: Set RIC0.FRE16 to 0b |
| 15 | FRD15 | 0 | R/W | Frame Received interrupt Disable 15<br>This bit supports interrupt enable. It controls set of **RIC0.FRE15**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE15<br>1: Set RIC0.FRE15 to 0b |
| 14 | FRD14 | 0 | R/W | Frame Received interrupt Disable 14<br>This bit supports interrupt enable. It controls set of **RIC0.FRE14**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE14<br>1: Set RIC0.FRE14 to 0b |
| 13 | FRD13 | 0 | R/W | Frame Received interrupt Disable 13<br>This bit supports interrupt enable. It controls set of **RIC0.FRE13**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE13<br>1: Set RIC0.FRE13 to 0b |
| 12 | FRD12 | 0 | R/W | Frame Received interrupt Disable 12<br>This bit supports interrupt enable. It controls set of **RIC0.FRE12**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE12<br>1: Set RIC0.FRE12 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 | FRD11 | 0 | R/W | Frame Received interrupt Disable 11<br>This bit supports interrupt enable. It controls set of **RIC0.FRE11**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE11<br>1: Set RIC0.FRE11 to 0b |
| 10 | FRD10 | 0 | R/W | Frame Received interrupt Disable 10<br>This bit supports interrupt enable. It controls set of **RIC0.FRE10**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE10<br>1: Set RIC0.FRE10 to 0b |
| 9 | FRD9 | 0 | R/W | Frame Received interrupt Disable 9<br>This bit supports interrupt enable. It controls set of **RIC0.FRE9**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE9<br>1: Set RIC0.FRE9 to 0b |
| 8 | FRD8 | 0 | R/W | Frame Received interrupt Disable 8<br>This bit supports interrupt enable. It controls set of **RIC0.FRE8**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE8<br>1: Set RIC0.FRE8 to 0b |
| 7 | FRD7 | 0 | R/W | Frame Received interrupt Disable 7<br>This bit supports interrupt enable. It controls set of **RIC0.FRE7**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE7<br>1: Set RIC0.FRE7 to 0b |
| 6 | FRD6 | 0 | R/W | Frame Received interrupt Disable 6<br>This bit supports interrupt enable. It controls set of **RIC0.FRE6**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE6<br>1: Set RIC0.FRE6 to 0b |
| 5 | FRD5 | 0 | R/W | Frame Received interrupt Disable 5<br>This bit supports interrupt enable. It controls set of **RIC0.FRE5**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE5<br>1: Set RIC0.FRE5 to 0b |
| 4 | FRD4 | 0 | R/W | Frame Received interrupt Disable 4<br>This bit supports interrupt enable. It controls set of **RIC0.FRE4**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE4<br>1: Set RIC0.FRE4 to 0b |
| 3 | FRD3 | 0 | R/W | Frame Received interrupt Disable 3<br>This bit supports interrupt enable. It controls set of **RIC0.FRE3**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE3<br>1: Set RIC0.FRE3 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | FRD2 | 0 | R/W | Frame Received interrupt Disable 2<br>This bit supports interrupt enable. It controls set of **RIC0.FRE2**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE2<br>1: Set RIC0.FRE2 to 0b |
| 1 | FRD1 | 0 | R/W | Frame Received interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **RIC0.FRE1**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE1<br>1: Set RIC0.FRE1 to 0b |
| 0 | FRD0 | 0 | R/W | Frame Received interrupt Disable 0<br>This bit supports interrupt enable. It controls set of **RIC0.FRE0**.<br>This bit is always read as 0b.<br>0: No change of RIC0.FRE0<br>1: Set RIC0.FRE0 to 0b |

### 100.1.2.61 Reception Interrupt Enable register 1 (RIE1)

The RIE1 register is used to control the Reception Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFWS | — | — | — | — | — | — | — | — | — | — | — | — | — | RWS17 | RWS16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RWS15 | RWS14 | RWS13 | RWS12 | RWS11 | RWS10 | RWS9 | RWS8 | RWS7 | RWS6 | RWS5 | RWS4 | RWS3 | RWS2 | RWS1 | RWS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RFWS | 0 | R/W | Rx-FIFO Warning interrupt Set<br>This bit supports interrupt enable. It controls set of **RIC1.RFWE**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RFWE<br>1: Set RIC1.RFWE to 1b |
| 30 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | RWS17 | 0 | R/W | Reception Warning interrupt Set 17<br>This bit supports interrupt enable. It controls set of **RIC1.RWE17**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE17<br>1: Set RIC1.RWE17 to 1b |
| 16 | RWS16 | 0 | R/W | Reception Warning interrupt Set 16<br>This bit supports interrupt enable. It controls set of **RIC1.RWE16**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE16<br>1: Set RIC1.RWE16 to 1b |
| 15 | RWS15 | 0 | R/W | Reception Warning interrupt Set 15<br>This bit supports interrupt enable. It controls set of **RIC1.RWE15**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE15<br>1: Set RIC1.RWE15 to 1b |
| 14 | RWS14 | 0 | R/W | Reception Warning interrupt Set 14<br>This bit supports interrupt enable. It controls set of **RIC1.RWE14**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE14<br>1: Set RIC1.RWE14 to 1b |
| 13 | RWS13 | 0 | R/W | Reception Warning interrupt Set 13<br>This bit supports interrupt enable. It controls set of **RIC1.RWE13**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE13<br>1: Set RIC1.RWE13 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 | RWS12 | 0 | R/W | Reception Warning interrupt Set 12<br>This bit supports interrupt enable. It controls set of **RIC1.RWE12**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE12<br>1: Set RIC1.RWE12 to 1b |
| 11 | RWS11 | 0 | R/W | Reception Warning interrupt Set 11<br>This bit supports interrupt enable. It controls set of **RIC1.RWE11**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE11<br>1: Set RIC1.RWE11 to 1b |
| 10 | RWS10 | 0 | R/W | Reception Warning interrupt Set 10<br>This bit supports interrupt enable. It controls set of **RIC1.RWE10**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE10<br>1: Set RIC1.RWE10 to 1b |
| 9 | RWS9 | 0 | R/W | Reception Warning interrupt Set 9<br>This bit supports interrupt enable. It controls set of **RIC1.RWE9**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE9<br>1: Set RIC1.RWE9 to 1b |
| 8 | RWS8 | 0 | R/W | Reception Warning interrupt Set 8<br>This bit supports interrupt enable. It controls set of **RIC1.RWE8**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE8<br>1: Set RIC1.RWE8 to 1b |
| 7 | RWS7 | 0 | R/W | Reception Warning interrupt Set 7<br>This bit supports interrupt enable. It controls set of **RIC1.RWE7**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE7<br>1: Set RIC1.RWE7 to 1b |
| 6 | RWS6 | 0 | R/W | Reception Warning interrupt Set 6<br>This bit supports interrupt enable. It controls set of **RIC1.RWE6**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE6<br>1: Set RIC1.RWE6 to 1b |
| 5 | RWS5 | 0 | R/W | Reception Warning interrupt Set 5<br>This bit supports interrupt enable. It controls set of **RIC1.RWE5**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE5<br>1: Set RIC1.RWE5 to 1b |
| 4 | RWS4 | 0 | R/W | Reception Warning interrupt Set 4<br>This bit supports interrupt enable. It controls set of **RIC1.RWE4**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE4<br>1: Set RIC1.RWE4 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | RWS3 | 0 | R/W | Reception Warning interrupt Set 3<br>This bit supports interrupt enable. It controls set of **RIC1.RWE3**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE3<br>1: Set RIC1.RWE3 to 1b |
| 2 | RWS2 | 0 | R/W | Reception Warning interrupt Set 2<br>This bit supports interrupt enable. It controls set of **RIC1.RWE2**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE2<br>1: Set RIC1.RWE2 to 1b |
| 1 | RWS1 | 0 | R/W | Reception Warning interrupt Set 1<br>This bit supports interrupt enable. It controls set of **RIC1.RWE1**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE1<br>1: Set RIC1.RWE1 to 1b |
| 0 | RWS0 | 0 | R/W | Reception Warning interrupt Set 0<br>This bit supports interrupt enable. It controls set of **RIC1.RWE0**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE0<br>1: Set RIC1.RWE0 to 1b |

### 100.1.2.62 Reception Interrupt Disable register 1 (RID1)

The RID1 register is used to control the Reception Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFWD | — | — | — | — | — | — | — | — | — | — | — | — | — | RWD17 | RWD16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RWD15 | RWD14 | RWD13 | RWD12 | RWD11 | RWD10 | RWD9 | RWD8 | RWD7 | RWD6 | RWD5 | RWD4 | RWD3 | RWD2 | RWD1 | RWD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RFWD | 0 | R/W | Rx-FIFO Warning interrupt Disable<br>This bit supports interrupt enable. It controls set of **RIC1.RFWE**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RFWE<br>1: Set RIC1.RFWE to 0b |
| 30 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | RWD17 | 0 | R/W | Reception Warning interrupt Disable 17<br>This bit supports interrupt enable. It controls set of **RIC1.RWE17**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE17<br>1: Set RIC1.RWE17 to 0b |
| 16 | RWD16 | 0 | R/W | Reception Warning interrupt Disable 16<br>This bit supports interrupt enable. It controls set of **RIC1.RWE16**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE16<br>1: Set RIC1.RWE16 to 0b |
| 15 | RWD15 | 0 | R/W | Reception Warning interrupt Disable 15<br>This bit supports interrupt enable. It controls set of **RIC1.RWE15**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE15<br>1: Set RIC1.RWE15 to 0b |
| 14 | RWD14 | 0 | R/W | Reception Warning interrupt Disable 14<br>This bit supports interrupt enable. It controls set of **RIC1.RWE14**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE14<br>1: Set RIC1.RWE14 to 0b |
| 13 | RWD13 | 0 | R/W | Reception Warning interrupt Disable 13<br>This bit supports interrupt enable. It controls set of **RIC1.RWE13**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE13<br>1: Set RIC1.RWE13 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 | RWD12 | 0 | R/W | Reception Warning interrupt Disable 12<br>This bit supports interrupt enable. It controls set of **RIC1.RWE12**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE12<br>1: Set RIC1.RWE12 to 0b |
| 11 | RWD11 | 0 | R/W | Reception Warning interrupt Disable 11<br>This bit supports interrupt enable. It controls set of **RIC1.RWE11**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE11<br>1: Set RIC1.RWE11 to 0b |
| 10 | RWD10 | 0 | R/W | Reception Warning interrupt Disable 10<br>This bit supports interrupt enable. It controls set of **RIC1.RWE10**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE10<br>1: Set RIC1.RWE10 to 0b |
| 9 | RWD9 | 0 | R/W | Reception Warning interrupt Disable 9<br>This bit supports interrupt enable. It controls set of **RIC1.RWE9**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE9<br>1: Set RIC1.RWE9 to 0b |
| 8 | RWD8 | 0 | R/W | Reception Warning interrupt Disable 8<br>This bit supports interrupt enable. It controls set of **RIC1.RWE8**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE8<br>1: Set RIC1.RWE8 to 0b |
| 7 | RWD7 | 0 | R/W | Reception Warning interrupt Disable 7<br>This bit supports interrupt enable. It controls set of **RIC1.RWE7**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE7<br>1: Set RIC1.RWE7 to 0b |
| 6 | RWD6 | 0 | R/W | Reception Warning interrupt Disable 6<br>This bit supports interrupt enable. It controls set of **RIC1.RWE6**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE6<br>1: Set RIC1.RWE6 to 0b |
| 5 | RWD5 | 0 | R/W | Reception Warning interrupt Disable 5<br>This bit supports interrupt enable. It controls set of **RIC1.RWE5**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE5<br>1: Set RIC1.RWE5 to 0b |
| 4 | RWD4 | 0 | R/W | Reception Warning interrupt Disable 4<br>This bit supports interrupt enable. It controls set of **RIC1.RWE4**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE4<br>1: Set RIC1.RWE4 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | RWD3 | 0 | R/W | Reception Warning interrupt Disable 3<br>This bit supports interrupt enable. It controls set of **RIC1.RWE3**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE3<br>1: Set RIC1.RWE3 to 0b |
| 2 | RWD2 | 0 | R/W | Reception Warning interrupt Disable 2<br>This bit supports interrupt enable. It controls set of **RIC1.RWE2**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE2<br>1: Set RIC1.RWE2 to 0b |
| 1 | RWD1 | 0 | R/W | Reception Warning interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **RIC1.RWE1**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE1<br>1: Set RIC1.RWE1 to 0b |
| 0 | RWD0 | 0 | R/W | Reception Warning interrupt Disable 0<br>This bit supports interrupt enable. It controls set of **RIC1.RWE0**.<br>This bit is always read as 0b.<br>0: No change of RIC1.RWE0<br>1: Set RIC1.RWE0 to 0b |

### 100.1.2.63 Reception Interrupt Enable register 2 (RIE2)

The RIE2 register is used to control the Reception Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFFS | — | — | — | — | — | — | — | — | — | — | — | — | — | QFS17 | QFS16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QFS15 | QFS14 | QFS13 | QFS12 | QFS11 | QFS10 | QFS9 | QFS8 | QFS7 | QFS6 | QFS5 | QFS4 | QFS3 | QFS2 | QFS1 | QFS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RFFS | 0 | R/W | Rx-FIFO Full interrupt Set<br>This bit supports interrupt enable. It controls set of **RIC2.RFFE**.<br>This bit is always read as 0b.<br>0: No change of RIC2.RFFE<br>1: Set RIC2.RFFE to 1b |
| 30 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | QFS17 | 0 | R/W | Queue Full interrupt Set 17<br>This bit supports interrupt enable. It controls set of **RIC2.QFE17**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE17<br>1: Set RIC2.QFE17 to 1b |
| 16 | QFS16 | 0 | R/W | Queue Full interrupt Set 16<br>This bit supports interrupt enable. It controls set of **RIC2.QFE16**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE16<br>1: Set RIC2.QFE16 to 1b |
| 15 | QFS15 | 0 | R/W | Queue Full interrupt Set 15<br>This bit supports interrupt enable. It controls set of **RIC2.QFE15**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE15<br>1: Set RIC2.QFE15 to 1b |
| 14 | QFS14 | 0 | R/W | Queue Full interrupt Set 14<br>This bit supports interrupt enable. It controls set of **RIC2.QFE14**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE14<br>1: Set RIC2.QFE14 to 1b |
| 13 | QFS13 | 0 | R/W | Queue Full interrupt Set 13<br>This bit supports interrupt enable. It controls set of **RIC2.QFE13**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE13<br>1: Set RIC2.QFE13 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 | QFS12 | 0 | R/W | Queue Full interrupt Set 12<br>This bit supports interrupt enable. It controls set of **RIC2.QFE12**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE12<br>1: Set RIC2.QFE12 to 1b |
| 11 | QFS11 | 0 | R/W | Queue Full interrupt Set 11<br>This bit supports interrupt enable. It controls set of **RIC2.QFE11**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE11<br>1: Set RIC2.QFE11 to 1b |
| 10 | QFS10 | 0 | R/W | Queue Full interrupt Set 10<br>This bit supports interrupt enable. It controls set of **RIC2.QFE10**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE10<br>1: Set RIC2.QFE10 to 1b |
| 9 | QFS9 | 0 | R/W | Queue Full interrupt Set 9<br>This bit supports interrupt enable. It controls set of **RIC2.QFE9**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE9<br>1: Set RIC2.QFE9 to 1b |
| 8 | QFS8 | 0 | R/W | Queue Full interrupt Set 8<br>This bit supports interrupt enable. It controls set of **RIC2.QFE8**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE8<br>1: Set RIC2.QFE8 to 1b |
| 7 | QFS7 | 0 | R/W | Queue Full interrupt Set 7<br>This bit supports interrupt enable. It controls set of **RIC2.QFE7**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE7<br>1: Set RIC2.QFE7 to 1b |
| 6 | QFS6 | 0 | R/W | Queue Full interrupt Set 6<br>This bit supports interrupt enable. It controls set of **RIC2.QFE6**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE6<br>1: Set RIC2.QFE6 to 1b |
| 5 | QFS5 | 0 | R/W | Queue Full interrupt Set 5<br>This bit supports interrupt enable. It controls set of **RIC2.QFE5**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE5<br>1: Set RIC2.QFE5 to 1b |
| 4 | QFS4 | 0 | R/W | Queue Full interrupt Set 4<br>This bit supports interrupt enable. It controls set of **RIC2.QFE4**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE4<br>1: Set RIC2.QFE4 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | QFS3 | 0 | R/W | Queue Full interrupt Set 3<br>This bit supports interrupt enable. It controls set of **RIC2.QFE3**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE3<br>1: Set RIC2.QFE3 to 1b |
| 2 | QFS2 | 0 | R/W | Queue Full interrupt Set 2<br>This bit supports interrupt enable. It controls set of **RIC2.QFE2**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE2<br>1: Set RIC2.QFE2 to 1b |
| 1 | QFS1 | 0 | R/W | Queue Full interrupt Set 1<br>This bit supports interrupt enable. It controls set of **RIC2.QFE1**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE1<br>1: Set RIC2.QFE1 to 1b |
| 0 | QFS0 | 0 | R/W | Queue Full interrupt Set 0<br>This bit supports interrupt enable. It controls set of **RIC2.QFE0**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE0<br>1: Set RIC2.QFE0 to 1b |

### 100.1.2.64 Reception Interrupt Disable register 2 (RID2)

The RID2 register is used to control the Reception Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFFD | — | — | — | — | — | — | — | — | — | — | — | — | — | QFD17 | QFD16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | QFD15 | QFD14 | QFD13 | QFD12 | QFD11 | QFD10 | QFD9 | QFD8 | QFD7 | QFD6 | QFD5 | QFD4 | QFD3 | QFD2 | QFD1 | QFD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | RFFD | 0 | R/W | Rx-FIFO Full interrupt Disable<br>This bit supports interrupt enable. It controls set of **RIC2.RFFE**.<br>This bit is always read as 0b.<br>0: No change of RIC2.RFFE<br>1: Set RIC2.RFFE to 0b |
| 30 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | QFD17 | 0 | R/W | Queue Full interrupt Disable 17<br>This bit supports interrupt enable. It controls set of **RIC2.QFE17**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE17<br>1: Set RIC2.QFE17 to 0b |
| 16 | QFD16 | 0 | R/W | Queue Full interrupt Disable 16<br>This bit supports interrupt enable. It controls set of **RIC2.QFE16**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE16<br>1: Set RIC2.QFE16 to 0b |
| 15 | QFD15 | 0 | R/W | Queue Full interrupt Disable 15<br>This bit supports interrupt enable. It controls set of **RIC2.QFE15**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE15<br>1: Set RIC2.QFE15 to 0b |
| 14 | QFD14 | 0 | R/W | Queue Full interrupt Disable 14<br>This bit supports interrupt enable. It controls set of **RIC2.QFE14**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE14<br>1: Set RIC2.QFE14 to 0b |
| 13 | QFD13 | 0 | R/W | Queue Full interrupt Disable 13<br>This bit supports interrupt enable. It controls set of **RIC2.QFE13**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE13<br>1: Set RIC2.QFE13 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 | QFD12 | 0 | R/W | Queue Full interrupt Disable 12<br>This bit supports interrupt enable. It controls set of **RIC2.QFE12**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE12<br>1: Set RIC2.QFE12 to 0b |
| 11 | QFD11 | 0 | R/W | Queue Full interrupt Disable 11<br>This bit supports interrupt enable. It controls set of **RIC2.QFE11**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE11<br>1: Set RIC2.QFE11 to 0b |
| 10 | QFD10 | 0 | R/W | Queue Full interrupt Disable 10<br>This bit supports interrupt enable. It controls set of **RIC2.QFE10**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE10<br>1: Set RIC2.QFE10 to 0b |
| 9 | QFD9 | 0 | R/W | Queue Full interrupt Disable 9<br>This bit supports interrupt enable. It controls set of **RIC2.QFE9**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE9<br>1: Set RIC2.QFE9 to 0b |
| 8 | QFD8 | 0 | R/W | Queue Full interrupt Disable 8<br>This bit supports interrupt enable. It controls set of **RIC2.QFE8**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE8<br>1: Set RIC2.QFE8 to 0b |
| 7 | QFD7 | 0 | R/W | Queue Full interrupt Disable 7<br>This bit supports interrupt enable. It controls set of **RIC2.QFE7**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE7<br>1: Set RIC2.QFE7 to 0b |
| 6 | QFD6 | 0 | R/W | Queue Full interrupt Disable 6<br>This bit supports interrupt enable. It controls set of **RIC2.QFE6**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE6<br>1: Set RIC2.QFE6 to 0b |
| 5 | QFD5 | 0 | R/W | Queue Full interrupt Disable 5<br>This bit supports interrupt enable. It controls set of **RIC2.QFE5**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE5<br>1: Set RIC2.QFE5 to 0b |
| 4 | QFD4 | 0 | R/W | Queue Full interrupt Disable 4<br>This bit supports interrupt enable. It controls set of **RIC2.QFE4**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE4<br>1: Set RIC2.QFE4 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | QFD3 | 0 | R/W | Queue Full interrupt Disable 3<br>This bit supports interrupt enable. It controls set of **RIC2.QFE3**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE3<br>1: Set RIC2.QFE3 to 0b |
| 2 | QFD2 | 0 | R/W | Queue Full interrupt Disable 2<br>This bit supports interrupt enable. It controls set of **RIC2.QFE2**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE2<br>1: Set RIC2.QFE2 to 0b |
| 1 | QFD1 | 0 | R/W | Queue Full interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **RIC2.QFE1**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE1<br>1: Set RIC2.QFE1 to 0b |
| 0 | QFD0 | 0 | R/W | Queue Full interrupt Disable 0<br>This bit supports interrupt enable. It controls set of **RIC2.QFE0**.<br>This bit is always read as 0b.<br>0: No change of RIC2.QFE0<br>1: Set RIC2.QFE0 to 0b |

### 100.1.2.65 Transmission Interrupt Enable register (TIE)

The TIE register is used to control the Transmission Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TDPS3 | TDPS2 | TDPS1 | TDPS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MFWS | MFUS | TFWS | TFUS | — | — | — | — | FTS3 | FTS2 | FTS1 | FTS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 19 | TDPS3 | 0 | R/W | Transmit Descriptor Processed interrupt Set 3<br>This bit supports interrupt enable. It controls set of **TIC.TDPE3**.<br>This bit is always read as 0b.<br>0: No change of TIC.TDPE3<br>1: Set TIC.TDPE3 to 1b |
| 18 | TDPS2 | 0 | R/W | Transmit Descriptor Processed interrupt Set 2<br>This bit supports interrupt enable. It controls set of **TIC.TDPE2**.<br>This bit is always read as 0b.<br>0: No change of TIC.TDPE2<br>1: Set TIC.TDPE2 to 1b |
| 17 | TDPS1 | 0 | R/W | Transmit Descriptor Processed interrupt Set 1<br>This bit supports interrupt enable. It controls set of **TIC.TDPE1**.<br>This bit is always read as 0b.<br>0: No change of TIC.TDPE1<br>1: Set TIC.TDPE1 to 1b |
| 16 | TDPS0 | 0 | R/W | Transmit Descriptor Processed interrupt Set 0<br>This bit supports interrupt enable. It controls set of **TIC.TDPE0**.<br>This bit is always read as 0b.<br>0: No change of TIC.TDPE0<br>1: Set TIC.TDPE0 to 1b |
| 15 to 12 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 11 | MFWS | 0 | R/W | E-MAC status FIFO Warning interrupt Set<br>This bit supports interrupt enable. It controls set of **TIC.MFWE**.<br>This bit is always read as 0b.<br>0: No change of TIC.MFWE<br>1: Set TIC.MFWE to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | MFUS | 0 | R/W | E-MAC status FIFO Updated interrupt Set<br>This bit supports interrupt enable. It controls set of **TIC.MFUE**.<br>This bit is always read as 0b.<br>0: No change of TIC.MFUE<br>1: Set TIC.MFUE to 1b |
| 9 | TFWS | 0 | R/W | Timestamp FIFO Warning interrupt Set<br>This bit supports interrupt enable. It controls set of **TIC.TFWE**.<br>This bit is always read as 0b.<br>0: No change of TIC.TFWE<br>1: Set TIC.TFWE to 1b |
| 8 | TFUS | 0 | R/W | Timestamp FIFO Updated interrupt Set<br>This bit supports interrupt enable. It controls set of **TIC.TFUE**.<br>This bit is always read as 0b.<br>0: No change of TIC.TFUE<br>1: Set TIC.TFUE to 1b |
| 7 to 4 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 3 | FTS3 | 0 | R/W | Frame Transmitted interrupt Set 3<br>This bit supports interrupt enable. It controls set of **TIC.FTE3**.<br>This bit is always read as 0b.<br>0: No change of TIC.FTE3<br>1: Set TIC.FTE3 to 1b |
| 2 | FTS2 | 0 | R/W | Frame Transmitted interrupt Set 2<br>This bit supports interrupt enable. It controls set of **TIC.FTE2**.<br>This bit is always read as 0b.<br>0: No change of TIC.FTE2<br>1: Set TIC.FTE2 to 1b |
| 1 | FTS1 | 0 | R/W | Frame Transmitted interrupt Set 1<br>This bit supports interrupt enable. It controls set of **TIC.FTE1**.<br>This bit is always read as 0b.<br>0: No change of TIC.FTE1<br>1: Set TIC.FTE1 to 1b |
| 0 | FTS0 | 0 | R/W | Frame Transmitted interrupt Set 0<br>This bit supports interrupt enable. It controls set of **TIC.FTE0**.<br>This bit is always read as 0b.<br>0: No change of TIC.FTE0<br>1: Set TIC.FTE0 to 1b |

### 100.1.2.66 Transmission Interrupt Disable register (TID)

The TID register is used to control the Transmission Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TDPD3 | TDPD2 | TDPD1 | TDPD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | MFWD | MFUD | TFWD | TFUD | — | — | — | — | FTD3 | FTD2 | FTD1 | FTD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 19 | TDPD3 | All 0 | R/W | Transmit Descriptor Processed interrupt Disable 3<br>This bit supports interrupt enable. It controls set of **TIC.TDPE3**.<br>This bit is always read as 0b.<br>0: No change of TIC.TDPE3<br>1: Set TIC.TDPE3 to 0b |
| 18 | TDPD2 | All 0 | R/W | Transmit Descriptor Processed interrupt Disable 2<br>This bit supports interrupt enable. It controls set of **TIC.TDPE2**.<br>This bit is always read as 0b.<br>0: No change of TIC.TDPE2<br>1: Set TIC.TDPE2 to 0b |
| 17 | TDPD1 | All 0 | R/W | Transmit Descriptor Processed interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **TIC.TDPE1**.<br>This bit is always read as 0b.<br>0: No change of TIC.TDPE1<br>1: Set TIC.TDPE1 to 0b |
| 16 | TDPD0 | All 0 | R/W | Transmit Descriptor Processed interrupt Disable 0<br>This bit supports interrupt enable. It controls set of **TIC.TDPE0**.<br>This bit is always read as 0b.<br>0: No change of TIC.TDPE0<br>1: Set TIC.TDPE0 to 0b |
| 15 to 12 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 11 | MFWD | 0 | R/W | E-MAC status FIFO Warning interrupt Disable<br>This bit supports interrupt enable. It controls set of **TIC.MFWE**.<br>This bit is always read as 0b.<br>0: No change of TIC.MFWE<br>1: Set TIC.MFWE to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 10 | MFUD | 0 | R/W | E-MAC status FIFO Updated interrupt Disable<br>This bit supports interrupt enable. It controls set of **TIC.MFUE**.<br>This bit is always read as 0b.<br>0: No change of TIC.MFUE<br>1: Set TIC.MFUE to 0b |
| 9 | TFWD | 0 | R/W | Timestamp FIFO Warning interrupt Disable<br>This bit supports interrupt enable. It controls set of **TIC.TFWE**.<br>This bit is always read as 0b.<br>0: No change of TIC.TFWE<br>1: Set TIC.TFWE to 0b |
| 8 | TFUD | 0 | R/W | Timestamp FIFO Updated interrupt Disable<br>This bit supports interrupt enable. It controls set of **TIC.TFUE**.<br>This bit is always read as 0b.<br>0: No change of TIC.TFUE<br>1: Set TIC.TFUE to 0b |
| 7 to 4 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 3 | FTD3 | 0 | R/W | Frame Transmitted interrupt Disable 3<br>This bit supports interrupt enable. It controls set of **TIC.FTE3**.<br>This bit is always read as 0b.<br>0: No change of TIC.FTE3<br>1: Set TIC.FTE3 to 0b |
| 2 | FTD2 | 0 | R/W | Frame Transmitted interrupt Disable 2<br>This bit supports interrupt enable. It controls set of **TIC.FTE2**.<br>This bit is always read as 0b.<br>0: No change of TIC.FTE2<br>1: Set TIC.FTE2 to 0b |
| 1 | FTD1 | 0 | R/W | Frame Transmitted interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **TIC.FTE1**.<br>This bit is always read as 0b.<br>0: No change of TIC.FTE1<br>1: Set TIC.FTE1 to 0b |
| 0 | FTD0 | 0 | R/W | Frame Transmitted interrupt Disable 0<br>This bit supports interrupt enable. It controls set of **TIC.FTE0**.<br>This bit is always read as 0b.<br>0: No change of TIC.FTE0<br>1: Set TIC.FTE0 to 0b |

### 100.1.2.67 Reception Interrupt Enable register 3 (RIE3)

The RIE3 register is used to control the Reception Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RDPS17 | RDPS16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDPS15 | RDPS14 | RDPS13 | RDPS12 | RDPS11 | RDPS10 | RDPS9 | RDPS8 | RDPS7 | RDPS6 | RDPS5 | RDPS4 | RDPS3 | RDPS2 | RDPS1 | RDPS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | RDPS17 | 0 | R/W | Receive Descriptor Processed interrupt Set 17<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE17**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE17<br>1: Set RIC3.RDPE17 to 1b |
| 16 | RDPS16 | 0 | R/W | Receive Descriptor Processed interrupt Set 16<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE16**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE16<br>1: Set RIC3.RDPE16 to 1b |
| 15 | RDPS15 | 0 | R/W | Receive Descriptor Processed interrupt Set 15<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE15**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE15<br>1: Set RIC3.RDPE15 to 1b |
| 14 | RDPS14 | 0 | R/W | Receive Descriptor Processed interrupt Set 14<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE14**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE14<br>1: Set RIC3.RDPE14 to 1b |
| 13 | RDPS13 | 0 | R/W | Receive Descriptor Processed interrupt Set 13<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE13**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE13<br>1: Set RIC3.RDPE13 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 | RDPS12 | 0 | R/W | Receive Descriptor Processed interrupt Set 12<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE12**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE12<br>1: Set RIC3.RDPE12 to 1b |
| 11 | RDPS11 | 0 | R/W | Receive Descriptor Processed interrupt Set 11<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE11**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE11<br>1: Set RIC3.RDPE11 to 1b |
| 10 | RDPS10 | 0 | R/W | Receive Descriptor Processed interrupt Set 10<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE10**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE10<br>1: Set RIC3.RDPE10 to 1b |
| 9 | RDPS9 | 0 | R/W | Receive Descriptor Processed interrupt Set 9<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE9**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE9<br>1: Set RIC3.RDPE9 to 1b |
| 8 | RDPS8 | 0 | R/W | Receive Descriptor Processed interrupt Set 8<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE8**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE8<br>1: Set RIC3.RDPE8 to 1b |
| 7 | RDPS7 | 0 | R/W | Receive Descriptor Processed interrupt Set 7<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE7**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE7<br>1: Set RIC3.RDPE7 to 1b |
| 6 | RDPS6 | 0 | R/W | Receive Descriptor Processed interrupt Set 6<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE6**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE6<br>1: Set RIC3.RDPE6 to 1b |
| 5 | RDPS5 | 0 | R/W | Receive Descriptor Processed interrupt Set 5<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE5**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE5<br>1: Set RIC3.RDPE5 to 1b |
| 4 | RDPS4 | 0 | R/W | Receive Descriptor Processed interrupt Set 4<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE4**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE4<br>1: Set RIC3.RDPE4 to 1b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | RDPS3 | 0 | R/W | Receive Descriptor Processed interrupt Set 3<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE3**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE3<br>1: Set RIC3.RDPE3 to 1b |
| 2 | RDPS2 | 0 | R/W | Receive Descriptor Processed interrupt Set 2<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE2**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE2<br>1: Set RIC3.RDPE2 to 1b |
| 1 | RDPS1 | 0 | R/W | Receive Descriptor Processed interrupt Set 1<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE1**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE1<br>1: Set RIC3.RDPE1 to 1b |
| 0 | RDPS0 | 0 | R/W | Receive Descriptor Processed interrupt Set 0<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE0**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE0<br>1: Set RIC3.RDPE0 to 1b |

### 100.1.2.68 Reception Interrupt Disable register 3 (RID3)

The RID3 register is used to control the Reception Interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RDPD17 | RDPD16 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDPD15 | RDPD14 | RDPD13 | RDPD12 | RDPD11 | RDPD10 | RDPD9 | RDPD8 | RDPD7 | RDPD6 | RDPD5 | RDPD4 | RDPD3 | RDPD2 | RDPD1 | RDPD0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 | RDPD17 | 0 | R/W | Receive Descriptor Processed interrupt Disable 17<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE17**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE17<br>1: Set RIC3.RDPE17 to 0b |
| 16 | RDPD16 | 0 | R/W | Receive Descriptor Processed interrupt Disable 16<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE16**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE16<br>1: Set RIC3.RDPE16 to 0b |
| 15 | RDPD15 | 0 | R/W | Receive Descriptor Processed interrupt Disable 15<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE15**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE15<br>1: Set RIC3.RDPE15 to 0b |
| 14 | RDPD14 | 0 | R/W | Receive Descriptor Processed interrupt Disable 14<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE14**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE14<br>1: Set RIC3.RDPE14 to 0b |
| 13 | RDPD13 | 0 | R/W | Receive Descriptor Processed interrupt Disable 13<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE13**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE13<br>1: Set RIC3.RDPE13 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 12 | RDPD12 | 0 | R/W | Receive Descriptor Processed interrupt Disable 12<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE12**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE12<br>1: Set RIC3.RDPE12 to 0b |
| 11 | RDPD11 | 0 | R/W | Receive Descriptor Processed interrupt Disable 11<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE11**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE11<br>1: Set RIC3.RDPE11 to 0b |
| 10 | RDPD10 | 0 | R/W | Receive Descriptor Processed interrupt Disable 10<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE10**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE10<br>1: Set RIC3.RDPE10 to 0b |
| 9 | RDPD9 | 0 | R/W | Receive Descriptor Processed interrupt Disable 9<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE9**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE9<br>1: Set RIC3.RDPE9 to 0b |
| 8 | RDPD8 | 0 | R/W | Receive Descriptor Processed interrupt Disable 8<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE8**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE8<br>1: Set RIC3.RDPE8 to 0b |
| 7 | RDPD7 | 0 | R/W | Receive Descriptor Processed interrupt Disable 7<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE7**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE7<br>1: Set RIC3.RDPE7 to 0b |
| 6 | RDPD6 | 0 | R/W | Receive Descriptor Processed interrupt Disable 6<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE6**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE6<br>1: Set RIC3.RDPE6 to 0b |
| 5 | RDPD5 | 0 | R/W | Receive Descriptor Processed interrupt Disable 5<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE5**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE5<br>1: Set RIC3.RDPE5 to 0b |
| 4 | RDPD4 | 0 | R/W | Receive Descriptor Processed interrupt Disable 4<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE4**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE4<br>1: Set RIC3.RDPE4 to 0b |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | RDPD3 | 0 | R/W | Receive Descriptor Processed interrupt Disable 3<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE3**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE3<br>1: Set RIC3.RDPE3 to 0b |
| 2 | RDPD2 | 0 | R/W | Receive Descriptor Processed interrupt Disable 2<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE2**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE2<br>1: Set RIC3.RDPE2 to 0b |
| 1 | RDPD1 | 0 | R/W | Receive Descriptor Processed interrupt Disable 1<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE1**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE1<br>1: Set RIC3.RDPE1 to 0b |
| 0 | RDPD0 | 0 | R/W | Receive Descriptor Processed interrupt Disable 0<br>This bit supports interrupt enable. It controls set of **RIC3.RDPE0**.<br>This bit is always read as 0b.<br>0: No change of RIC3.RDPE0<br>1: Set RIC3.RDPE0 to 0b |

### 100.1.2.69 E-MAC Mode Register (ECMR)

ECMR is used to specify the operating mode of the E-MAC. The settings in this register are normally made in the initialization process following a reset.

The operating mode settings must not be changed while transmission or reception is enabled (i.e. while the RE or TE bit in this register is 1).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | TRCCM | — | — | RCSC | — | DPAD | RZPF | TZPF | PFR | RXF | TXF |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | MPDE | — | — | RE | TE | — | — | — | DM | PRM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 26 | TRCCM | 0 | R/W | Counter Clear Mode<br>This bit sets the method for clearing the counter register. Refer to the descriptions of the counter registers.<br>0: Writing to a counter register leads to the register being cleared to 0.<br>1: Reading from a counter register leads to the register being cleared to 0. |
| 25, 24 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 23 | RCSC | 0 | R/W | Checksum Calculation<br>Setting this bit to 1 enables automatic calculation of checksums for data in received frames.<br>Only the data field of an Ethernet frame without a VLAN field is in the scope of checksum calculation. Specifically, the checksum is calculated from the data field, which follows the length/type field and is followed by the CRC field. Calculation only involves 16-bit addition; it does not involve bit inversion.<br>0: Checksums are not automatically calculated.<br>1: Checksums are automatically calculated. |
| 22 | — | 0 | R/W | Reserved<br>This bit is read as 0. The write value should be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 21 | DPAD | 0 | R/W | Data Padding<br>This bit specifies padding or non-padding of data when less than 60 bytes are to be transmitted.<br>When this bit is set to 1, data are transmitted without padding; when it is set to 0, data are padded to make up 60-byte units for transmission.<br>0: Padding to make up 60 bytes is inserted in data for transmission when fewer than 60 bytes are to be transmitted.<br>1: Padding is not inserted in data for transmission when fewer than 60 bytes are to be transmitted and the data are transmitted without being changed. |
| 20 | RZPF | 0 | R/W | PAUSE Frame Reception with Time = 0<br>When the RZPF bit is set to 0, received PAUSE frames with the Timer value 0 are discarded.<br>When the RZPF bit is set to 1, release from the transmission wait state follows reception of a PAUSE frame with the Timer value 0.<br>0: Reception of PAUSE frames with the TIME parameter value 0 is disabled.<br>1: Reception of PAUSE frames with the TIME parameter value 0 is enabled. |
| 19 | TZPF | 0 | R/W | Transmit Zero PAUSE Frame<br>When setting CCC.FCE=1, and Rx FIFO become Warning Level.<br>0: Not transmit the PAUSE frame of TIME parameter value 0.<br>1: Transmit the PAUSE frame of TIME parameter value 0. |
| 18 | PFR | 0 | R/W | PAUSE Frame Receive Mode<br>This bit specifies whether PAUSE frames are transferred to the AVB-DMAC.<br>0: PAUSE frames are not transferred to the AVB-DMAC.<br>1: PAUSE frames are transferred to the AVB-DMAC. |
| 17 | RXF | 0 | R/W | Operating Mode for Flow Control in Reception<br>When the RXF bit is set to 1 and a PAUSE frame is received, a next frame to be transmitted is not transmitted until the time indicated by the Timer value in the PAUSE frame has elapsed. However, the transmission of a current frame is continued. The number of received PAUSE frames is also counted. For details, see section 100.1.2.95 PAUSE Frame Receive Counter (PFRCR).<br>Setting this bit to 0 disables PAUSE frame detection.<br>0: Detection of PAUSE frames is disabled.<br>1: Flow control for the receiving port is enabled. |
| 16 | TXF | 0 | R/W | Transmit Flow control mode<br>0: Flow control for the transmitting port is disabled.<br>1: Flow control for the transmitting port is enabled. |
| 15 to 10 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 9 | MPDE | 0 | R/W | Magic Packet™ Detection Enable<br>The MPDE bit enables or disables Magic Packet™ detection by hardware to allow activation via the Ethernet connection.<br>0: Magic Packet™ detection is not enabled.<br>1: Magic Packet™ detection is enabled. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8, 7 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 6 | RE | 0 | R/W | Reception Enable<br>If this bit is switched from reception being enabled (RE = 1) to reception being disabled (RE = 0) while a frame is being received, reception will continue until reception of the frame is completed.<br>0: Reception is disabled.<br>1: Reception is enabled. |
| 5 | TE | 0 | R/W | Transmission Enable<br>If this bit is switched from transmission being enabled (TE = 1) to transmission being disabled (TE = 0) while a frame is being transmitted, transmission will continue until transmission of that frame is completed.<br>0: Transmission is disabled.<br>1: Transmission is enabled. |
| 4 to 2 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 1 | DM | 0 | R/W | Duplex Mode<br>This bit should always be set to 1. The value after reset is 0.<br>0: Value after reset<br>1: Full-duplex operation<br>This bit should always be set to 1. |
| 0 | PRM | 0 | R/W | Promiscuous Mode<br>Setting the PRM bit enables all Ethernet frames to be received. All Ethernet frames mean all receivable frames, irrespective of differences of destination address (exclude PAUSE frame).<br>0: Normal operation<br>1: Promiscuous mode operation |

### 100.1.2.70 Receive Frame Length Register (RFLR)

The RFLR register specifies the maximum length (in bytes) of frames that can be received by this LSI. Settings in this register must not be changed while reception is enabled (while the RE bit in the E-MAC mode register (ECMR) is 1).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RFL[17:16] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFL[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 17 to 0 | RFL[17:0] | H'0 0000 | R/W | Receive Frame Length<br>Frame data described here refers to all fields from the destination address up to the CRC data. Frame contents from the destination address up to the data are actually transferred to memory. CRC data are not included in the transfer. When more data than the specified number of bytes are received, the portion of data that exceeds the specified value and more 8 byte length is discarded.<br>CAUTION: The prepared descriptor data size is just the specified value (RFLR.RFL). Therefore descriptor data size must be more than RFLR.RFL + 8 if you will receive such the long frame.<br>H'0 0000 to H'0 05EE: 1,518 bytes<br>H'0 05EF: 1,519 bytes<br>H'0 05F0: 1,520 bytes<br>: :<br>H'0 07FF: 2,047 bytes<br>H'0 0800: 2,048 bytes<br>: :<br>H'0 1000: 4,096 bytes<br>: :<br>H'1 0000: 65,535 bytes<br>: :<br>H'2 0000 to H'3 FFFF: 131,072 bytes |

### 100.1.2.71 E-MAC Status Register (ECSR)

The ECSR register indicates the state of the E-MAC. The CPU can be notified of the state. For bits associated with interrupts, the interrupt can be enabled or disabled by the corresponding bit in the E-MAC Interrupt Permission Register (ECSIPR) described in section 100.1.2.88.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | PFRI | PHYI | LCHNG | MPD | ICD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 | PFRI | 0 | R/W | PAUSE Frame Retry Interrupt<br>This bit indicate that reached to the number of PAUSE frame retransmission for register PFTTLR setting.<br>Procedure for 0-clear to this bit is follows.<br>0: Read the PFTTCR register to clear the PFTTCR register.<br>1: Write to this bit 1. |
| 3 | PHYI | 0 | R | PHY interrupt terminal state bit<br>This bit indicates the state of input to the PHY interrupt pin (AVB_PHY_INT) from the PHY-LSI.<br>0: PHY interrupt terminal (AVB_PHY_INT) is not asserted.<br>1: PHY interrupt terminal (AVB_PHY_INT) is asserted. |
| 2 | LCHNG | 0 | R/W | Link signal change bit<br>This bit indicates a transition of the link status signal (AVB_LINK) input from the PHY-LSI from high to low or low to high.<br>However, signal changes may also be detected at times when the link status signal (AVB_LINK) function is selected.<br>To check the current link state, refer to the link status pin state bit in the PHY status register (PSR.LMON).<br>0: The change of Link status signal (AVB_LINK) is not detected.<br>1: The change of Link status signal (AVB_LINK) is detected. |
| 1 | MPD | 0 | R/W | Magic Packet™ Detection<br>This bit indicates that a Magic Packet™ has been detected on the line.<br>Writing 1 to this bit clears it to 0.<br>0: A Magic Packet™ has not been detected.<br>1: A Magic Packet™ has been detected. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | ICD | 0 | R/W | Illegal Carrier Detection<br>This bit indicates that the PHY-LSI has detected an illegal carrier on the line. If a change in the signal input from the PHY-LSI occurs in a period shorter than the software recognition period, the correct information may not be obtained. Refer to the timing specification for the PHY-LSI used.<br>Writing 1 to this bit clears it to 0.<br>0: PHY-LSI has not detected an illegal carrier on the line.<br>1: PHY-LSI has detected an illegal carrier on the line. |

### 100.1.2.72 E-MAC Interrupt Permission Register (ECSIPR)

The ECSIPR register enables or disables the states indicated by the ECSR register as interrupt sources. Each effective bit disables or enables interrupts corresponding to the bits in ECSR.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | PFRIM | PHYIM | LINKIM | MPDIP | ICDIP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 | PFRIM | 0 | R/W | PAUSE Frame Retry Interrupt Mask<br>Setting this bit to 1 selects interrupt generation on setting of the PAUSE Frame Retry Interrupt bit (ECSR. PFRI) in the E-MAC status register to 1.<br>0: Interrupts on setting of the PFRI bit is disabled.<br>1: Interrupts on setting of the PFRI bit is enabled. |
| 3 | PHYIM | 0 | R/W | PHY Interrupt Mask<br>Setting this bit to 1 selects interrupt generation on setting of the PHY interrupt terminal state bit (ECSR. PHYI) in the E-MAC status register to 1.<br>0: Interrupts on setting of the PHYI bit is disabled.<br>1: Interrupts on setting of the PHYI bit is enabled. |
| 2 | LINKIM | 0 | R/W | LINK Interrupt Mask<br>Setting this bit to 1 selects interrupt generation on setting of the Link signal change bit (ECSR. LCHNG) in the E-MAC status register to 1.<br>0: Interrupts on setting of the LCHNG bit is disabled.<br>1: Interrupts on setting of the LCHNG bit is enabled. |
| 1 | MPDIP | 0 | R/W | Magic Packet™ Detect Interrupt Enable<br>Setting this bit to 1 selects interrupt generation on setting of the Magic PacketTM detection bit (ECSR.MPD) in the E-MAC status register to 1.<br>0: Interrupts on setting of the MPD bit is disabled.<br>1: Interrupts on setting of the MPD bit is enabled. |
| 0 | ICDIP | 0 | R/W | Illegal Carrier Detect Interrupt Enable<br>Setting this bit to 1 selects interrupt generation on setting of the illegal carrier detection bit (ECSR.ICD) in the E-MAC status register to 1.<br>0: Interrupts on setting of the ICD bit is disabled.<br>1: Interrupt on setting of the ICD bit is enabled.<br>Not support for 1000Mbps mode. |

### 100.1.2.73 PHY Interface Register (PIR)

The PIR register provides a means of access to the PHY-LSI internal registers via the MII.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MDI | MDO | MMD | MDC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 3 | MDI | — | R | MII Management Data-In<br>This bit indicates the level of the AVB_MDIO pin. |
| 2 | MDO | 0 | R/W | MII Management Data-Out<br>This bit holds data for output from the AVB_MDIO pin.<br>The AVB_MDIO pin outputs data when the MMD bit is set to 1 (to specify writing as the direction). Data are not output while the MMD bit is set to 0 (to specify reading as the direction). |
| 1 | MMD | 0 | R/W | MII Management Mode<br>This bit specifies the direction for data through MDIO (reading or writing).<br>0: Read direction is specified.<br>1: Write direction is specified. |
| 0 | MDC | 0 | R/W | MII Management Data Clock<br>Values set in this bit are output on the AVB_MDC pin to supply the MII with the management data clock. For the method of access to the MII registers, see section 100.1.3.13 Connection to PHY-LSI. |

### 100.1.2.74 PHY Status Register (PSR)

The PSR register is a read-only register that can read interface signals from the PHY-LSI.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | LMON |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are read as 0. |
| 0 | LMON | 0 | R | Link Status Pin State<br>The Link state can be read by connecting the Link signal output from the PHY-LSI to the link status pin (AVB_LNK) pin.<br>For the active sense, refer to the specifications of the PHY-LSI to be connected.<br>0: The link status signal (AVB_LNK) is at the low level.<br>1: The link status signal (AVB_LNK) is at the high level. |

### 100.1.2.75 PHY_INT Polarity Register (PIPR)

The PIPR register is used to set the active sense of the AVB_PHY-INT pin.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

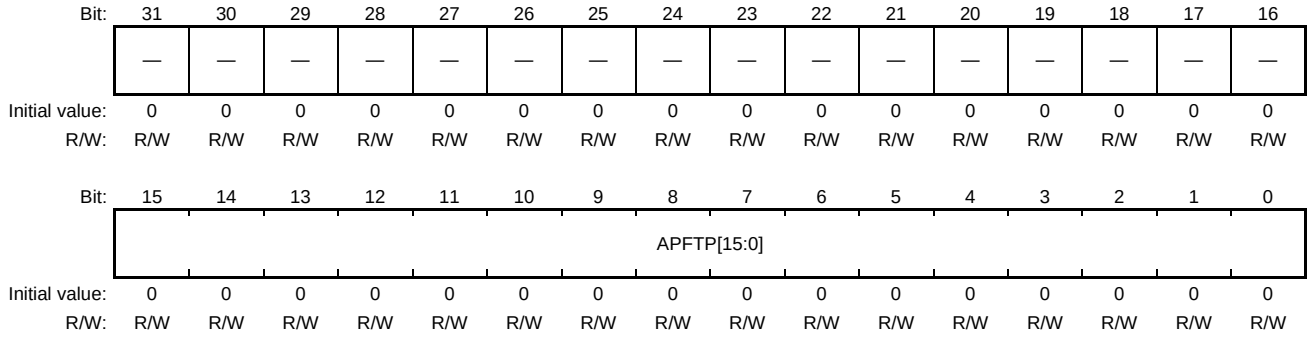
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PHYIP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 0 | PHYIP | 0 | R/W | PHY Interrupt Input Pin Polarity<br>This bit sets the active sense of the PHY interrupt pin (AVB_PHY-INT).<br>For the active sense, refer to the specifications of the PHY-LSI to be connected.<br>0: PHY interrupt pin (AVB_PHY-INT) is active low (the low level triggers the interrupt state)<br>1: PHY interrupt pin (AVB_PHY-INT) is active high (the high level triggers the interrupt state) |

### 100.1.2.76 Automatic PAUSE frame register (APR)

The APR register is used to set the value for the TIME parameter of auto generated PAUSE frames.

When ECMR.TXF=b'1, the value shouldn't set all 0 in this register.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | APFTP[15:0] | H'0000 | R/W | Auto Pause Frame Time Parameter<br>When a PAUSE frame is auto transmitted, the value set in this register is used as its TIME parameter.<br>H'0000: —<br>H'0001: 1 × 512 bit time<br>H'0002: 2 × 512 bit time<br>: :<br>H'FFFF: 65535 × 512 bit time<br>A bit time changes relative to the transfer speed as follows.<br>1000 Mbps:1 bit time = 1 ns<br>100 Mbps: 1 bit time = 10 ns |

### 100.1.2.77 Manual PAUSE Frame Register (MPR)

The MPR register is used to set the value for the TIME parameter of manually generated PAUSE frames. When a PAUSE frame is manually transmitted, the value set in this register is used as its TIME parameter.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MP[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | MP[15:0] | H'0000 | R/W | Manual PAUSE<br>These bits set the value of the TIME parameter in manually generated PAUSE frames.<br>The unit for the setting is 512 bit time.<br>These bits set the TIME parameter value of a manual PAUSE frame.<br>H'0000: —<br>H'0001: 1 × 512 bit time<br>H'0002: 2 × 512 bit time<br>: :<br>H'FFFF: 65535 × 512 bit time<br>A bit time changes relative to the transfer speed as follows.<br>1000 Mbps:1 bit time = 1 ns<br>100 Mbps: 1 bit time = 10 ns |

### 100.1.2.78 PAUSE Frame Transmit Counter (PFTCR)

The PFTCR register is a counter that indicates the number of times PAUSE frames have been transmitted.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFTXC[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | PFTXC[15:0] | H'0000 | R | PAUSE Frame Transmit Counter<br>Counter for counting the number of transmitted PAUSE frames<br>The bits are cleared to 0 when they are read.<br>If counting up and clearing of the counter coincide, clearing the counter takes priority. |

### 100.1.2.79 PAUSE Frame Receive Counter (PFRCR)

The RFRCR register is a counter that indicates the number of times PAUSE frames have been received.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFRXC[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | PFRXC[15:0] | H'0000 | R | PAUSE Frame Receive Counter<br>These bits indicate the number of PAUSE frames that have been received when flow control in reception is enabled (the RXF bit in ECMR = 1).<br>The bits are cleared to 0 when they are read.<br>If counting up and clearing the counter coincide, clearing the counter takes priority. |

### 100.1.2.80 Automatic PAUSE frame retransmit count register (TPAUSER)

The TPAUSER register is used to set the value for upper limit number of auto generated PAUSE frames.

The setting in this register must not be changed while transmission is enabled.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFRTULMT[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | PFRTULMT[15:0] | H'0000 | R/W | Pause Frame Retry Upper LiMiT<br>These bit's set the value for upper limit time of auto generated PAUSE frames while Receive FIFO caution.<br>H'0000: unlimited<br>H'0001: 1 time<br>H'0002: 2 times<br>: :<br>H'FFFF: 65535 times |

### 100.1.2.81 PAUSE frame transmit times counter (PFTTCR)

The PFTTCR register is a counter that indicates the number of times auto PAUSE frames have been transmit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFRTC[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | PFRTC[15:0] | H'0000 | R | Pause Frame Retry Counter<br>These bits indicate the number of auto PAUSE frames that have been transmit while Receive FIFO caution.<br>The bits are cleared to 0 when they are read and Receive FIFO caution start. |

### 100.1.2.82 E-MAC Mode Register 2 (GECMR)

The GECMR register specifies the operating mode for the E-MAC.

The setting in the GECMR register must not be changed while transmission or reception is enabled.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

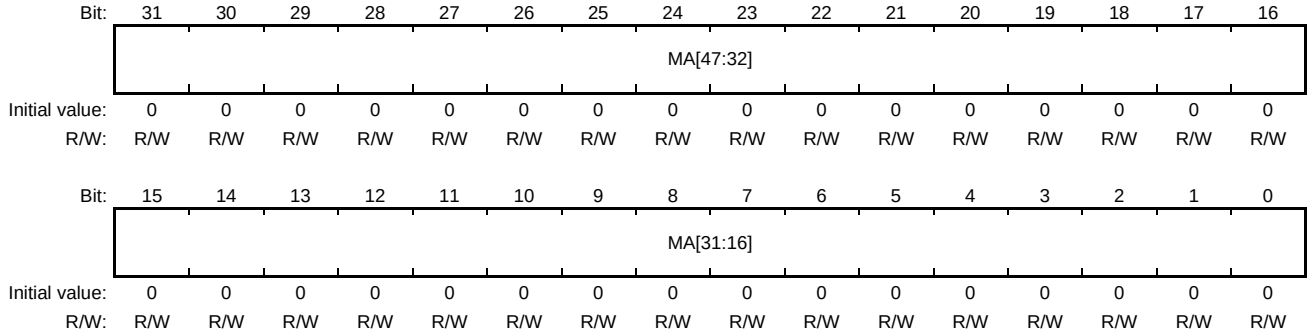
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SPEED |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 0 | SPEED | 0 | R/W | Transfer Speed Setting<br>This bit sets the transfer rate.<br>0: Transfer is at 100 Mbps.<br>1: Transfer is at 1000 Mbps. |

### 100.1.2.83 E-MAC Address High Register (MAHR)

The MAHR register specifies the 32 higher-order bits of the 48-bit E-MAC address. The settings in this register are normally made in the initialization process after a reset.

The settings in this register must not be changed while transmission or reception is enabled.

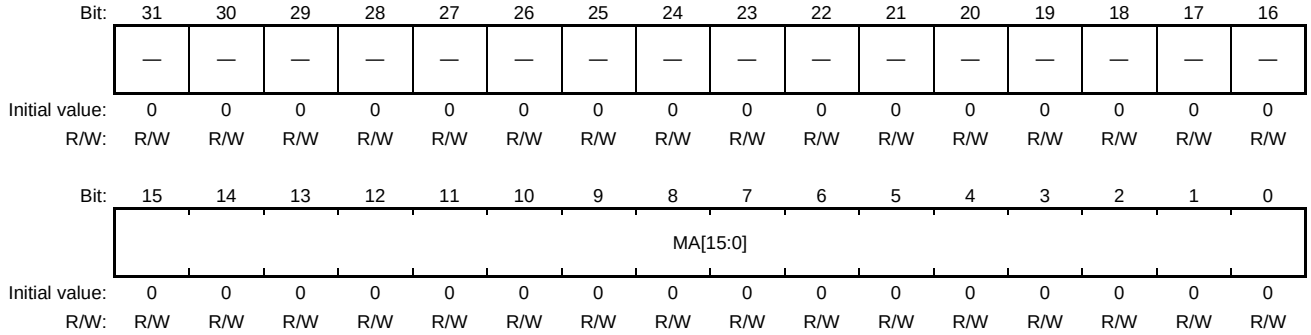



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | MA[47:16] | H'0000 0000 | R/W | E-MAC Address Bits 47 to 16<br>These bits are used to set the 32 higher-order bits of the E-MAC address.<br>For example, if the E-MAC address is 01-23-45-67-89-AB (hexadecimal), set H'0123 4567 in the MAHR register. |

### 100.1.2.84 E-MAC Address Low Register (MALR)

The MALR register specifies the 16 lower-order bits of the 48-bit E-MAC address. The settings in this register are normally made in the initialization process after a reset.

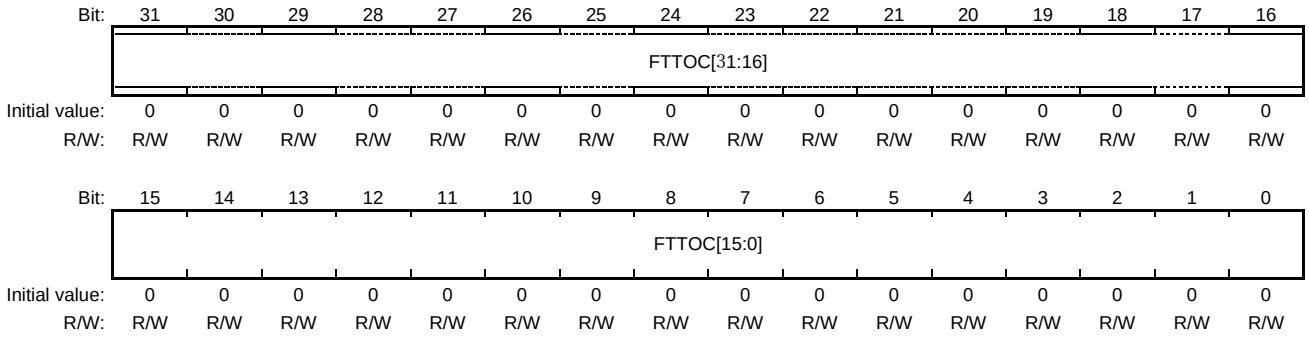
The settings in this register must not be changed while transmission or reception is enabled.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | MA[15:0] | H'0000 | R/W | E-MAC Address Bits 15 to 0<br>These bits are used to set the 16 lower-order bits of the E-MAC address.<br>For example, if the E-MAC address is 01-23-45-67-89-AB (hexadecimal), set H'89AB in the MALR register. |

### 100.1.2.85 Transmit retry over counter register (TROCR)

The TROCR register is a counter that indicates the number of times frames with time-out were transmit. Counting up stops when the value in this register reaches H'FFFF FFFF.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | FTTOC [31:0] | H'0000 0000 | R/W | Frame transmit time-out counter<br>These bits indicate the number of transmit frames having time-out.<br>The bits are cleared to 0 when they are read while the counter clear mode bit (TRCCM) in the E-MAC mode register is set to 1.<br>When TRCCM = 0, they are cleared to 0 by the writing of any value to this register. |

### 100.1.2.86 CRC Error Frame Receive Counter Register (CEFCR)

The CEFCR register is a counter that indicates the number of times frames with CRC errors were received. Counting up stops when the value in this register reaches H'0000 FFFF.

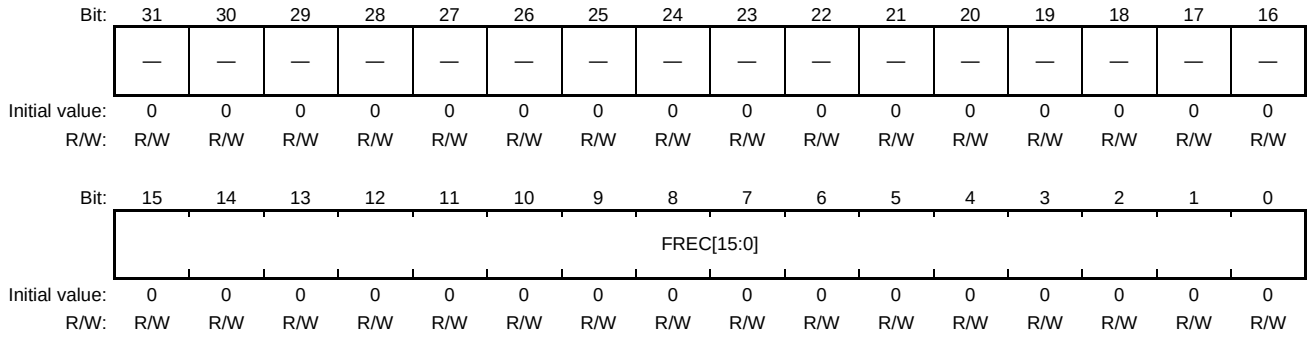
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CEFC[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | CEFC[15:0] | H'0000 | R/W | CRC Error Frame Counter<br>These bits indicate the number of received frames having CRC errors.<br>The bits are cleared to 0 when they are read while the counter clear mode bit (TRCCM) in the E-MAC mode register is set to 1.<br>When TRCCM = 0, they are cleared to 0 by the writing of any value to this register. |

### 100.1.2.87 Frame Receive Error Counter Register (FRECR)

The FRECR register is a counter that indicates the number of frames for which receive errors were generated by input on the RX_ER from the PHY-LSI. Counting up stops when the value in this register reaches H'0000 FFFF.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | FREC[15:0] | H'0000 | R/W | Frame Receive Error Counter<br>These bits indicate the number of errors during frame reception.<br>The bits are cleared to 0 when they are read while the counter clear mode bit (TRCCM) in the E-MAC mode register is set to 1.<br>When TRCCM = 0, they are cleared to 0 by the writing of any value to this register. |

### 100.1.2.88 Too-Short Frame Receive Counter Register (TSFRCR)

The TSFRCR register is a counter that indicates the number of received frames that were fewer than 64 bytes in length. Counting stops when the value in this register reaches H'0000 FFFF.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSFRC[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | TSFRC[15:0] | H'0000 | R/W | Too-Short Frame Receive Counter<br>These bits indicate the number of received frames that were fewer than 64 bytes in length.<br>The bits are cleared to 0 when they are read while the counter clear mode bit (TRCCM) in the E-MAC mode register is set to 1.<br>When TRCCM = 0, they are cleared to 0 by the writing of any value to this register. |

### 100.1.2.89 Too-Long Frame Receive Counter Register (TLFRCR)

The TLFRCR register is a counter that indicates the number of received frames that were longer than the value specified in the receive frame length register (RFLR). Counting up stops when the value in the TLFRCR register reaches H'0000 FFFF.

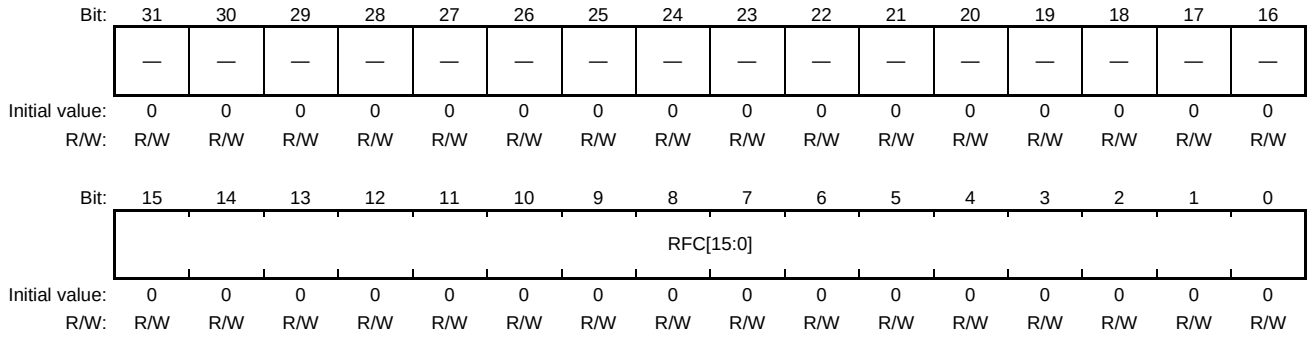
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TLFC[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | TLFC[15:0] | H'0000 | R/W | Too-Long Frame Receive Counter<br>These bits indicate the number of received frames that were longer than the value in RFLR.<br>The bits are cleared to 0 when they are read while the counter clear mode bit (TRCCM) in the E-MAC mode register is set to 1.<br>When TRCCM = 0, they are cleared to 0 by the writing of any value to this register. |

### 100.1.2.90 Residual-Bit Frame Receive Counter Register (RFCR)

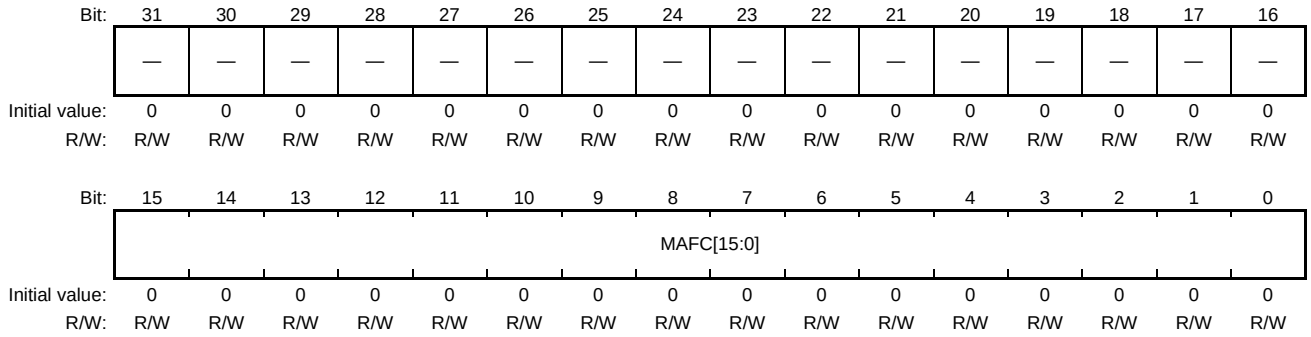
The RFCR register is a counter that indicates the number of received frames containing “residual bits” (trailing bits not making up an 8-bit unit). Counting up stops when the value in this register reaches H'0000 FFFF.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | RFC[15:0] | H'0000 | R/W | Residual-Bit Frame Receive Counter<br>These bits indicate the number of received frames containing residual bits.<br>The bits are cleared to 0 when they are read while the counter clear mode bit (TRCCM) in the E-MAC mode register is set to 1.<br>When TRCCM = 0, they are cleared to 0 by the writing of any value to this register. |

### 100.1.2.91 Multicast Address Frame Receive Counter Register (MAFCR)

The MAFCR register is a counter that indicates the number of received frames for which a multicast address was specified. Counting up stops when the value in this register reaches H'0000 FFFF.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 15 to 0 | MAFC[15:0] | H'0000 | R/W | Multicast Address Frame Counter<br>These bits indicate the number of multicast frames that have been received.<br>The bits are cleared to 0 when they are read while the counter clear mode bit (TRCCM) in the E-MAC mode register is set to 1.<br>When TRCCM = 0, they are cleared to 0 by the writing of any value to this register. |

### 100.1.2.92 Checksum Operating Mode Register (CSR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | H'0000 0200 | R/W | Reserved<br>These bits are read as H'0000 0200. The write value should be H'0000 0200. |

### 100.1.2.93 Rx Frame Checksum Enable Register (CSR2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.94 Unsupported Rx Frame Counter Register (CSR30)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.95 Rx IPv4 Header checksum Error Counter Register (CSR31)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

### 100.1.2.96 Rx TCP/UDP/ICMP checksum Error Counter register (CSR32)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |

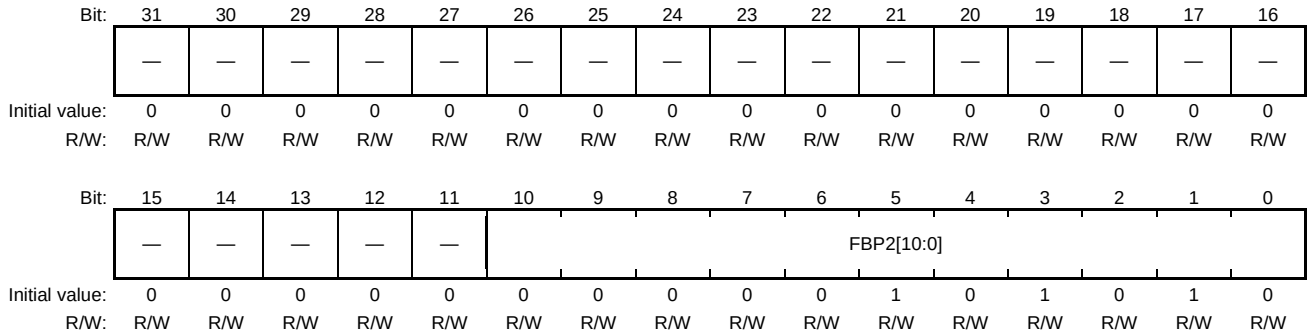
### 100.1.2.97 UDP/IP Support Mode Register (USMR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | EI6F | EI4F | EMOM | EEOM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 3 | EI6F | 0 | R/W | Enable UDP/IPv6 Filtering<br>0: UDP/IPv6 Filtering disabled<br>1: UDP/IPv6 Filtering enabled |
| 2 | EI4F | 0 | R/W | Enable UDP/IPv4 Filtering<br>0: UDP/IPv4 Filtering disabled<br>1: UDP/IPv4 Filtering enabled |
| 1 | EMOM | 0 | R/W | Enable Manual Offset Mode<br>0: Manual Offset Mode disabled<br>1: Manual Offset Mode enabled<br>When this bit is 0, These bits set the position in Ethernet frames of the first byte of the bytes(see section 100.1.3.15(1)(d) Disable Manual Offset Mode) to be used by the separation filter.<br>When this bit is 1, These bits set the position in Ethernet frames of the first byte of the bytes**(I4SFO.FBP, I4DFO.FBP, I6SFO.FBP, I6DFO.FBP and UPFO.FBP)** to be used by the separation filter. |
| 0 | EEOM | 0 | R/W | Enable Extended Offset Mode<br>0: Extended Offset Mode disabled<br>1: Extended Offset Mode enabled<br>When this bit is 0, enables the position in Ethernet frames of the first byte of the bytes(SFO.FBP) to be used by the separation filter.<br>When this bit is 1, enables the position in Ethernet frames of the first byte of the bytes(SFO2.FBP2) to be used by the separation filter. |

### 100.1.2.98 Separation Filter Offset Register 2 (SFO2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 to 0 | FBP2[10:0] | H'02A | R/W | First Byte Position2<br>These bits set the position in Ethernet frames of the first byte of the bytes to be used by the separation filter.<br>When these bits are 0, the separation filter starts from the start of each Ethernet frame (first byte of the destination address). For bytes in Ethernet frames, see Figure 100.1.3.1.1 Operating Mode of AVB-DMAC, in section 100.1.3.1(1) Operating Modes.<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>[Changing condition]<br>These bits are updated to the value of **SFV2.LV**[11:0] when **SFL2.LC** changes from 30 (Load SFO2.FBP2) to 31 while **CSR.OPS** is OPERATION.<br>For separation filtering, see section 100.1.3.15(1)(a) Separation Filtering. |

### 100.1.2.99 IPv4 Source Address Filter Offset Register (I4SFO)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | FBP[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 to 0 | FBP[10:0] | H'01A | R/W | First Byte Position<br>These bits set the position in Ethernet frames of the first byte of the bytes to be used by the separation filter.<br>When these bits are 0, the separation filter starts from the start of each Ethernet frame (first byte of the destination address). For bytes in Ethernet frames, see Figure 100.1.3.1.1 Operating Mode of AVB-DMAC, in section 100.1.3.1(1) Operating Modes.<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>[Changing condition]<br>These bits are updated to the value of **I4SFVR.LV**[11:0] when **I4SFLR.LC** changes from 30 (Load I4SFO.FBP) to 31 while **CSR.OPS** is OPERATION.<br>For separation filtering, see section 100.1.3.15(1)(a) Separation Filtering. |

### 100.1.2.100 IPv4 Destination Address Filter Offset Register (I4DFO)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | FBP[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 to 0 | FBP[10:0] | H'01E | R/W | First Byte Position<br>These bits set the position in Ethernet frames of the first byte of the bytes to be used by the separation filter.<br>When these bits are 0, the separation filter starts from the start of each Ethernet frame (first byte of the destination address). For bytes in Ethernet frames, see Figure 100.1.3.1.1 Operating Mode of AVB-DMAC, in section 100.1.3.1(1) Operating Modes.<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>[Changing condition]<br>These bits are updated to the value of **I4DFVR.LV**[11:0] when **I4DFLR.LC** changes from 30 (Load I4DFO.FBP) to 31 while **CSR.OPS** is OPERATION.<br>For separation filtering, see section 100.1.3.15(1)(a) Separation Filtering. |

### 100.1.2.101 IPv6 Source Address Filter Offset Register (I6SFO)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | FBP[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 to 0 | FBP[10:0] | H'016 | R/W | First Byte Position<br>These bits set the position in Ethernet frames of the first byte of the bytes to be used by the separation filter.<br>When these bits are 0, the separation filter starts from the start of each Ethernet frame (first byte of the destination address). For bytes in Ethernet frames, see Figure 100.1.3.1.1 Operating Mode of AVB-DMAC, in section 100.1.3.1(1) Operating Modes.<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>[Changing condition]<br>These bits are updated to the value of **I6SFVR.LV**[11:0] when **I6SFLR.LC** changes from 30 (Load I6SFO.FBP) to 31 while **CSR.OPS** is OPERATION.<br>For separation filtering, see section 100.1.3.15(1)(a) Separation Filtering. |

### 100.1.2.102 IPv6 Destination Address Filter Offset Register (I6DFO)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | FBP[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 to 0 | FBP[10:0] | H'026 | R/W | First Byte Position<br>These bits set the position in Ethernet frames of the first byte of the bytes to be used by the separation filter.<br>When these bits are 0, the separation filter starts from the start of each Ethernet frame (first byte of the destination address). For bytes in Ethernet frames, see Figure 100.1.3.1.1 Operating Mode of AVB-DMAC, in section 100.1.3.1(1) Operating Modes.<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>[Changing condition]<br>These bits are updated to the value of **I6DFVR.LV**[11:0] when **I6DFLR.LC** changes from 30 (Load I6DFO.FBP) to 31 while **CSR.OPS** is OPERATION.<br>For separation filtering, see section 100.1.3.15(1)(a) Separation Filtering. |

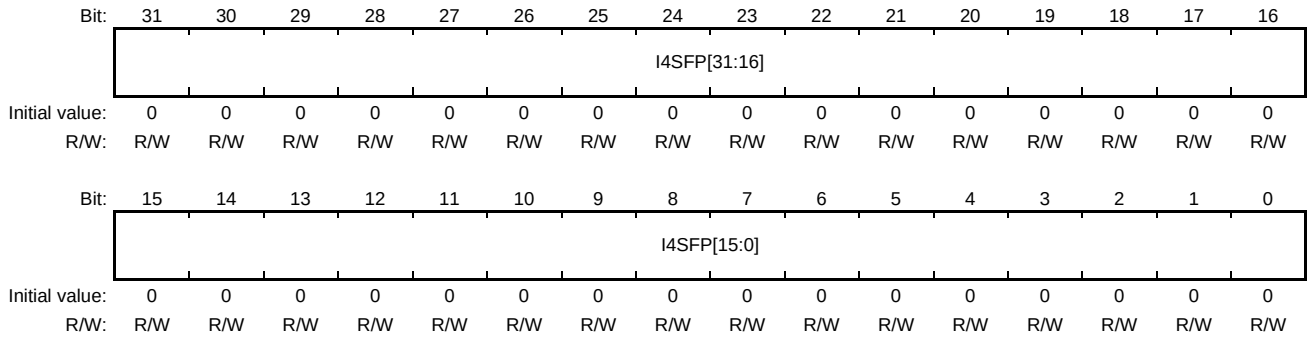
### 100.1.2.103 UDP Port Number Filter Offset Register (UPFO)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | FBP[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

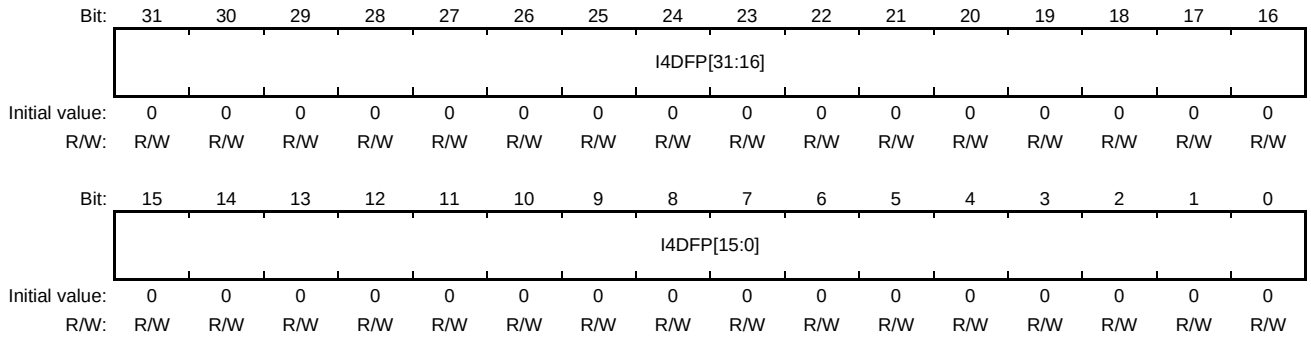
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should be 0. |
| 10 to 0 | FBP[10:0] | H'022 | R/W | First Byte Position<br>These bits set the position in Ethernet frames of the first byte of the bytes to be used by the separation filter.<br>When these bits are 0, the separation filter starts from the start of each Ethernet frame (first byte of the destination address). For bytes in Ethernet frames, see Figure 100.1.3.1.1 Operating Mode of AVB-DMAC, in section 100.1.3.1(1) Operating Modes.<br>Writing to the bits is only possible when the current operating mode is configuration mode.<br>[Changing condition]<br>These bits are updated to the value of **UPFVR.LV**[5:0] when **UPFLR.LC** changes from 30 (Load UPFO.FBP) to 31 while **CSR.OPS** is OPERATION.<br>For separation filtering, see section 100.1.3.15(1)(a) Separation Filtering. |

### 100.1.2.104 IPv4 Source Address Filter Pattern Register i (I4SFPRi) (i = 0 to 15)



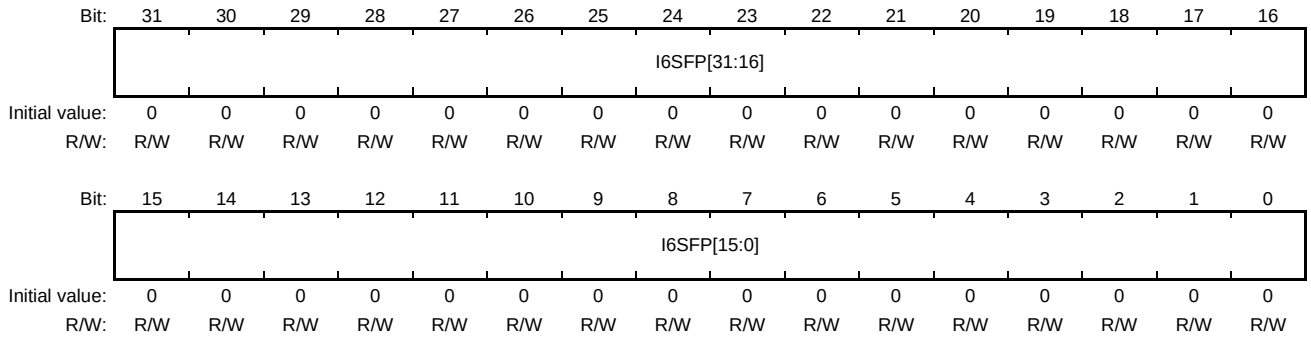

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | I4SFP[31:0] | H'0000 0000 | R/W | IP v4 Source Address Filter Pattern<br>These bits set the pattern of the address filter.<br>The 32-bit filter pattern is set for each queue. |

### 100.1.2.105 IPv4 Destination Address Filter Pattern Register i (I4DFPRi) (i = 0 to 15)



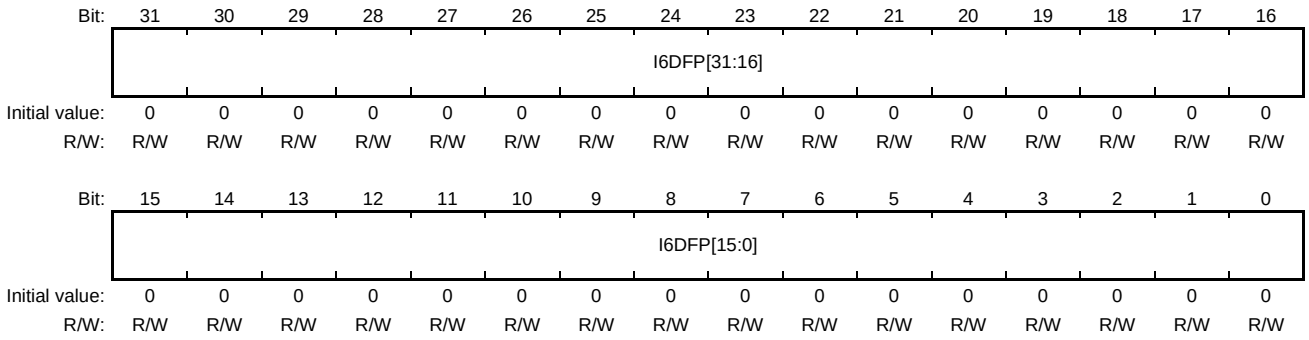

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | I4DFP[31:0] | H'0000 0000 | R/W | IP v4 Destination Address Filter Pattern<br>These bits set the pattern of the address filter.<br>The 32-bit filter pattern is set for each queue. |

### 100.1.2.106 IPv6 Source Address Filter Pattern Register i (I6SFPRi) (i = 0 to 63)



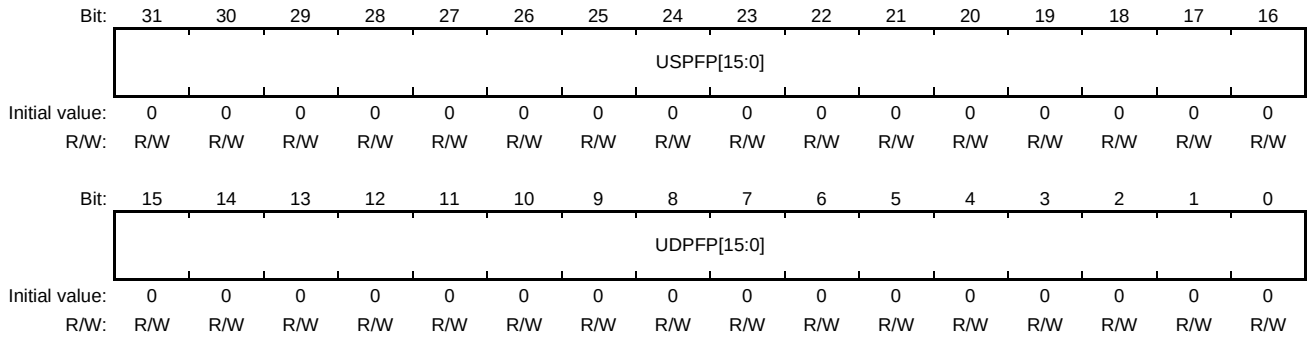

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | I6SFP[31:0] | H'0000 0000 | R/W | IPv6 Source Address Filter Pattern<br>These bits set the pattern of the address filter.<br>The 128-bit filter pattern is set for each queue. |

### 100.1.2.107 IPv6 Destination Address Filter Pattern Register i (I6DFPRi) (i = 0 to 63)



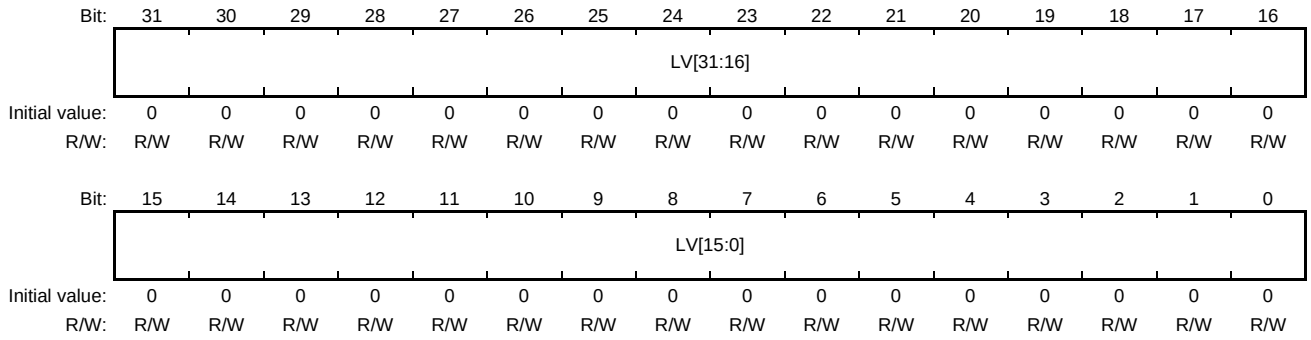

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | I6DFP[31:0] | H'0000 0000 | R/W | IPv6 Destination Address Filter Pattern<br>These bits set the pattern of the address filter.<br>The 128-bit filter pattern is set for each queue. |

### 100.1.2.108 UDP Port Number Filter Pattern Register i (UPFPRi) (i = 0 to 15)



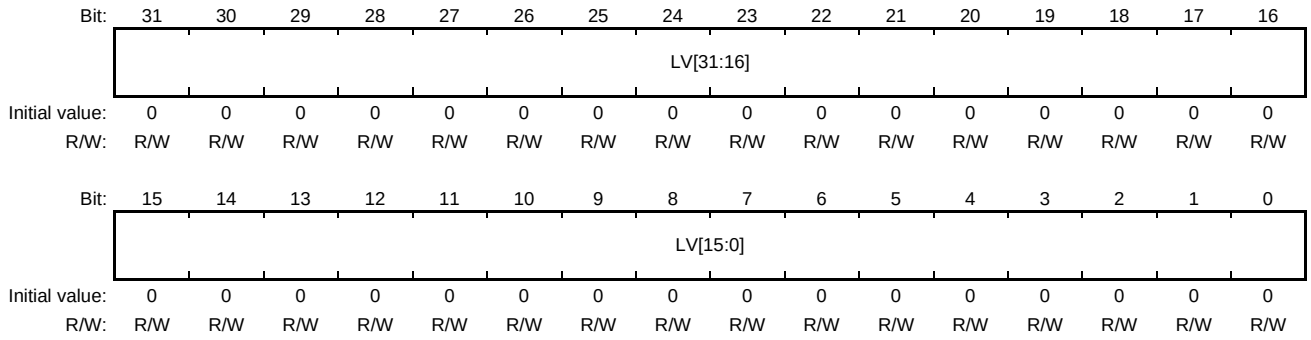

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | USPFP[15:0] | H'0000 | R/W | UDP Source Port Number Filter Pattern<br>These bits set the pattern of the port filter.<br>The 16-bit filter pattern is set for each queue. |
| 15 to 0 | UDPFP[15:0] | H'0000 | R/W | UDP Destination Port Number Filter Pattern<br>These bits set the pattern of the port filter.<br>The 16-bit filter pattern is set for each queue. |

### 100.1.2.109 Separation Filter Value Register 2 (SFV2)



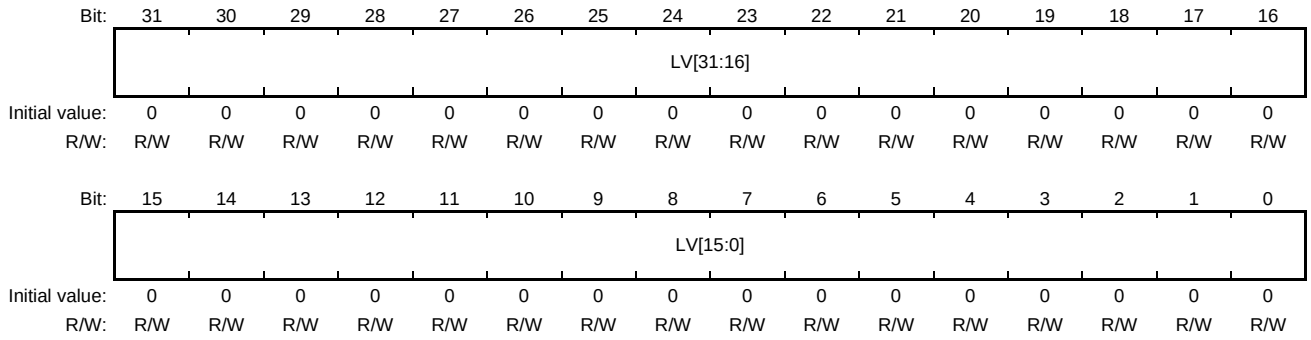

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LV[31:0] | H'0000 0000 | R/W | Separation Filter Value<br>These bits set the Load value of the separation filter. |

### 100.1.2.110 IPv4 Source Address Filter Value Register (I4SFVR)



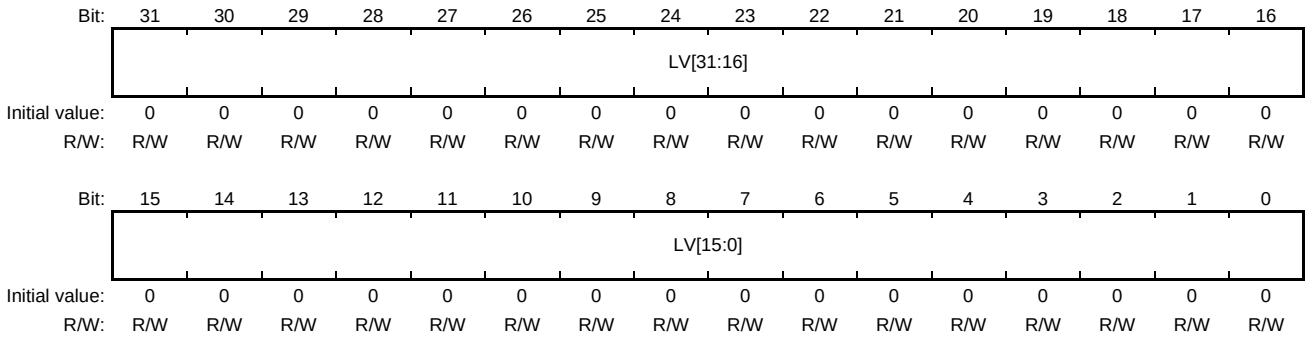

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LV[31:0] | H'0000 0000 | R/W | Separation Filter Value<br>These bits set the Load value of the separation filter. |

### 100.1.2.111 IPv4 Destination Address Filter Value Register (I4DFVR)



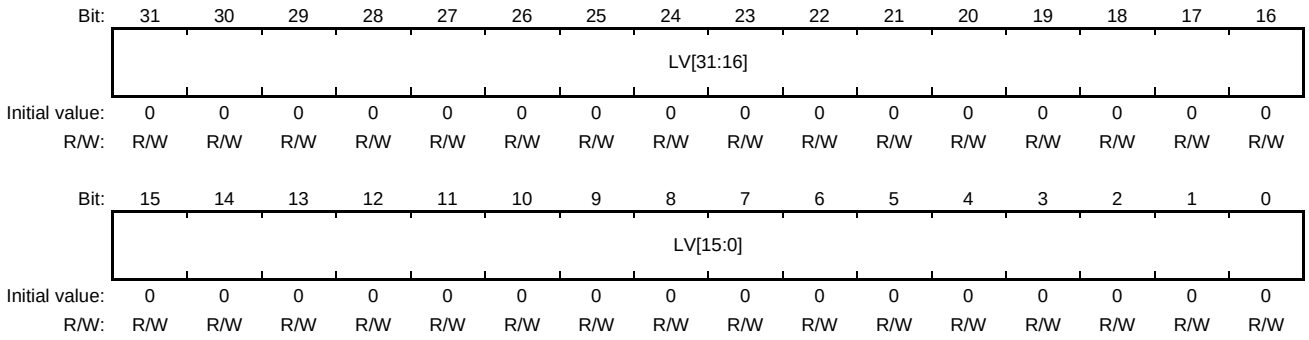

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LV[31:0] | H'0000 0000 | R/W | Separation Filter Value<br>These bits set the Load value of the separation filter. |

### 100.1.2.112 IPv6 Source Address Filter Value Register i (I6SFVRi) (i = 0 to 3)



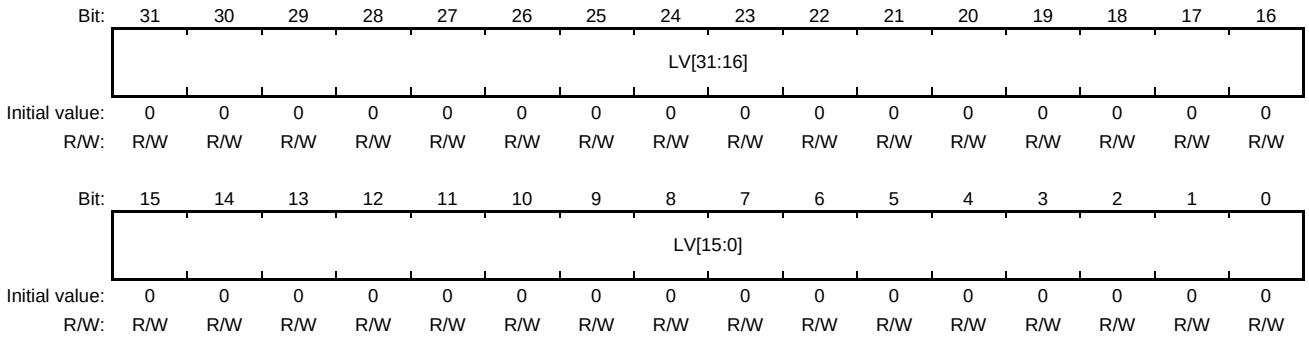

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LV[31:0] | H'0000 0000 | R/W | Separation Filter Value<br>These bits set the Load value of the separation filter. |

### 100.1.2.113 IPv6 Destination Address Filter Value Register i (I6DFVRi) (i = 0 to 3)



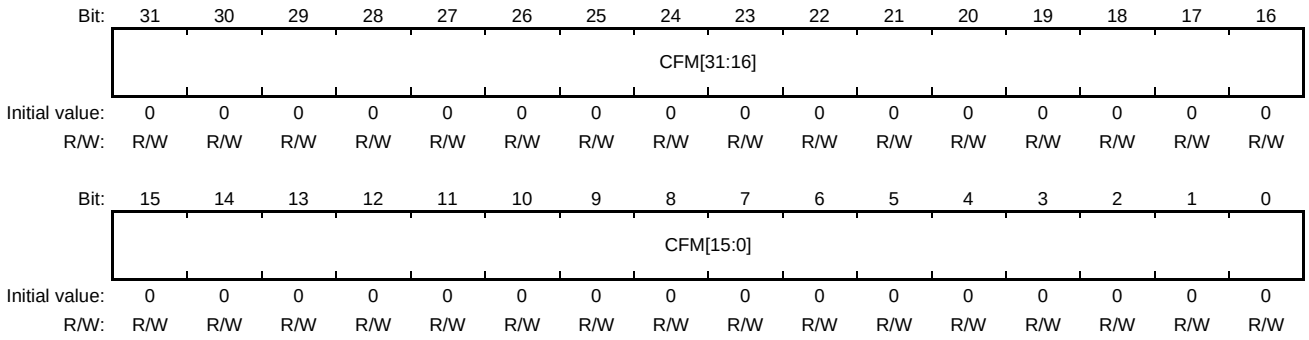

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LV[31:0] | H'0000 0000 | R/W | Separation Filter Value<br>These bits set the Load value of the separation filter. |

### 100.1.2.114 UDP Port Number Filter Value Register (UPFVR)



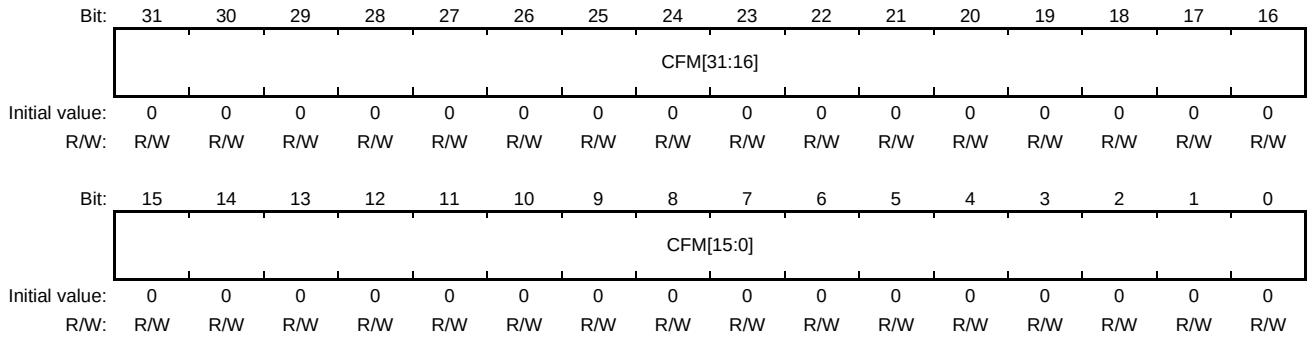

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LV[31:0] | H'0000 0000 | R/W | Separation Filter Value<br>These bits set the Load value of the separation filter. |

### 100.1.2.115 IPv4 Source Address Filter Mask Register (I4SFMR)



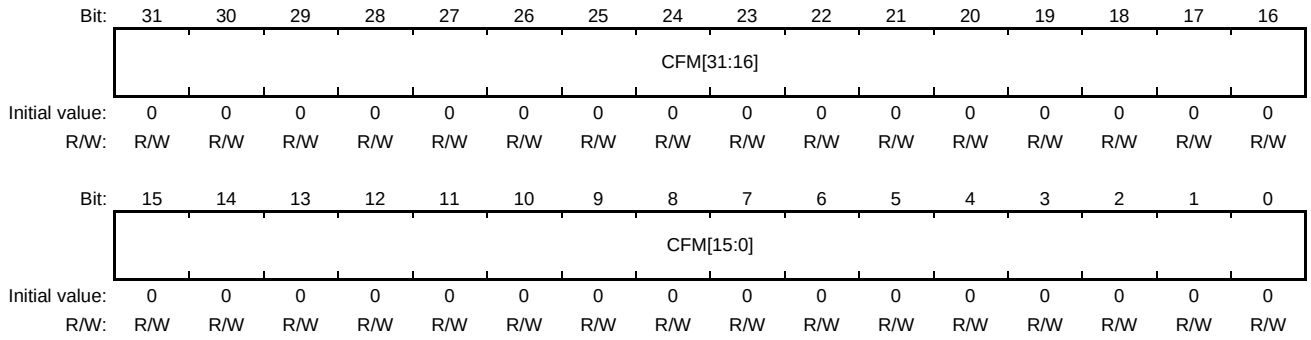

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CFM[31:0] | H'0000 0000 | R/W | Separation Filter Mask<br>These bits set the mask value for the separation filter. |

### 100.1.2.116 IPv4 Destination Address Filter Mask Register (I4DFMR)



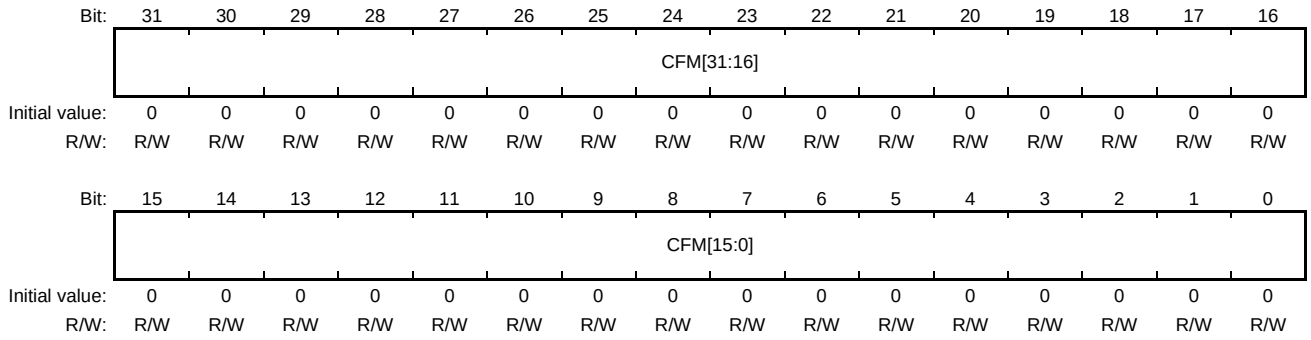

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CFM[31:0] | H'0000 0000 | R/W | Separation Filter Mask<br>These bits set the mask value for the separation filter. |

### 100.1.2.117 IPv6 Source Address Filter Mask Register i (I6SFMRi) (i = 0 or 3)



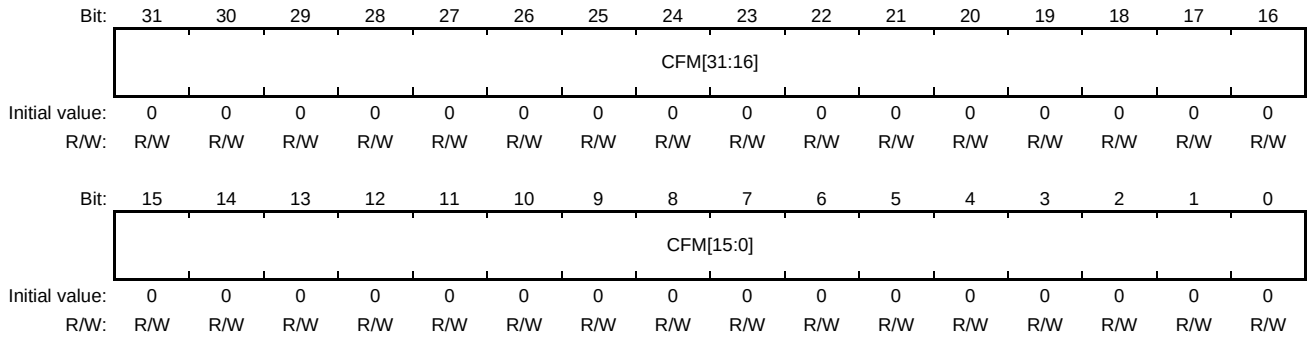

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CFM[31:0] | H'0000 0000 | R/W | Separation Filter Mask<br>These bits set the mask value for the separation filter. |

### 100.1.2.118 IPv6 Destination Address Filter Mask Register i (I6DFMRi) (i = 0 or 3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CFM[31:0] | H'0000 0000 | R/W | Separation Filter Mask<br>These bits set the mask value for the separation filter. |

### 100.1.2.119 UDP Port Number Filter Mask Register (UPFMR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CFM[31:0] | H'0000 0000 | R/W | Separation Filter Mask<br>These bits set the mask value for the separation filter. |

### 100.1.2.120 Separation Filter Load Register 2 (SFL2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | LC[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 to 0 | LC[4:0] | H'1F | R/W | Load Command<br>These bits control update of separation filter configuration in OPERATION mode.<br>By writing 30 to these bits CPU triggers an update of First Byte Position (**SFO2.FBP2**) value.<br>The CPU can only write to these bits if **CSR.OPS** is OPERATION.<br>The CPU can only write to these bits if **SFL2.LC** is 31.<br>[Changing condition]<br>These bits are set to 31 when leaving OPERATION mode.<br>These bits are set to 31 when the requested update has been processed.<br>0-29: Invalid<br>30: Load of SFO2.FBP2<br>31: No pending load request |

### 100.1.2.121 IPv4 Source Address Filter Load Register (I4SFLR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | LC[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 to 0 | LC[4:0] | H'1F | R/W | Load Command<br>These bits control update of separation filter configuration in OPERATION mode.<br>By writing 30 to these bits CPU triggers an update of First Byte Position (I4**SFO.FBP**) value.<br>By writing 29 to these bits CPU triggers an update of Common Filter Mask **(I4SFMRi.CFM**) value.<br>By writing number between 0 and 15 to these bits CPU triggers an update of Filter Pattern s (I4**SFPRi.I4SFPs**) value. The number written to **I4SFLR.LC** defines the index s of the updated Filter Pattern.<br>The CPU can only write to these bits if **CSR.OPS** is OPERATION.<br>The CPU can only write to these bits if **I4SFLR.LC** is 31.<br>[Changing condition]<br>These bits are set to 31 when leaving OPERATION mode.<br>These bits are set to 31 when the requested update has been processed.<br>0-15: Load of I4SFPRi.I4SFPs with s = I4SFLR.LC<br>16-28: Invalid<br>29: Load of I4SFMRi.CFM<br>30: Load of I4SFO.FBP<br>31: No pending load request |

### 100.1.2.122 IPv4 Destination Address Filter Load Register (I4DFLR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | LC[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 to 0 | LC[4:0] | H'1F | R/W | Load Command<br>These bits control update of separation filter configuration in OPERATION mode.<br>By writing 30 to these bits CPU triggers an update of First Byte Position (**I4DFO.FBP**) value.<br>By writing 29 to these bits CPU triggers an update of Common Filter Mask (**I4DFMRi.CFM**) value.<br>By writing number between 0 and 15 to these bits CPU triggers an update of Filter Pattern s (**I4DFPRi.I4DFPs**) value. The number written to **I4DFLR.LC** defines the index s of the updated Filter Pattern.<br>The CPU can only write to these bits if **CSR.OPS** is OPERATION.<br>The CPU can only write to these bits if **I4DFLR.LC** is 31.<br>[Changing condition]<br>These bits are set to 31 when leaving OPERATION mode.<br>These bits are set to 31 when the requested update has been processed.<br>0-15: Load of I4DFPRi.I4DFPs with s = I4DFLR.LC<br>16-28: Invalid<br>29: Load of I4DFMRi.CFM<br>30: Load of I4DFO.FBP<br>31: No pending load request |

### 100.1.2.123 IPv6 Source Address Filter Load Register (I6SFLR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | LC[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 to 0 | LC[4:0] | H'1F | R/W | Load Command<br>These bits control update of separation filter configuration in OPERATION mode.<br>By writing 30 to these bits CPU triggers an update of First Byte Position (**I6SFO.FBP**) value.<br>By writing 29 to these bits CPU triggers an update of Common Filter Mask (**I6SFMRi.CFM**) value.<br>By writing number between 0 and 15 to these bits CPU triggers an update of Filter Pattern s (**I6SFPRi.I6SFPs**) value. The number written to **I6SFLR.LC** defines the index s of the updated Filter Pattern.<br>The CPU can only write to these bits if **CSR.OPS** is OPERATION.<br>The CPU can only write to these bits if **I6SFLR.LC** is 31.<br>[Changing condition]<br>These bits are set to 31 when leaving OPERATION mode.<br>These bits are set to 31 when the requested update has been processed.<br>0-15: Load of I6SFPRi.FPs with s = I6SFLR.LC<br>16-28: Invalid<br>29: Load of I6SFMRi.CFM<br>30: Load of I6SFO.FBP<br>31: No pending load request |

### 100.1.2.124 IPv6 Destination Address Filter Load Register (I6DFLR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | LC[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 to 0 | LC[4:0] | H'1F | R/W | Load Command<br>These bits control update of separation filter configuration in OPERATION mode.<br>By writing 30 to these bits CPU triggers an update of First Byte Position (**I6DFO.FBP**) value.<br>By writing 29 to these bits CPU triggers an update of Common Filter Mask (**I6DFMRi.CFM**) value.<br>By writing number between 0 and 15 to these bits CPU triggers an update of Filter Pattern s (**I6DFPRi.I6DFPs**) value. The number written to **I6DFLR.LC** defines the index s of the updated Filter Pattern.<br>The CPU can only write to these bits if **CSR.OPS** is OPERATION.<br>The CPU can only write to these bits if **I6DFLR.LC** is 31.<br>[Changing condition]<br>These bits are set to 31 when leaving OPERATION mode.<br>These bits are set to 31 when the requested update has been processed.<br>0-15: Load of I6DFPRi.FPs with s = I6DFLR.LC<br>16-28: Invalid<br>29: Load of I6DFMRi.CFM<br>30: Load of I6DFO.FBP<br>31: No pending load request |

### 100.1.2.125 UDP Port Number Filter Load Register (UPFLR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | LC[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | — | All 0 | R/W | Reserved<br>These bits are read as 0. The write value should be 0. |
| 4 to 0 | LC[4:0] | H'1F | R/W | Load Command<br>These bits control update of separation filter configuration in OPERATION mode.<br>By writing 30 to these bits CPU triggers an update of First Byte Position (**UPFO.FBP**) value.<br>By writing 29 to these bits CPU triggers an update of Common Filter Mask (**UPFMR.CFM**) value.<br>By writing number between 0 and 15 to these bits CPU triggers an update of Filter Pattern s (**UPFPRi.USPFP and UPFPRi.UDPFP**) value. The number written to **UPFLR.LC** defines the index s of the updated Filter Pattern.<br>The CPU can only write to these bits if **CSR.OPS** is OPERATION.<br>The CPU can only write to these bits if **UPFLR.LC** is 31.<br>[Changing condition]<br>These bits are set to 31 when leaving OPERATION mode.<br>These bits are set to 31 when the requested update has been processed.<br>0-15: Load of UPFPRi.USPFP and UPFPRi.UDPFP with s = UPFLR.LC<br>16-28: Invalid<br>29: Load of UPFMR.CFM<br>30: Load of UPFO.FBP<br>31: No pending load request |

### 100.1.3 Operation

The EthernetAVB-IF consists of the following functional units:

- DMA transfer controller (AVB-DMAC): Handles DMA transfer between the data storage areas for reception and transmission in the URAM and the reception and transmission FIFO buffers
- MAC controller (E-MAC): Handles transfer between the reception and transmission FIFO buffers and the RGMII

Using its direct memory access (DMA) function, the AVB-DMAC handles DMA transfer of frame data between the destinations for storing Ethernet frame data for transmission and reception in the URAM and the FIFO buffers for reception and transmission. Data cannot be directly read from or written to the FIFO buffers.

To handle DMA transfer, the AVB-DMAC requires information that includes the addresses for storage of data for transmission and received data. These data are referred to as descriptors. The AVB-DMAC reads data for transmission from the storage area for data to be transmitted according to the information in descriptors and writes received data to the storage area for received data accompanied by information in descriptors. The descriptors are placed in the URAM. Arranging multiple descriptors in descriptor lists allows the continuous reception or transmission of multiple Ethernet frames.

The E-MAC supports a RGMII, which provides an interface format for the externally connected PHY-LSI. The E-MAC constructs Ethernet frames from data written to the transmission FIFO and transmits these frames to the RGMII. It also performs CRC checking of Ethernet frames received from the RGMII and writes the frames to the reception FIFO.

### 100.1.3.1 AVB-DMAC Operating Modes

Figure 100.1.3.1.1 illustrates the operating modes of the AVB-DMAC.

Transitions of AVB-DMAC operating mode are under the control of the items listed below.

- CPU operating mode (hardware reset and power-down mode)
- Configuration of the operating mode configuration bits (CCC.OPC) in the AVB-DMAC mode register

The current operating mode can be confirmed by reading the operating mode status bits in the AVB-DMAC status register (CSR.OPS).

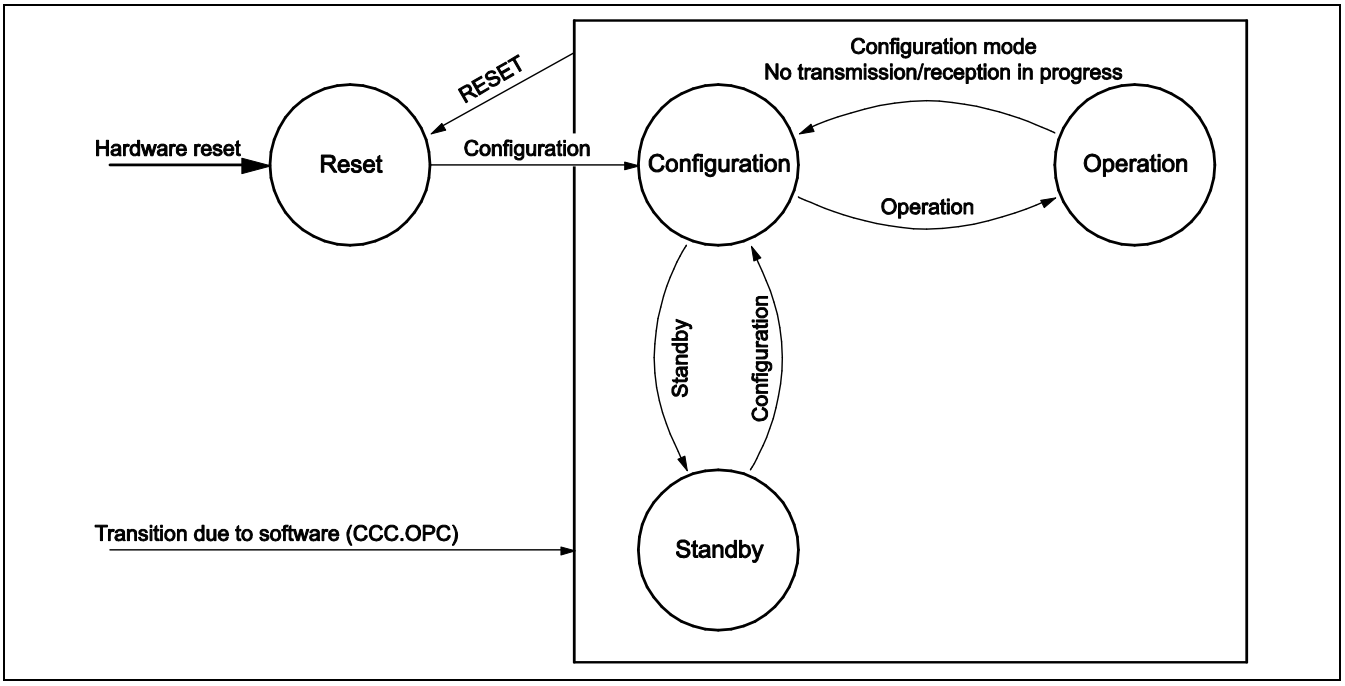



**Figure 100.1.3.1.1 Operating Mode of AVB-DMAC**

#### (1) Operating Modes

**(a) Reset mode**

After a hardware reset, the AVB-DMAC enters reset mode.
In reset mode, only the AVB-DMAC operating mode control function is controllable and other functions are all stopped. This mode is designed for reduced power when the Ethernet function is not necessary.

**(b) Configuration mode**

In configuration mode, various settings for the AVB-DMAC can be made.
The operation of most functions is stopped and all status registers are initialized to their reset values. The E-MAC functions in this mode.
By **CCC.GAC** it is possible to enable gPTP support already in CONFIG mode.

**(c) Operation mode**

In operation mode, all functions of the AVB-DMAC can operate. Ethernet communications can only proceed in this mode.

**(d) Standby mode**

In standby mode, the E-MAC can only be used to control the operating mode. Other functions cannot be used.
When **CCC.GAC** is 1b CPU should not enter STANDBY mode.

### (2) How to Set the Operating Mode

Set the operating mode configuration bits in the AVB-DMAC mode register (CCC.OPC) to select the operating mode. Furthermore, the current operating mode can be checked by reading the operating mode status bits in the AVB-DMAC status register (CSR.OPS).

Transitions other than from operation mode to configuration mode are made after the value is written to the operating mode configuration bits (CCC.OPC) (Figure 100.1.3.1.2).

For transitions from operation mode to configuration mode, follow the procedure in Figure 100.1.3.1.3 because any transmission and reception in progress will be executed before the transition to configuration mode.

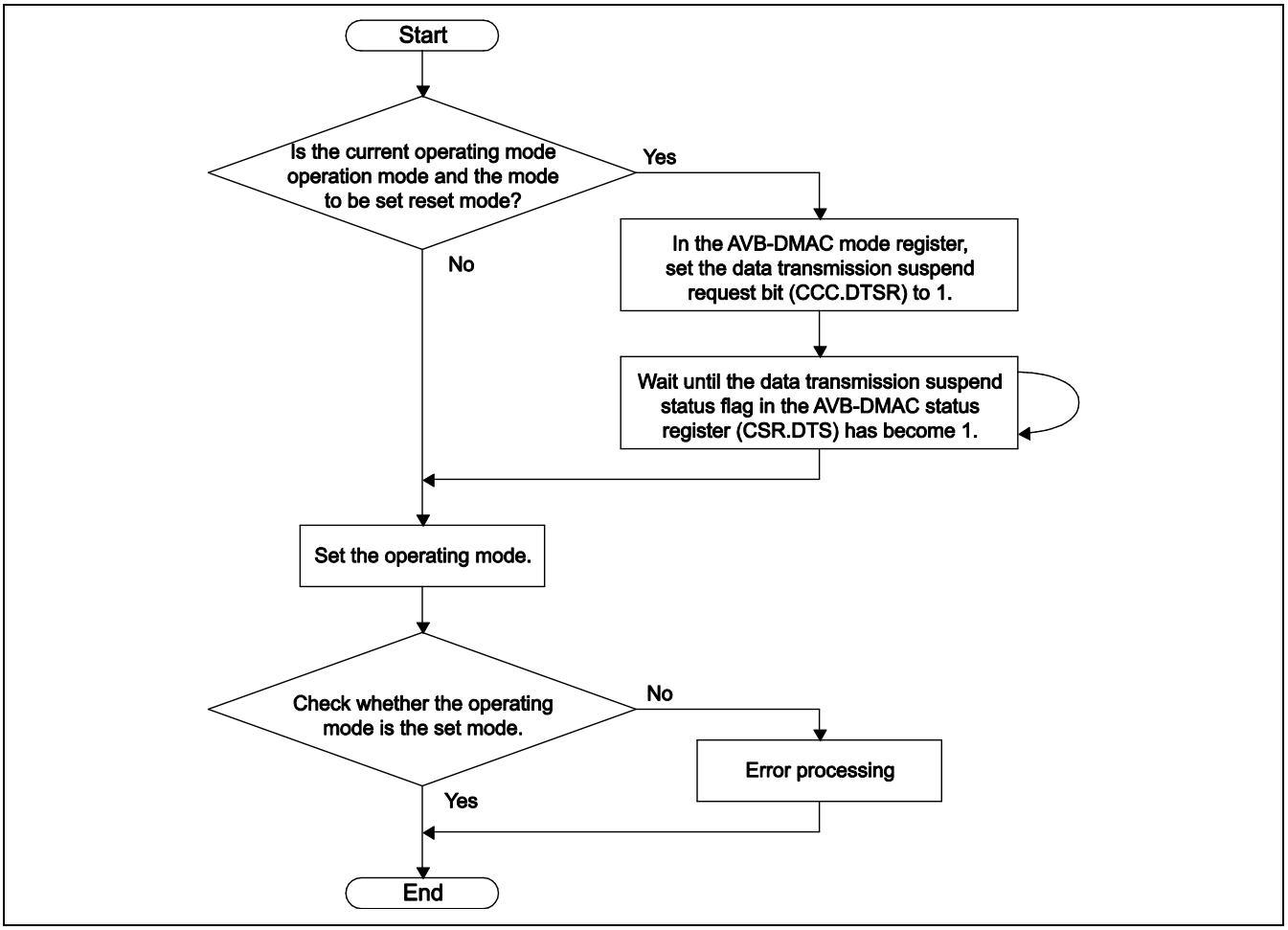



**Figure 100.1.3.1.2 Flow for Transitions of Operating Mode (Other than from Operation Mode to Configuration Mode)**

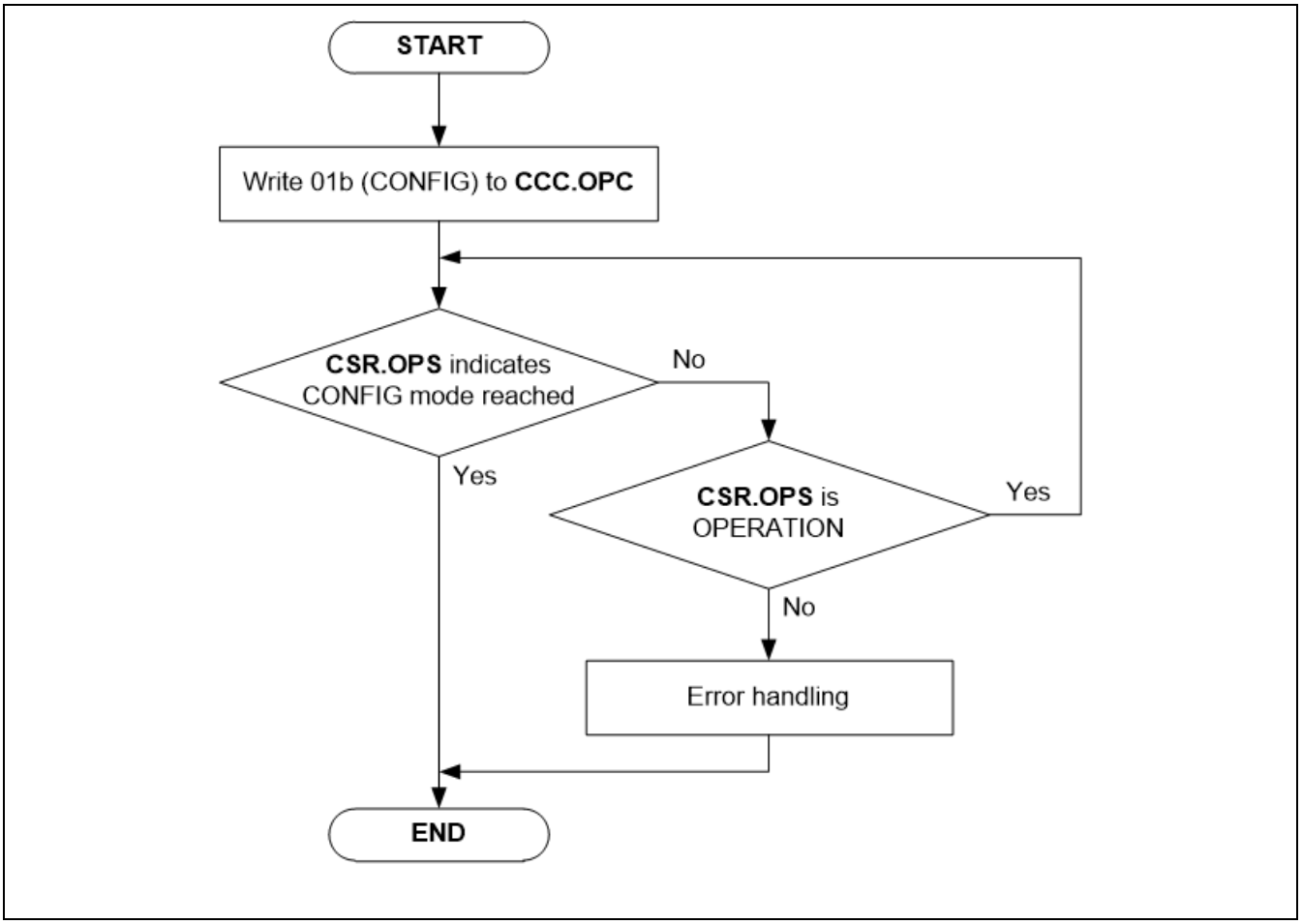



**Figure 100.1.3.1.3 Flow for Transitions of Operating Mode (from Operation Mode to Configuration Mode)**

In the transition from operation mode to configuration mode, the AVB-DMAC executes the following operations before the transition is completed. Read the operating mode status bits in the AVB-DMAC status register (CSR.OPS) to check that the transition to configuration mode has been completed.

- If the transfer of a frame between the reception FIFO and URAM is in progress, this is completed (other received frames remaining in the FIFO and any frames that are subsequently received by the E-MAC are discarded).
- If the transfer of a frame is in progress between the transmission FIFO and URAM, this is completed (frames for transmission remaining in the URAM will not be transmitted).
- All frames for transmission in the transmission FIFO are transferred to the E-MAC.

The CPU should not disable URAM transfers by CCC.DTSR nor prevent transmission by MAC when requesting transition from OPERATION to CONFIG, else EthernetAVB-IF cannot perform requested mode change.

Notes: When the operating mode shifts to configuration mode, all status registers are cleared.
When need status register information, the reception and transmission path can be individually disabled and can check the status before leaving operating mode with follow.
We recommend following the procedure below in the case of this transition.

1. Disable reception.
2. Since reception actually stopping after being disabled requires time, wait for an interval equivalent to that for reception of a maximum length packet.
3. Stop the software task that is generating data for transmission.
4. Wait until the receive process status bit (CSR.RPO) and the transmit process status bits (CSR.TPO0 to 3) in the AVB-DMAC status register are set to 0.

5. Capture all of the required status information.
6. Set the operating mode configuration bits in the AVB-DMAC mode register (CCC.OPC) to initiate the transition to configuration mode.

### (3) Leave RESET mode with additional configuration

When writing 01b (CONFIG) to **CCC.OPC** while EthernetAVB-IF is in RESET mode, it is possible to configure availability of gPTP support within CONFIG mode

To enable gPTP support in CONFIG mode use a 32 bit write access when setting **CCC.OPC** to 01b. This write should also set **CCC.GAC** to 1b and **CCC.CSEL** to intended configuration.

### (4) Operating Mode Transitions Due to Hardware

The following hardware factors can also initiate transitions of the AVB-DMAC operating mode.

#### (a) Hardware reset

Resetting of the LSI chip leads to resetting of the entire EthernetAVB-IF module. The operating mode shifts to reset mode.

#### (b) Transition during power-off by module standby

This transition is triggered by module standby of CPG during the power-off sequence.

The AVB-DMAC completes the bus master access in progress, and then shifts to reset mode. At this time, the operating mode configuration bits in the AVB-DMAC mode register (CCC.OPC) are set to B'00.

### 100.1.3.2 Common Control for Transmission and Reception

#### (1) Initialization Procedure

Figure 100.1.3.2.1 shows the overall initialization procedure in outline.

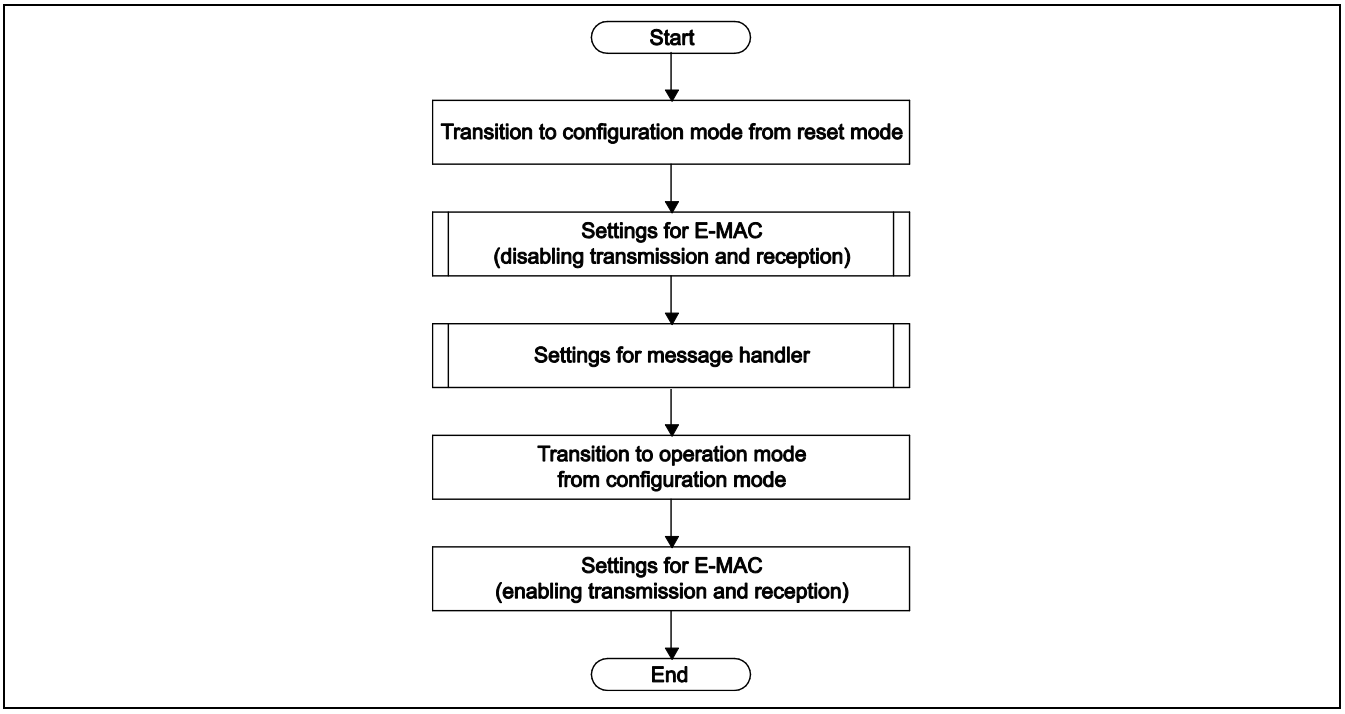



**Figure 100.1.3.2.1 Outline of the Initialization Procedure**

**(a) Initializing the Receiver Section**

Before starting reception, follow the procedure below.

Keep the operating mode to configuration mode, and do not enable reception until the settings for the AVB-DMAC are completed. However, there is a limitation that if it can be determined that no data is received, it is no problem to change the setting.

- Set the operating mode to configuration mode.
- Set AVB filtering for network control frames and AVB stream frames to suit the specifications of the product the chip will be used in.
- Create a descriptor chain for each queue to be used.
- Set the base address for table address in the descriptor base address table register (DBAT).
- Specify the maximum frame length with the receive frame length upper limit register (RFLR).
- Specify whether padding is to be used with the receive padding configuration register (RPC).
- Set the unread frame counter for each queue with unread frame counter registers (UFCVs) 0 to 4.

**(b) Initializing the Transmitter Section**

Figure 100.1.3.2.2 illustrates initialization of the transmitter section.

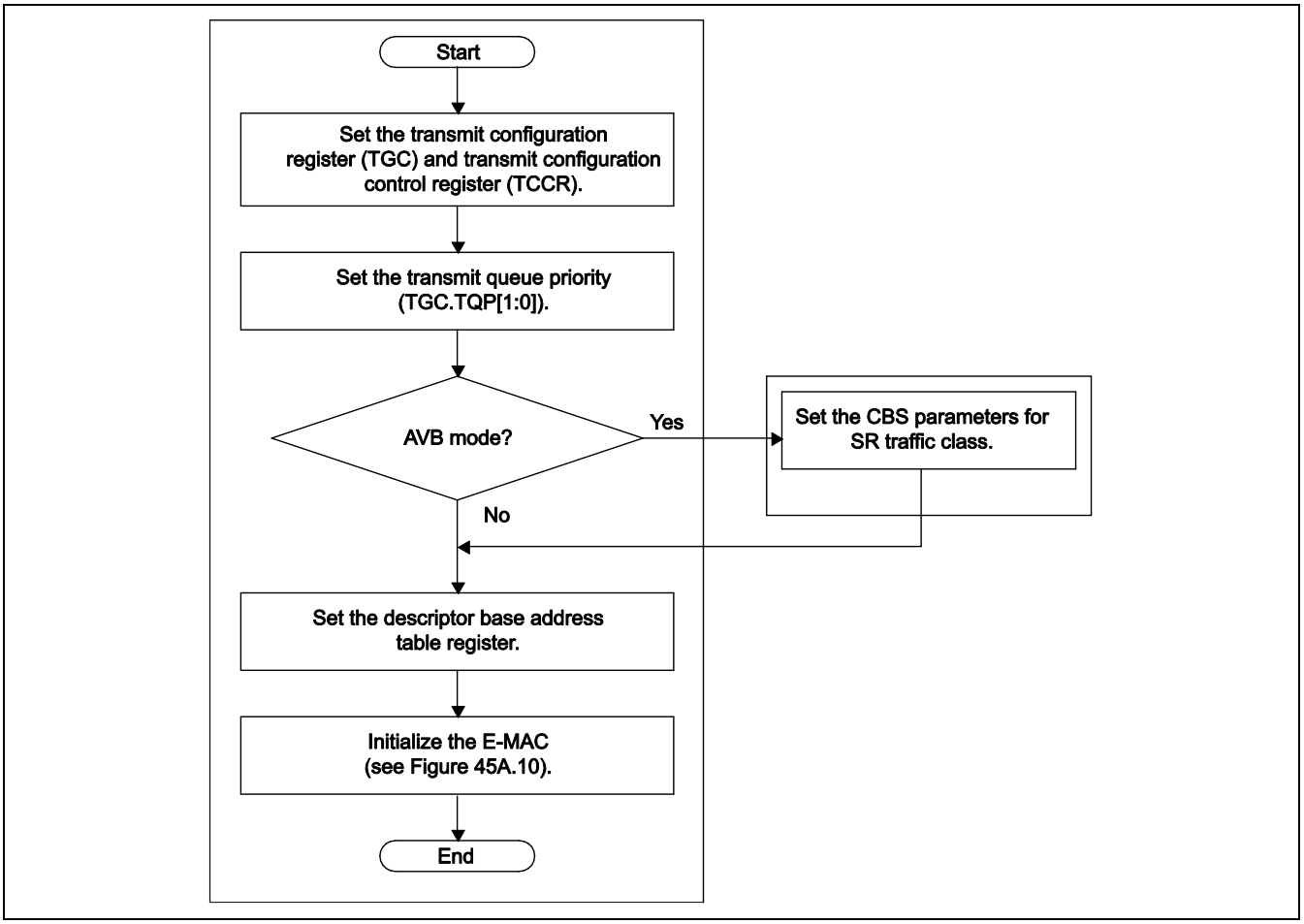



**Figure 100.1.3.2.2 Procedure for Initializing the Transmitter Section**

**(c) Initializing the E-MAC Section**

Figure 100.1.3.2.3 illustrates initialization of the E-MAC section.

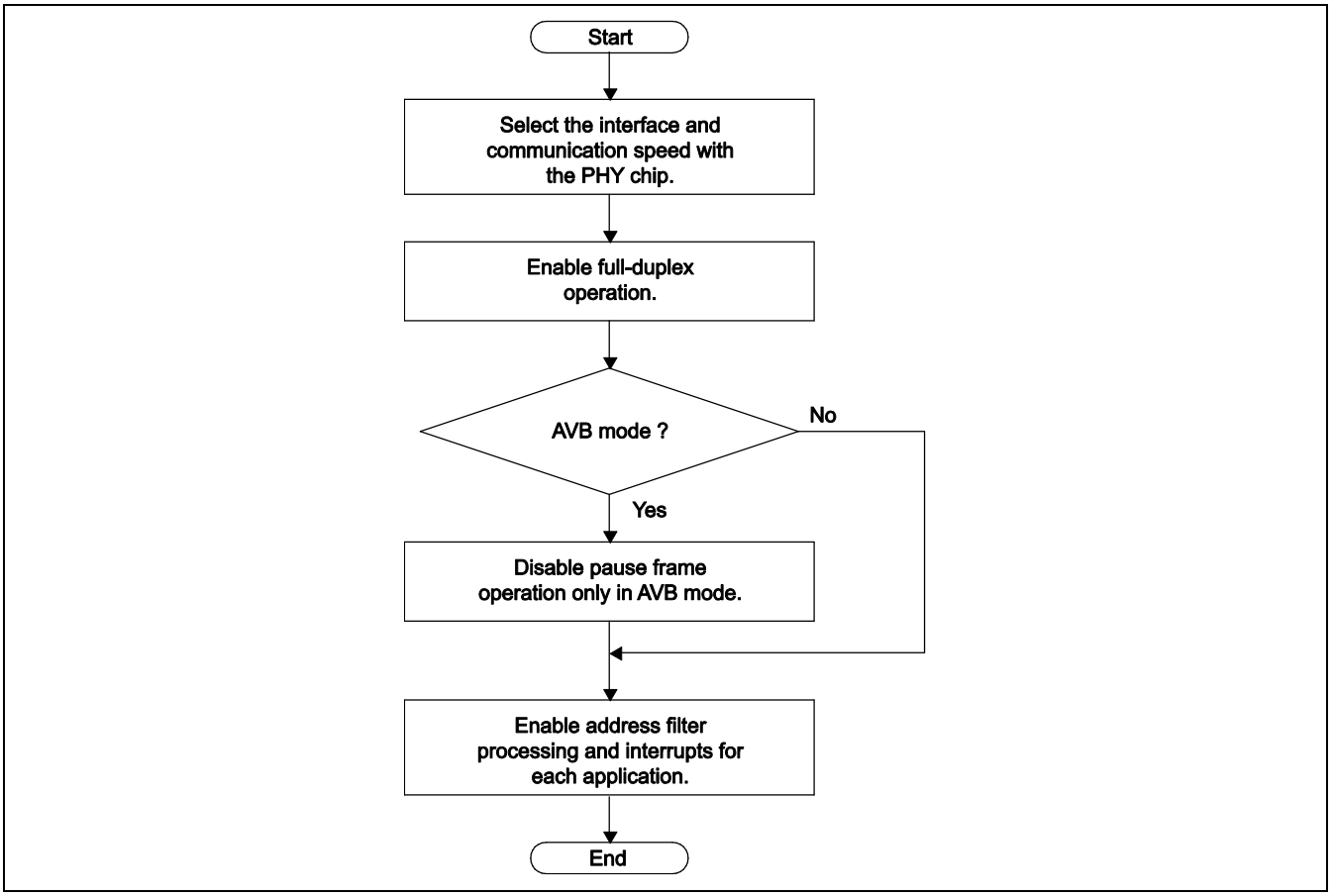



**Figure 100.1.3.2.3 Procedure for Initializing the E-MAC Section**

**(d) Initialization of the Application Unit**

Figure 100.1.3.2.4 illustrates initialization of the application unit.

For a description of how to set up the descriptors and the CBS traffic shaping parameters, see section 100.1.3.3 Descriptors, and section 100.1.3.6 CBS (Credit-Based Shaping).

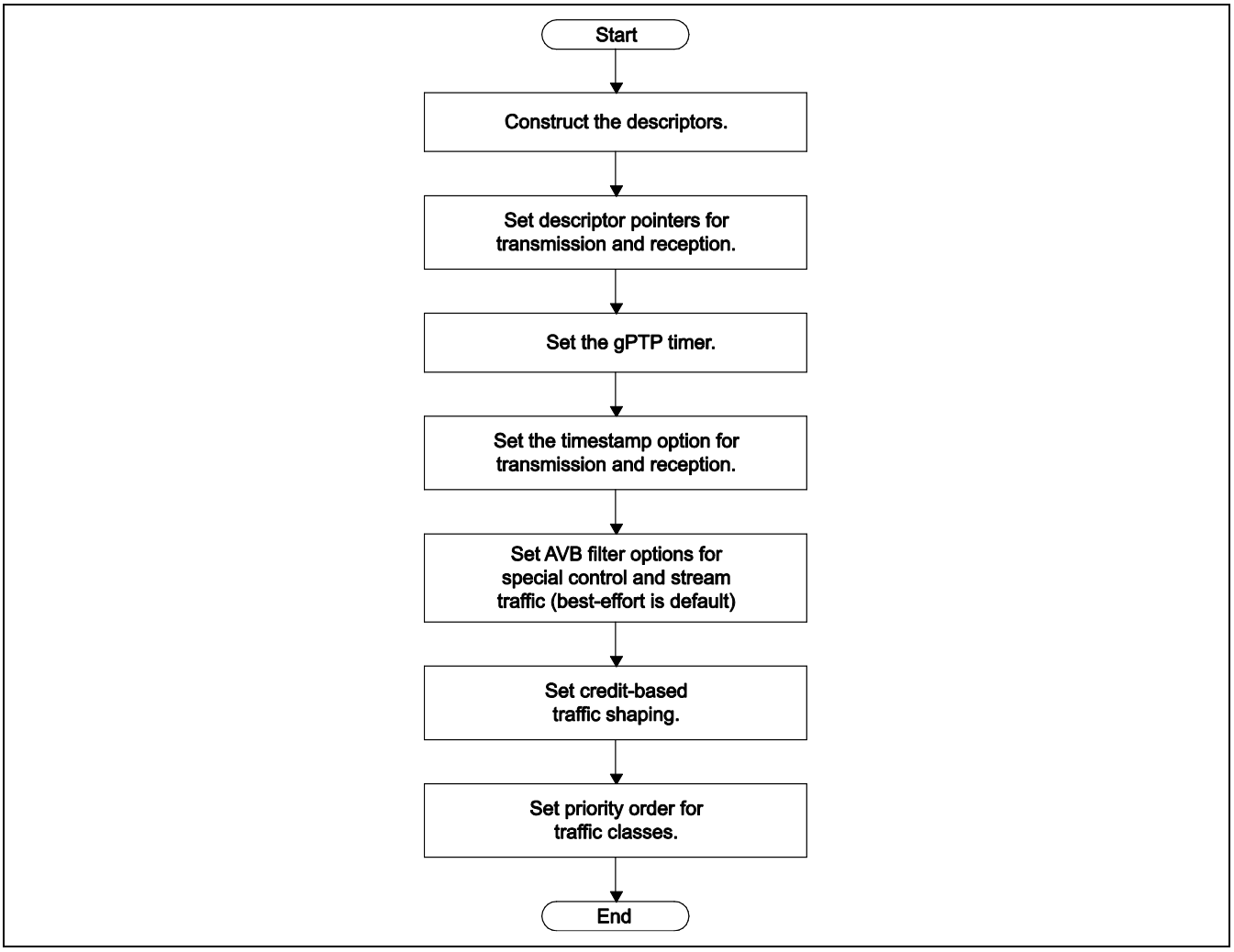



**Figure 100.1.3.2.4 Initializing the Sections for Use by the Application**

**(e) Relationship between Transmission Queue Numbers and Traffic Classes**

In fetching, the relationships between the transmission queues and traffic classes are fixed, so the priority specified by the transmit queue priority bits in the transmit configuration register (TGC.TQP) has no effect.

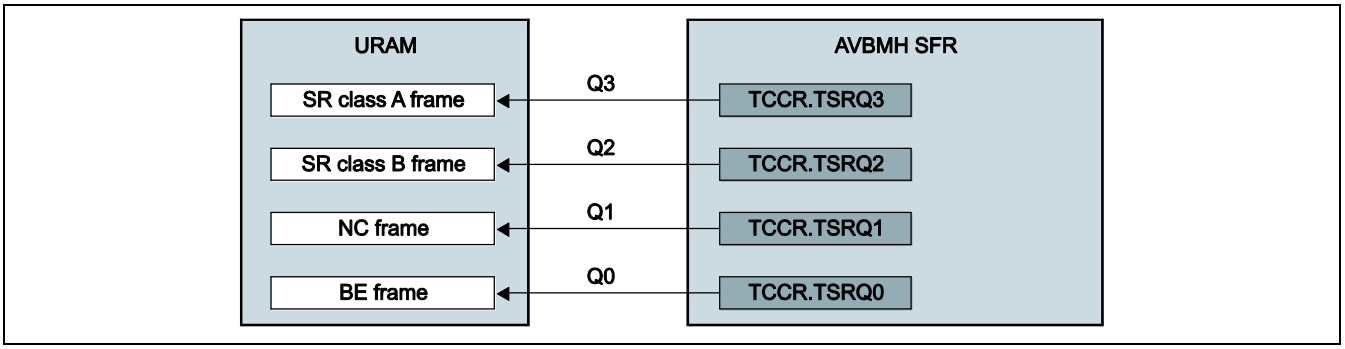



**Figure 100.1.3.2.5 Class Associations of Queues for the Scheduler**

In fetching, the credit values for stream classes A and B are not taken into account. Behavior depends on the setting of the transfer FIFO size configuration bits in the transfer configuration registers (TGC.TBDt) and on the frame size that can be fetched to the transmission FIFO.

When the transmit queue priority bit and enable credit based shaping bit in the transmit configuration register {TGC.TQP, TGC.ECBS} are B'00 or B'01, the priority order is Q3 → Q 2 → Q1 → Q0.

When the transmit queue priority bit and enable credit based shaping bit in the transmit configuration register {TGC.TQP, TGC.ECBS} are B'11, the priority order is Q1 → Q3 → Q2 → Q0.

### (2) Checking Integrity

The AVB-DMAC is capable of detecting and identifying errors produced in the processing of Ethernet frames and in the transfer of frame data for transmission and reception.

#### (a) Concept of Integrity Checking in Reception

The aim of integrity checking in reception is preventing the storage of error frames in the URAM. If an error frame is stored in the URAM, software can get information to identify the frame as an error frame.

Note: If a special descriptor chain is to be used for separation of received headers from the associated data, an error that breaks the sequence may lead to storage space for synchronization running out. In such cases, software interaction or re-synchronization via the EOS descriptor is required.

#### (b) Concept of Integrity Checking in Transmission

The purpose of integrity checking in transmission is to prevent the transmission of broken frames.

Since transmission of a frame by the E-MAC can neither be stopped nor disabled once it has started, this check involves intensive monitoring for problems that can arise during fetching.

#### (c) Items for Monitoring in Both Reception and Transmission

**Errors in access to the URAM for reading of descriptors**

The same descriptor may be processed again because the current descriptor address (CDARq.CDA) was not changed. If this problem occurs in a divided frame, the sequence may be broken.

**In reception**

- The received frame will be lost.
- The same problem will occur for the next frame of data received for the same queue.

**In transmission**

- The transmit start request bit in the transmit configuration control register (TCCR.TSRQt) is set to 0.
- The frame will be lost from the transmission FIFO.

Errors in access to read descriptors from the URAM are detected from the response signal of the AXI-Bus. EthernetAVB-IF flags only one access error per descriptor read.

**Illegal configuration of a descriptor by an application**

The same descriptor may be processed again because the current descriptor address (CDARq.CDA) was not changed. If this problem occurs in a divided frame, the sequence may be broken.

**In reception**

- The received frame will be lost.
- The same problem will occur for the next frame of data received for the same queue.

**In transmission**

- The transmit start request bit in the transmit configuration control register (TCCR.TSRQt) is set to 0.
- The frame will be lost from the transmission FIFO.

**Errors in access to the URAM for writing of descriptors**

As in the case where no error occurs, the current descriptor address (CDARq.CDA) and the transmit start request bit in the transmit configuration control register (TCCR.TSRQt) are updated.

As DESCR.DT was not updated, hardware and software synchronization may have been destroyed.

Errors in access to write descriptors to the URAM are detected from the response signal of the AXI-Bus.

**(d) Items for Monitoring in Reception**

**Errors in access to the URAM for writing of data or time stamps**

- As in the case where no error occurs, the current descriptor address (CDARq.CDA) is updated.
- DESCR.EI is set to indicate incorrect contents.
- This problem occurring in a divided frame may break the descriptor sequence, making the queue unusable.

Errors in access to write data or descriptors to the URAM are detected from the response signal of the AXI-Bus.

EthernetAVB-IF flags each access error individually, even they are related to same frame.

**Error of the Reception FIFO**

- Received frames are all invalidated.
- All frames stored as received frames are discarded. At this time, the number of frames and queue information cannot be captured.

If the reception FIFO RAM faults, AVB-DMAC detects the fault as error of the reception FIFO.

**Queue Synchronisation Fault of the Reception**

EthernetAVB-IF will not continue splitting frame across EOS descriptor. A queue synchronisation fault is detected when read descriptor with DESCR.DT = EOS at position where FMID, or FEND should be written

- RIS2.QFFr is set 1b
- CDARq.CDA is updated as in fault free operation.
- Neither CIARr.CIA nor LIARr.LIA is changed.
- Neither UFCVi.CVr nor RIS0.FRFr is changed.
- Received frame is lost. There is an incomplete frame (no FEND) in descriptor chain.

Note: Reading descriptor may be one or more storage elements ahead in descriptor chain.

**(e) Items for Monitoring in Transmission**

**Errors in Access for Reading Data from the URAM**

- Data that have already been fetched are discarded from the transmission FIFO.
- When an error of this type occurs during processing of an FSINGLE or FEND descriptor:
  As in the case where no error occurs, the current descriptor address (CDARq.CDA) and the transmit start bit in the transmit configuration control register (TCCR.TSRQt) are updated. Fetching resumes after the error frame.
- When an error of this type occurs during processing of an FSTART or FMID descriptor:
  The current descriptor address (CDARq.CDA) is not updated.
  The transmit start bit in the transmit configuration control register (TCCR.TSRQt) is set to 0.

Errors in access to read data from the URAM are detected from the response signal of the AXI-Bus.

**Overflow of the Transmission FIFO**

- As in the case where no error occurs, the current descriptor address (CDARq.CDA) and the transmit start bit in the transmit configuration control register (TCCR.TSRQt) are updated. Fetching resumes after the error frame.
- The frame will be discarded from the FIFO.

**Frame size error during transmission**

— As in the case where no error occurs, the current descriptor address (CDARq.CDA) and the transmit start bit in the transmit configuration control register (TCCR.TSRQt) are updated. Fetching resumes after the error frame.

A transmit frame size error is detected when the size setting in one or more (in the case of a divided frame) descriptors for frame transmission is 1966 or more bytes. Such frames are cut out and transmitted.

**Damaged Data in the Transmission FIFO**

— Fetching is not affected by damaged data.
— Since damaged data from the FIFO is only detected during frame transmission, an error frame may be transmitted.

If damaged data in the transmission FIFO is an error due to the transmission FIFO, this is detected by the AVB-DMAC.

**(f) Items for Monitoring in gPTP**

**Access error while reading AVTP timestamps from Tx-Buffer RAM**

— AVTP timestamps are invalidated.
— gPTP unit ignores the issue. It uses the value as read for AVTP comparison.

An access error when reading timestamps from Tx-Buffer RAM is flagged to EthernetAVB-IF by external RAM protection logic.

### 100.1.3.3 Descriptors

#### (1) Data Representation in URAM

The AVB-DMAC transfers data for transmission and received data to and from the application software via the URAM.

The memory in the URAM for use by the AVB-DMAC is configured with control structures referred to as descriptors and associated areas to which the frame data are allocated. Dividing the memory into a control area and data area allows the flexible allocation of frame data to the URAM. This enables sharing of the areas to which frame data are allocated and the use of non-contiguous areas. Frame data can be copied without using the CPU. Arbitration that ensures hardware and software access to the memory area is also available without access to registers of the AVB-DMAC.

Figure 100.1.3.3.1 shows an example of the memory maps for descriptors and the descriptor data area in the URAM.

A descriptor consists of its type (DESCR.DT), which controls the descriptor functions, a descriptor pointer (DESCR.DPTR) indicating the start address for storage of the frame data in the descriptor area, and the data size field (DESCR.DS), indicating the amount of frame data. Post-processing interrupt generation can be set up for each descriptor. Enabling and disabling of the interrupt is controlled by the descriptor interrupt enable bits (DESCR.DIE).

The descriptor may also hold information related to content. This information does not affect general descriptor functions. It provides information other than the frame data proper, such as on the state of reception.

For details, see section 100.1.3.4(2) Setting Up Reception Descriptors, and section 100.1.3.5(2) Setting Up Transmission Descriptors.

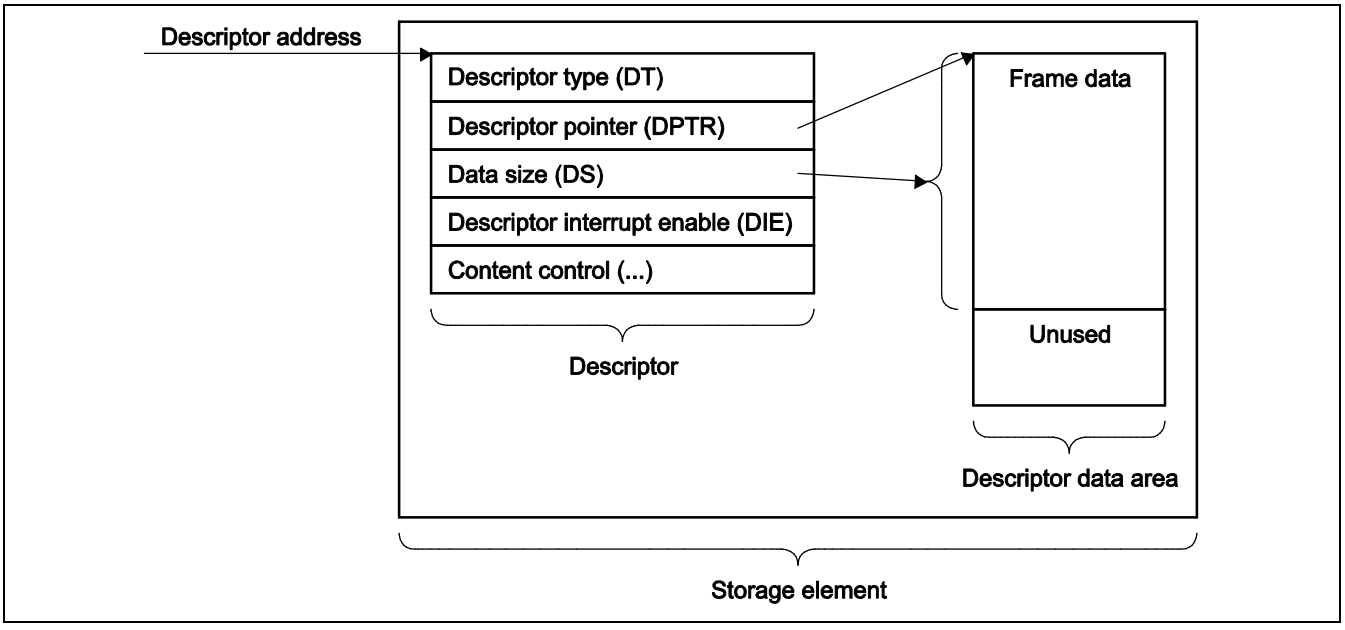



**Figure 100.1.3.3.1 Outline of Storage Element Used for Receive and Transmit Queues**

The descriptor must be aligned with a 8-bit boundary in the URAM.

Descriptors are generally configured of 64 bits, but are configured of 160 bits when reception and storage of gPTP time stamps is enabled.

The amount of data in the frame is defined by the data size bits (DESCR.DS). In reception, these bits indicate the upper limit on the size of frames to be received. If the data size is not aligned with a 32-bit boundary, the bytes to the next 32-bit boundary in the data area will be an unused area.

### (2) Using Descriptor Chains in Queues

Transmission and reception descriptors in the URAM are grouped into queues. Each queue handles frames so that they are transmitted in order of priority and received separately. A queue is capable of controlling one or more frames. Accordingly, multiple descriptors can be assigned to one queue. A combination of multiple descriptors is referred to as a descriptor chain.

For a descriptor chain, the three general descriptor types listed below are defined. For details on these descriptor types, see section 100.1.3.3(6) Descriptor Type.

- Descriptors that define frame data
- Descriptors that control the descriptor chain itself (e.g. LINK, EOS).
- Descriptors that arbitrate access by hardware or software

Figure 100.1.3.3.2 shows the two basic topologies for descriptor chains. In the simplified examples in the figure, all descriptors allocated to the chain are stored in the array.

- For a linear descriptor chain, the last descriptor in the array is a control descriptor indicating the end of the descriptors (e.g. EEMPTY).
- For a cyclic descriptor chain, the last descriptor in the array is a control descriptor that returns to the first descriptor in the array (e.g. LINK).

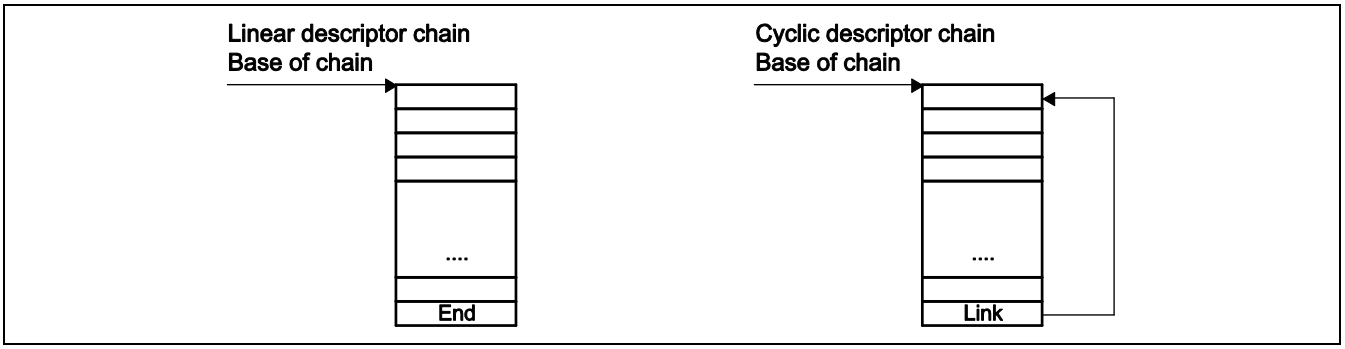



**Figure 100.1.3.3.2 Outline of the Basic Descriptor Chains**

The relationship between queues and descriptor chains is defined by the base addresses of chains. A queue is connected to one descriptor chain over one round of processing. There is also a method of switching to a different chain while in operation mode.

There are no restrictions on the number of link descriptors and their locations within the chain. The last descriptor of a designed chain determines the topology.

Which chain structure is to be used or which topology is suitable depends on the application. A description of how to design descriptor chains to suit various applications is given in section 100.1.3.4(2) Setting Up Reception Descriptors, and section 100.1.3.5(2) Setting Up Transmission Descriptors.

### (3) Descriptor Base Address Table

The base address table in the URAM contains the address of the first descriptor of all chains to be handled by the respective queues.

Entries 0 to 3 are used to access transmission queues 0 to 3. Subsequent entries are used to access reception queues. Entry 4 thus corresponds to reception queue 0.

The configuration of entries in the base address table is the same as the configuration of link descriptors. We recommend using the descriptor type (DESCR.DT) LINKFIX. Processing of this link descriptor does not change it, so it does not require updating. The first descriptor of a chain performs hardware and software synchronization. If the application requires hardware and software synchronization for the base addresses, use the descriptor type (DESCR.DT) LINK.

The CPU is only capable of using LINKFIX and LINK as descriptor types (DESCR.DT) of descriptors in the base address table.

Set the location of the base address table in the URAM in the descriptor base address table register (DBAT).

Figure 100.1.3.3.3 shows an example of a base address table for controlling four transmission and three reception queues. The boxes to the right of the table represent descriptor chains with the desired topologies.

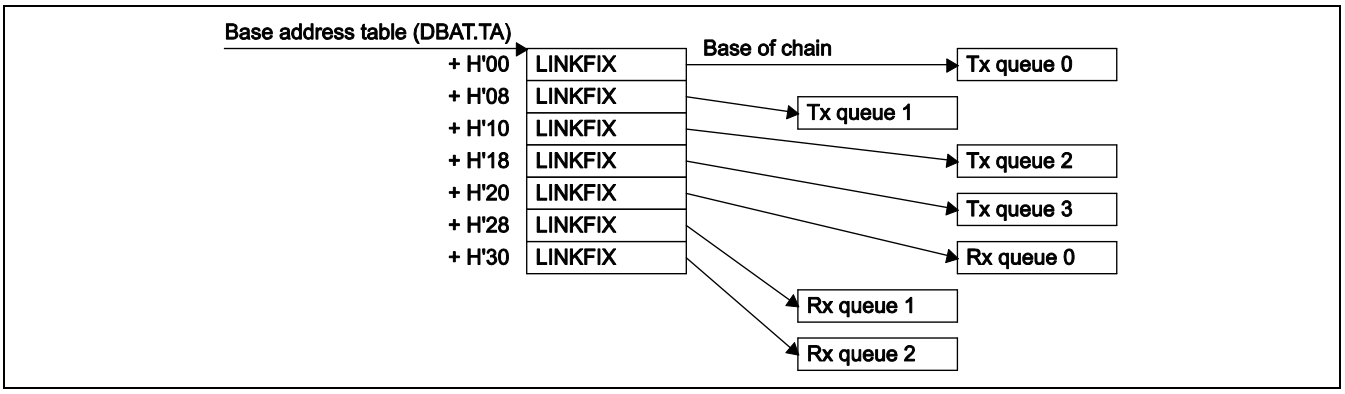



**Figure 100.1.3.3.3 Example of a Base Address Table for Reception and Transmission Queues**

Note: The size of the descriptors in the base address table is always eight bytes even if the queue itself includes extended descriptors.

### (4) Descriptor Chain Processing

The descriptor that is currently processed or will be processed when the related queue gets active is the current descriptor. The current descriptor address for use by a queue q can be checked in the current descriptor address register q (CDARq).

Current descriptors are stored in registers or in descriptors as described below in the given situations.

- In the descriptor base address table registers for all q queues (DBAT) (DBAT.TA+8*q) when the operating mode shifts to operation mode.
- In the descriptor base address table register (DBAT) (DBAT.TA+8*q) when a base address load request is issued for a queue q by setting the corresponding bit (DLR.LBAq) in the descriptor base address load request register (DLR).
- In DESCR.DPTR for a link descriptor (LINK, LINKFIX) to be processed.
- After a descriptor has been processed, the current descriptor for the same queue is incremented by the size of the descriptors being handled by the queue (8 bytes for normal descriptors and 20 bytes for extended descriptors). The AVB-DMAC updates the descriptor type and informs the CPU that the descriptor has been processed.

### (5) Descriptor Interrupts

A descriptor is able to issue a descriptor interrupt on completion of its processing. The setting of the descriptor interrupt enable bits (DESCR.DIE) in each descriptor selects disabling or generation of the descriptor interrupt.

EthernetAVB-IF provides two kinds of descriptor interrupts:
- Queue specific descriptor interrupt (1 per receive/transmit queue)
- Universal descriptor interrupt (15 shared between all receive/transmit queues)

The descriptor interrupt is a common resource that is shared between reception and transmission queues. Software control of the descriptor interrupt provides a flexible method of application-specific flag processing.

Figure 100.1.3.3.4 illustrates the way in which the AVB-DMAC generates descriptor interrupts (or sets bits in the descriptor interrupt status register (DIS.DPFi)). Processing of a descriptor with the value i in the descriptor interrupt enable bits (DESCR.DIE) leads to the corresponding bit in the descriptor interrupt status register (DIS.DPFi) being set.

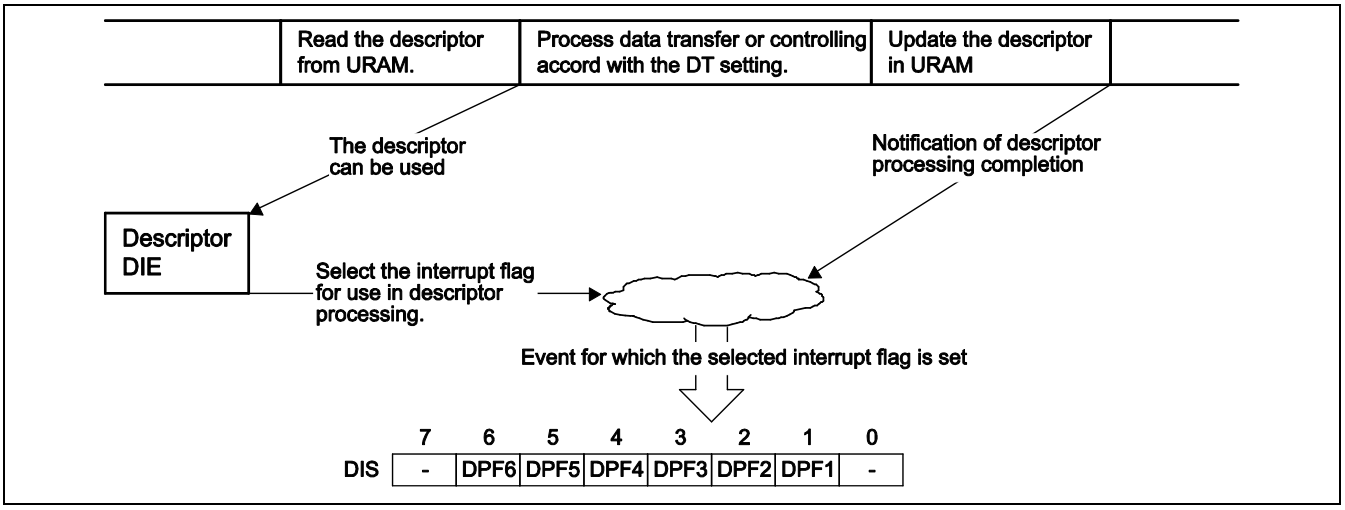



**Figure 100.1.3.3.4 Method of Descriptor Interrupt Generation**

### (6) Descriptor Type

The descriptor types (indicated by the DESCR.DT bits) supported by the AVB-DMAC fall into the following three categories.

- Definitions of frame data
- Control of descriptor chains
- Hardware and software arbitration

Table 100.10 is a summary of the descriptor types available for the AVB-DMAC. Entries under “Name” are the names of the descriptor types and the values under “DT” are the corresponding values to be set in the descriptor type field (DESCR.DT). A given descriptor may be handled differently according to whether it is in a transmission or reception queue, so the transmission and reception columns list the scopes of control and processing of the descriptor types.

The abbreviations defined below are used in the transmission and reception columns.

**Definition of SW:**

— The descriptor is processed by software.
— Software has access to and may modify the descriptor and descriptor data area.
— This descriptor cannot be changed by hardware (AVB-DMAC).

**Definition of HW:**

— The descriptor is processed by hardware (AVB-DMAC).
— Software must modify neither the descriptor nor the descriptor data area.
— Hardware (AVB-DMAC) processes this descriptor and subsequently changes the descriptor type.

**Invalid:**

This descriptor type is not used in transfer in the given direction (transmission or reception).

Do not write this value to the descriptor type (DESCR.DT) field for transfer in the given direction.

Hardware does not process these descriptor types in the cases listed as invalid. The current descriptor address (CDARq.CDA) will not be changed when processing of a queue for the given direction arrives at a descriptor with this type setting.

**Table 100.10 Summary of Descriptor Type**

| Name | DT | Description | Reception | Transmission |
|---|---|---|---|---|
| Frame data | | | | |
| FSTART | 5 | Frame Start<br>The descriptor points to valid data for a frame. The frame starts with the given data and continues with that indicated by the next descriptor. | SW | HW |
| FMID | 4 | Frame Middle<br>The descriptor points to valid data for a frame. The frame started with a previous descriptor and continues to the data indicated by the next descriptor. | SW | HW |
| FEND | 6 | Frame End<br>The descriptor points to valid data for a frame. The frame continues from the previous descriptor and ends with the data indicated by in this descriptor. | SW | HW |
| FSINGLE | 7 | Frame Single<br>The descriptor points to valid data for a complete frame. | SW | HW |
| Chain control | | | | |
| LINK | 8 | Link<br>Defines the next descriptor in the chain. | HW | HW |
| LINKFIX | 9 | Fixed Link<br>Same as LINK, but not changed by AVB-DMAC after processing. | SW | SW |
| EOS | 10 | End Of Set<br>Control element to split a descriptor chain. The chain stops and waits for user interaction. | HW | HW |
| HW/SW arbitration | | | | |
| FEMPTY | 12 | Frame Empty<br>A descriptor related to frame data but not containing valid data for a frame | HW | SW |
| FEMPTY_IS | 13 | Frame Empty Incremental Start<br>A descriptor related to frame data but not containing valid data for a frame.<br>DESCR.DPTR sets the base address of an “incremental data area” in the URAM. | HW | Invalid |
| FEMPTY_IC | 14 | Frame Empty Incremental Continue<br>A descriptor related to frame data but not containing valid data for a frame.<br>Data is stored to the incremental data area in the URAM. | HW | Invalid |
| FEMPTY_ND | 15 | Frame Empty No Data storage<br>A descriptor related to frame data but not containing valid data for a frame.<br>The descriptor is processed in the same way as FEMPTY but data are not stored in the URAM. | HW | Invalid |
| LEMPTY | 2 | Link Empty<br>A link descriptor for processing by the AVB-DMAC | SW | SW |
| EEMPTY | 3 | EOS Empty<br>An EOS descriptor for processing by the AVB-DMAC | SW | SW |
| DT0 | 0 | Reserved | Invalid | Invalid |
| DT1 | 1 | Reserved | Invalid | Invalid |
| DT11 | 11 | Reserved | Invalid | Invalid |

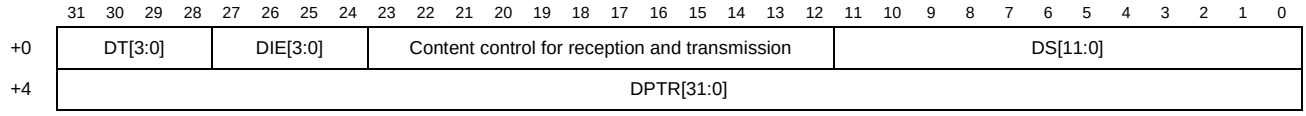
### (7) Layout of General Descriptors in the URAM

The AVB-DMAC updates processed descriptors in the URAM. The field to be changed in the descriptor being updated depends upon whether the direction is transmission or reception and the queue mode. Other fields will not be changed. There are no restrictions on the values set in unused descriptor fields (indicated by "—" in the figure).

#### (a) Frame Data Descriptors

The allocation of bits in the frame data descriptors (FSTART, FMID, FEND, and FSINGLE) is shown below.

- Normal descriptor (usable in both reception and transmission)




- Extended descriptor (usable only in reception)

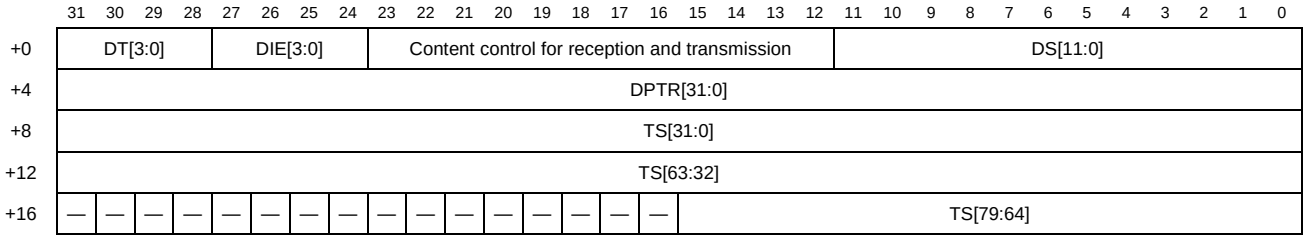



**Table 100.11 Contents of Frame Data Descriptors (DESCR)**

| Bit Name | Function |
|---|---|
| DT[3:0] | Descriptor Type<br>5: FSTART<br>4: FMID<br>6: FEND<br>7: FSINGLE<br>For details, see section 100.1.3.4(2) Setting Up Reception Descriptors, and section 100.1.3.5(2) Setting Up Transmission Descriptors. |
| DIE[3:0] | Descriptor Interrupt Enable<br>B'0000: Descriptor interrupt is disabled.<br>B'0001 to B'1111: The corresponding descriptor interrupt is generated (DIS.DPFi). |
| — | Content Control<br>For details, see section 100.1.3.4(2) Setting Up Reception Descriptors, and section 100.1.3.5(2) Setting Up Transmission Descriptors. |
| DS[11:0] | Data Size<br>Size of the data area/frame data for the descriptor (in bytes) |
| DPTR[31:0] | Descriptor Pointer<br>Pointer to the data area for the descriptor |
| TS[79:0] | Time Stamp<br>Time stamp of the received frame (only available in extended descriptors) |

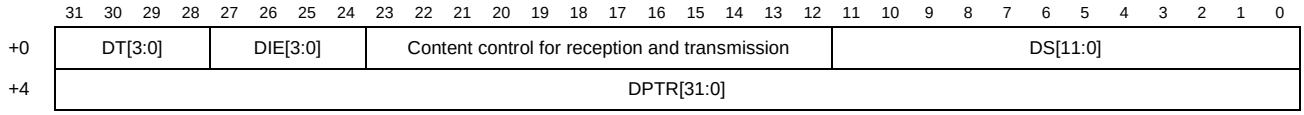
Note: Register an address aligned with a 32-bit boundary as the descriptor pointer (DESCR.DPTR).
Some bits in extended descriptors are reserved (The reserved bits (above DESCR.TS[79:64]) in an extended descriptor are set to H'0000 after the time stamp is stored).

**(b) Hardware/Software Arbitration Descriptors (Only for Reception)**

The allocation of bits in the descriptors for hardware/software arbitration (FEMPTY, FEMPTY_IS, FEMPTY_IC, and FEMPTY_ND) is shown below.

The allocation of bits in the arbitration descriptors for use in reception is the same as in frame data descriptors.

- Normal descriptor




- Extended descriptor (usable only in reception)

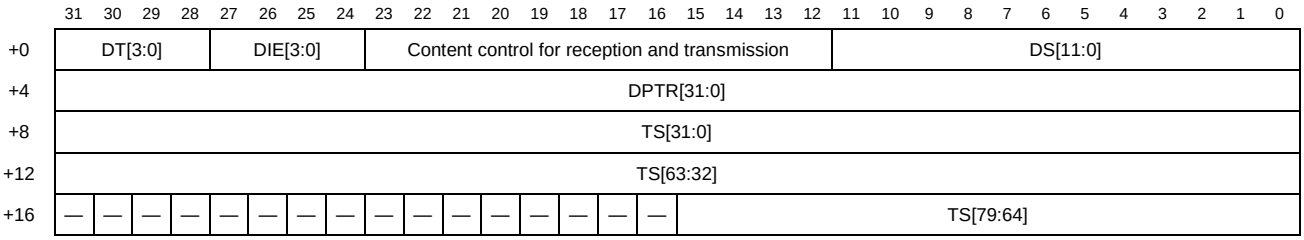



**Table 100.12 Contents of Hardware/Software Arbitration Descriptors (DESCR)**

| Bit Name | Function |
|---|---|
| DT[3:0] | Descriptor Type<br>12: FEMPTY<br>13: FEMPTY_IS<br>14: FEMPTY_IC<br>15: FEMPTY_ND<br>For details, see Table 100.10, Summary of Descriptor Types. |
| DIE[3:0] | Descriptor Interrupt Enable<br>B'0000: Descriptor interrupt is disabled.<br>B'0001 to B'1111: The corresponding descriptor interrupt is generated (DIS.DPFi). |
| — | Content Control<br>For details, see section 100.1.3.4(2) Setting Up Reception Descriptors, and section 100.1.3.5(2) Setting Up Transmission Descriptors. |
| DS[11:0] | Data Size<br>Size of the data area/frame data for the descriptor (in bytes) |
| DPTR[31:0] | Descriptor Pointer<br>Pointer to the data area for the descriptor |
| TS[79:0] | Time Stamp<br>Time stamp of the received frame (only available in extended descriptors) |

Note: Register an address aligned with a 32-bit boundary as the descriptor pointer (DESCR.DPTR).
When the descriptor is an extended descriptor, it has a 12-byte unused area.

In an FEMPTY descriptor, the descriptor type (DT), descriptor interrupt enable (DIE), data size (DS), and descriptor pointer (DPTR) fields are used.

In an FEMPTY_IS descriptor, the descriptor type (DT), descriptor interrupt enable (DIE), and descriptor pointer (DPTR) fields are used.

In an FEMPTY_IC descriptor, the descriptor type (DT) and descriptor interrupt enable (DIE) are used.

In an FEMPTY_ND descriptor, the descriptor type (DT), descriptor interrupt enable (DIE), and data size (DS) are used.

**(c) Link Descriptors**

The allocation of bits in the link descriptors (LINK and LINKFIX) is shown below.

- Normal descriptor

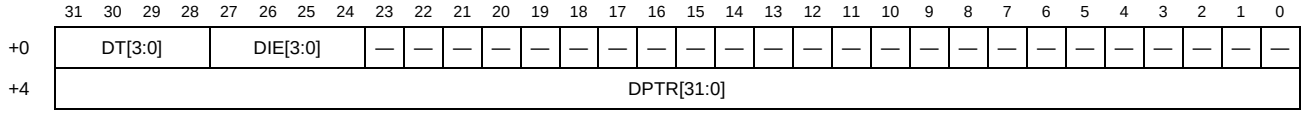



**Table 100.13 Contents of Link Descriptors (DESCR)**

| Bit Name | Function |
|---|---|
| DT[3:0] | Descriptor Type<br>8: LINK<br>9: LINKFIX<br>For details, see Table 100.10, Summary of Descriptor Types. |
| DIE[3:0] | Descriptor Interrupt Enable<br>B'0000: Descriptor interrupt is disabled.<br>B'0001 to B'1111: The corresponding descriptor interrupt is generated (DIS.DPFi). |
| DPTR[31:0] | Descriptor Pointer<br>Pointer to the data area for the descriptor<br>Register an address on a 32-bit boundary. |

Note: Register an address aligned with a 32-bit boundary as the descriptor pointer (DESCR.DPTR).

### (d) Other Descriptors

The allocation of bits in the other descriptors (EOS, FEMPTY (only for transmission), LEMPTY, and EEMPTY) is shown below.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| +0 | DT[3:0] | | | | DIE[3:0] | | | | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| +4 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

**Table 100.14 Contents of Other Descriptors (DESCR)**

| Bit Name | Function |
|---|---|
| DT[3:0] | Descriptor Type<br>10: EOS<br>12: FEMPTY (only for transmission)<br>2: LEMPTY<br>3: EEMPTY<br>For details, see Table 100.10, Summary of Descriptor Types. |
| DIE[3:0] | Descriptor Interrupt Enable<br>B'0000: Descriptor interrupt is disabled.<br>B'0001 to B'1111: The corresponding descriptor interrupt is generated (DIS.DPFi). |

### (8) How to Use Frame Data Descriptors

The descriptor data area size field (DESCR.DS) can specify up to 2048 bytes of Ethernet frame data per data area. Settings higher than 2048 (bytes) cannot be made.

In general, Ethernet frames are not of uniform length. The AVB-DMAC is capable of dividing frame data into multiple descriptors in order to minimize the memory capacity for frame data. This function allows processing of frames that are longer than the limit for descriptor data areas. Division can also be applied to frames on the basis of their data structures.

To handle frames and descriptors, four descriptor types (DESCR.DT) as FSTART, FEND, FMID and FSINGLE are defined.

Figure 100.1.3.3.5 shows the mapping of frame data by frame data descriptors. The descriptor data areas are allocated to the URAM. For frames that require division into four or more data areas, additional FMID descriptors can be added as required.

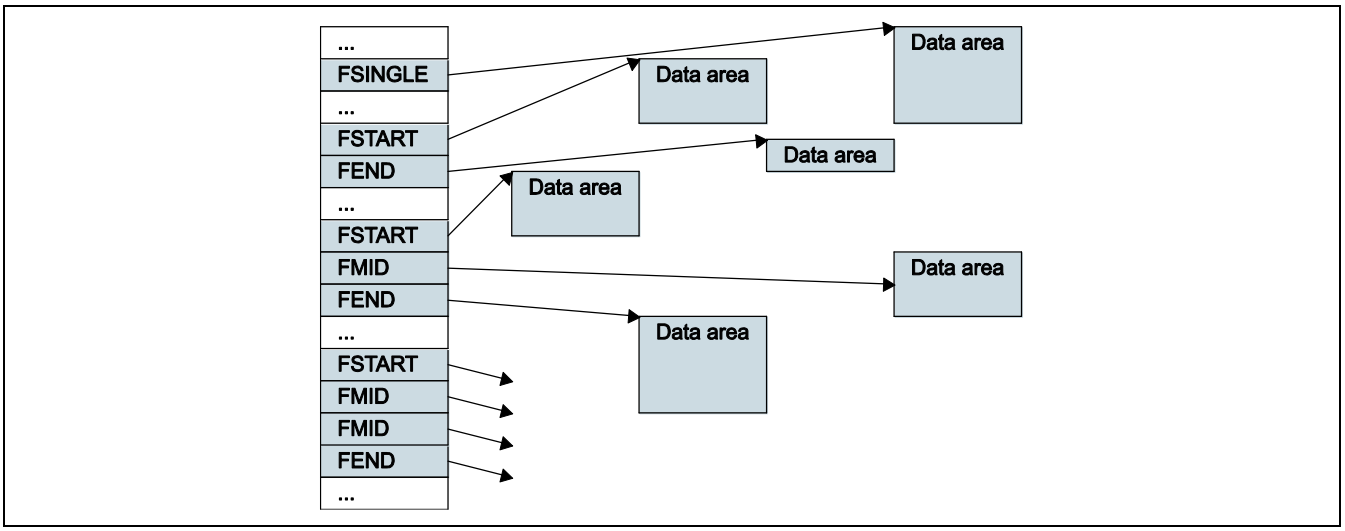



**Figure 100.1.3.3.5 Mapping of Frame Data**

For reception, set the descriptor data areas to the maximum size (i.e. give DESCR.DS its maximum value). The AVB-DMAC will store received frame data in the given area. If a received frame has more data than the maximum size, the AVB-DMAC will divide the data up.

For transmission, set the descriptor data area size field to the actual data size. The AVB-DMAC modifies the descriptor type (DESCR.DT) to FEMPTY after processing the relevant descriptor. The data size (DESCR.DS) and descriptor pointer (DESCR.PTR) fields retain their settings.

A descriptor data area including unused space produces an empty space between data areas. In reception, an “incremental data area” can be used to prevent empty spaces. For incremental data areas, see section 100.1.3.4(3)(b) Incremental Data Areas.

As well as reducing the memory capacity taken up by the descriptor area in the URAM, division into frames can be used to identify different sections of data (e.g. for separating a header and data).

### (9) How to Use Chain Control Descriptors

**Link Descriptors**

The link descriptors can be used to set up cyclic descriptor chains (for details, see section 100.1.3.3(2) Using Descriptor Chains in Queues)

After a LINK descriptor is processed, its descriptor type (DESCR.DT) is changed to LEMPTY. The descriptor pointer (DESCR.PTR) retains its setting.

After processing of a LINKFIX descriptor, the descriptor type (DESCR.DT) is not updated. Software can change the descriptor type (DESCR.DT), descriptor interrupt enable (DESCR.DIE), and descriptor pointer (DESCR.DPTR). However, DESCR.DT should not be modified by software. Take care to check the current descriptor address register (CDARq.CDA) before changing the descriptor pointer (DESCR.DPTR).

**EOS Descriptor**

Use the EOS descriptor to divide a descriptor chain into various segments. The queue can continue even after an EOS descriptor.

In transmission, the response to an EOS descriptor is clearing of the transmit start request bit in the transmit configuration control register (TCCR.TSRQq) to 0.

In reception, the response is generation of a receive queue full interrupt (RIS2.QFFr), although if the frame currently being received is being divided for storage (received data such as those where some storage is in FMID- or FEND-type frames), the data are not completely stored.

### (10) How to Use Hardware and Software Arbitration Descriptors

In hardware processing of descriptors, the empty descriptor types (FEMPTY, LEMPTY, and EEMPTY) are used to distinguish various descriptors. For software, they can be used to initiate checking for empty spaces, etc.

**FEMPTY, FEMPTY_IS, FEMPTY_IC, and FEMPTY_ND**

These descriptor types (DESCR.DT) are used for descriptors that do not contain effective data. Of these, only FEMPTY is used in transmission. The descriptor pointer of a FEMPTYxxx descriptor refers to a descriptor data area.

**LEMPTY**

This descriptor type (DESCR.DT) is assigned to LINK descriptors after they have been processed. The descriptor pointer (DESCR.DPTR) of an LEMPTY descriptor still points to the linked descriptor.

**EEMPTY**

This descriptor type (DESCR.DT) is assigned to EOS descriptors after they have been processed. The descriptor pointer (DESCR.DPTR) of an EEMPTY descriptor is not used.

### (11) Synchronization between Descriptor Access by Hardware and Software

The allocation of descriptor types (DESCR.DT) to the URAM can be used to set up the primary synchronization between hardware and software. By this, the number of CPU accesses to registers of EthernetAVB-IF can be minimized and performance can be increased.

Basic concepts of synchronization:

- Each descriptor type in the set is exclusively for processing by hardware or software, depending on the direction of transfer (see Table 100.10 Summary of Descriptor Type).
- Software must not change a descriptor assigned to hardware processing (the hardware does not change descriptors assigned to software processing).

In the case of software processing, the software must process the information in the descriptor and the corresponding frame data before changing the descriptor type. If a descriptor type for hardware is set in DESCR.DT, the software should not change any part of the descriptor or of the corresponding frame data.

### (12) Tips for Optimizing Performance in Handling Descriptors

The following items are recommended as ways to ensure the optimal use of data structures in the URAM.

They are not requirements, but using a different approach may increase the load on the system bus within the LSI chip.

- Register descriptors with 64-bit alignment (this does not apply to extended descriptors).
- While in operation mode, use LINKFIX instead of LINK whenever a descriptor need not be changed. Hardware modifies the descriptor type (DESCR.DT) fields of LINK descriptors.
- Frame data is accessed in blocks up to 128 bytes.
- The number of 128 byte borders (addresses H'x xx00 and H'x xx80) and frame data inside should be minimized.
- Design the descriptor chains in ways that minimize parallelism of processing. This helps in dividing the chains into segments allocated to different cache pages, and in arranging the different segments exclusively for access by software or hardware.
- Minimize the number of divided frames. This can reduce the overhead of descriptor handling.

### 100.1.3.4 Control in Reception

The point of the AVB-DMAC is to transfer data between the E-MAC and URAM without intervention by the CPU.

AVB-DMAC needs descriptors that define the amounts of frame data to be stored and the locations. After the E-MAC receives a frame, it stores the received frame data and the conditions of reception as the E-MAC state. If the descriptor is extended, the time stamp is also stored. For a description of how to set up descriptors for use in reception, see section 100.1.3.4(2) Setting Up Reception Descriptors.

The AVB-DMAC filters received frames to separate them into various classifications (separation filtering). More specifically, this is done to separate received frames into the various reception queues and to set the priorities of different classes of received frames. For more on separation filtering, see section 100.1.3.4(1)(a) Separation Filtering.

Figure 100.1.3.4.1 shows the reception data bus and the selection of queues for use in reception.

Each frame received from the E-MAC is stored in the reception FIFO; in parallel with this, the frame is analyzed to identify its type and the target queue number. After the E-MAC completes reception, the target queue number is generated and stored in the reception FIFO. Appending of a reception flag depends on the storage of one frame among the reception queues in the URAM, and the unread frame counter (UFC) is also associated with frame storage.

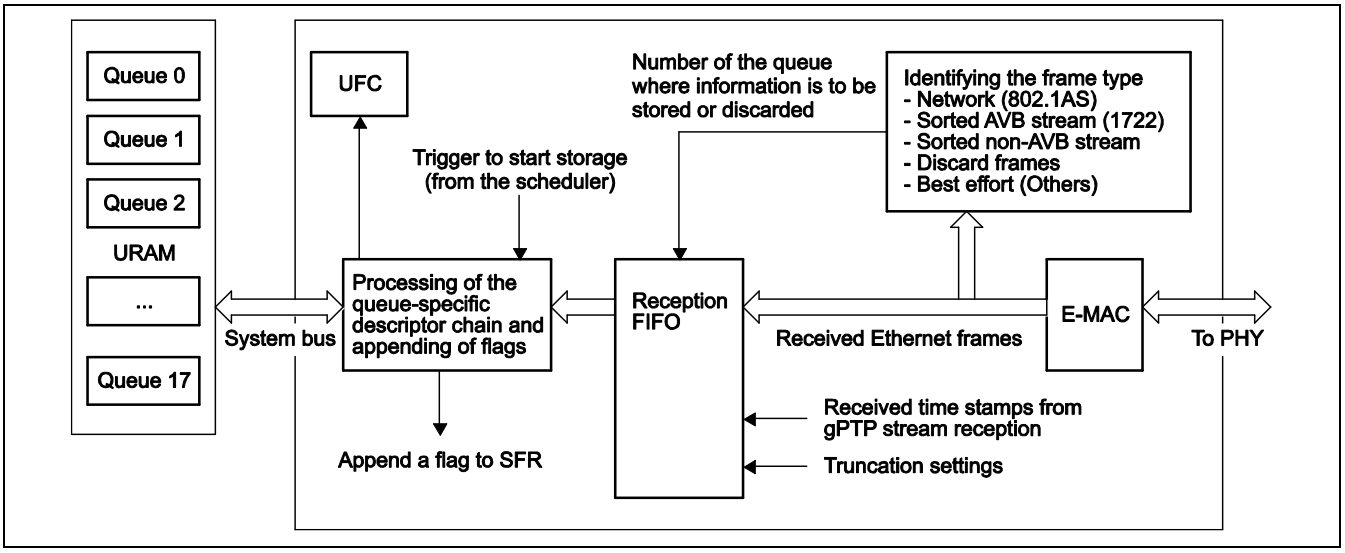



**Figure 100.1.3.4.1 Mechanism of General Reception Queue Selection**

### (1) Reception Queues

The AVB-DMAC applies its separation filtering mechanism to select the reception queue for storing a received frame. The AVB-DMAC stores all received frames in the URAM.

There are two conditions for the AVB-DMAC to discard a received frame.

- Detection of an error during reception by the E-MAC
  - Whether error frames are discarded or stored in reception queue 0 (best effort) depends on the setting of the error frame enable bit in the receive configuration register (RCR.EFFS). If error frames are to be stored (RCR.EFFS = 1), they are always stored in queue 0 (best effort). In this case, characteristics specific to the queue (e.g. truncation) will vary. If the storage of time stamps for reception queue 0 (best effort) is enabled (the time stamp enable bit in the receive configuration register RCR.ETS0 = 1), time stamps are stored even for error frames.
- Frame fails the separation filter
  - It depends on RCR.ESF if such frame is discarded or stored in receive queue 0 (best effort).

The flowchart in Figure 100.1.3.4.2 shows how the AVB-DMAC selects the reception queue in accord with the frame type, including judgment by the separation filter. Selection of the queue starts when the E-MAC completes frame reception. The result is storage of the frame in the proper queue or the frame being discarded.

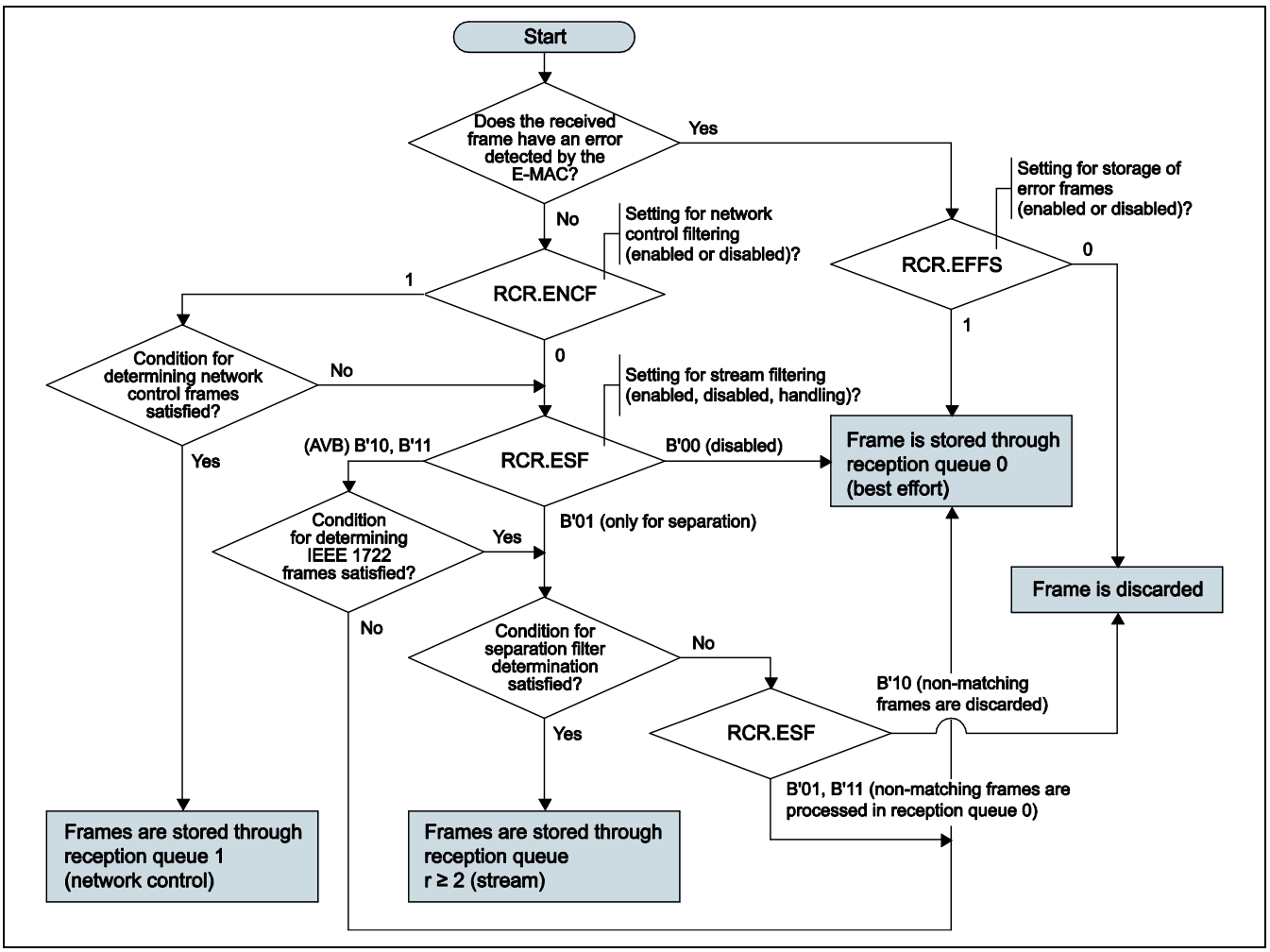



**Figure 100.1.3.4.2 Mechanism of Reception Queue Selection**

Notes on the meanings of entries in the flowchart

- "Condition for determining network control frames"
  The Ethernet destination address (DA) is 01:80:C2:00:00:0E.
  The Ethernet type (ET) is 88:F7.
- "Condition for determining IEEE 1722 frames"
  The Ethernet destination address (DA) is within the range from 91:E0:F0:00:00:00 to 91:E0:F0:00:FE:FF.
  The VLAN tagged TPID (tag protocol identifier) field (VL) is 81:00.
  The Ethernet type (ET) is 22:F0.
- "Condition for separation filter determination"
  See section 100.1.3.4(1)(a) Separation Filtering.

Figure 100.1.3.4.3 shows the allocation of bits related to the network and stream types in Ethernet frames. The preambles of Ethernet frames are not taken into account.

| Data bytes | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Network type | DA1 | DA2 | DA3 | DA4 | DA5 | DA6 | SA1 | SA2 | SA3 | SA4 | SA5 | SA6 | ET1 | ET2 | ... | ... | ... | ... | ... | ... |
| Stream type | DA1 | DA2 | DA3 | DA4 | DA5 | DA6 | SA1 | SA2 | SA3 | SA4 | SA5 | SA6 | VL1 | VL2 | - | - | ET1 | ET2 | ... | ... |

**Figure 100.1.3.4.3 Data Bytes of Ethernet Frames Used in Classification**

**(a) Separation Filtering**

Separation filtering involves the checking of up to 64 bits (eight successive bytes) in received Ethernet frames. The setting for the first byte (i.e. the setting of the separation filter offset configuration register (SFO.FBP)), selects the part of frames to be used in separation filtering. There is also a common filter mask (set in the separation filter mask configuration register (SFMi.CFM)) that can be freely set to reduce the number of bytes used in separation filtering or to mask particular bits.

**Examples**

To use one byte in separation, set separation filter mask configuration register 0 (SFM0.CFM) to H'0000 00FF and separation filter mask configuration register 1 (SFM1.CFM) to H'0000 0000.

To use seven bytes in separation, set separation filter mask configuration register 0 (SFM0.CFM) to H'FFFF FFFF and separation filter mask configuration register 1 (SFM1.CFM) to H'00FF FFFF.

Note: If bits at some positions are set to 0 in the separation mask, in order to match with the pattern, the bits at the corresponding positions of the pattern must also be set to 0. Only those bits in which the separation filter pattern configuration register (SFPi.FPs) setting is equal to the separation filter mask configuration register (SFMi.CFM) are sorted by matching with received data.

Figure 100.1.3.4.4 shows separation filtering. The selected data from a received frame (Rx_Frame[63:0]) are masked by the common filter mask. As a result, the selected frame data can be obtained. This value is compared with all filter patterns. The separation filter circuit in the AVB-DMAC selects the filter pattern that matches the queue having the lowest index s or selects a flag to indicate that there is no matching separation pattern.

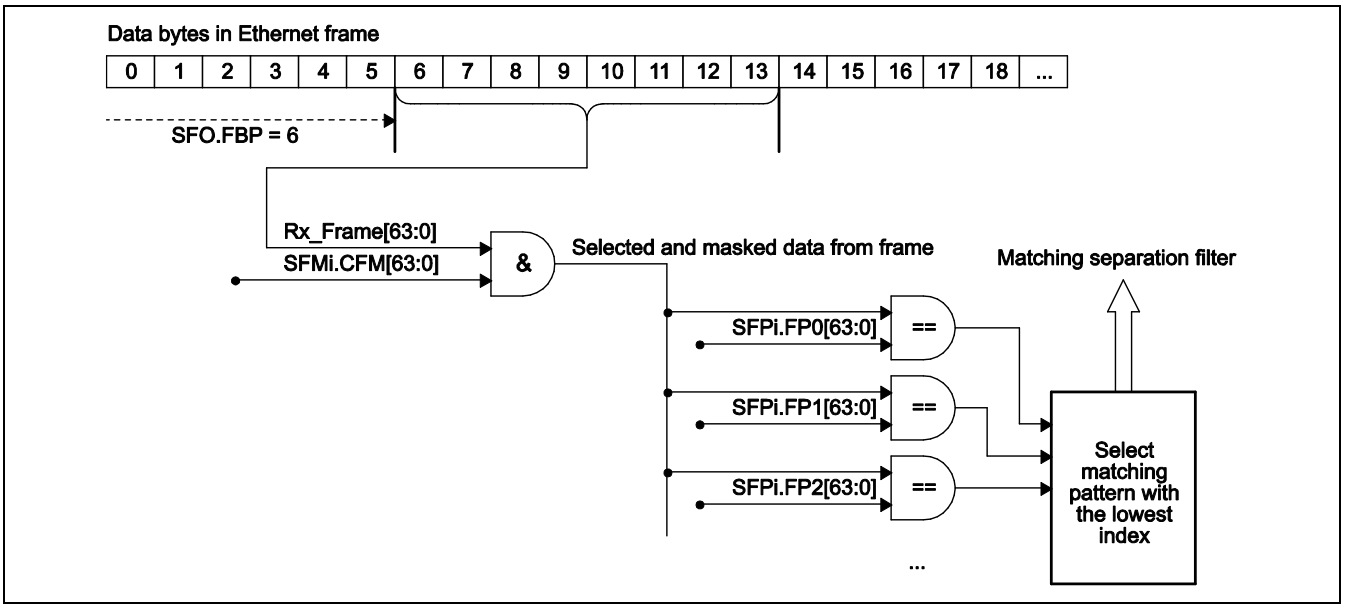



**Figure 100.1.3.4.4 Separation Filtering**

**(b) Separating Streams**

The AVB-DMAC applies separation filtering to sort frames received in streams. An AVB network has a concept of "Talker" and "Listener". A Talker is an end station that generates one or more streams. A Listener is an end station that has the role of being a sink for at least one stream. The various A/V streams are identified by 8-byte stream IDs.

The number of end stations within an AVB network and their roles differ with the application.

The stream ID is a general pattern of the AVB network for identifying one stream. Figure 100.1.3.4.5 shows the bit allocation of bits in IEEE1722 Ethernet frames and stream ID fields.

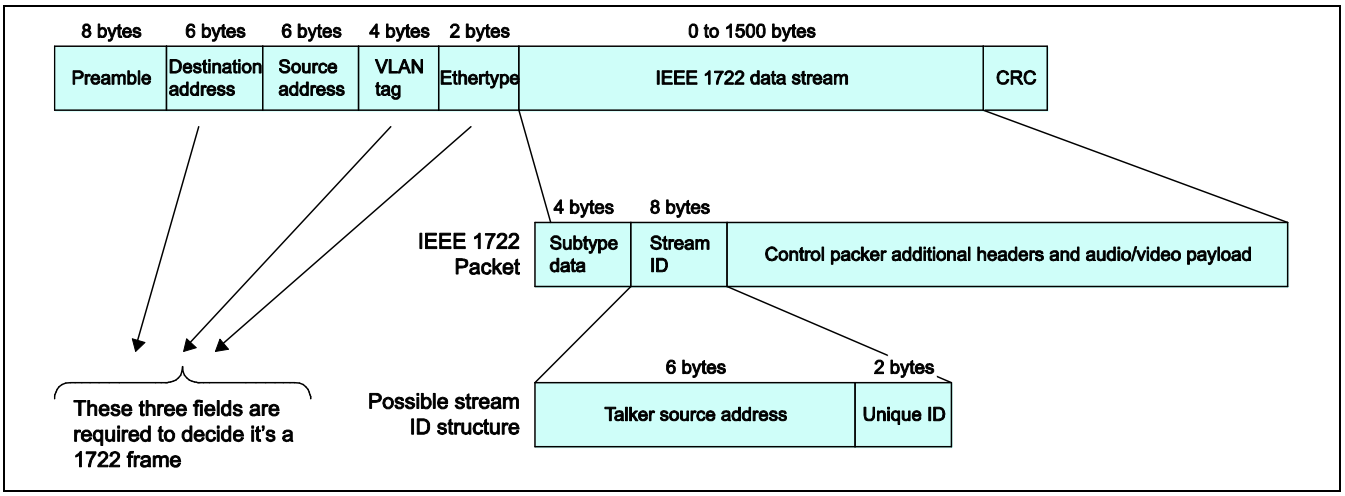



**Figure 100.1.3.4.5 IEEE 1722 Frame Layout and Stream ID**

The IEEE 1722 standard stipulates that the stream ID field starts from the 23rd byte (not counting the preamble). Accordingly, set the separation filter offset (SFO.FBP) to 22 in operations on IEEE 1722 streams. Set the separation filter mask (SFMi) and separation filter pattern (SFPi) in accord with the specification of the product in which the chip is being used.

Example: In the example of a stream ID shown in Figure 100.1.3.4.5, the current application divides the field into the talker source address and the unique stream ID. The unique ID is used to differentiate between multiple streams from the same talker. Based on this, there are two settings for separation filter masking:

- To divide various streams into individual queues, set SFM0.CFM to H'FFFF FFFF and SFM1.CFM to H'FFFF FFFF.
- To divide streams from various talkers into individual queues, set SFM0.CFM to H'FFFF FFFF and SFM1.CFM to H'0000 FFFF. This excludes the unique ID from the filter condition.

**(c) Reconfiguration of separation filter during OPERATION**

In CONFIG mode the initial separation filter configuration is defined. To prevent inconsistent filter results during OPERATION direct parameter change is not possible; instead update mechanism by **SFVi.LV** and **SFL.LC** should be used.

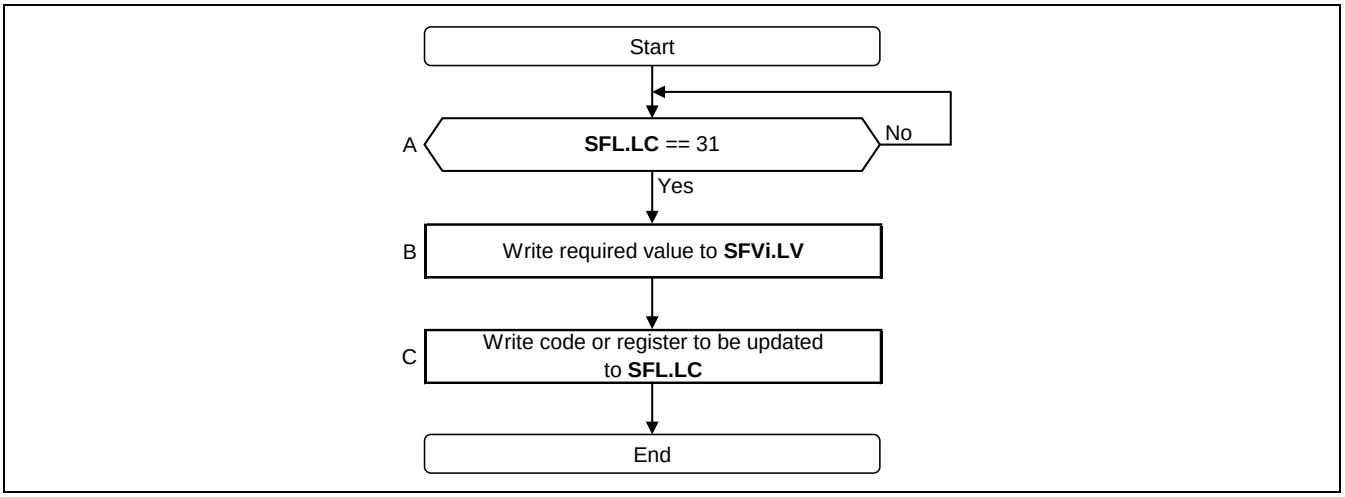



**Figure 100.1.3.4.6 SW flow to update separation filter during OPERATION**

When reconfigure separation filter, EthernetAVB-IF guarantees consistent classification for one received frame. If reconfiguration requires a sequence of parameter changes, CPU has to order reconfiguration based on application demand.

EthernetAVB-IF updates filter configurations when no frame reception is handled. In worse case the update is delayed by one frame reception. Exact duration depends on PHY speed and frame size. By reading **SFL.LC** CPU can observe when a previous update has been processed.

Note: Frame classification happens during frame reception and storage into internal Rx-FIFO. There may be frames with old classification pending in Rx-FIFO or under storage to URAM when filter reconfiguration is accepted.

### (2) Setting Up Reception Descriptors

For reception, the descriptor mechanism is essentially as described in section 100.1.3.3 Descriptors.

This section describes memory operations that are especially required in handling reception queues.

#### (a) Reception Descriptor Type

The type of a descriptor is defined by the descriptor type (DESCR.DT) field.

Table 100.15 shows the descriptor types used in reception.

**Table 100.15 Descriptor Types in Reception**

| Descriptor Type (DESCR.DT) | Operation | Write-back |
|---|---|---|
| Frame Start (FSTART) | Condition for data not being stored in a reception queue:<br>The RIS2.QFFr bit indicates that queue r is full and the received frame is not stored.<br>Descriptor processing proceeds again in response to further reception. | Not changed |
| Frame Middle (FMID) | Same as FSTART | Not changed |
| Frame End (FEND) | Same as FSTART | Not changed |
| Frame Single (FSINGLE) | Same as FSTART | Not changed |
| Link (LINK) | Processing proceeds to the descriptor specified by DESCR.DPTR. | LEMPTY |
| Fixed Link (LINKFIX) | Same as LINK | Not changed |
| End Of Set (EOS) | A stop point defined by software has been reached.<br>A descriptor of this type within a divided frame (writing of FMID or FEND) stops the frame being stored and the frame is lost. RIS2.QFFr indicates that the frame has been lost.<br>If this happens at the start of a frame (writing of FSTART or FSINGLE), storing of frames starts from the next descriptor.<br>In either case, processing shifts to the next descriptor in the chain. | EEMPTY |
| Frame Empty (FEMPTY) | The descriptor can be used to store received data.<br>Up to DESCR.DS bytes are stored in the descriptor data area.<br>For details, see section 100.1.3.4(3)(a) Storing Frame Data in the Descriptor Data Area. | FSTART, FMID, FEND, or FSINGLE |
| Frame Empty Incremental Start (FEMPTY_IS) | The descriptor can be used to store received data.<br>All data for the frame are stored in the descriptor data area.<br>DESCR.DPTR indicates the base address of the incremental data area.<br>For details, see section 100.1.3.4(3)(b) Incremental Data Areas. | FEND or FSINGLE |
| Frame Empty Incremental Continue (FEMPTY_IC) | The descriptor can be used to store received data.<br>The remaining bytes of frame data are stored in the descriptor data area.<br>DESCR.DPTR is undefined, but is written back at the start position within the incremental data area after processing.<br>For details, see section 100.1.3.4(3)(b) Incremental Data Areas. | FEND or FSINGLE |
| Frame Empty No Data storage (FEMPTY_ND) | The descriptor can be used to store received data.<br>Up to DESCR.DS bytes are captured from the reception FIFO but not stored. After processing, DESCR.DS is written back as 0.<br>For details, see section 100.1.3.4(3)(b) Incremental Data Areas. | FSTART, FMID, FEND or FSINGLE |
| Link Empty (LEMPTY) | Same as FSTART | Not changed |
| EOS Empty (EEMPTY) | Same as FSTART | Not changed |

**(b) Configuration of Reception Frame Data Descriptors**

Figure 100.1.3.4.7 shows the configuration of descriptors for use with reception queues. The reception-specific fields are the same whether the descriptor is normal or extended. The reception-specific fields (DESCR.MSC, DESCR.PS, DESCR.EI, and DESCR.TR) are described in Table 100.16.

For the other fields and the descriptor types, see section 100.1.3.3(6) Descriptor Type.

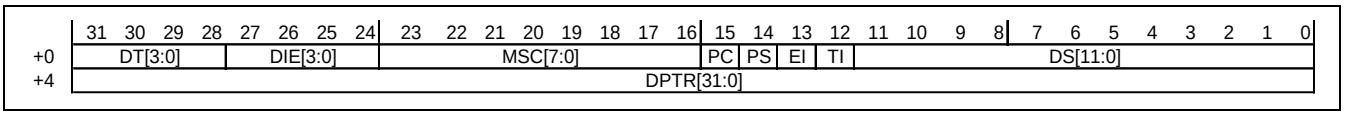



**Figure 100.1.3.4.7 Configuration of Descriptor for a Received Frame**

**Table 100.16 Configuration of a Received Descriptor**

| Bit Name | Function |
|---|---|
| MSC | E-MAC Status Code<br>These bits indicate errors in reception detected by the E-MAC.<br>In the case of a divided frame, these bits are set to the same value within all descriptors for the frame data. Details of the bits are as follows.<br>MSC[7]: Received frame has a multicast address.<br>MSC[6]: Carrier extend error<br>MSC[5]: Carrier sense error<br>MSC[4]: Received frame has alignment error.<br>MSC[3]: Received frame is too long.<br>MSC[2]: Received frame is too short<br>MSC[1]: Error in frame reception (PHY detects an error)<br>MSC[0]: Received frame has a CRC error. |
| PC | Payload Crc status<br>This register is functional safety use only. |
| PS | Padding Selection<br>These bits specify whether frame data are to be padded when stored in the incremental data area.<br>Insertion of padding data is in accord with the settings in the RPC register.<br>0: Padding is not to be inserted.<br>1: Padding data may be inserted. This depends on the RPC settings. |
| EI | Error Indication<br>This bit indicates the detection of an error in frame data while a frame was being stored.<br>The bit is set to 1 for a descriptor in which an error has been detected. If the descriptor is for a divided frame, storage of the frame is aborted.<br>0: No error<br>1: Error is detected |
| TI | Truncation Indication<br>This bit indicates whether frame data received from the E-MAC have been truncated before being stored.<br>These bits are set to the same value within all frame data descriptors for a divided frame.<br>0: Data have not been truncated.<br>1: Data have been truncated. |

Note: The RCR.EFFS bit specifies whether or not frames with errors detected by the E-MAC are to be stored in the URAM. When the storing of error frames is disabled, error codes are not written to DESCR.MSC.

### (3) Reception Processing

After initialization, the AVB-DMAC is able to select the proper reception queue and store received frames in the data area in the URAM as indicated by the descriptor. The AVB-DMAC continues to store received data in the URAM as long as space is available for descriptors and data areas.

Received frames are classified and stored in the reception FIFO in accord with the algorithm described in section 100.1.3.4(1)(a) Separation Filtering.

If the reception FIFO contains even one frame, storing is executed to the reception queue.

If there is even one empty data descriptor in a queue for which reception has started, the storage of frame data starts. Received frames for a queue that is already full (there is no empty frame descriptor or the UFC stop level has been reached) are discarded from the reception FIFO. This ensures that one queue being full does not prevent the storage of data in the other queues.

#### (a) Storing Frame Data in the Descriptor Data Area

Frame data for storage are assumed to be in either of the two patterns described below.

- The data for an entire frame will fit in the descriptor data area.
  - In this case, the descriptor type (DESCR.DT) is FSINGLE.
- Frame data to be stored in the descriptor data area arrive in divided form.
  - In this case, FSTART is written to the descriptor type (DESCR.DT) bits of the first descriptor of the frame data to arrive and FMID and FEND are written to the type bits of descriptors for subsequent data.

The descriptor type is updated by the AVB-DMAC in the last step of descriptor processing, so software can always access the descriptor assigned to DESCR.DT.

When normal synchronization mode is used, the CPU can write FEMPTYxxx directly to the descriptor type field after processing the stored element. Do not change the descriptor or any part of the descriptor data area after FEMPTYxxx is written to DESCR.DT.

**Storing Frame Data for a Whole Single Frame**

For a frame with an FSINGLE descriptor, all data for the frame are held at the position defined by DESCR.DPTR. DESCR.DS indicates the length of the received frame.

If DESCR.DS is bigger than the actual size of a received frame, the FSINGLE descriptor is stored in place of the FEMPTY or FEMPTY_ND descriptor after processing.

Also, the FSINGLE descriptor is stored in place of the FEMPTY_IS or FEMPTY_IC descriptor, which always hold the descriptor data area greater than all frame data in Rx-FIFO after processing.

**Storing Frame Data as Divided Frames**

Divided frames are handled in the same way as a single frame. A frame stored with divided descriptors must be recombined before use. DESCR.EI and DESCR.DS are only valid in the last descriptor of the sequence for a divided frame.

Note: If the data area size setting in DESCR.DS is not a multiple of four, the number of bytes set in DESCR.DS is fetched from the reception FIFO and the remaining bytes are used as the next storage area.

After a received frame is divided into different descriptors, each storage element is handled separately, and the descriptor type is assigned by software after processing. Accordingly, an error frame (FEMPTYxxx instead of FMID or FEND) may exist while a descriptor chain is being processed. In such a case, the CPU must postpone processing of the error frame to the next trigger point.

**No Data are Stored**

The application specification may lead to some types of received frames being unimportant (for example, when the application only requires stream data from the Ethernet frames). Storing frames in divided form makes separating out the unnecessary parts of Ethernet frames possible.

If part of a divided frame is not required, use the FEMPTY_ND descriptor for that part so that it is not stored in the URAM. Not storing the data negates the need for bandwidth on the data bus, improving the overall performance.

When an FEMPTY_ND descriptor is processed, DESCR.DS is set to 0. This brings the frame data section of the descriptor into agreement with the FEMPTY type. DESCR.DS = 0 is for the unique identification of the descriptor after writing.

**(b) Incremental Data Areas**

Secure space in the URAM for storing received data. Even when data are placed in the URAM area such that all descriptor data areas of a chain are contiguous, a received frame being shorter than the descriptor data area will lead to an empty space. Figure 100.1.3.4.8 shows an example of settings and the memory map.

Certain applications require that data areas be contiguous (e.g. when received data are to be processed other than by hardware as the A/V codec module). When the length of received frames differs (e.g. when payloads vary between having one or two A/V packages), the use of a static pointer in the descriptor produces empty spaces in the data area. This may necessitate direct additional processing to remove the empty spaces.

Accordingly, and to reduce the CPU load imposed by copying data, the AVB-DMAC supports an “incremental data area” function.

When incremental data areas are in use, all descriptors use a common data area for storage. One descriptor (FEMPTY_IS) defines the base address of the incremental data area and the next descriptor (FEMPTY_IC) within the descriptor chain holds received data. Figure 100.1.3.4.9 shows an example of settings and the memory map.

Use of an incremental data area does not reduce the memory space in the individual descriptor data areas.

The hardware and software synchronization strategy and performance are also not changed.

It is also possible to divide a frame up among various descriptors in a way that reflects its structure (e.g. one descriptor for the Ethernet header and one for the data payload).

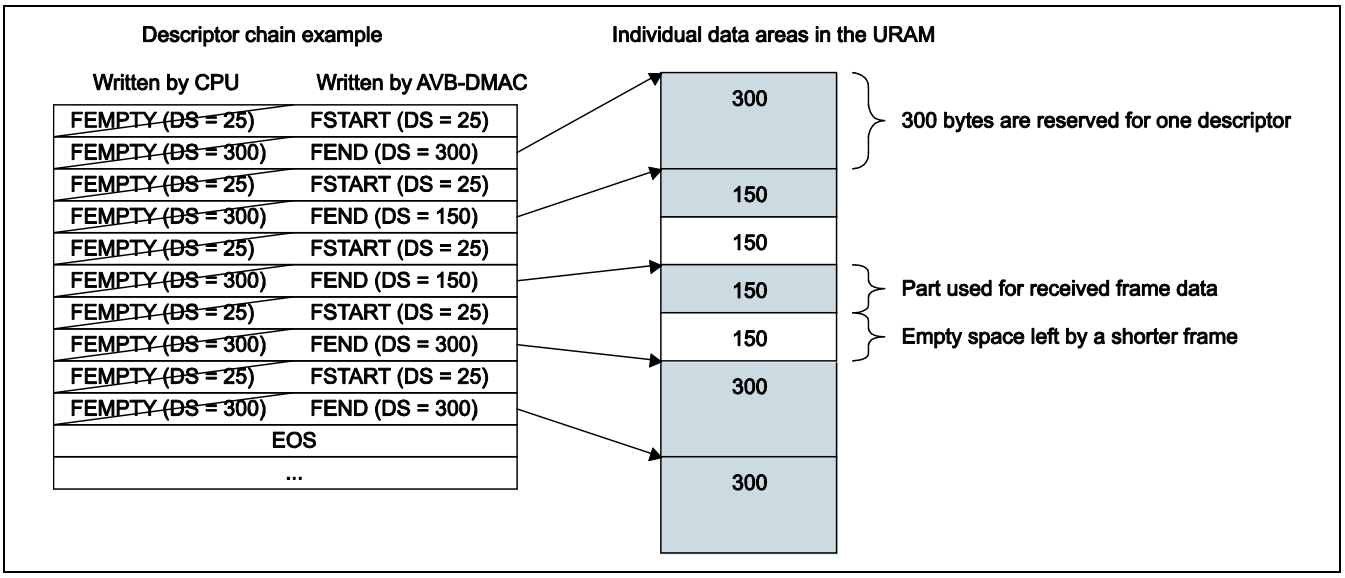



**Figure 100.1.3.4.8 A Reception Queue Using Individual Descriptor Data Areas**

Figure 100.1.3.4.8 and Figure 100.1.3.4.9 show how control of the data storage areas by a descriptor chain varies according to whether individual or incremental data areas are in use. The chains are configured for storing received frames consisting of a 25-byte header (which is treated as one descriptor; and is outside the scope) and a 150- or 300-byte payload (whether one or two 150-byte payload packages are transmitted with one Ethernet frame depends on the data source).

In Figure 100.1.3.4.8, the EOS descriptor is added as an example of a re-synchronization point. If the frame source transmits a frame containing more than 325 bytes, the frame will be divided among three descriptors, meaning that synchronization of the header and data sequences is lost. Despite this, however, the frame is not divided across the EOS descriptor, so recovery is automatic without software interaction. The EOS is not required with the incremental descriptors because all data being processed are always stored while an incremental data area is in use.

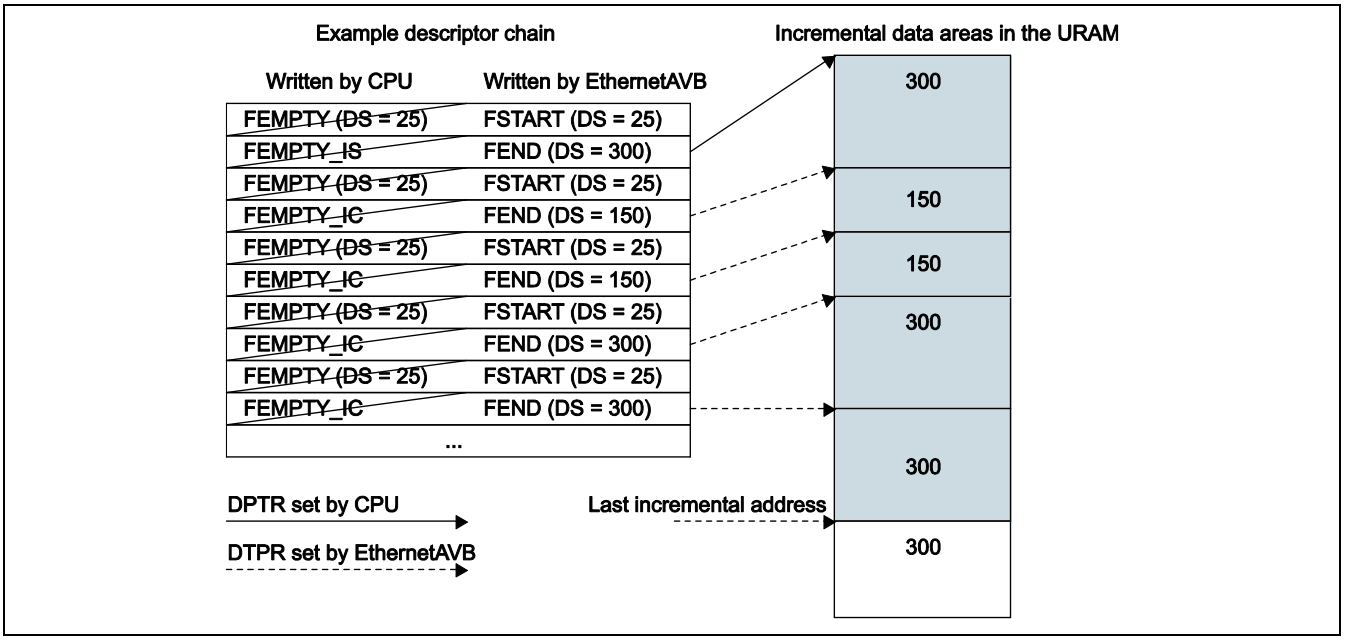



**Figure 100.1.3.4.9 Reception Queue Using a Common Incremental Data Area**

As Figure 100.1.3.4.9 shows, when data are stored in an incremental data area, the descriptor pointers in the FEMPTY_IC descriptors (DESCR.DPTR) are updated. Accordingly, the resulting FEND or FSINGLE descriptor is in the same format as after writing to an FEMPTY descriptor.

Software captures received data from an incremental data area, which has no empty storage areas between frame data. The only empty space is that at the end of the incremental data area. The sizes of incremental data areas and of blocks of data for storage in incremental data areas must be multiples of four bytes. When the amount of data for storage in an incremental data area is not a multiple of four bytes, from one to three bytes of empty space will be produced. DESCR.DS can be read to check for such empty spaces.

CPU cannot restrict directly the amount of received data stored by an incremental descriptor (FEMPTY_IS, FEMPTY_IC) as this is possible for other descriptors (FEMPTY, FEMPTY_ND) by DESCR.DS. Incremental descriptors store always all received data.

**Setting Up an Incremental Data Area**

A descriptor chain in the incremental data area having N descriptors (one FEMPTY_IS and N-1 FEMPTY_IC) means that the incremental storage area of N times of maximum frame size is needed.

As Figure 100.1.3.4.9 shows, DESCR.DPTR of an FEMPTY_IS descriptor indicates the base address of the incremental data area. The next FEMPTY_IS descriptor in the chain indicates the processing step where data must be stored in the incremental data area.

**Processing an Incremental Data Area Based on Descriptors**

Since data processing by the CPU is the same regardless of how the AVB-DMAC stores the data, data stored in an incremental data area do not require any special handling.

**Padding**

Use padding for received frame data that are not aligned correctly in the specified memory structure. Padding can be set individually for each descriptor. Accordingly, in the reception of divided frames, padding can be restricted to only those frames that require it (e.g. A/V payload data.)

Padding can also be used to optimize system performance in an incremental data area (e.g. to prevent inefficient access by aligning received data with 32-byte boundaries in the incremental data area), as well as to fulfill application-specific requirements for specified memory structures (e.g. formats required by other modules that will be processing the received data).

Padding can only be used in an incremental data area.

The value H'0000 0000 is always used in padding.

Padding is the addition of the number of words (from one to seven 32-bit words) set in the stored padding counter in the receive padding configuration register (RPC.PCNT). This padding is repeatedly inserted in accord with the value in the stored data counter (RPC.DCNT) (from one to 255 32-bit words). When the stored data counter (RPC.DCNT) reaches 0, however, padding is not repeated.

The first word of padding is always inserted at the position specified by DESCR.DPTR. When divided frames are in use, a padding word can be inserted at any byte position, and padding is handled on a 32-bit basis (e.g. an incremental data area where the first descriptor is for a 42-byte header data and the second descriptor holds padded payload data).

The next figure shows a general example of how padding is inserted and an example of setting up padding. A indicates frame data A received by the E-MAC, and B indicates the frame data stored in the descriptor data area (32-bit word units).

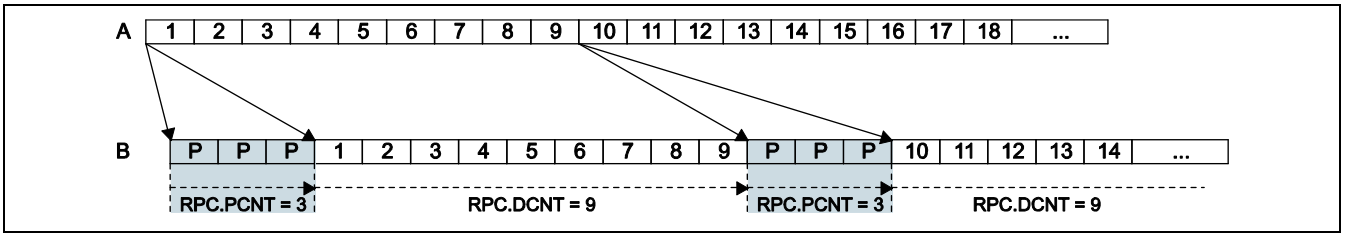



**Figure 100.1.3.4.10 Example of a Padding Setting**

Both padding and received frame data are counted in the descriptor size (DESCR.DS).

**(c) Mode with Write-Back**

Constructing a descriptor chain requires software (see Figure 100.1.3.4.11).

In the example in the figure, the variable SWdescr (software descriptor pointer) is a structure to identify a descriptor being processed. SWdescr must be initialized after operation mode is entered and a descriptor base address load request (DLR.LBAq) is executed (condition for starting the flow of software operations).

The frame_processing() function processes the stored data. The function can use SWdescr.DT to check whether processing of a frame is completed. How frame data are processed differs with the application, so create functions that handle processing in accord with the specification.

The processing section is common to all modes of reception. The number of frames processed in response to each trigger can be restricted. When multiple frames have to be processed in a batch, waiting for individual trigger boxes must be skipped for these frames.

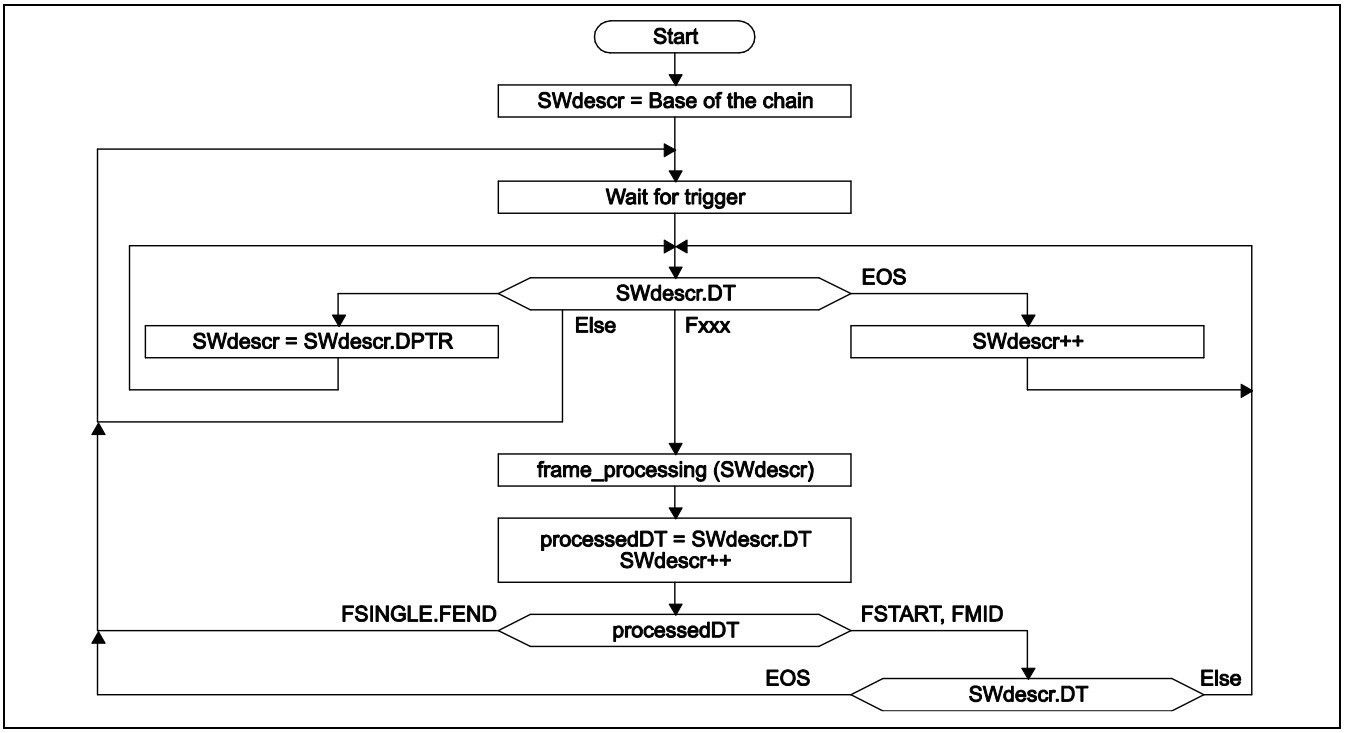



**Figure 100.1.3.4.11 Flow of Reception Descriptor Processing (with Write-Back)**

**(d) Support for Reception Time Stamps**

Capturing reception time stamps is essential for IEEE 802.1AS time synchronization. Other types of received frames may also require that a reception time stamp be appended; this depends on the application. The AVB-DMAC supports reception time stamps based on the gPTP timer by storing time stamps, which have been captured when Start Frame Delimiter (SFD) of a received frame is arrived, in the last frame data descriptor (FEND or FSINGLE). For the gPTP timer, see section 100.1.3.7(1) gPTP Timer.

When time stamps are to be stored, use extended descriptors for the entire reception queue. Furthermore, time stamps are always stored for reception queue 1 (network control). Time stamps for reception queue 0 (best effort) and reception queue r ($r \geq 2$; for stream data) can be selected by the time stamp enable bits in the receive configuration register (RCR.ETS0 or RCR.ETS2).

### (4) Unread Frame Counters

Each reception queue has an unread frame counter (UFCVi). Use the unread frame counter configuration bits in the receive queue configuration register (RQCi.UFCCr) to select from among the four warning and stop levels for each unread frame counter. The 0 setting disables the stop and warning functions. For how to set this up, see Figure 100.1.3.4.12.

Operations of the AVB-DMAC (hardware) and CPU (software) drive an unread frame counter (UFC) in the following ways.

- The hardware indicates that it has added a new frame to the descriptor chain for the queue (this increments the counter).
- Software indicates how many frames from the descriptor chain it has processed by writing to the corresponding bits of the unread frame counter decrement register for the queue (this decrements the register by the number written).

The unread frame counter is based on the number of frames stored in the URAM and is only incremented by one even when a received frame is divided into different descriptors. Failure in storing a descriptor chain requires care because this may unread frame counter may fail in synchronization as described in section 100.1.3.4(4)(a) Unread Frame (UFC) Synchronization Failure.

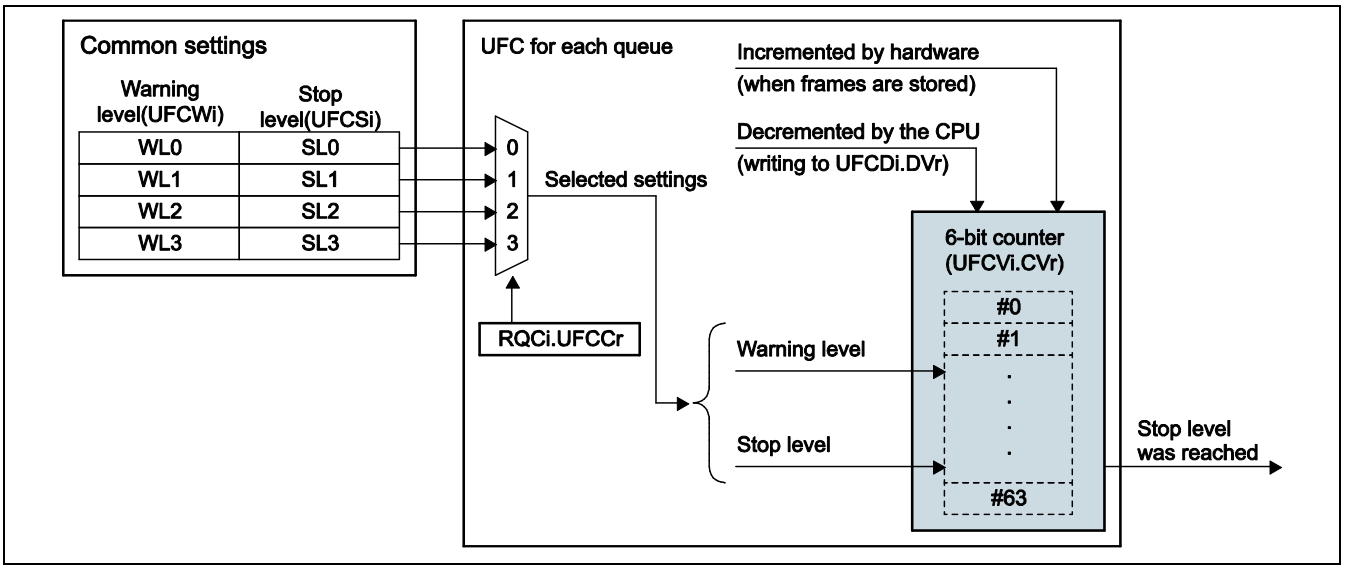



**Figure 100.1.3.4.12 Overview of an Unread Frame Counter**

Unless synchronization of hardware and software is lost, the current unread frame counter value (UFCVi.CVr) indicates the number of unread frames in the queue.

The indicator that the stop level has been reached prevents the storage of further received frames in the descriptor chain. Selecting 0 as the stop level disables this function. Otherwise, further received frames for the queue are discarded once its unread frame counter reaches the stop level. Activation of the unread frame counter stop function is indicating by setting of the receive queue full interrupt flag in the receive interrupt status register 2 (RIS2.QFFr).

Set the unread frame counter warning level configuration register (UFCW) and the unread frame counter stop level configuration register (UFCS) for each reception queue that will use the unread frame counter function while the current operating mode is configuration mode.

**(a) Unread Frame (UFC) Synchronization Failure**

The unread frame counters do not recognize failure to store a frame in the URAM. In other words, the AVB-DMAC increments the counter for a queue each time it captures a frame for that queue from the reception FIFO whether or not it succeeds in storing the frame normally in the descriptor chain.

In general, synchronization of hardware and software fails under the following conditions.

- An unread frame counter reaching its maximum value
  When the value of a counter in an unread frame counter register i (UFCVi) (i = 0 to 4) reaches 63, synchronization for the corresponding queue can fail.
  The CPU can only judge that a failure in synchronization has not occurred when the stop level is set to 63.
- A queue not having enough space for a descriptor
  In this case, the corresponding receive queue full interrupt flag (RIS2.QFFr) in receive interrupt status register 2 (RIS2) is set.
  If an unread frame counter reaches its stop level while synchronization remains normal, the receive queue full interrupt flag (RIS2.QFFr) in the receive interrupt status register 2 (RIS2) is set. Software must respond to this.
- A problem occurring during access to memory

The result of a failure in synchronization is the unread frame counter indicating that the corresponding descriptor chain may contain more available frames than it actually can. To retrieve the correct starting point for operations, use the descriptor base address load request (DLR.LBAq) for the given queue.

### 100.1.3.5 Transmission Control

Areas in the URAM for storing transmission descriptors must also be secured (for descriptors, see section 100.1.3.3 Descriptors).

The AVB-DMAC fetches data from the URAM in accord with the procedure the descriptor describes. The descriptor also retains tag information once the frame has been fetched for transmission. The tag information is used to maintain the relationships between status and time stamps for the software and the AVB-DMAC. The status and time stamp information for transmitted frames remains accessible after their transmission is completed.

#### (1) Transmission Modes

The AVB-DMAC has two modes of transmission.

- AVB transmission mode
  This mode is selected by the priority level setting for the transmission queue in the transmit configuration register (setting of the {TGC.TQP, TGC.ECBS} bits) being B'01 or B'11.
- Non-AVB transmission mode
  This mode is selected by the priority level setting for the transmission queue in the transmit configuration register (setting of the {TGC.TQP, TGC.ECBS} bits) being B'00.

**(a) AVB Transmission Mode**

AVB transmission supports the control of traffic through the output port to implement various traffic classes.

**Support for Traffic Classes and Associated Priority**

When transmission is in AVB transmission mode, streams of traffic are transmitted in accord with the part of the AVB specification called Forwarding and Queuing for Time Sensitive Streams (FQTSS; for details on this, see the IEEE 802.1Q standard).

In the AVB specification, at least one queue for a reserving stream under the Stream Reservation Protocol (SR stream) and at least one queue for a non-SR stream are present, and the queues for SR traffic is Highest priority queue.

The AVB-DMAC supports four traffic classes: SR class A, SR class B, network control (NC) traffic (gPTP frames), and best effort (BE) traffic. Allocating a specific queue to network control (NC) frames ensures the control of synchronization.

The AVB-DMAC realizes compliance with the AVB standards by handling queues with the following architecture (in terms of traffic classes).

- Four transmission queues (Q3, Q2, Q1, and Q0) are available.
- Q3 and Q2 are for SR streams (one each for class A and class B).
- Q1 is for low-bandwidth network control (NC) traffic (gPTP frames)
- Q0 is for other types of traffic (MSRPDU*1, MVRPDU*2, best effort (BE), etc.)

Notes: 1. MSRPDU: Multiple Stream Registration Protocol Data Unit
2. MVRPDU: Multiple VLAN Registration Protocol Data Unit

Fetching from queues proceeds in order of priority of the above traffic types. Three systems of priority are available through the setting of the transmit queue priority bit and enable credit based shaping bit in the transmit configuration register (TGC.TQP and TGC.ECBS). In the default priority scheme, which is called AVB mode 1 (selected by {TGC.TQP, TGC.ECBS} = B'01), operation of the AVB-DMAC is fully in accord with the AVB specification. AVB mode 2 (transmit queue priority bits {TGC.TQP, TGC.ECBS} = B'11) is an alternative priority scheme and varies from the AVB specification. Using this scheme thus requires more care. The other setting is for non-AVB-mode transmission.

**Table 100.17 Default and Alternative Priority Orders in AVB Transmission Mode**

| Priority Schemes (AVB Mode) | Priority Order of Queues |
|---|---|
| Default | Q3 (SR class A) > Q2 (SR class B) > Q1 (NC) > Q0 (BE) |
| Alternative | Q1 (NC) > Q3 (SR class A) > Q2 (SR class B) > Q0 (BE) |

**Transmission Selecting Algorithm and CBS**

The algorithm the AVB-DMAC applies to select frames for transmission is in accord with the specifications under section 8.6.8, Transmission selection, of the IEEE 802.1Q standard. For AVB mode, the CBS (credit-based shaping) algorithm is applied to the class A and class B SR queues (Q3 and Q2). Use of the CBS enables correct handling of the priorities of transmission from the SR queues. For the CBS algorithm, see section 100.1.3.6 CBS (Credit-Based Shaping).

When the following conditions both hold, transmission from an SR queue (Q3 or Q2) proceeds at the specified time.

- The queue contains at least one frame ready for transmission.
- The queue has available credit.
- Unless an SR queue satisfies the above conditions, a higher priority queue is not present (not ready for transmission).

A non-SR queue (Q1 or Q0) is selected if the conditions below both hold.

- The queue contains at least one frame ready for transmission.
- As well as the above condition, a higher priority queue is not present (not ready for transmission).

Figure 100.1.3.5.1 and Figure 100.1.3.5.2 are flowcharts of selection for transmission in AVB mode 1 (default) and AVB mode 2 (alternative).

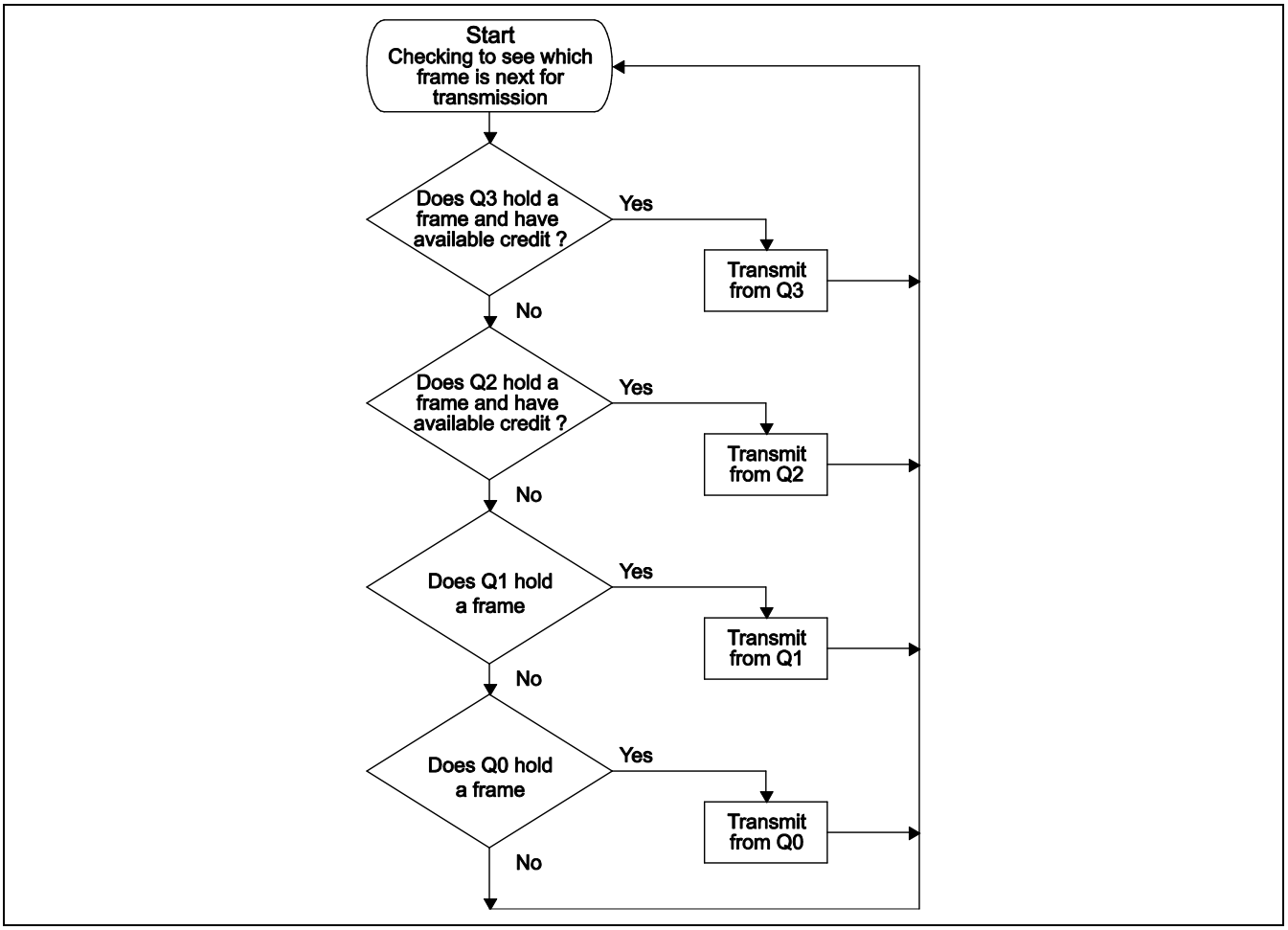



**Figure 100.1.3.5.1 Flow of Selection for Transmission in AVB Mode 1 (Default)**

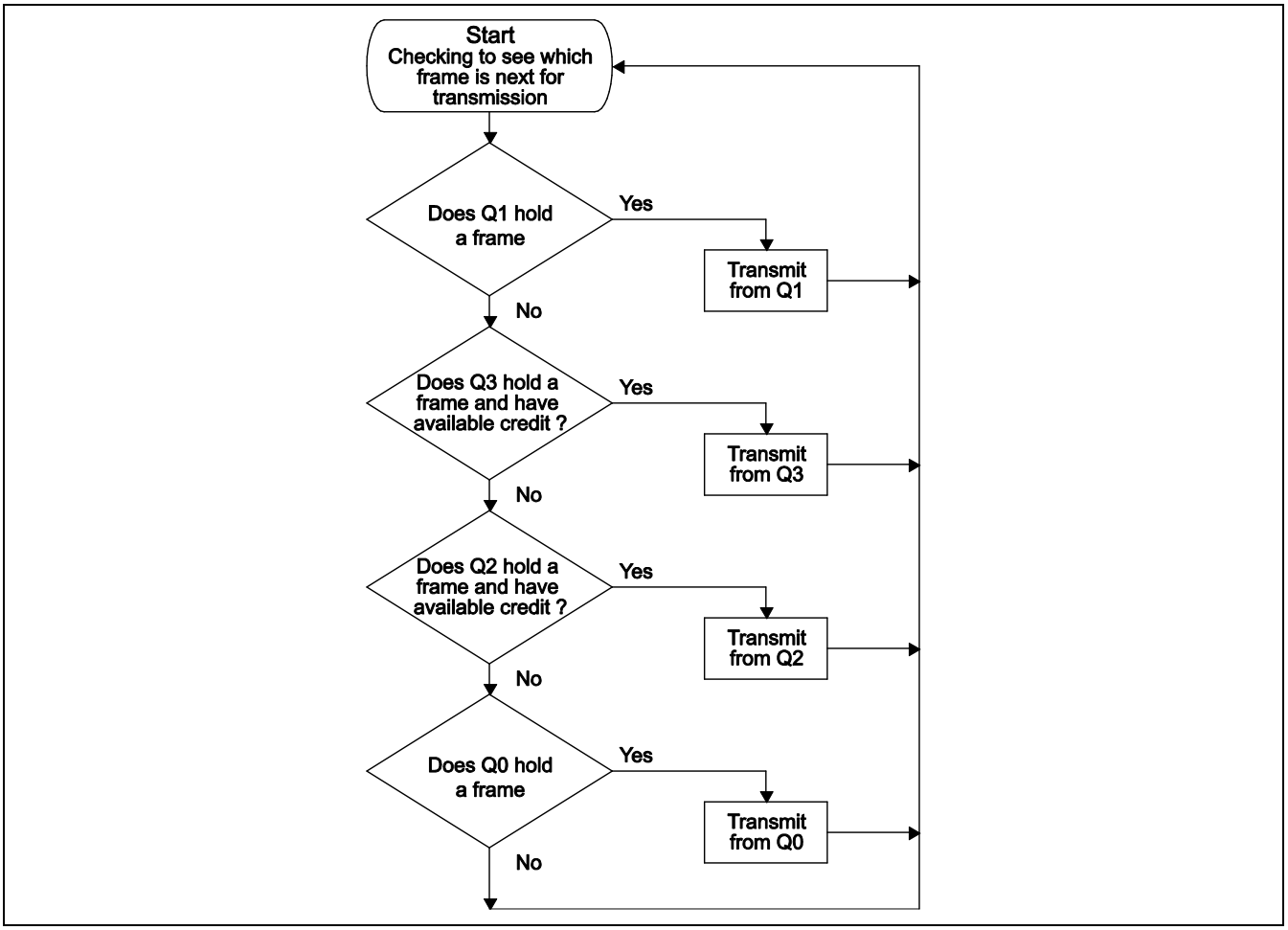



**Figure 100.1.3.5.2 Flow of Selection for Transmission in AVB Mode 2 (Alternative)**

**(b) Non-AVB Transmission Mode**

In non-AVB transmission mode, an absolute priority scheme is used. The SR class is not supported and the CBS algorithm is not used.

In non-AVB transmission mode (when the transmit queue priority bit and enable credit based shaping bit in the transmit configuration register {TGC.TQP, TGC.ECBS} are B'00), data is fetched for transmission in a strict order of priority (Q3 > Q2 > Q1 > Q0).

Figure 100.1.3.5.3 shows the flow of selection in non-AVB transmission mode.

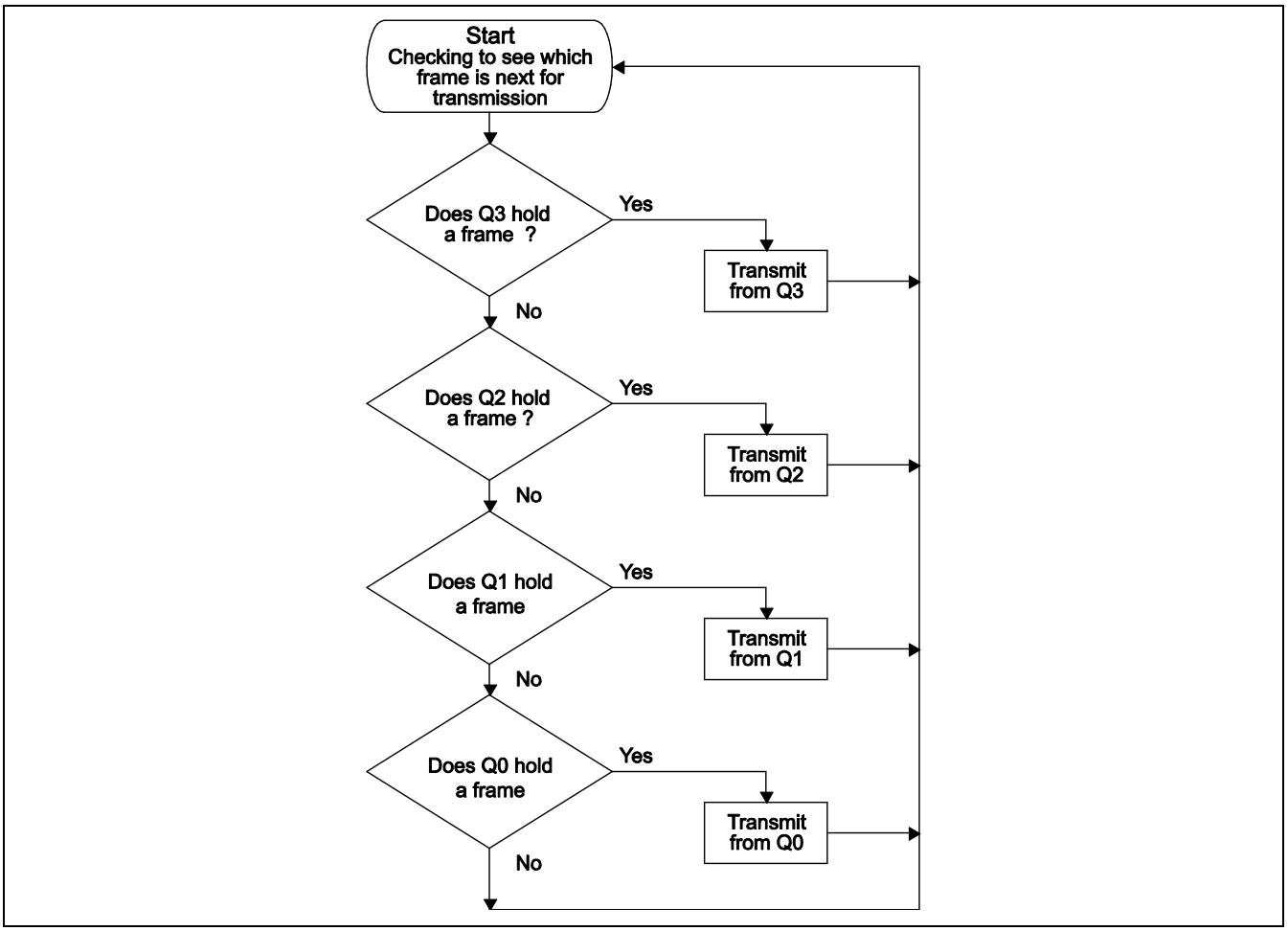



**Figure 100.1.3.5.3 Flow of Selection for Transmission in Non-AVB Mode**

**(c) Setting the Size of the Transmission FIFO**

The transmission FIFO is made up of 122 clusters. Each cluster can hold up to 128 bytes.

The size of the part of the transmission FIFO for use by each of the four transmission queues can be set by the corresponding transmit queue configuration q bits in the transmit control register (TGC.TBDq). The setting of TGC.TBDq is fixed to 2.

**General Usage Examples:**

Q0: Frames containing up to 1500 bytes → 1500/128 = 11.7 → 12 clusters
Q1: Frames containing up to 1024 bytes → 1024/128 = 8.0 → 8 clusters
Q3: Frames containing up to 1996 bytes → 1996/128 = 15.6 → 16 clusters
Q4: Frames containing up to 1996 bytes → 1996/128 = 15.6 → 16 clusters
When the depth of all transmission queues is 2, only the following number of clusters is required.
$2 \times (12 + 8 + 16 + 16) + 16 = 2 \times 52 + 16 = 120$

In the worst case (each of the transmission queues holds 1996 bytes), $2 \times 64 + 16 = 144$ clusters are required, so the maximum length of transmission frames must be less than 1996 bytes for some queues.

### (2) Setting Up Transmission Descriptors

#### (a) Transmission Descriptor Type

The type of a descriptor is defined by the descriptor type (DESCR.DT) field.

Table 100.18 shows the descriptor types used in transmission.

**Table 100.18 Descriptor Types in Transmission**

| Descriptor Type (DESCR.DT) | Operation | Write-back |
|---|---|---|
| Frame Start (FSTART) | The AVB-DMAC fetches the first of the data for the divided frame and proceeds to processing of the next descriptor. | FEMPTY |
| Frame Middle (FMID) | The AVB-DMAC fetches the second or subsequent data for the divided frame and proceeds to processing of the next descriptor. | FEMPTY |
| Frame End (FEND) | The AVB-DMAC fetches the last of the data for the divided frame.<br>The frame of data that has been fetched to the transmission FIFO is ready for transmission by the E-MAC, and then the AVB-DMAC proceeds to processing of the next descriptor. | FEMPTY |
| Frame Single (FSINGLE) | The AVB-DMAC fetches the frame of data.<br>The frame of data that has been fetched to the transmission FIFO is ready for transmission by the E-MAC, and then the AVB-DMAC proceeds to processing of the next descriptor. | FEMPTY |
| Link (LINK) | Processing proceeds to the descriptor specified by DESCR.DPTR. | LEMPTY |
| Fixed Link (LINKFIX) | Same as LINK | Not changed |
| End Of Set (EOS) | This is a transmission stop point defined by software<br>This leads to clearing of the transmit start request bit (TCCR.TSRQt), which stops transmission.<br>When the TCCR.TSRQt is again set to 1 (a new transmission start request is issued), processing proceeds to the next descriptor. | EEMPTY |
| Frame Empty (FEMPTY) | No frame data are ready for transmission<br>This leads to clearing of the transmit start request bit (TCCR.TSRQt), which stops transmission.<br>When the TCCR.TSRQt is again set to 1 (a new transmission start request is issued), processing starts at this descriptor. | Not changed |
| Link Empty (LEMPTY) | Same as FEMPTY | Not changed |
| EOS Empty (EEMPTY) | Same as FEMPTY | Not changed |

**(b) Configuration of Transmission Frame Data Descriptors**

Figure 100.1.3.5.4 shows the configuration of descriptors for use with transmission queues. The transmission-specific fields (DESCR.MSR, DESCR.TSR, and DESCR.TAG) are described in Table 100.19.

For the other fields and the descriptor types, see section 100.1.3.3(6) Descriptor Type.

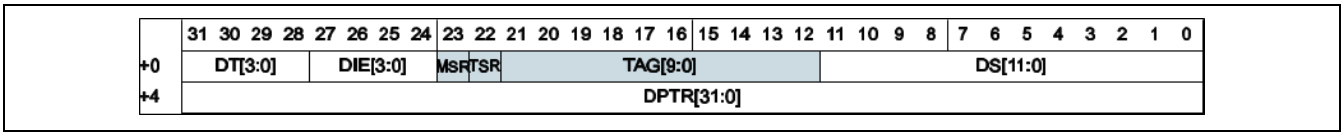



**Figure 100.1.3.5.4 Configuration of Descriptor for a Transmitted Frame**

**Table 100.19 Configuration of a Transmission Descriptor**

| Bit Name | Function |
|---|---|
| MSR | MAC Status storage Request<br>This bit configures if E-MAC transmission status of the frame is stored in E-MAC status FIFO.<br>0b: No status information is stored<br>1b: Status information is to be kept for this frame in MAC status FIFO<br>This bit is considered only if DESCR.DT = FEND or FSINGLE.<br>Note: If there is error during transmission of frame by E-MAC then status is always stored. |
| TSR | Time Stamp Store Request<br>This bit specifies whether the transmission time stamp is to be stored within the EthernetAVB-IF module.<br>0: The time stamp status FIFO within the EthernetAVB-IF module does not retain a transmission time stamp.<br>1: The time stamp status FIFO within the EthernetAVB-IF module retains a transmission time stamp.<br>This bit is available while the current DESCR.DT is FEND or FSINGLE. |
| TAG | Frame Tag<br>This TAG field is used to associate each frame data with a time stamp. Frame TAG is not required but is recommended.<br>This bit is available while the current DESCR.DT is FEND or FSINGLE. |

For the time stamp FIFO function, see section 100.1.3.5(4) Time Stamping in Transmission.

### (3) Transmission

#### (a) Transmitting Frames

Setting the transmit start request bit in the transmit configuration control register (TCCR.TSRQt) starts the transfer of frames from the corresponding transmission queue.

The descriptor in the current descriptor address (CDARq.CDA) for the queue t (with q = t) is read first.

If this descriptor is a descriptor for frame transmission (FSINGLE, etc.), the AVB-DMAC fetches the frame data from the data area indicated by the descriptor, writes FEMPTY back to the descriptor type (DESCR.DT) bits to indicate completion of this processing, then proceeds to processing of the next descriptor.

If the descriptor is not for transmission, processing is as dictated by the given descriptor (for these descriptors, see the descriptions in section 100.1.3.3 Descriptors).

If a base address load request is issued for a descriptor chain while it is being processed (by setting 1 in the LBAq bit for transmission queue q that is currently being processed in the descriptor base address load request register, DLR), processing proceeds to the new descriptor chain. Changing the chain does not interrupt frame fetching, but note that frames that have not been fetched from the old chain remain where they are.

Figure 100.1.3.5.5 shows descriptor processing during transmission.

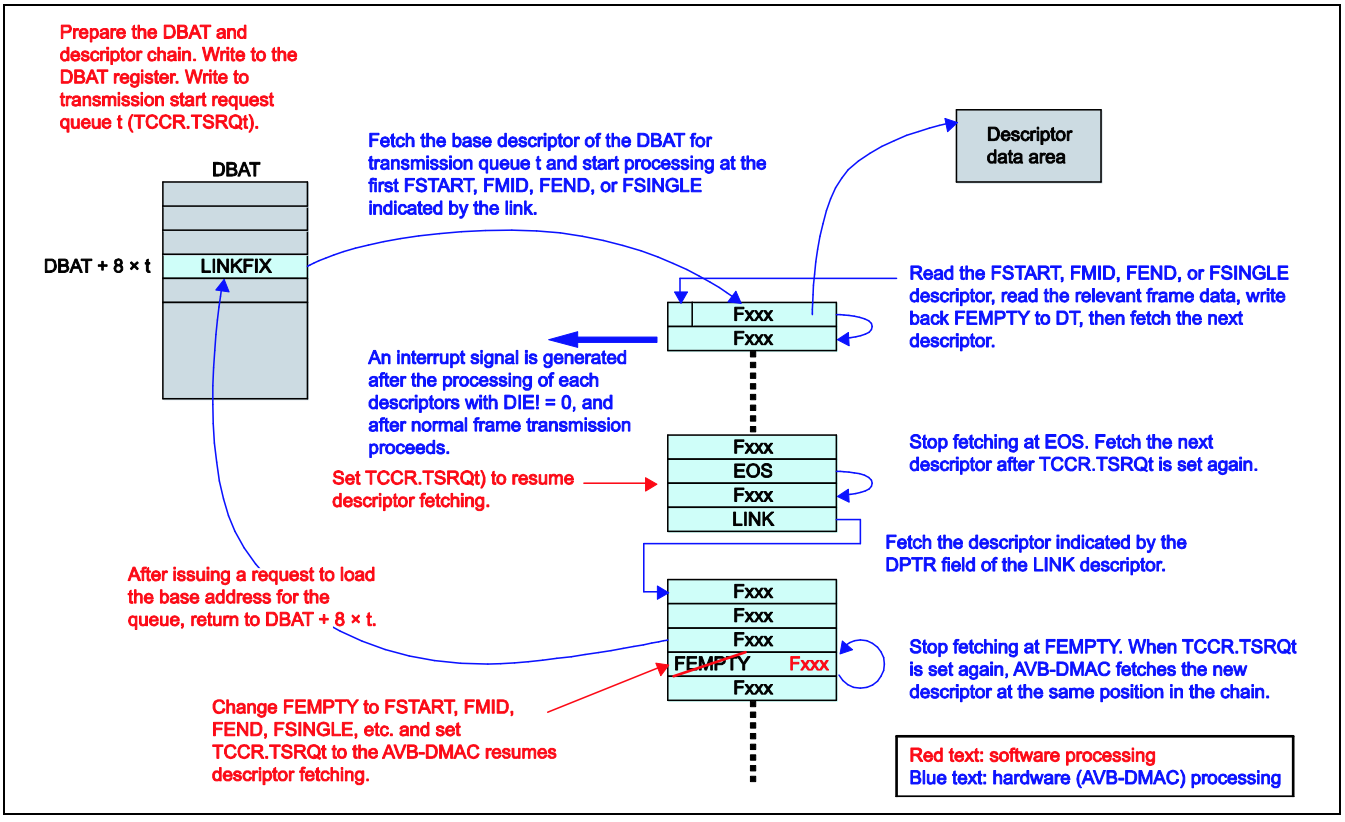



**Figure 100.1.3.5.5 Descriptor Processing During Transmission**

**(b) Examples of Descriptor Usage**

**Immediate Frame Transmission**

Immediate frame transmission is a pattern in which fetching by the AVB-DMAC starts whenever software adds data to a queue. FEMPTY descriptors are used as stop points to keep the hardware and software in synchronization.

Create descriptor chains that have FEMTPY descriptors at the stop points.

Figure 100.1.3.5.6 shows the flow for software implementing this pattern. SW should read back written descriptor between changing FEMPTY descriptor before writing to TCCR.TSRQt.

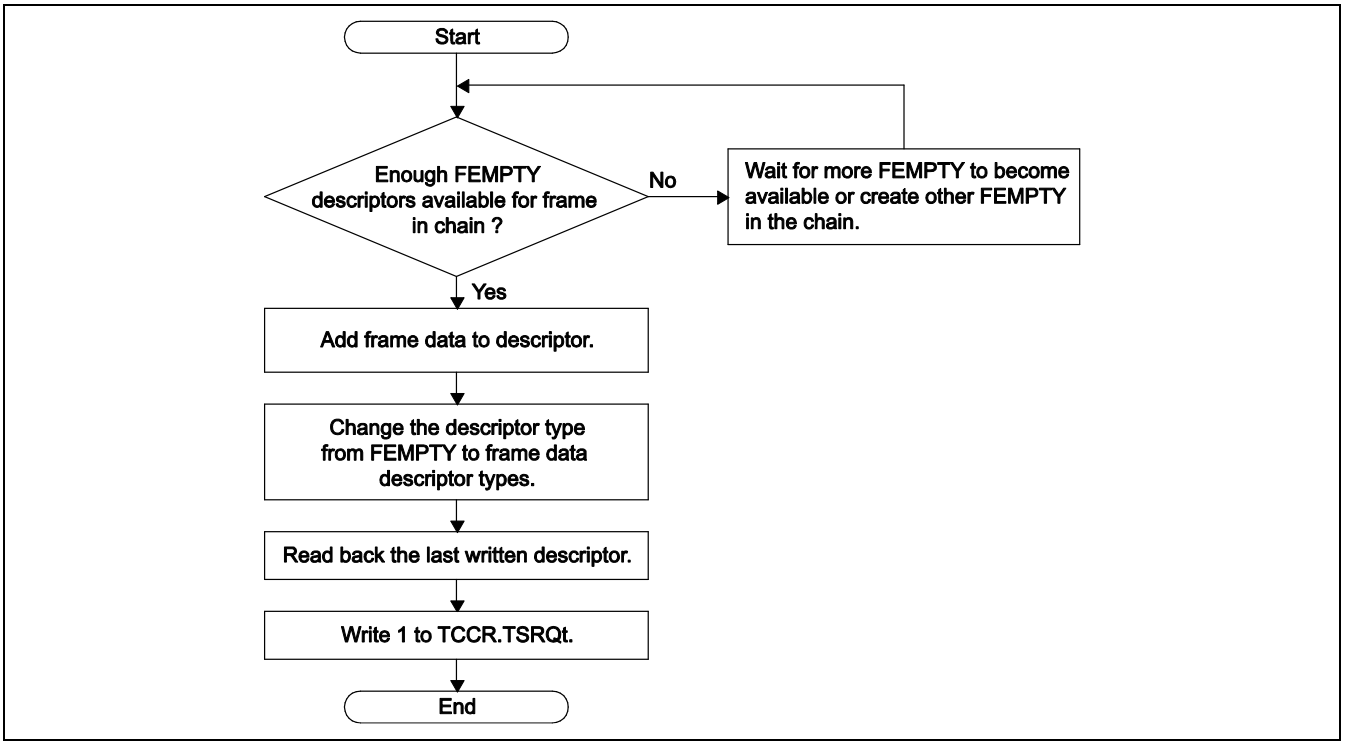



**Figure 100.1.3.5.6 Software Flow for Immediate Frame Transmission**

Figure 100.1.3.5.7 shows software and AVB-DMAC operations for immediate frame transmission.

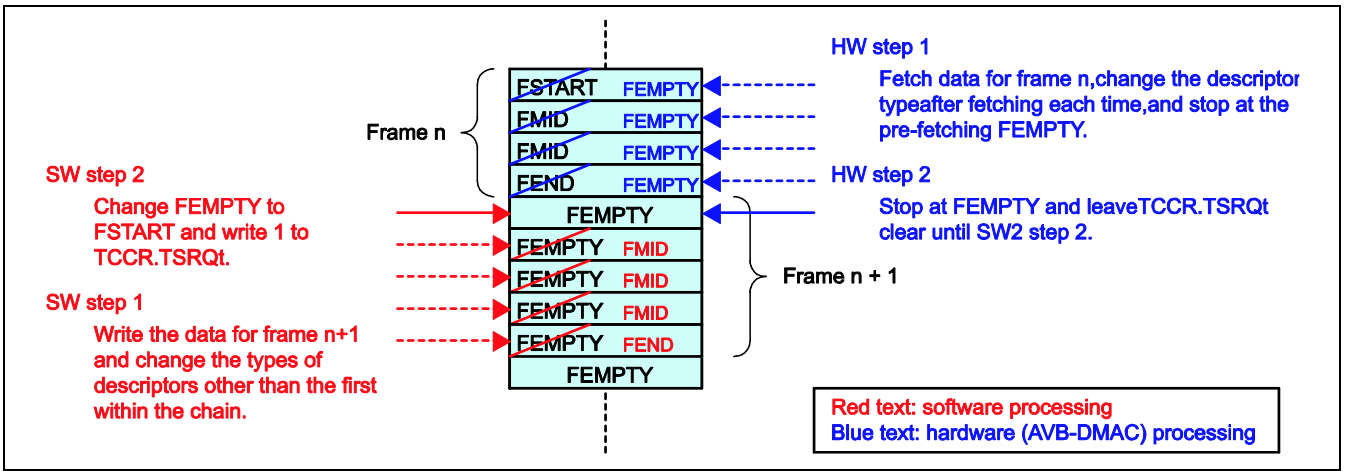



**Figure 100.1.3.5.7 Software and AVB-DMAC Operations for Immediate Frame Transmission**

**Frame Set Transmission with Changing of the Active Descriptor Chain**

This pattern is used when data are transmitted with a delay for software control to secure bandwidth or for other reasons, rather than immediately transmitted. EOS descriptors are used for the stop points. Start by creating a descriptor chain that has a FEMPTY descriptor as its stop point.

Figure 100.1.3.5.8 shows the software flow in this pattern.

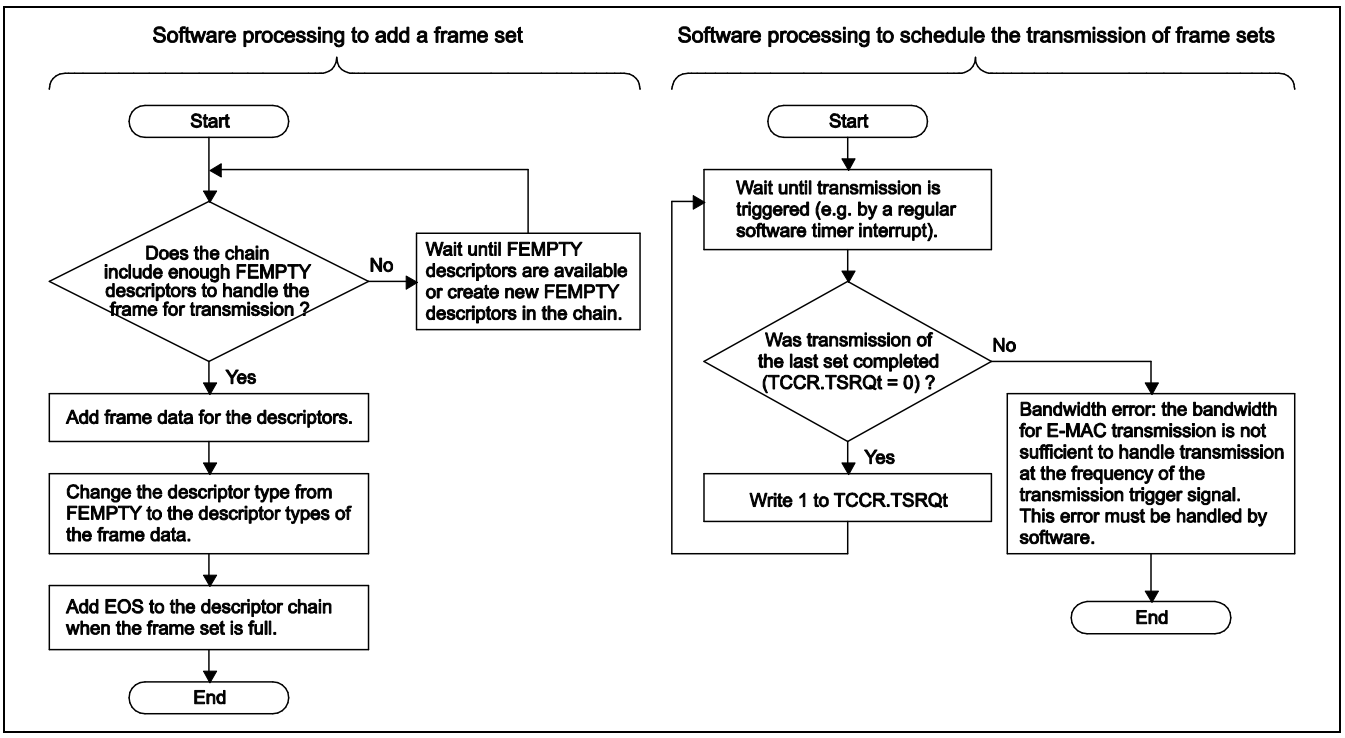



**Figure 100.1.3.5.8 Software Flow for Frame Set Transmission with Changing of the Active Descriptor Chain**

Figure 100.1.3.5.9 shows software and AVB-DMAC operations for frame set transmission.

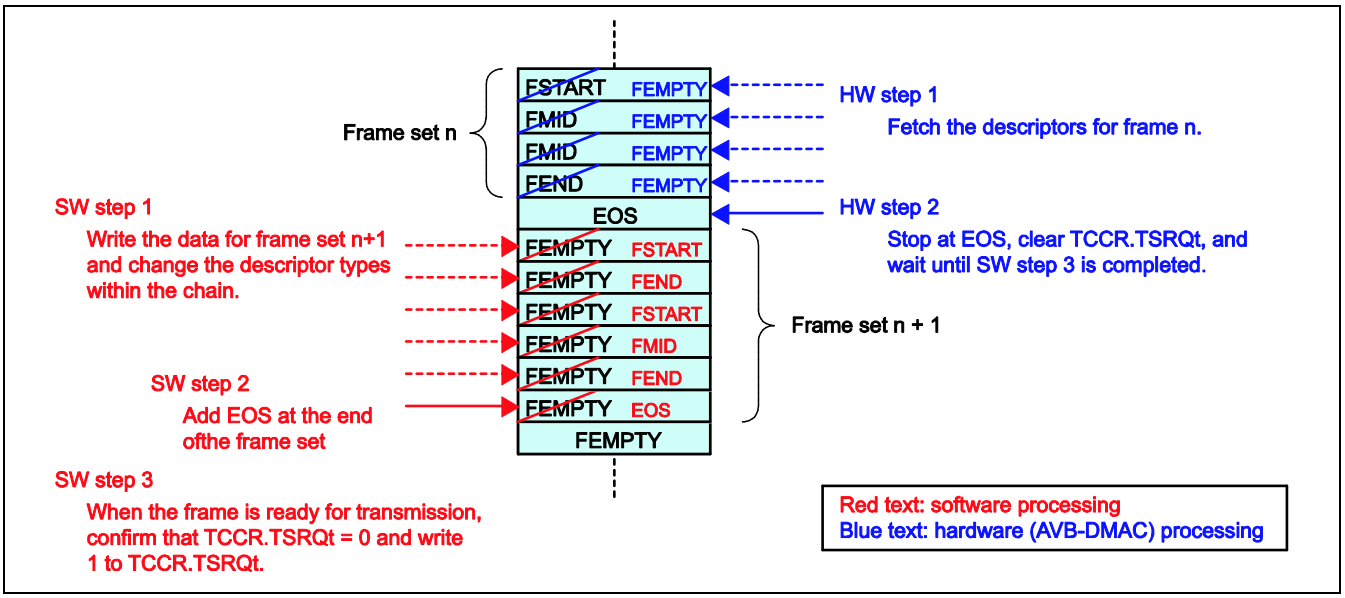



**Figure 100.1.3.5.9 SW and AVB-DMAC Operations for Frame Set Transmission with Changing of the Active Descriptor Chain**

**Frame Set Transmission Using a Shadow Descriptor Chain**

This pattern is used when data are transmitted with a delay for software control to secure bandwidth or for other reasons, rather than immediately transmitted. Two or more descriptor chains are used. The chains are classified as active or shadow chains. EOS descriptors are used for the stop points.

Create descriptor chains that have FEMPTY descriptors at the stop points.

Figure 100.1.3.5.10 shows the flow for software implementing this pattern.

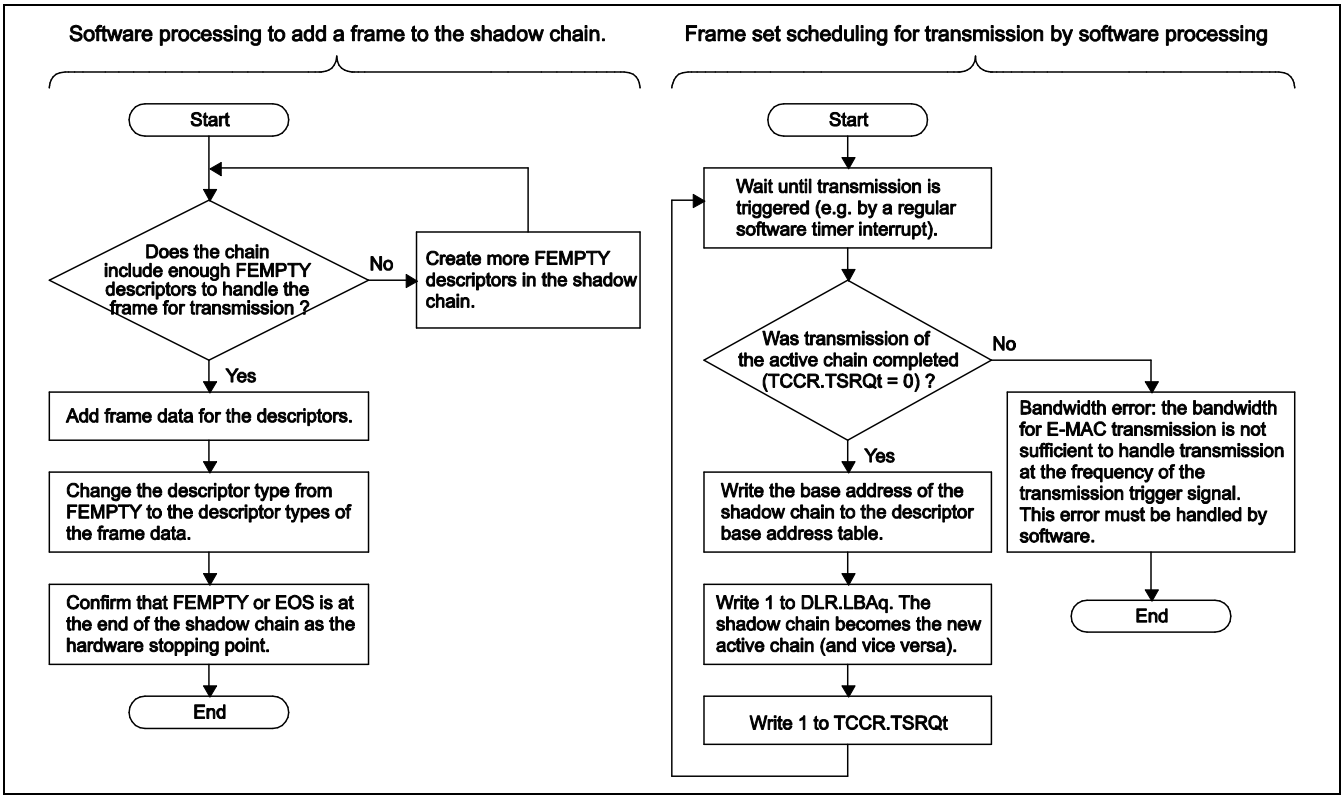



**Figure 100.1.3.5.10 Software Flow for Frame Set Transmission Using the Shadow Descriptor Chain**

Figure 100.1.3.5.11 shows software and AVB-DMAC operations for frame set transmission.

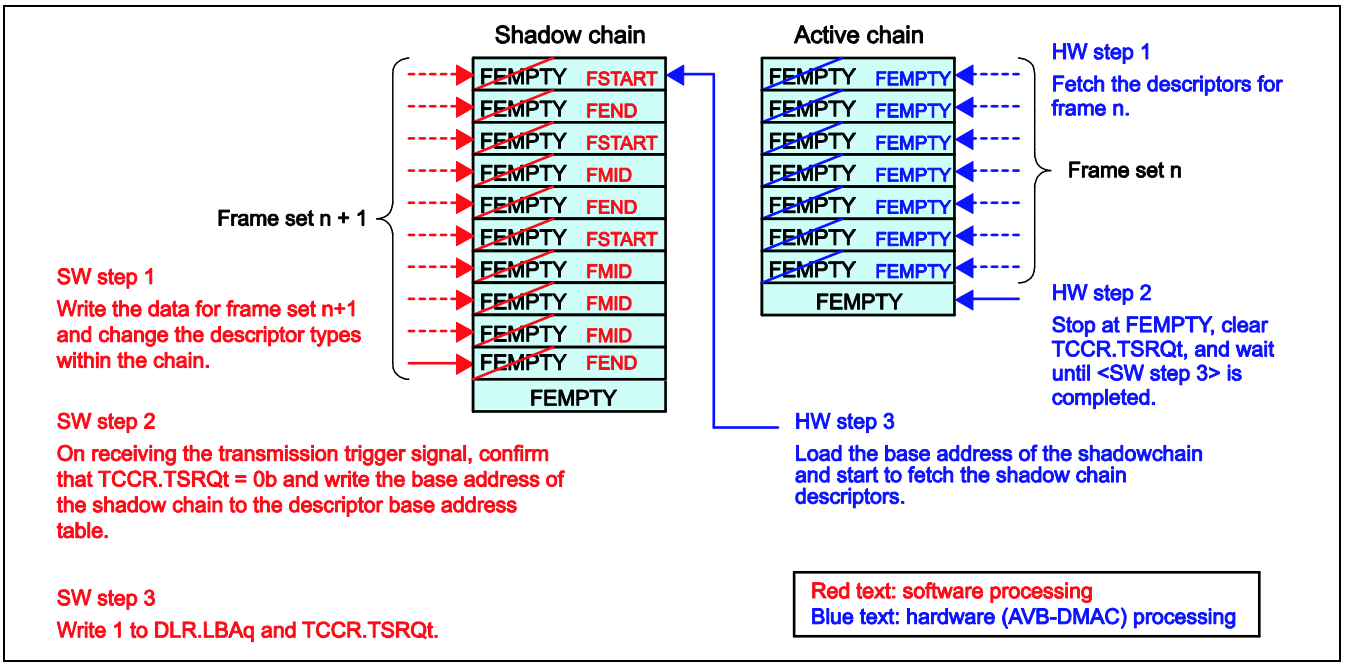



**Figure 100.1.3.5.11 SW and AVB-DMAC Operations for Frame Set Transmission Using the Shadow Descriptor Chain**

### (4) Time Stamping in Transmission

Transmission time stamps are important in satisfying the requirements for timing and synchronization of the IEEE 802.1AS standard. Reference to this information can also be useful to other applications and in testing. The AVB-DMAC supports the storage of time stamps for transmitted frames. The time-stamp values are based on the gPTP timer and are captured at the same time as sending of the Start of Frame Delimiter (SFD) for transmitted frames.

When the time stamp storage request field (DESCR.TSR) is set to 1, selecting storage of a time stamp, the tag number defined in the tag field (DESCR.TAG) of the last descriptor (FEND or FSINGLE) for the frame being transmitted is stored with the time stamp. This eases identification and association. The time-stamp FIFO is accessible at any time.

Figure 100.1.3.5.12 shows the mechanism supporting transmission time stamping.

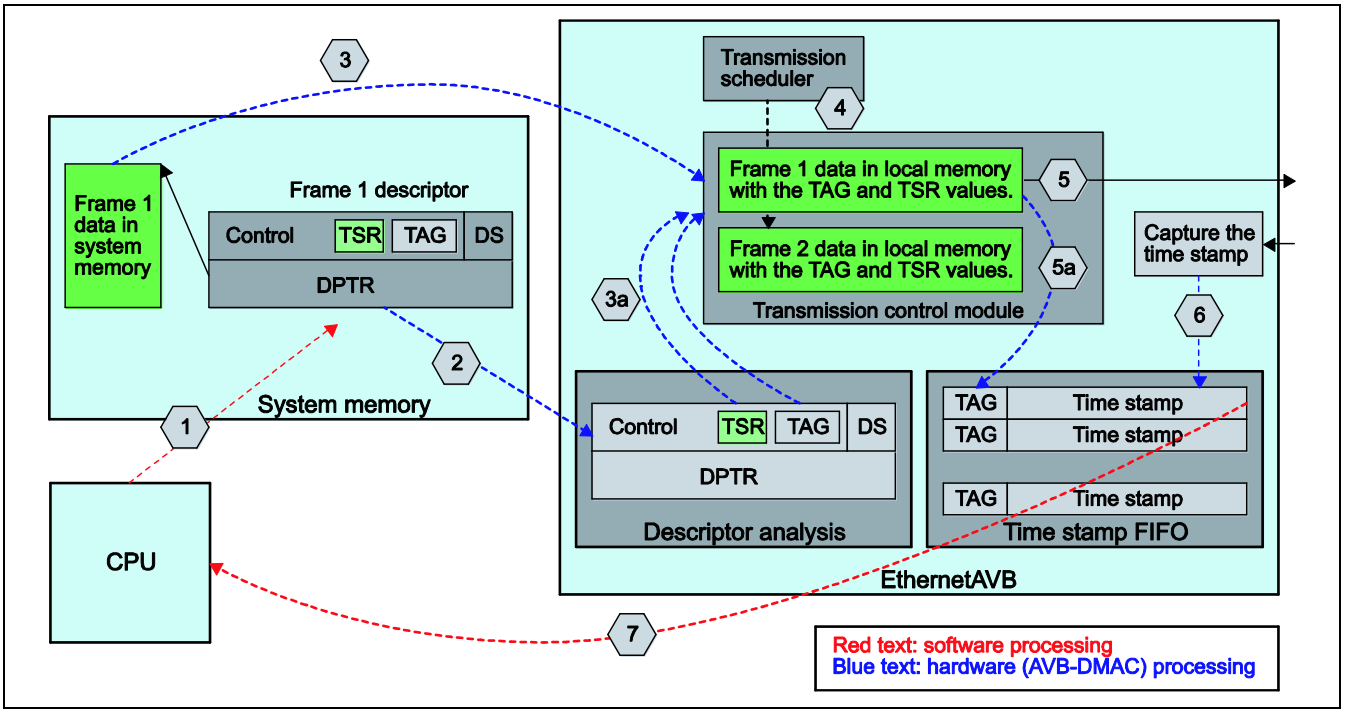



**Figure 100.1.3.5.12 Mechanism to Support Transmission Time Stamps**

The method of using this function is described below:

1. Secure space in the URAM for the frame requiring time stamping.
   Write the tag number of the frame to the frame tag field (DESCR.TAG) and set the time stamp storage request field (DESCR.TSR) to 1.
2. The AVB-DMAC fetches and analyzes the descriptor. The time stamp storage request field (DESCR.TSR) is 1, so it recognizes that transmitting this frame also requires storage of the time stamp.
3. The AVB-DMAC fetches the data for frame 1 and temporarily stores the frame in internal memory for scheduling.
   3a: The frame tag field (DESCR.TAG) and time stamp storage request field (DESCR.TSR) are stored with the fetched data.)
4. Under the control of priority settings according to credit-based shaping (CBS) or another scheme, the transmission scheduler decides it is time to transmit frame 1.
5. Transmission of frame 1 starts.
   5a: Frame 1's tag is stored in the time-stamp FIFO.
6. The gPTP time stamp is captured at the start of frame delimiter (SFD) for transmission and stored with the tag in the time-stamp FIFO when the frame is completely transmitted. An interrupt is generated to indicate existence of new timestamp information. For this to happen, the descriptor interrupt control register (DIC) must be set beforehand.
7. The entry can now be read from the time-stamp FIFO.

Use the time-stamp FIFO for the timing and synchronization of frames with IEEE 802.1AS compliance.

Time stamping can also be used with other frames, but take care not to allow the time-stamp FIFO to overflow. When the FIFO is full, further time stamps supplied to it are lost.

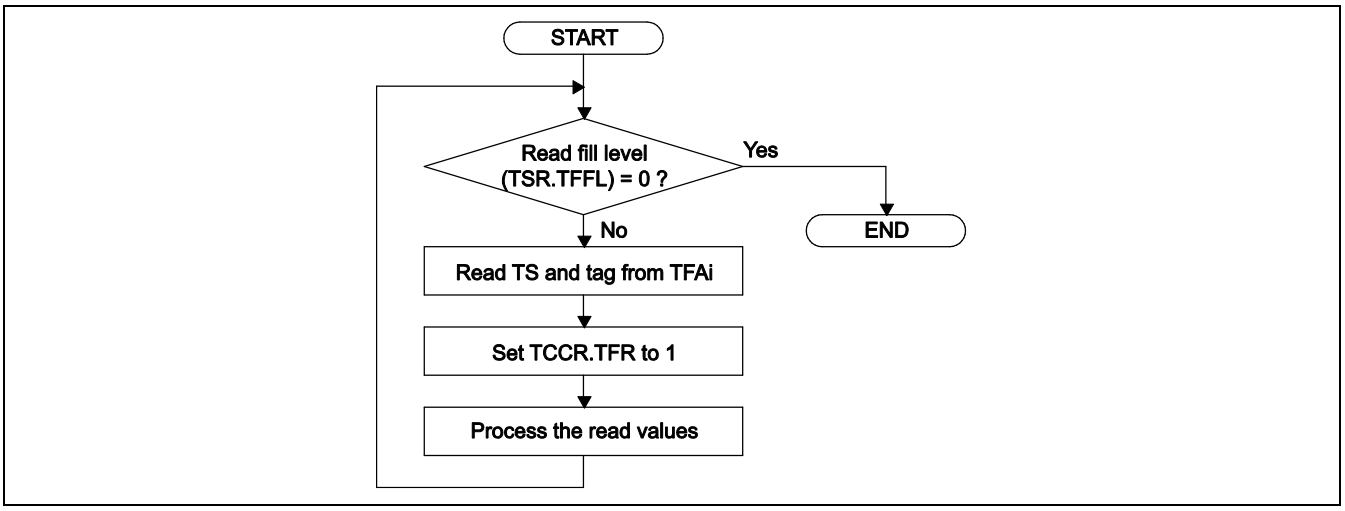



**Figure 100.1.3.5.13 Flow of Transmission Time Stamping**

### (5) Ending Transmission

Figure 100.1.3.5.14 shows the procedure for ending transmission.

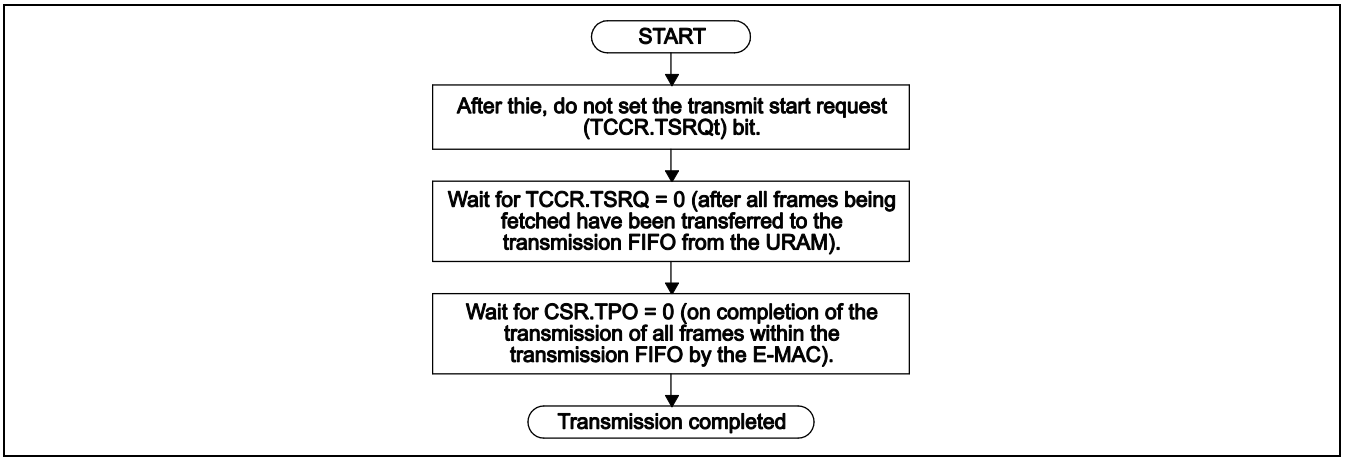



**Figure 100.1.3.5.14 Procedures for Ending Transmission**

### 100.1.3.6 CBS (Credit-Based Shaping)

In AVB transmission mode (i.e. when the transmit queue priority bit and enable credit based shaping bit in the transmit configuration register {TGC.TQP, TGC.ECBS} are B'01 or B'11), transmission queues Q3 and Q2 are respectively assigned to class A and class B stream traffic and the CBS (Credit Based Shaping) algorithm is used to select the transmission queues in order to satisfy the Forwarding and Queuing for Time Sensitive Streams (FQTSS) specification (see section 8.6.8 or section 34 in IEEE 802.1Q).

The CBS algorithm is based on the concept of transmission credit for each queue. Credit can be thought of as the degree to which a queue has the “right” to transmit at a given time. Actually, in AVB transmission mode as specified in IEEE 802.1Q, queues that are subject to the CBS algorithm are able to transmit when the following conditions are met.

At least one frame is stored in the queue.

The credit for the queue is 0 or a positive value.

The credit for a transmission queue is incremented while one or more frames from the queue are present in the transmission FIFO but transmission of these frames is not proceeding. This state is indicated by the transmission process status bit for queue t in the AVB-DMAC status register (CSR.TPOt). The credit is decremented while transmission of a frame from the queue is in progress. This mechanism is used to control transmission so that the transmission of frames from the queues for each of the traffic classes does not exceed the specified maximum bandwidths.

IEEE 802.1Q defines the following parameters for queues under the control of the CBS algorithm.

**portTransmitRate:** Maximum transmission data rate of an external port. The E-MAC determines this parameter.

**bandwidthFraction:** Maximum fraction of portTransmitRate that can be used for a queue.

**idleSlope**: Rate of change of credit for a queue when transmission of frames from the queue is not proceeding so the credit value (in bits per second) is increasing. idleSlope is also equal to the maximum fraction of the total bandwidth (portTransmitRate) that is available to the given queue under a specified condition (frames from the queue can be placed in a continuous stream. See Annex L of IEEE 802.Q.
idleSlope = bandwidthFraction × portTransmitRate

**sendSlope**: Rate of change of credit for a queue while transmission of a frame from the queue is in progress so the credit value is decreasing.
The value of sendSlope is defined as follows:
sendSlope = idleSlope – portTransmitRate

Furthermore, the values below are used to define individual traffic classes (or queues for the classes) under control of the algorithm. See Annex L of IEEE 802.Q.

**maxFrameSize:** Maximum size of frames (in bits) of the corresponding traffic class that can be transmitted from a port

**maxInterferenceSize:** Maximum burst size (in bits) by which delays for the corresponding traffic class can be allowed

**hiCredit:** Maximum credit value (positive number). Can be calculated by using the following equation: hiCredit = maxInterferenceSize × (idleSlope / portTransmitRate)

**loCredit:** Minimum credit value (negative number). Can be calculated by using the following equation:
loCredit = maxFrameSize × (sendSlope / portTransmitRate)

Figure 100.1.3.6.1 shows how the CBS algorithm works and the meaning of the above parameters.

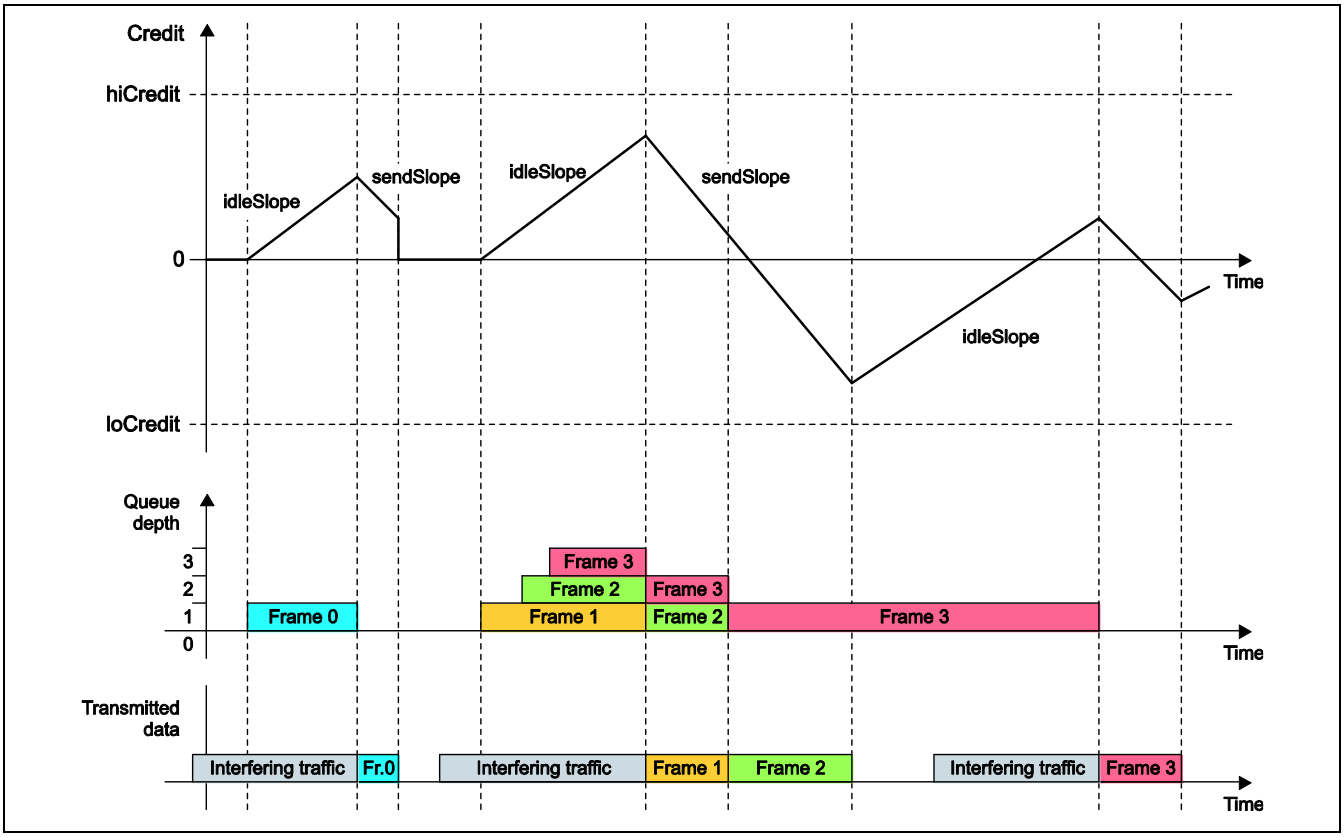



**Figure 100.1.3.6.1 CBS (Credit-Based Shaping) Operation**

Figure 100.1.3.6.2 shows the implementation of CBS in the AVB-DMAC.

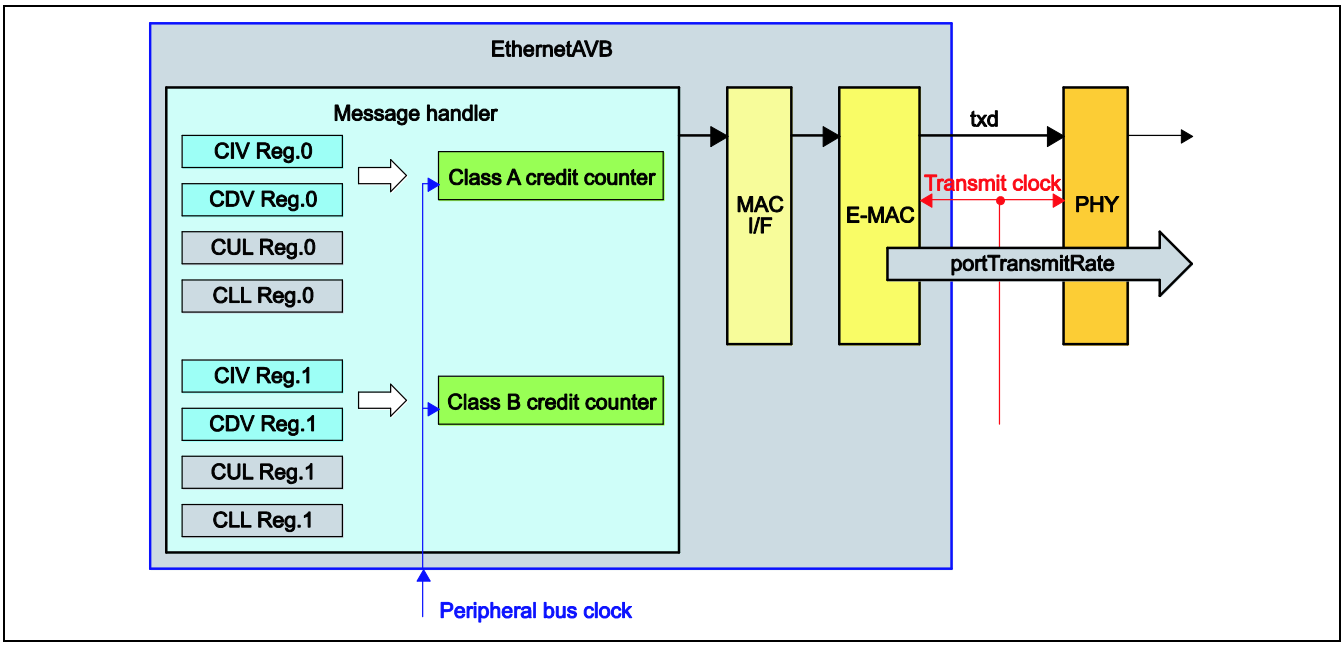



**Figure 100.1.3.6.2 CBS (Credit-Based Shaping) Operation in the AVB-DMAC**

The above implementation is based on “credit counters” for the respective traffic classes (SR class A and class B). The following parameters apply for these classes.

CBS increment value (CIV): Signed positive number
The credit is incremented by this amount every high-speed peripheral bus clock cycle while a frame from the queue is pending but transmission has not started (idleSlope).

CBS decrement value (CDV): Signed negative number
The credit is decremented by this amount every high-speed peripheral bus clock cycle while transmission of a frame from the queue is proceeding (sendSlope).

The CBS increment value (CIV) and CBS decrement value (CDV) are defined as follows.

CIV = idleSlope × Mfactor
CDV = sendSlope × Mfactor

Mfactor is a multiplier factor to ensure accuracy for CIV and CDV. CIV and CDV are calculated by using the following equations.

CIV = (portTransmitRate/HPϕ) × bwFraction × Mfactor
CDV = (portTransmitRate/HPϕ) × (bwFraction – 1) × Mfactor

HPϕ is the frequency of the high-speed peripheral bus clock. The credit counters are driven by the high-speed peripheral bus clock, so calculating the slope parameters for CBS requires (1/HPϕ).

Use software to prepare Mfactor for the CBS parameters. All queues for the same class must have the same Mfactor for all CBS parameters. Mfactor for a specified class c can be changed during operation, unless transmission is pending for that class (i.e. the transmit process status bit in the AVB-DMAC status register (CSR.TPOt) = 0). At that time, the credit counter values for class A and class B are 0. Note that the credit value will not match a new incrementation or decrementation parameter if Mfactor is changed while the credit counter value is non-zero. Mfactor is not present in the AVB-DMAC registers.

Set the CIV and CDV parameters in the CBS increment value registers (CIVRc) and the CBS decrement value registers (CDVRc). These are treated as dynamic settings since they should be updated when streams are registered and erased in accord with IEEE 802.1Qat.

The AVB-DMAC also has CBS upper limit registers (CULc) (the upper limit registers for classes A and B) and CBS lower limit registers (CLLc) (the lower limit registers for classes A and B). Set Mfactor to match the credit value and set the upper limit (hiCredit) and the lower limit (loCredit) for each class as defined above.

CUL = hiCredit × Mfactor = maxInterferenceSize × bwFraction × Mfactor
CLL = loCredit × Mfactor = maxFrameSize × (bwFraction – 1) × Mfactor

**Example:**

Assume that portTransmitRate = 100 Mbps, HPϕ = 133.33 MHz and bwFraction = 3%. Then idleSlope and sendSlope represented as one bit vs. cycles of the high-speed peripheral bus clock are as follows.

idleSlope = (portTransmitRate/HPϕ) × bwFraction = 100/133.33 (Mbps/MHz) × 3% = 0.023 bit per high-speed peripheral bus clock cycle
sendSlope = idleSlope – (portTransmitRate / HPϕ) = –0.727 bits per high-speed peripheral bus clock cycle
Let Mfactor be 100, then CIV and CDV parameters are determined as follows.

CIV = idleSlope × Mfactor = 2.3
CDV = sendSlope × Mfactor = –72.7

Note : 133MHz is calculation example. Need to replace with S0D8_HSC $\phi$ and calculate.

### (1) Restrictions on CIV, CDV and Mfactor

The maximum value (the minimum value for negative numbers) up to which the credit counter will not overflow determines the maximum values of CIV and CDV that can be set in the CBS registers. This maximum credit value is equivalent to the worst case of the hiCredit value, and the maximum values for class A and class B are calculated as follows.

**<Conditions>**

- Class A maximum value (hiCredit_max_classA)
  classA bwFraction ≅ 100%
  Maintaining the proper relations in the transmission priority order requires waiting for a period equivalent to the maximum frame size.
  hiCredit_max_classA ≅ maxInterferenceSize for class A = Interference due to one max. sized frame = header + max. size payload + CRC (2000 bytes) + preamble (8 bytes) + IFG (12 bytes) + processing_delay (≅80 bytes) ≅ 2100 bytes
- Class B maximum value (hiCredit_max_classB)
  classB bwFraction ≅ 100%
  Maintaining the proper relations in the transmission priority order requires waiting for a period equivalent to the maximum size of frames in the class A transmission queue and other transmission queues.
  hiCredit_max_classB ≅ maxInterferenceSize for class B = Interference due to two max-size frames = 2 × hiCredit_max_classA ≅ 4200 bytes

hiCredit_max_classA = 16800

hiCredit_max_classB = 33600

The maximum values that can be selected with Mfactor for the 32-bit signed counter without overflow are:

Mfactor_max_classA = $2^{31}$–1 / hiCredit_max_classA ≅ 127826 and
Mfactor_max_classB = $2^{31}$–1 / hiCredit_max_classB ≅ 63913.

A high degree of accuracy can be achieved even with a low bandwidth. In class B, bandwidthFraction = 0.05% and the bandwidth error < 0.1%.

The maximum value of CIV is calculated from the following equation.

CIV = idleSlope x Mfactor = (portTransmitRate / HPϕ) × bandwidthFraction × Mfactor

When Mfactor is the maximum value and bandwidthFraction is the maximum value (up to 100%):

CIV_max_classA = (portTransmitRate / HPϕ) × Mfactor_max_classA and
CIV_max_classB = (portTransmitRate / HPϕ) × Mfactor_max_classB.

The maximum values when portTransmitRate = 1000 Mbps and HPϕ = 133.33 MHz are as follows:

CIV_max_classA ≅ 958719

CIV_max_classB ≅ 479359

These values in the table are the limits of CIV up to which the 32-bit credit counter will not overflow. The CIV parameters are implemented as 16 bits + a sign bit, so a further limit of CIV ≤ 65535 applies to both class A and class B.

Note : 133MHz is calculation example. Need to replace with S0D8_HSCφ and calculate.

### (2) Credit Incrementation during Inter-Frame Gaps (IFGs)

The inter-frame gap (IFG) after a frame is transmitted is not treated as part of frame transmission by the CBS credit counter. During an IFG, the credit is incremented for all SR queues that have pending frames or negative credit. Figure 100.1.3.6.3 illustrates credit operations during IFGs.

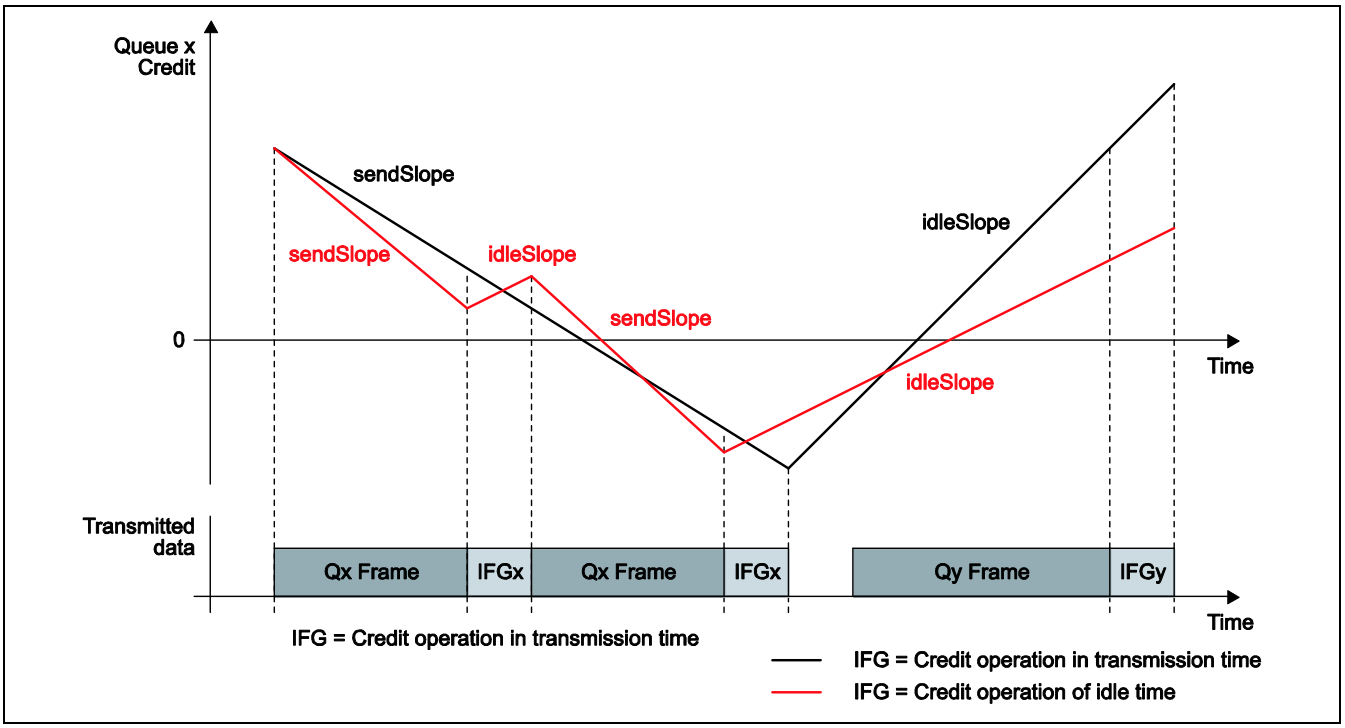



**Figure 100.1.3.6.3 Credit Operations during IFGs**

Accordingly, the IFG need not be included in calculation of the bandwidth requirements for the specified SR class when deciding the idleSlope, sendSlope, and CIV and CDV parameters. However, IFG must also be included in the calculation in order to confirm that the total bandwidth allocated to all SR classes does not exceed 100% of portTransmitRate. This is described in *Section 35.2.2.8.4* of *IEEE 802.1Q*.

### (3) Example

The case of a class A 48-kHz stereo audio stream among Ethernet frames is described as an example.

After every class A measurement interval (125 µs), 80 octets consisting of two sets of six 32-bit samples plus a 32-octet header are stored as audio data within a frame. The IEEE 802.3 also imposes a 42-octet media-specific framing overhead (an 8-octet preamble, 14-octet IEEE 802.3 header, 4-octet IEEE 802.1Q priority/VID Tag, 4-octet CRC, and 12-octet IFG) are also added. Accordingly, the total frame size is 80 + 42 = 122, and one such frame is transmitted after every class measurement interval.

This represents a total bandwidth of about 7.8 Mbits per second (122 octets × 8 bits per octet × 8000 frames per second) for this class. If the E-MAC is assumed to run at 1 Gbps (portTransmitRate), this is equivalent to the allocation of about 0.78% of the total bandwidth to each class A queue. If other traffic classes are to share the total transmission bandwidth, checking that this 0.78% allocation does not lead to the total allocation of bandwidth being greater than 100% of portTransmitRate is required.

To obtain the CIV and CDV parameters for a given class, the IFG must not be taken into account in calculation of the frame size must not include the IFG. For this case, therefore, we obtain an 80-bit payload + 30-bit overhead = 110-octet measurement interval for the class → the total bandwidth for the class = 7.04 Mbps = 0.704% of portTransmitRate.

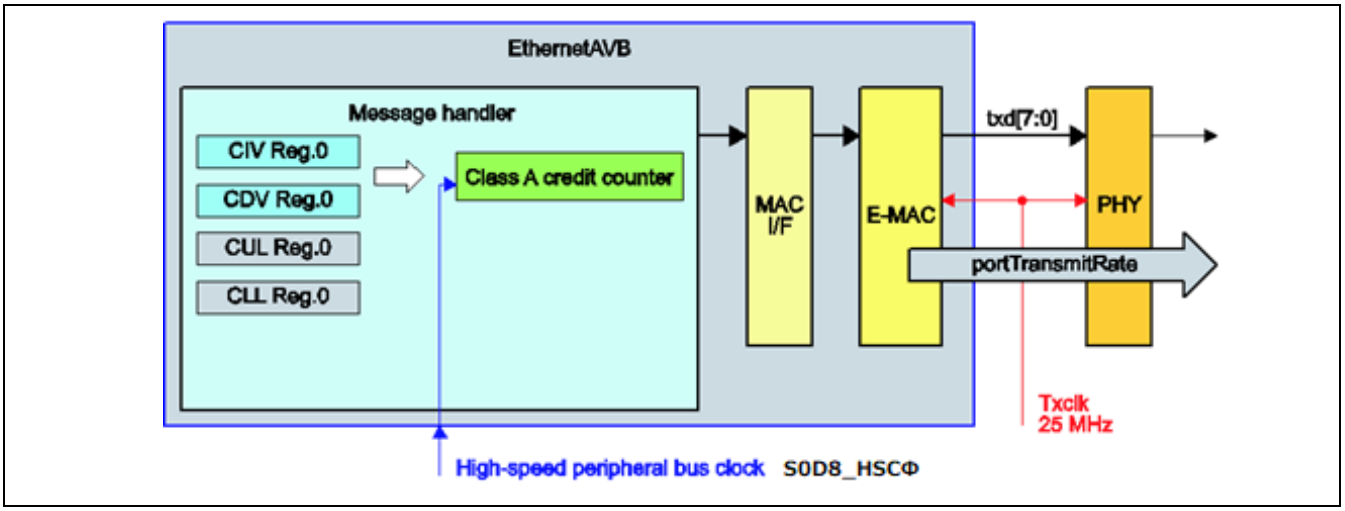



**Figure 100.1.3.6.4 Example of CBS Settings**

Given that:

- the E-MAC runs at 1000 Mbps, so portTransmitRate = 1000 Mbps and
- high-speed peripheral bus clock (operating clock for the credit counter) frequency = 133.33 MHz, securing a bandwidth of 7.04 Mbits/sec for class A requires configuring the CBS parameters as follows.
  bandwidthFraction = 0.704%
  idleSlope = (portTransmitRate / HPϕ) × bandwidthFraction ≅ 0.0528 bits per high-speed peripheral bus clock cycle
  sendSlope = idleSlope – (portTransmitRate / HPϕ) ≅ –7.44739 bits per high-speed peripheral bus clock cycle

When Mfactor = 100000, the parameters are as follows.

- CIV = idleSlope × Mfactor = 5280 bits per high-speed peripheral bus clock cycle
- CDV = sendSlope × Mfactor = 744739bits per high-speed peripheral bus clock cycle

These are the final values for setting in the CIVR1 and CDVR1 registers.

Note : 133MHz is calculation example. Need to replace with S0D8_HSCφ and calculate.

### 100.1.3.7 Flow Control

The EthernetAVB-IF does not support flow control in the specification. Flow control is only effective when transmitting and receiving only normal (non-AVB) packets.

The E-MAC supports flow control for full-duplex operation in compliance with the IEEE 802.3 standards. This flow control is applicable to both reception and transmission. In regard to the transmission of PAUSE frames, flow control operates in the following ways.

(1) PAUSE Frame Transmission

PAUSE frames can also be transmitted in response to software operations. Writing a timer value to the manual PAUSE frame register (MPR) starts the transmission of a PAUSE frame. This only causes the transmission of one PAUSE frame.

(2) PAUSE Frame Reception

After reception of a PAUSE frame, transmission of the next frame does not proceed until the time indicated by the Timer value elapses. However, transmission of a frame currently being transmitted continues. PAUSE frames are only received while the RXF bit in the E-MAC mode register (ECMR) is set to 1. The number of received PAUSE frames is counted.

(3) PAUSE Frames with the Timer Value 0

The setting of the 0-time PAUSE frame enable bit (ECMR.ZPF) enables or disables the reception of PAUSE frames with the TIME parameter value 0.

- When control of PAUSE frames with the TIME parameter value 0 is disabled

  PAUSE frames with the TIME parameter value 0 are not transmitted. Received PAUSE frames with the TIME parameter value 0 are discarded.

### 100.1.3.8 Magic Packet Detection

The E-MAC has a Magic Packet detection function. This function provides a facility for host devices and other sources to start other peripheral devices connected to a LAN. A peripheral device that handles Magic Packets starts itself in response to receiving a Magic Packet.

When a Magic Packet is detected, data from broadcast packets that were previously being received are stored in the FIFO and the E-MAC is notified of the receiving status. To return to normal operation from the associated interrupt processing, the E-MAC and AVB-DMAC must be initialized by using the operating mode configuration bit in the AVB-DMAC mode register (CCC.OPC) to set the operating mode to reset mode.

Magic Packets are received regardless of the destination address. As a result, the AVB_MAGIC pin is only enabled in the case of a match with the destination address specified in the Magic Packet.

The procedure for using the Magic Packet detection function with this LSI chip is as follows.

1. Use the various interrupt enabling and masking registers to disable the output of interrupts from interrupt sources.
2. Set the Magic Packet detection enable bit in the E-MAC mode register (ECMR.MPDE).
3. Set the Magic Packet detection interrupt enable bit in the E-MAC interrupt enable register (ECSIPR.MPDIP) to enable the interrupt.
4. Place the CPU in sleep mode as required.
5. An interrupt is conveyed to the CPU on detection of a Magic Packet.
6. The AVB_MAGIC pin notifies connected devices of Magic Packet detection.

Note: The Magic Packet detection interrupt status can be read in the E-MAC status register (ECSR.MPD). The bit can be cleared by setting the E-MAC status register Magic Packet TM detection bit (ECSR.MPD) to 1.

But EthernetAVB-IF can’t detect 2nd or later Magic PacketTM and the AVB_MAGIC pin can be negated though the bit is cleared.

The AVB_MAGIC pin can be negated and EthernetAVB-IF can detect MAGIC PacketTM again according to below way.

1) Hardware reset — set reset pin in the R-Car series to reset signal.
2) Software reset — set mstp/srst register for EthernetAVB-IF to software reset mode.
3) Software reset register in AVB — set AVB DMAC mode register Operating Mode Configuration bits (CCC.OPC) to reset mode.

#### 100.1.3.9 Interrupts

For details, refer to the “Interrupt Controller” section in the manual.

The interrupts of EthernetAVB-IF are indicated by individual flags **DIS**, **EIS**, **RISi**, **TIS**, and **GIS**. Each interrupt is controllable by the corresponding interrupt enable bit. The status flags are independent from this enable bits.

Additionally grouped interrupt status available in **ISS** and **EIS.QFS**. The summary status allows CPU to locate the interrupt source with a minimum number of read accesses to EthernetAVB-IF.

EthernetAVB-IF interrupts are combined to 4 groups of interrupt lines for request an interrupt service by CPU. Interrupt requests in group 0 and 1 are only generated in OPERATION mode. Interrupt requests in group 2 are only generated in OPERATION mode and in CONFIG mode if gPTP support is enabled (**CCC.GAC**). Interrupt request on line 3 depends on E-MAC configuration and can be generated in CONFIG, OPERATION and STANDBY mode.

Interrupt lines are level output and each line is asserted when at least one of the interrupt status registers bits belonging to the group associated with that line is set to 1b, and the corresponding enable bit in the related interrupt control register is also set to 1b.

For **ISS** bits the combined flag/enable is used. The green dots in detailed interrupt figures in detailed interrupt line description illustrates the point where status information for **ISS** is taken.

**Table 100.25 Interrupt assignment**

| Interrupt Name | Cause of interrupt |
|---|---|
| Ethernet AVB.ch0 | Line0_Rx[0] |
| Ethernet AVB.ch1 | Line0_Rx[1] |
| Ethernet AVB.ch2 | Line0_Rx[2] |
| Ethernet AVB.ch3 | Line0_Rx[3] |
| Ethernet AVB.ch4 | Line0_Rx[4] |
| Ethernet AVB.ch5 | Line0_Rx[5] |
| Ethernet AVB.ch6 | Line0_Rx[6] |
| Ethernet AVB.ch7 | Line0_Rx[7] |
| Ethernet AVB.ch8 | Line0_Rx[8] |
| Ethernet AVB.ch9 | Line0_Rx[9] |
| Ethernet AVB.ch10 | Line0_Rx[10] |
| Ethernet AVB.ch11 | Line0_Rx[11] |
| Ethernet AVB.ch12 | Line0_Rx[12] |
| Ethernet AVB.ch13 | Line0_Rx[13] |
| Ethernet AVB.ch14 | Line0_Rx[14] |
| Ethernet AVB.ch15 | Line0_Rx[15] |
| Ethernet AVB.ch16 | Line0_Rx[16] |
| Ethernet AVB.ch17 | Line0_Rx[17] |
| Ethernet AVB.ch18 | Line0_Tx[0] |
| Ethernet AVB.ch19 | Line0_Tx[1] |

| Interrupt Name | Cause of interrupt |
|---|---|
| Ethernet AVB.ch20 | Line0_Tx[2] |
| Ethernet AVB.ch21 | Line0_Tx[3] |
| Ethernet AVB.ch22 | Line0_DiA \| Line1_A \| Line2_A |
| Ethernet AVB.ch23 | Line0_DiB \| Line1_B \| Line2_B |
| Ethernet AVB.ch24 | Line 3 |
| EtherAVB TX RAM EDC Error | * |
| EtherAVB RX RAM EDC Error | * |

*This interrupt is functional safety use only.

### (1) Concurrent interrupt enable support

The interrupt enable bits can be directly modified by writing the interrupt control registers (**xxC.yyE**). If concurrent processes enable/disable interrupts located in same interrupt control register inter-process communication is required. To reduce SW overhead there are dedicated registers available controlling conflict free enable/disable of interrupts.

To enable an interrupt CPU should write 1b to the corresponding bit (**xxE.yyS**) in the enable register. Because written 0b have no effect, there is no side effect to concurrent processes.

To disable an interrupt CPU should write 1b to the corresponding bit (**xxD.yyD**) in the disable register. Because written 0b have no effect, there is no side effect to concurrent processes.

The current interrupt enable status is readable in **xxC.yyE**.

Grouping of interrupts for each interrupt line is as described in following sub-chapters.

### (2) Group 0 (Data related interrupts)

This group consists on:

Frame received interrupts **RIS0.FRFr** (for each received frame stored in URAM)
Receive warning interrupts **RIS1.RWFr** (when Unread Frame Counter reaches warning level)
Descriptor processed interrupts **RIS3.RDPFr** (for reception)Frame transmitted interrupts **TIS.FTFt** (for completion of frame transmission by E-MAC)
Descriptor processed interrupts **TIS.TDPFr** (for transmission)
Descriptor processed interrupts **DIS.DPFi** (for reception and transmission)

By reading **ISS**, CPU get an overview of all data related interrupts (**ISS.FRM**, **ISS.RWM**, **ISS.DPMi, ISS.FTM**).

Interrupt group 0 consists on these 24 interrupt lines:

Line0_Rx[17:0]
Line0_Tx[3:0]
Line0_DiA (Share the interrupt line : Group0.Line0_DiA, Group1.Line1_A, Group2.Line2_A)
Line0_DiB (Share the interrupt line : Group0.Line0_DiB, Group1.Line1_B, Group2.Line2_B)

**(a) Reception interrupts**

There are 18 dedicated interrupt lines flagging frame reception. For the queue full flag it is configurable by **CIE.RQFM** if these notification are part of the queue specific reception interrupt lines or part of the error interrupt lines (group 1).

There is a common interrupt enable (**CIE.CRIE**) for all reception lines.

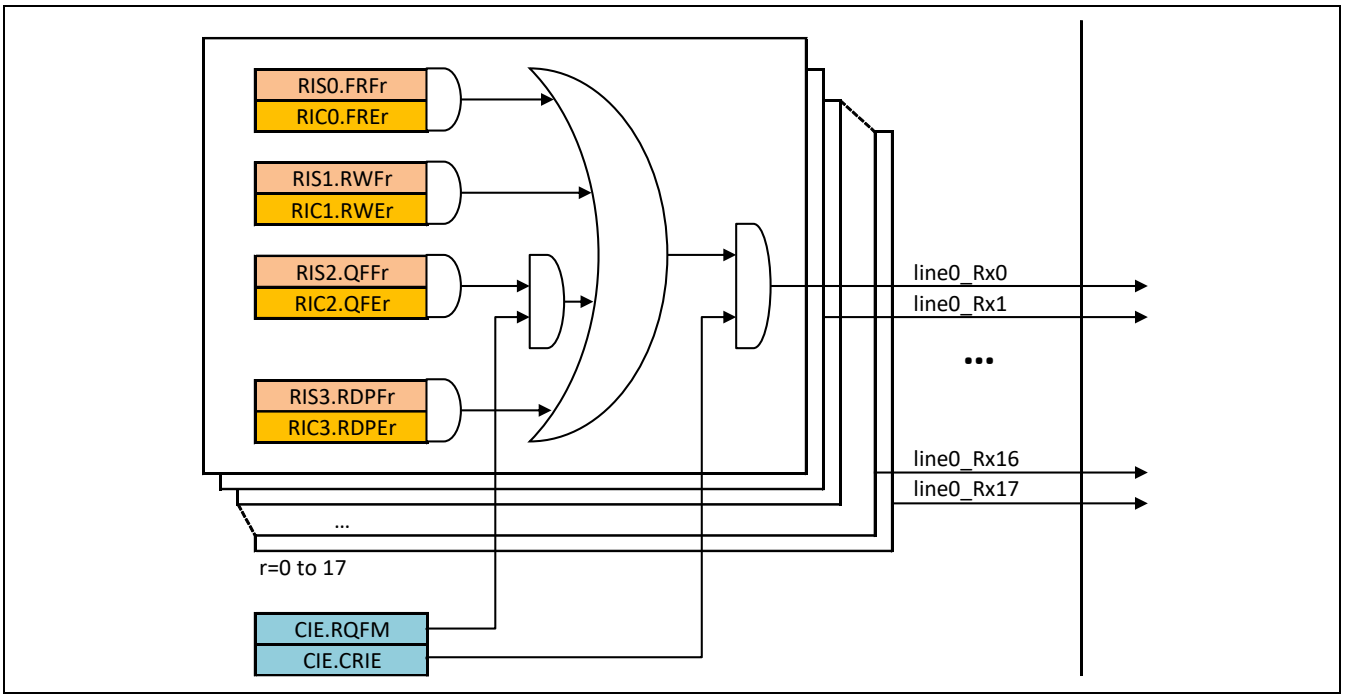



**Figure 100.1.3.11.1 Interrupt group 0, reception interrupt lines**

**(b) Transmission interrupts**

There are 4 dedicated interrupt lines flagging frame transmission.

There is a common interrupt enable (**CIE.CTIE**) for all reception lines.

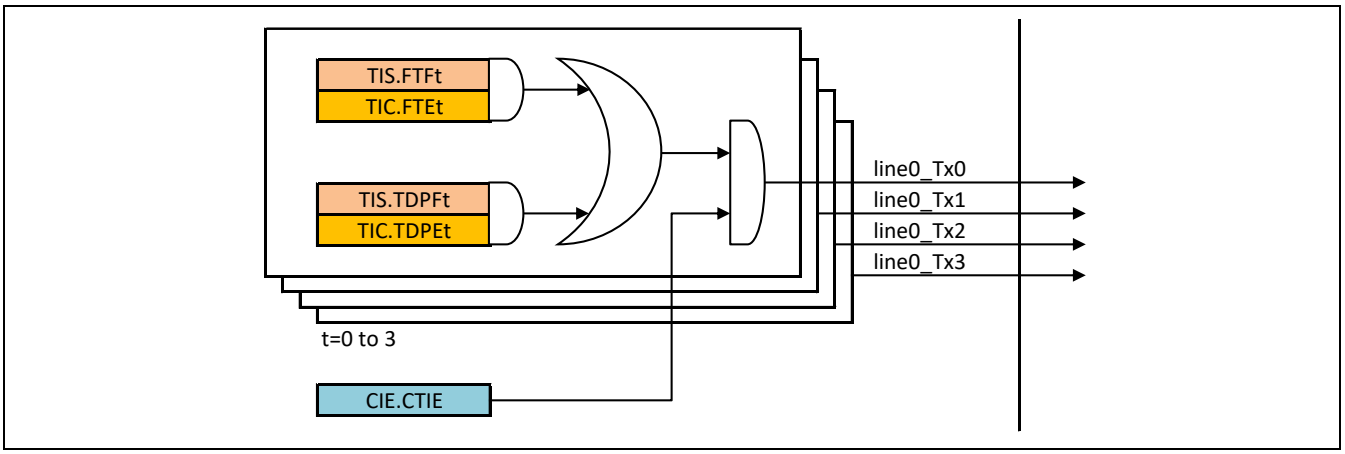



**Figure 100.1.3.11.2 Interrupt group 0, transmission interrupt lines**

**(c) Descriptor and merged interrupts**

There are 2 interrupt lines flagging descriptor interrupts. Line **line0_DiA** has also the capability to generate a merged data interrupt (controlled by **CIE.CL0M**).

Descriptor interrupt 1 is not available as common descriptor interrupt; this descriptor interrupt is split into queue specific descriptor interrupts.

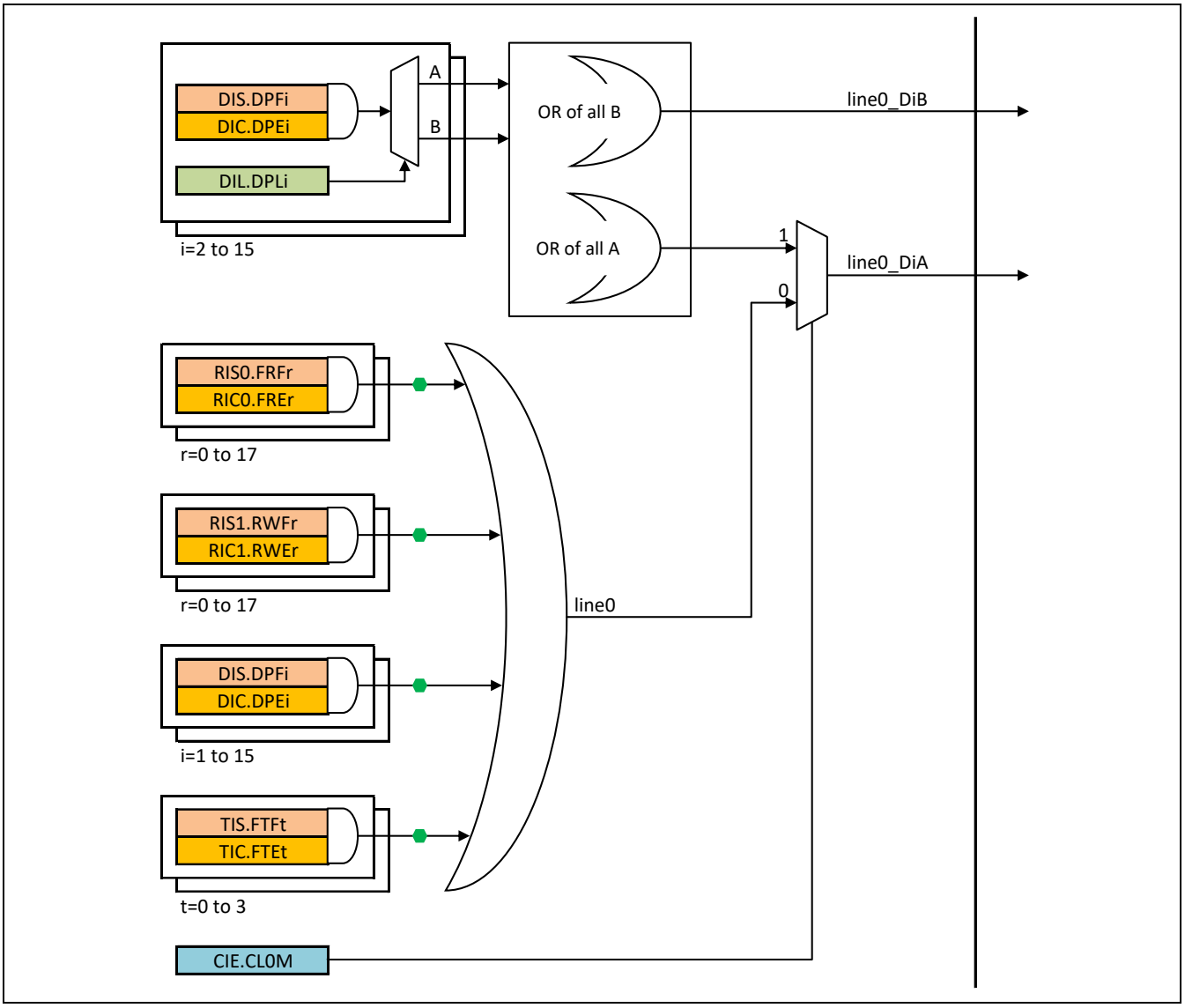



**Figure 100.1.3.11.3 Interrupt group 0, common descriptor and merged interrupt lines**

### (3) Group 1 (Error related interrupts)

This group consists on:

Tx-Buffer full interrupt **EIS.TBFF** (when delaying fetching of transmit frame)
Interrupt for E-MAC Status FIFO overwrite (**EIS.MFFF**)
Interrupt for Timestamp FIFO overrun (**EIS.TFFF**)
Interrupts for CBS limits (**EIS.CULF1**, **EIS.CULF0**, **EIS.CLLF1**, **EIS.CLLF0**)
Interrupt for separation filtering error (**EIS.SEF**)
Interrupt for queue error (error while processing receive or transmit queue) (**EIS.QEF**)

Interrupt for transmission error detected by E-MAC (**EIS.MTEF, EIS.MREF**)
Rx-FIFO full interrupt **RIS2.RFFF** (when Rx-FIFO overflows)
Receive queue full interrupts **RIS2.QFFr** (when no sufficient space is in receive queue)

By reading **EIS**, CPU get an overview of all error interrupts. Additionally **ISS.EM** informs about pending error related interrupt.

Interrupt group 1 consists on these 2 interrupt lines:

Line1_A (Share the interrupt line : Group0.Line0_DiA, Group1.Line1_A, Group2.Line2_A)
Line1_B (Share the interrupt line : Group0.Line0_DiB, Group1.Line1_B, Group2.Line2_B)

For the queue full flag it is configurable by **CIE.RQFM** if these notification are part of the queue specific reception interrupt lines or part of the error interrupt line **line1_A.**

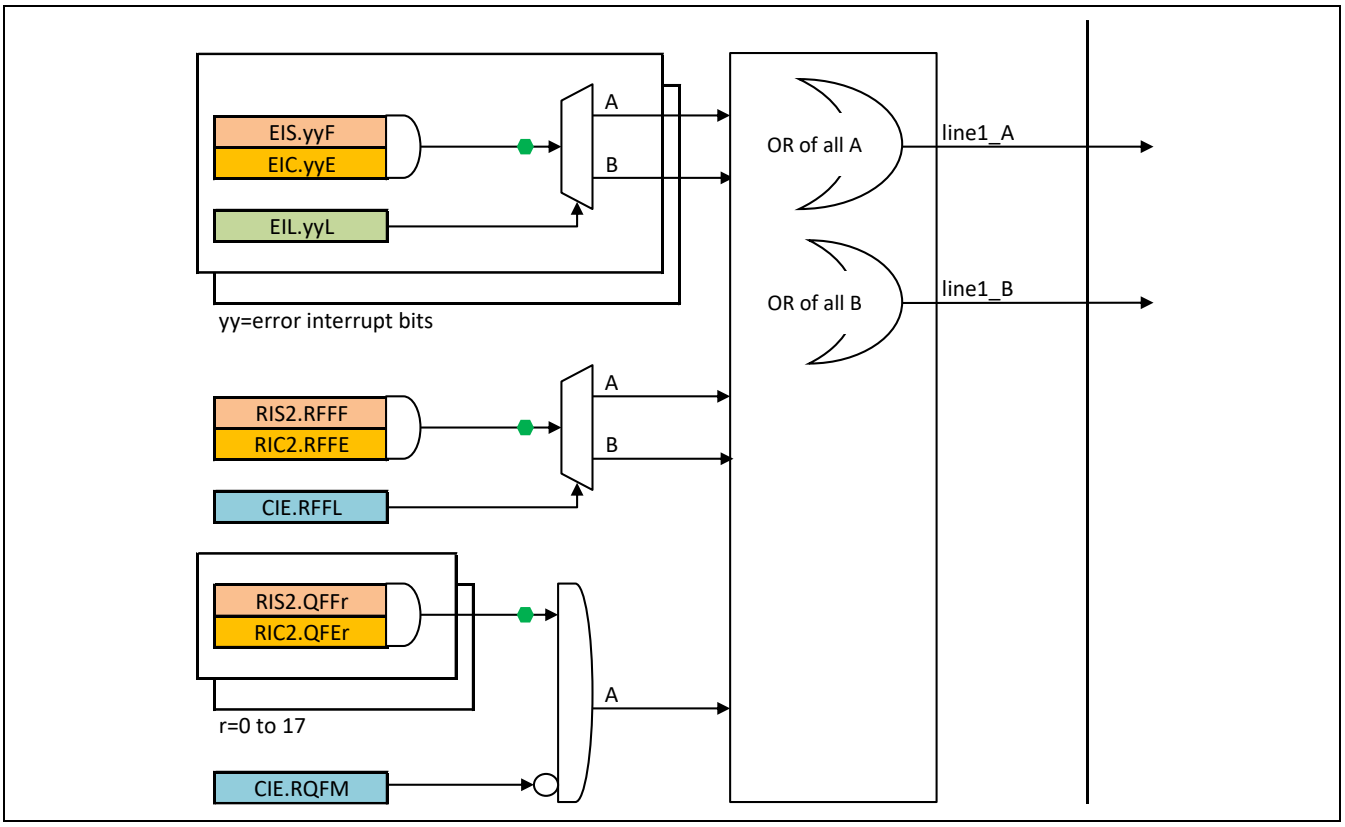



**Figure 100.1.3.11.4 Interrupt group 1, error interrupt lines**

The abbreviation **yy** in figure indicates error interrupts: **TBF**, **MFF**, **TFF**, **CULFc**, **CLLFc**, **SE**, **QE**, **MTE**, and **MRE**.

### (4) Group 2 (Management related interrupts)

Reception

Rx-FIFO Warning interrupt **RIS1.RFWF** (Rx-FIFO has reached configured warning level)

Transmission

E-MAC status FIFO warning interrupt **TIS.MFWF**
E-MAC Status FIFO updated interrupt **TIS.MFUF**
Timestamp FIFO warning interrupt **TIS.TFWF**
Timestamp FIFO updated interrupt **TIS.TFUF**

gPTP

AVTP time captured interrupt **GIS.ATCFi**
AVTP presentation target matched interrupt **GIS.PTMFi**
AVTP presentation time overrun interrupt **GIS.PTOF** (when external trigger before old value is processed)
AVTP presentation time captured interrupt **GIS.PTCF**

By reading **ISS**, CPU get an overview of all management related interrupts (**ISS.RFWM**, **ISS.MFWM**, **ISS.MFUM**, **ISS.TFWM**, **ISS.TFUM**, **ISS.CGIM**).

Interrupt group 2 consists on these 2 interrupt lines:

Line2_A (Share the interrupt line : Group0.Line0_DiA, Group1.Line1_A, Group2.Line2_A)
Line2_B (Share the interrupt line : Group0.Line0_DiB, Group1.Line1_B, Group2.Line2_B)

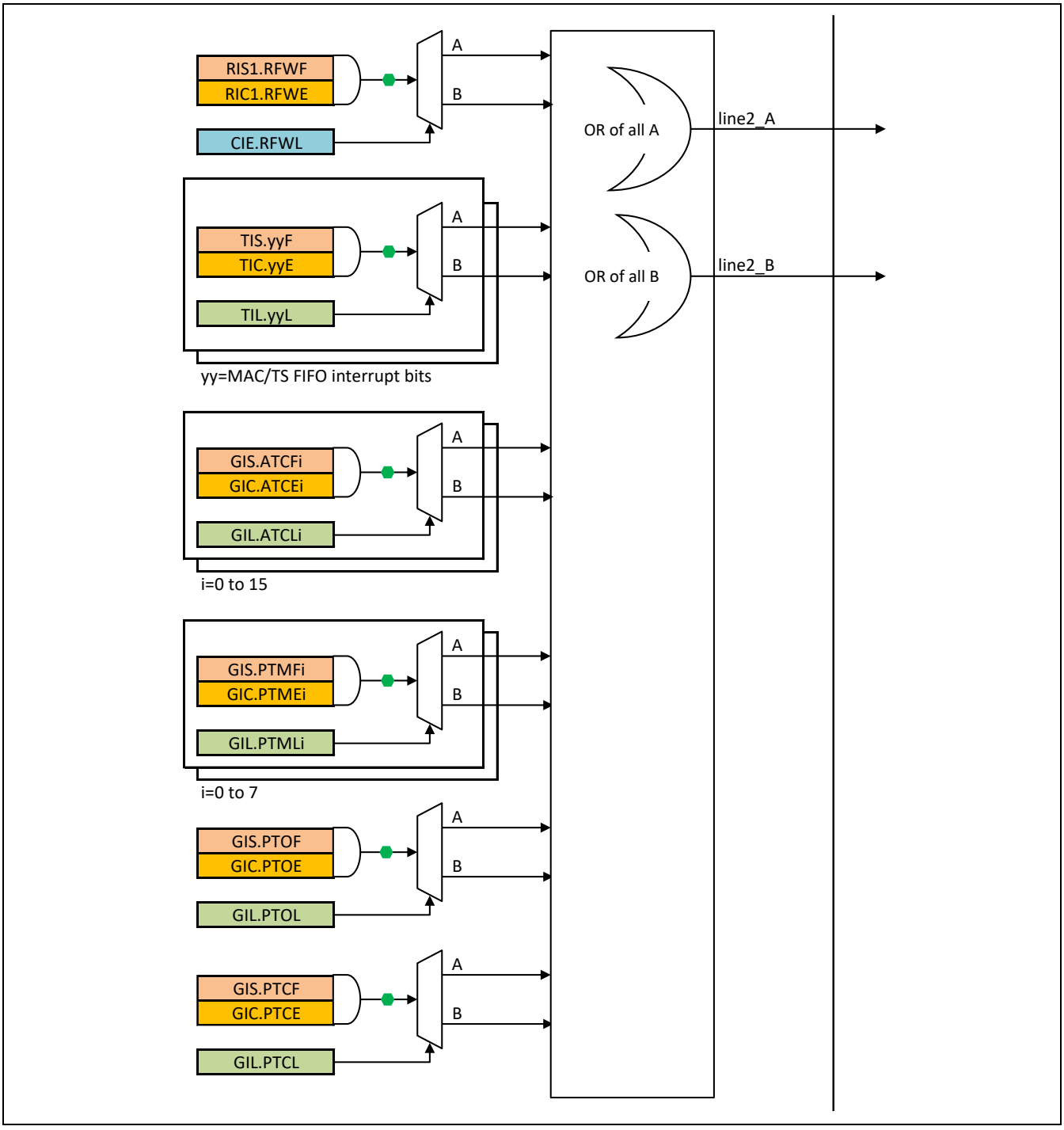



**Figure 100.1.3.11.5 Interrupt group 2, management interrupt lines**

The abbreviation **yy** in figure indicates error interrupts: **MFW**, **MFU**, **TFW**, and **TFU**.

### (5) Line 3 (E-MAC interrupt)

The E-MAC interrupt is conveyed when the E-MAC interrupt source is generated.

The general error interrupt state can be checked by reading the E-MAC interrupt summary bit in the interrupt summary status register (ISS.MS).

### 100.1.3.10 Flows of Operations

#### (1) Flow of E-MAC Initialization

Figure 100.1.3.12.1 shows the flow of E-MAC initialization (for AVB mode and full-duplex operation).

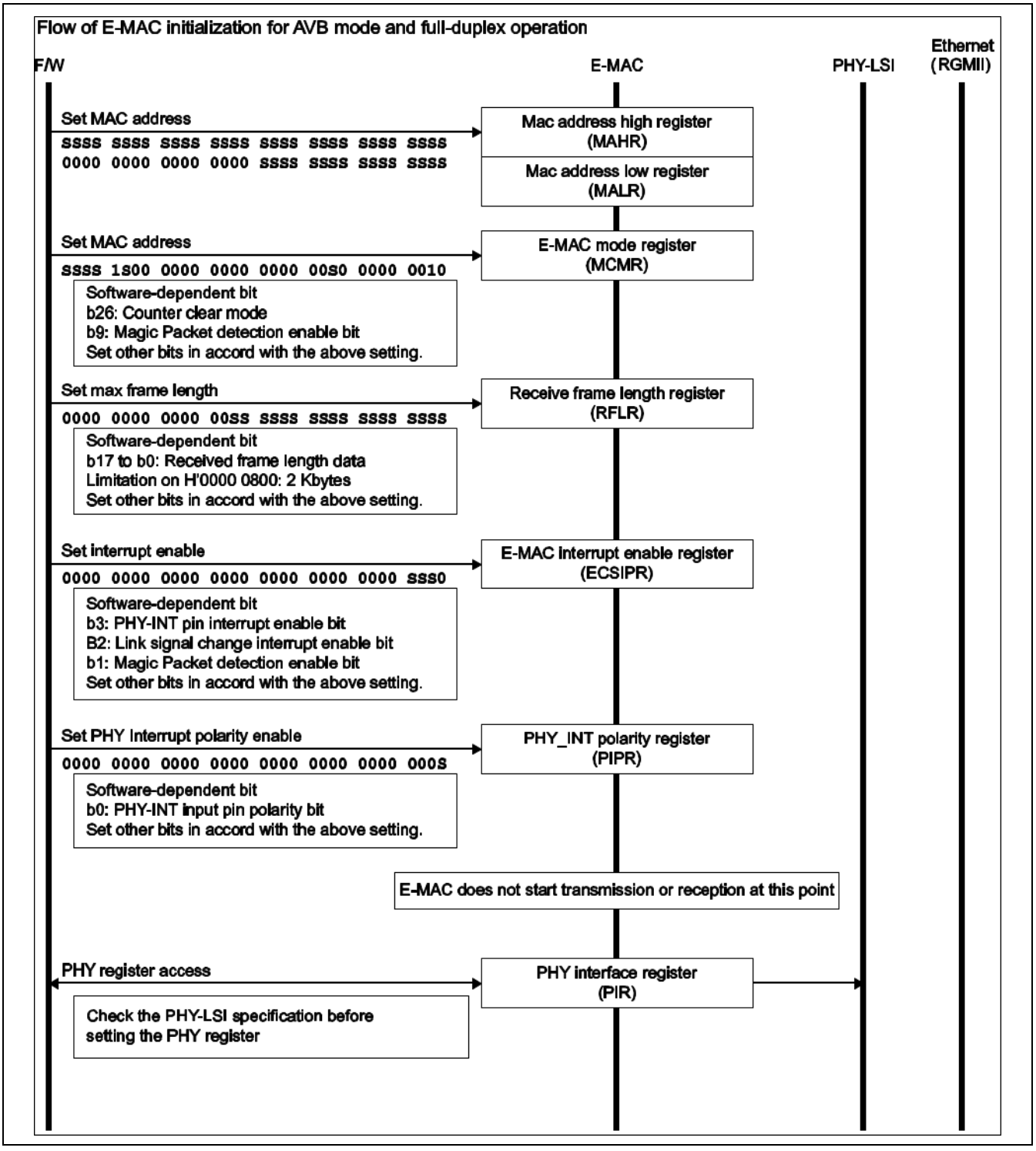



**Figure 100.1.3.12.1 Flow of E-MAC Initialization (for AVB Mode and Full-Duplex Operation)**

### (2) Flow of AVB-DMAC Initialization

Figure 100.1.3.12.2 shows the flow of AVB-DMAC initialization (for AVB mode and full-duplex operation).

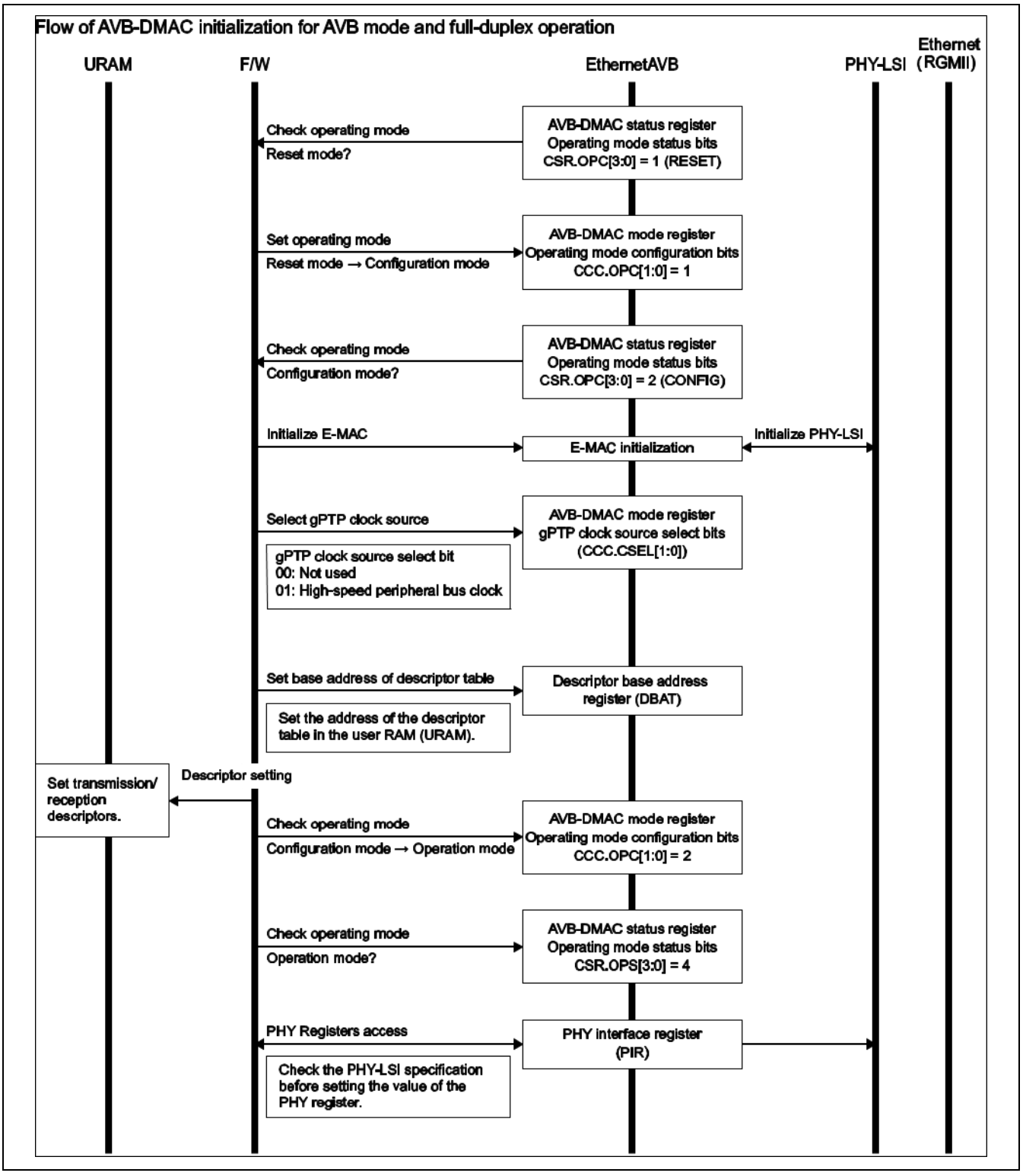



**Figure 100.1.3.12.2 Flow of AVB-DMAC Initialization (for AVB Mode and Full-Duplex Operation)**

### (3) Flow for the AVB-DMAC in Reception

Figure 100.1.3.12.3 shows the flow for the AVB-DMAC in reception (in AVB mode and full-duplex operation).

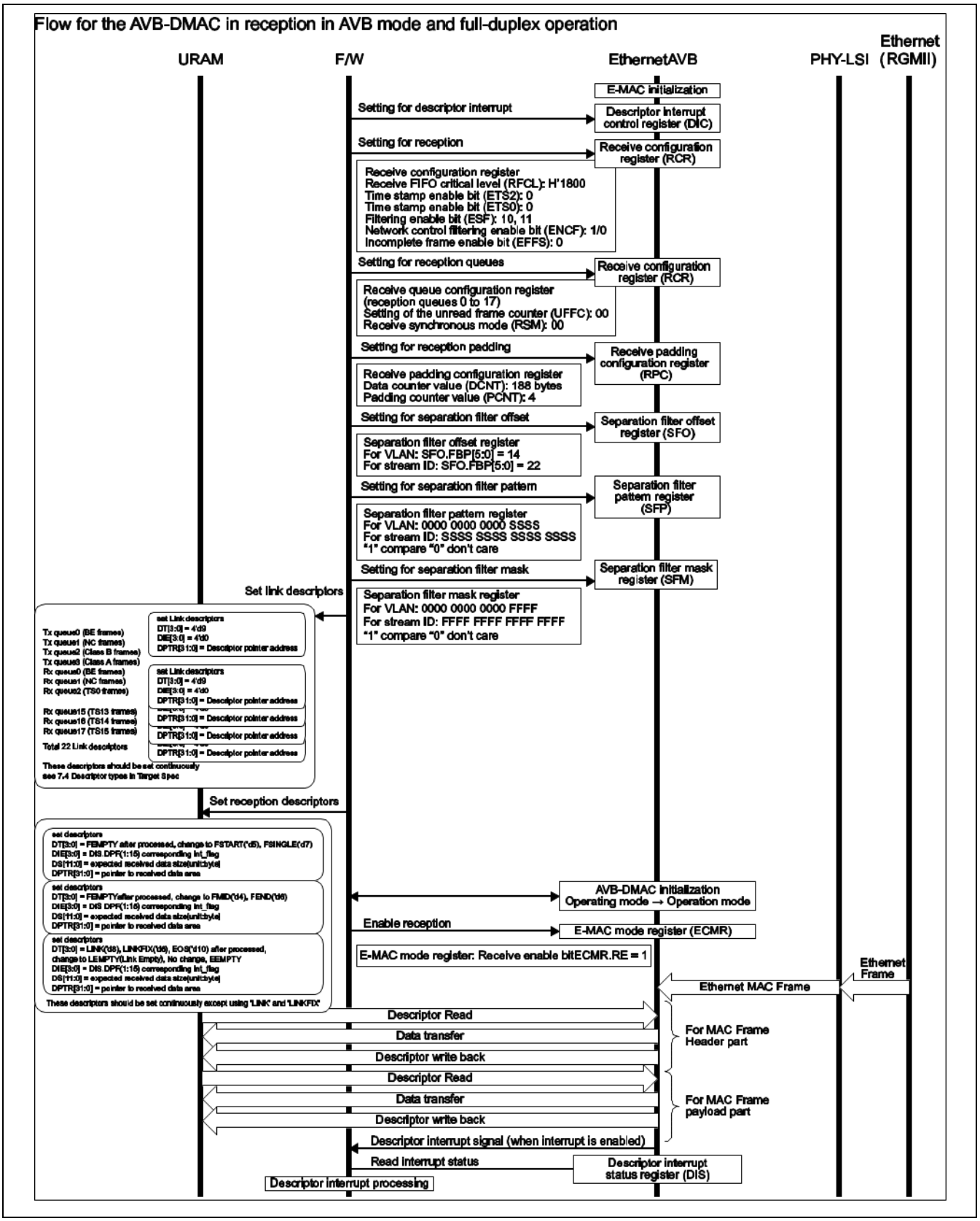



**Figure 100.1.3.12.3 Flow for the AVB-DMAC in Reception (in AVB Mode and Full-Duplex Operation)**

### (4) Flow for the AVB-DMAC in Transmission

Figure 100.1.3.12.4 shows the flow for the AVB-DMAC in transmission (in AVB mode and full-duplex operation).

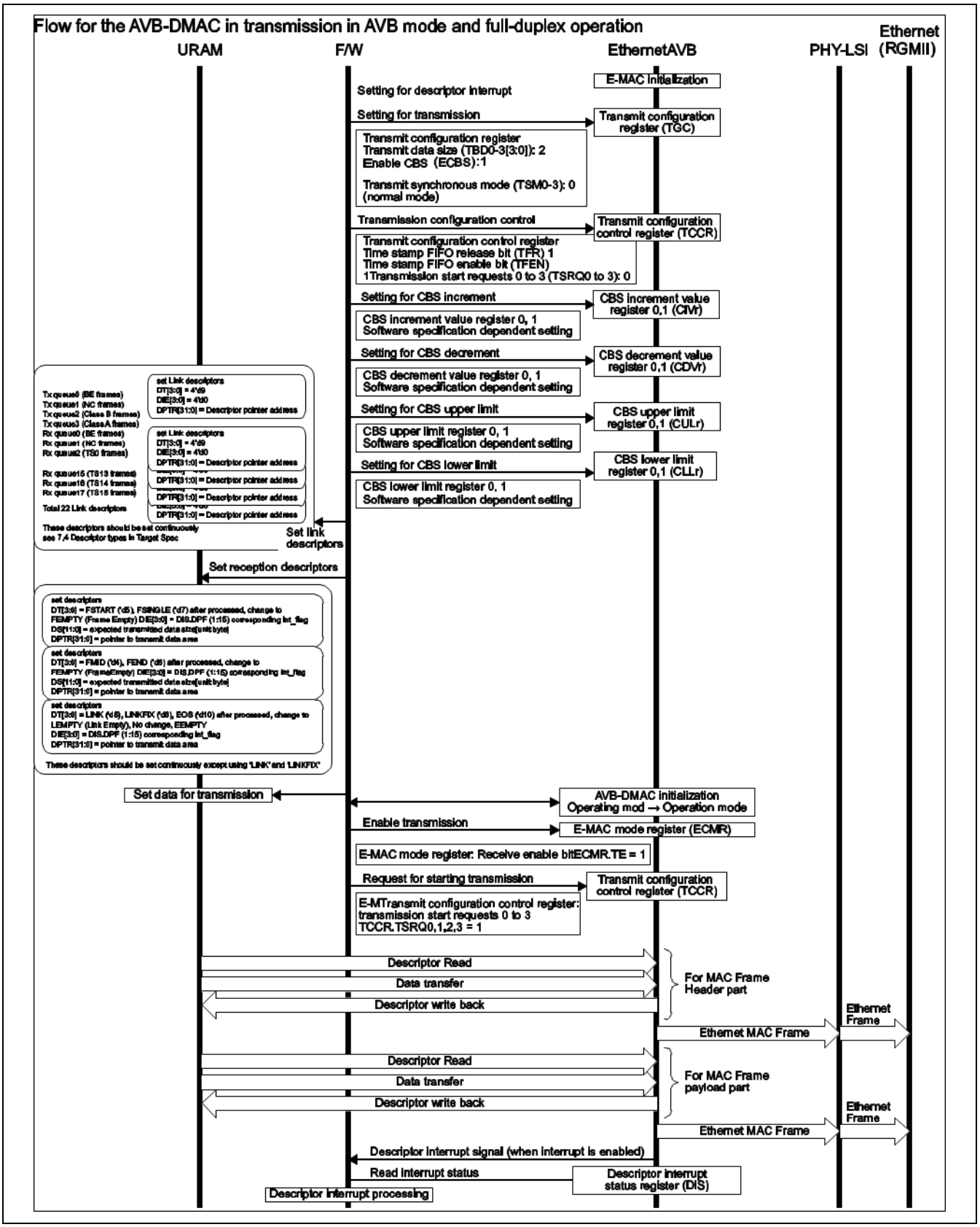



**Figure 100.1.3.12.4 Flow for the AVB-DMAC in Transmission (in AVB Mode and Full-Duplex Operation)**

### (5) Flow for Stopping AVB-DMAC Operation in Reception

Figure 100.1.3.12.5 shows the flow for stopping AVB-DMAC operation in reception (normal, common to all modes).

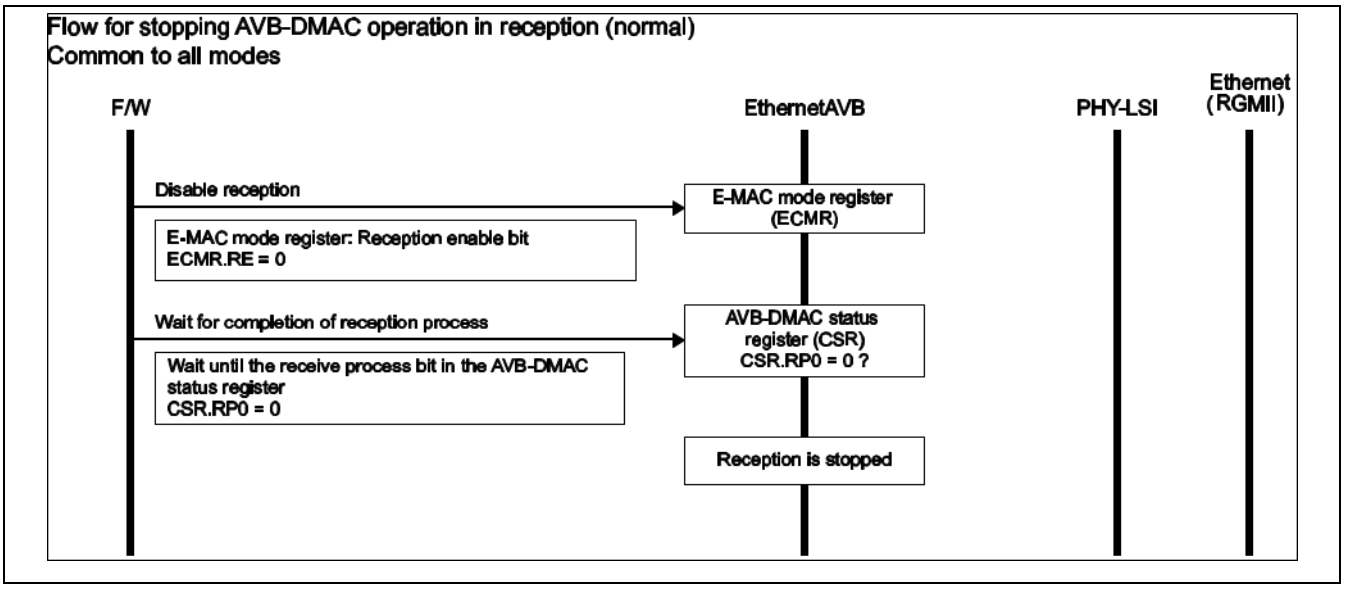



**Figure 100.1.3.12.5 Flow for Stopping AVB-DMAC Operation in Reception (Normal, Common to All Modes)**

### (6) Flow for Stopping AVB-DMAC Operation in Transmission

Figure 100.1.3.12.6 shows the flow for stopping AVB-DMAC operation in transmission (normal, common to all modes).

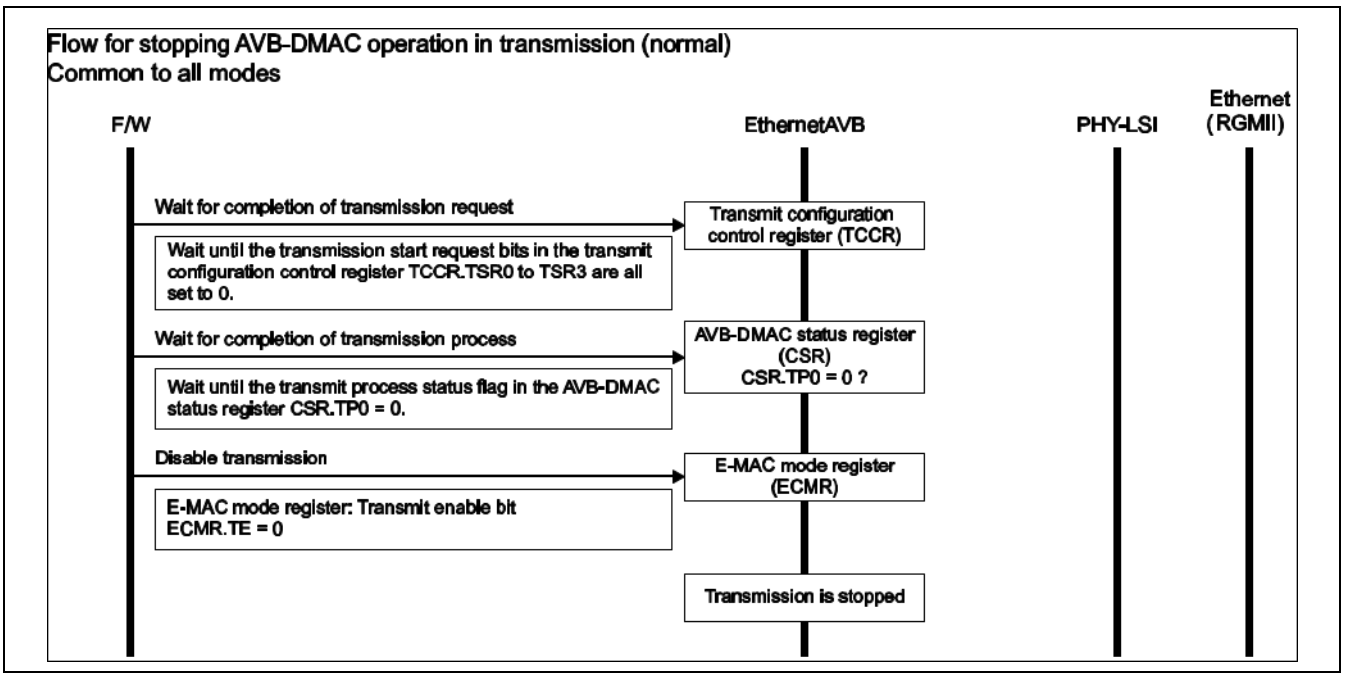



**Figure 100.1.3.12.6 Flow for Stopping AVB-DMAC Operation in Transmission (Normal, Common to All Modes)**

### (7) Flow for Stopping and Resetting the AVB-DMAC

Figure 100.1.3.12.7 shows the flow for stopping and resetting the AVB-DMAC (normal, common to all modes).

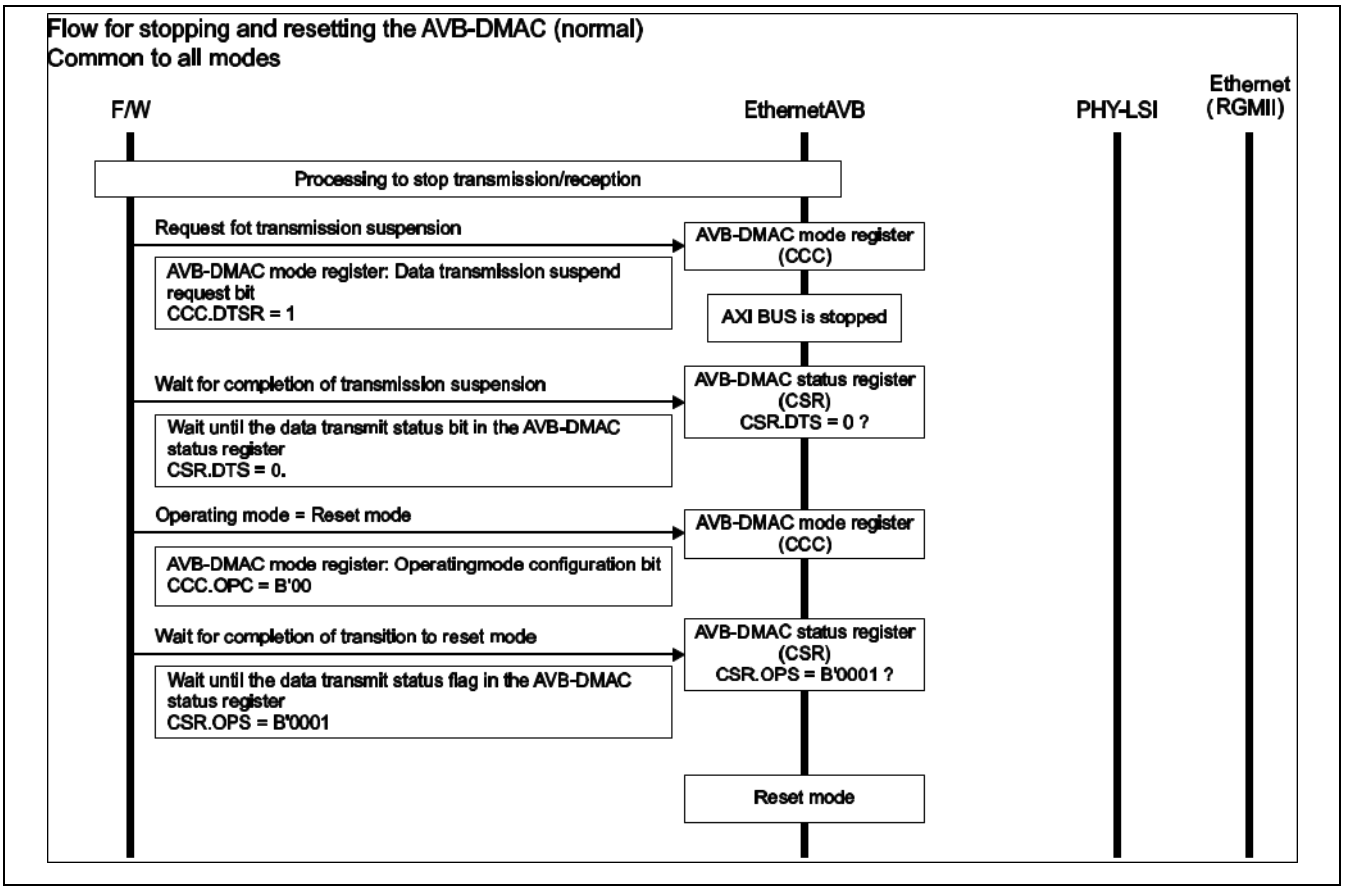



**Figure 100.1.3.12.7 Flow for Stopping and Resetting the AVB-DMAC (Normal, Common to All Modes)**

### (8) Flow for Emergency Stopping the AVB-DMAC

Figure 100.1.3.12.8 shows the flow for emergency stopping the AVB-DMAC (normal, common to all modes).

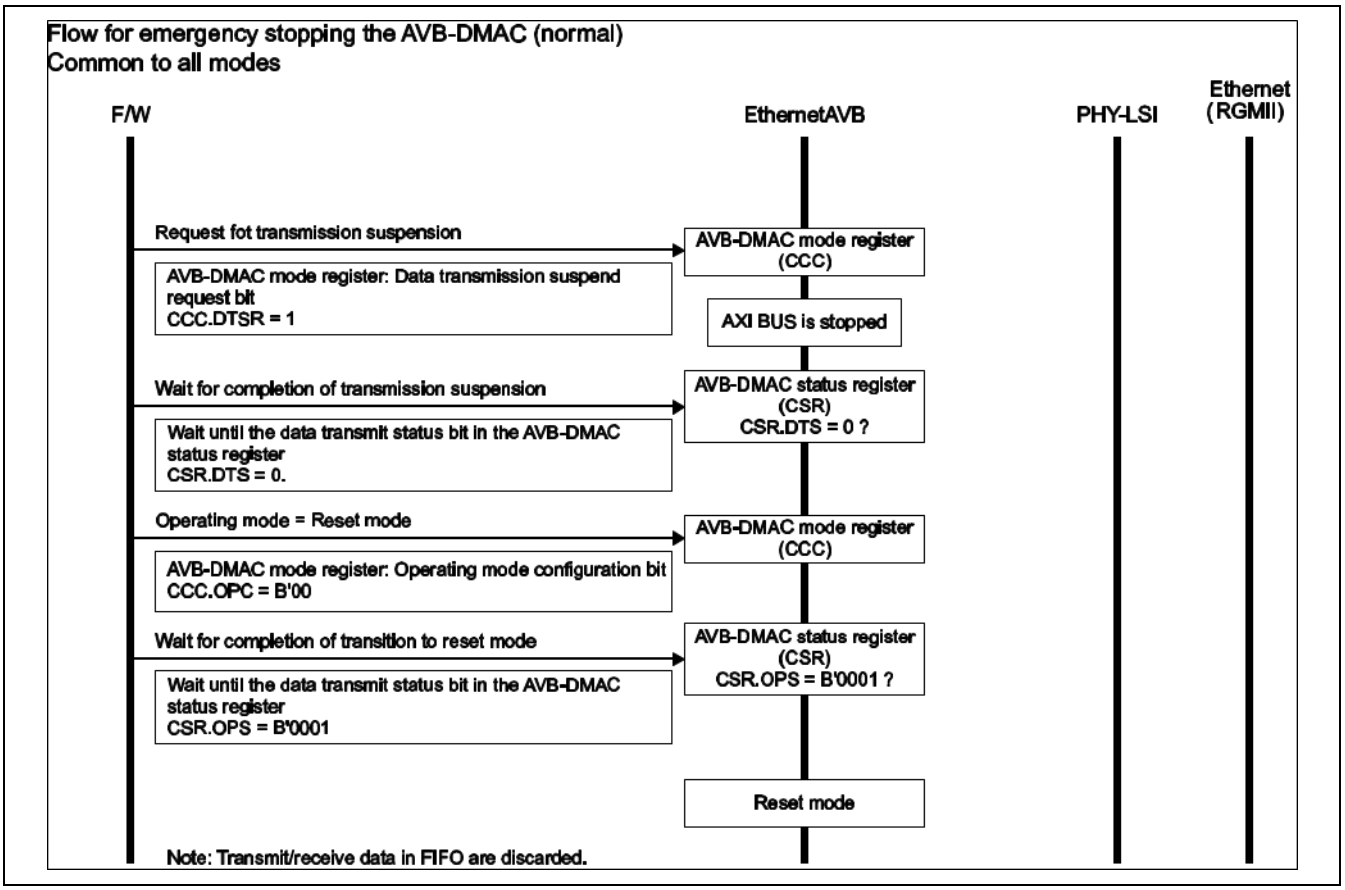



**Figure 100.1.3.12.8 Flow for Emergency Stopping the AVB-DMAC (Normal, Common to All Modes)**

### (9) Flow of gPTP Initialization

Figure 100.1.3.12.9 shows the flow of gPTP initialization (normal, common to all modes).

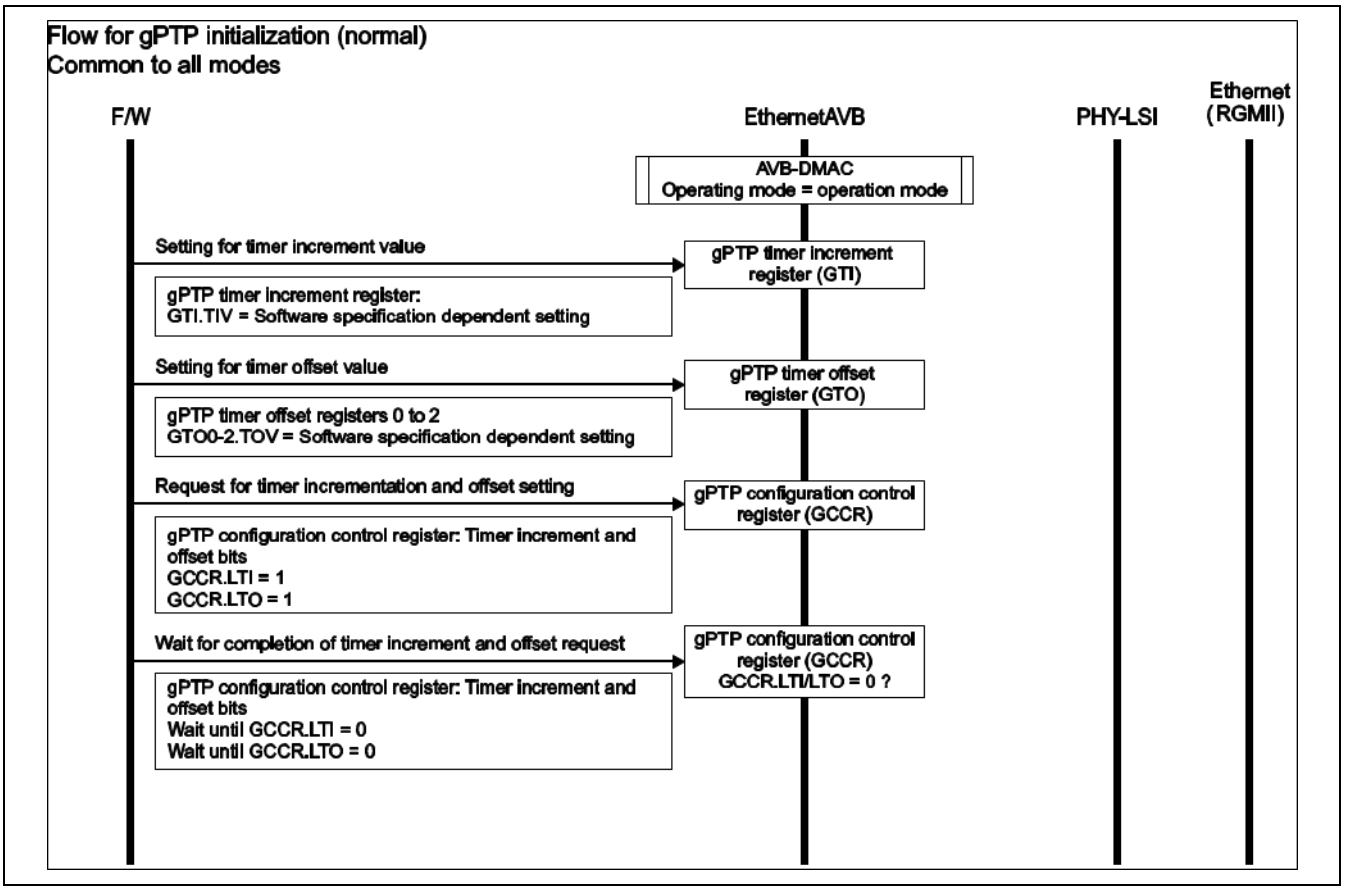



**Figure 100.1.3.12.9 Flow of gPTP Initialization (Normal, Common to All Modes)**

### (10) Flow of gPTP Time Stamping in Transmission

Figure 100.1.3.12.10 shows the flow of gPTP time stamping in transmission (normal, common to all modes).

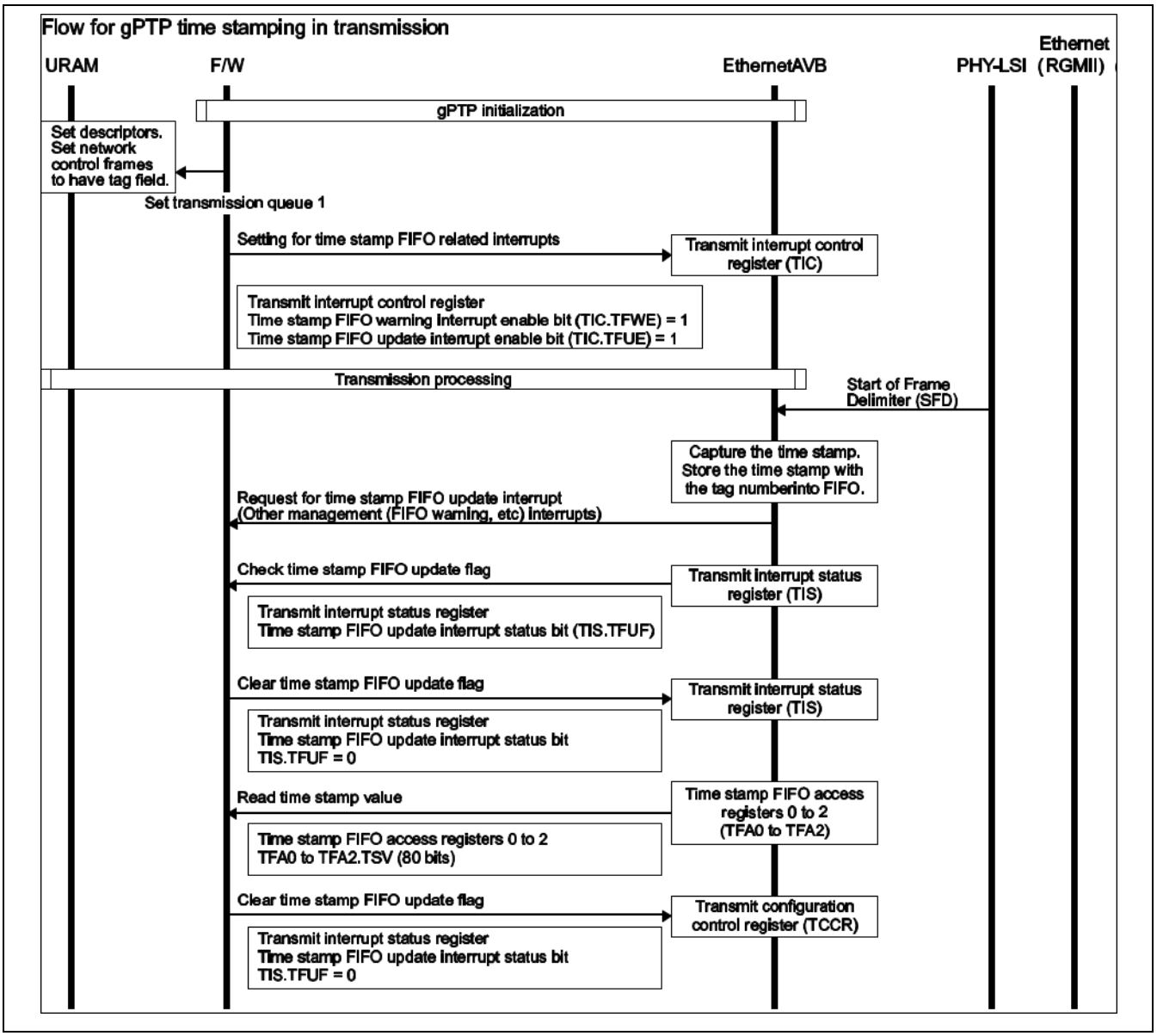



**Figure 100.1.3.12.10 Flow of gPTP Time Stamping in Transmission (Normal, Common to All Modes)**

### (11) Flow of gPTP Time Stamping and Synchronization in Reception

Figure 100.1.3.12.11 shows the flow of gPTP time stamping and synchronization in reception (normal, common to all modes).

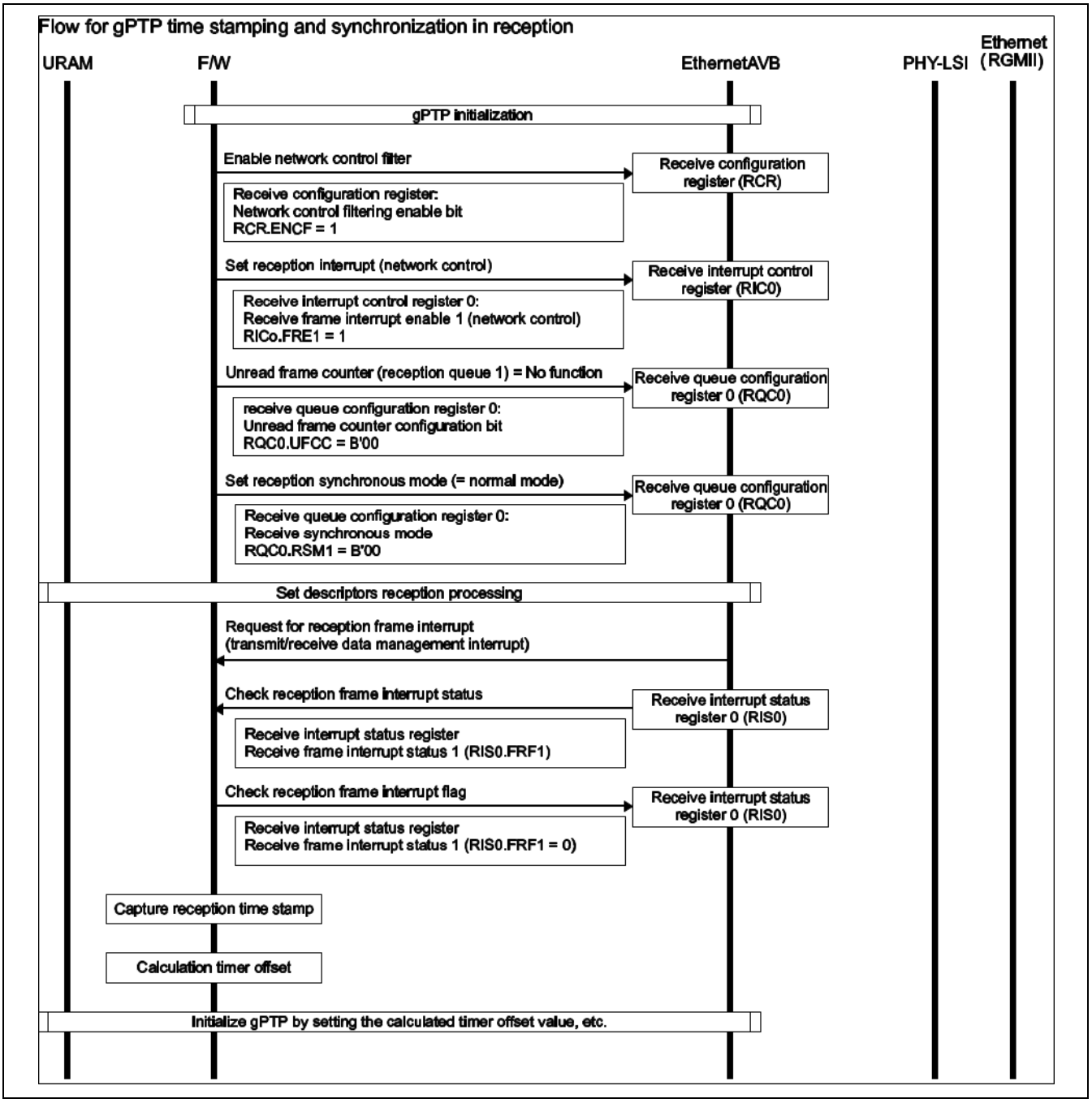



**Figure 100.1.3.12.11 Flow of gPTP Time Stamping and Synchronization in Reception (Normal, Common to All Modes)**

### (12) Flow of Capturing gPTP Presentation Times

Figure 100.1.3.12.12 shows the flow of capturing gPTP presentation times (common to all modes).

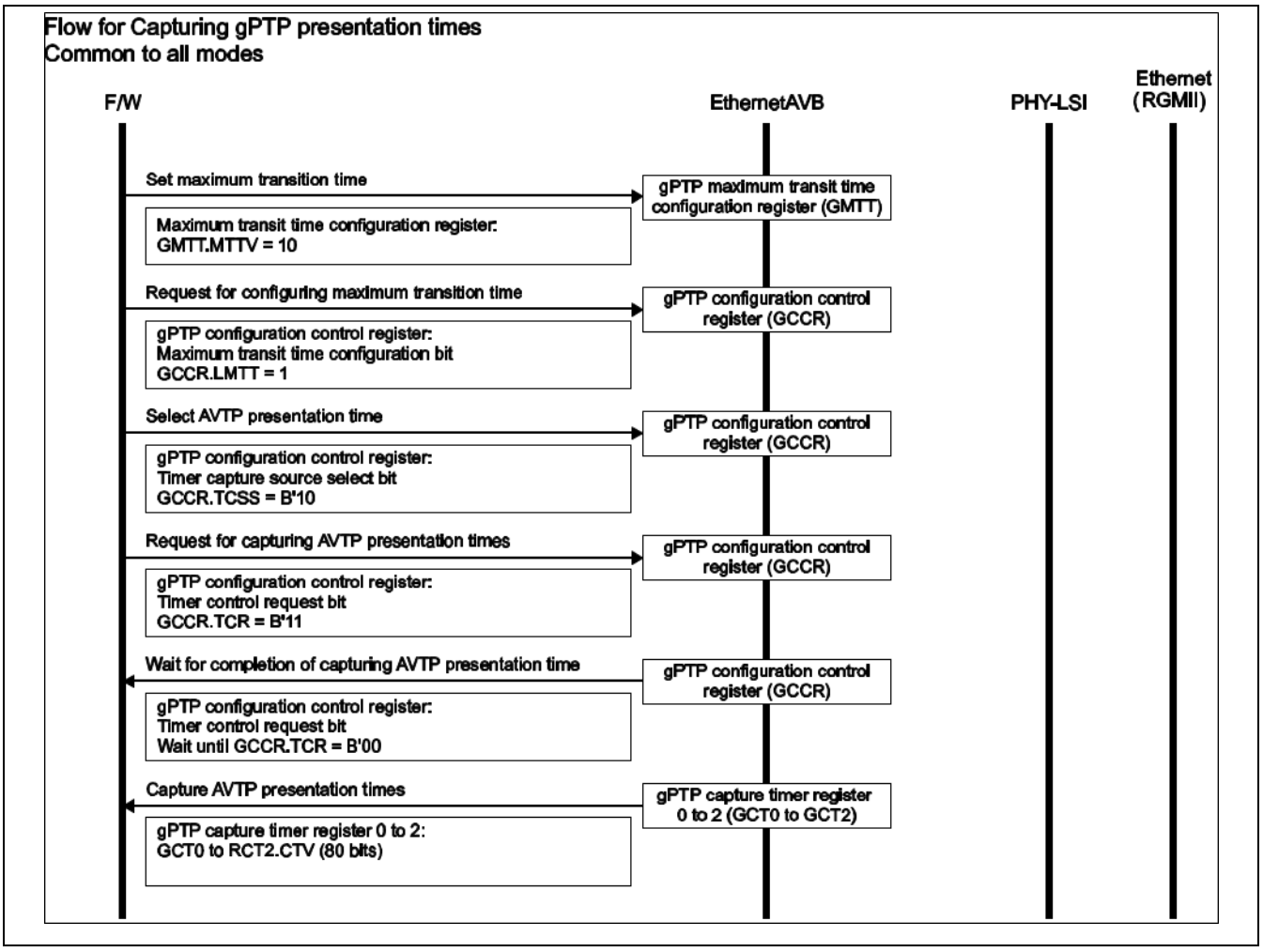



**Figure 100.1.3.12.12 Flow of Capturing gPTP Presentation Times (Common to All Modes)**

### (13) Flow of AVTP Presentation Time Comparison

Figure 100.1.3.12.13 shows the flow of AVTP presentation time comparison (common to all modes).

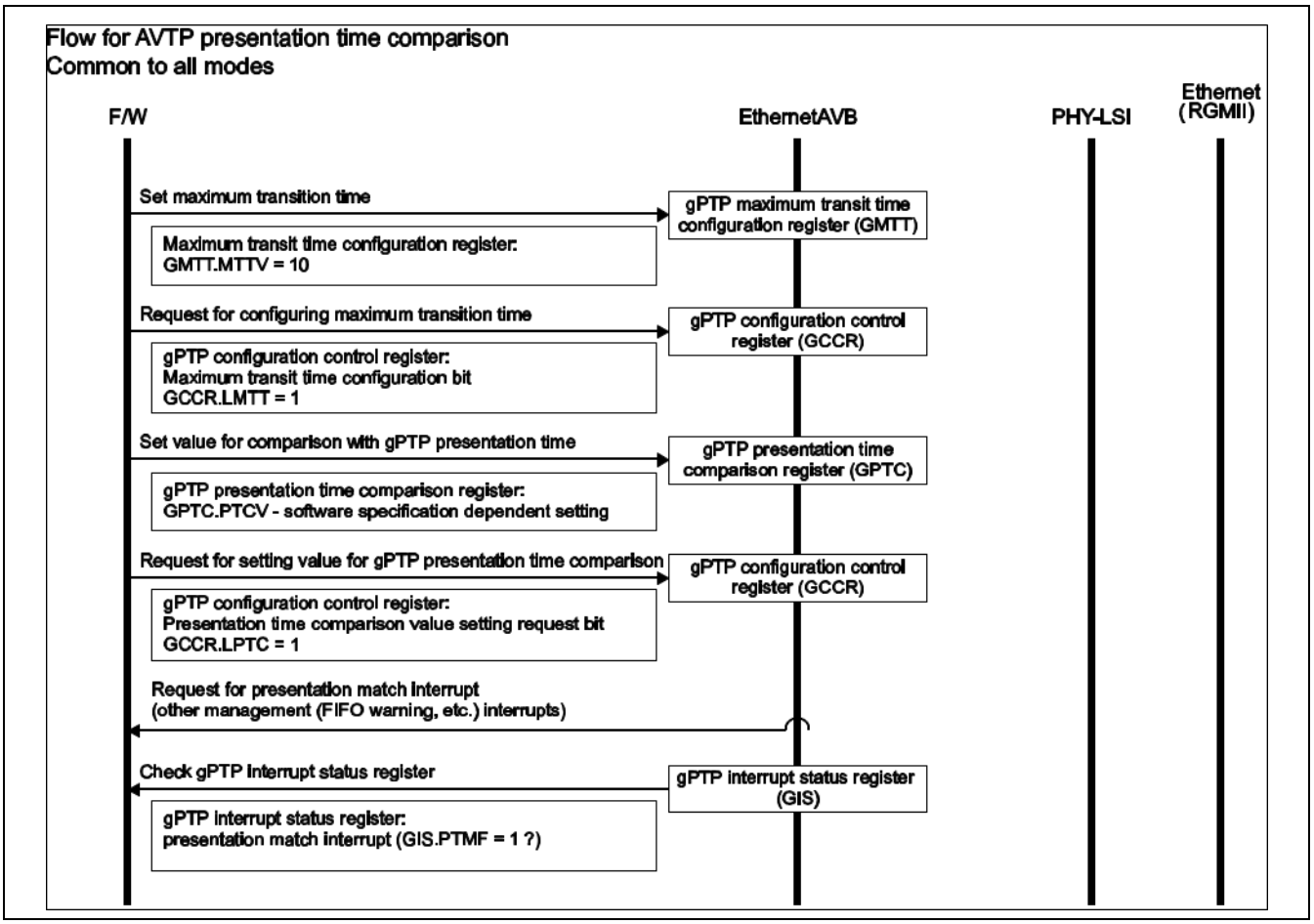



**Figure 100.1.3.12.13 Flow of AVTP Presentation Time Comparison (Common to All Modes)**

### (14) Flow of Loopback Mode Operation

Figure 100.1.3.12.14 shows the flow of loopback mode operation.

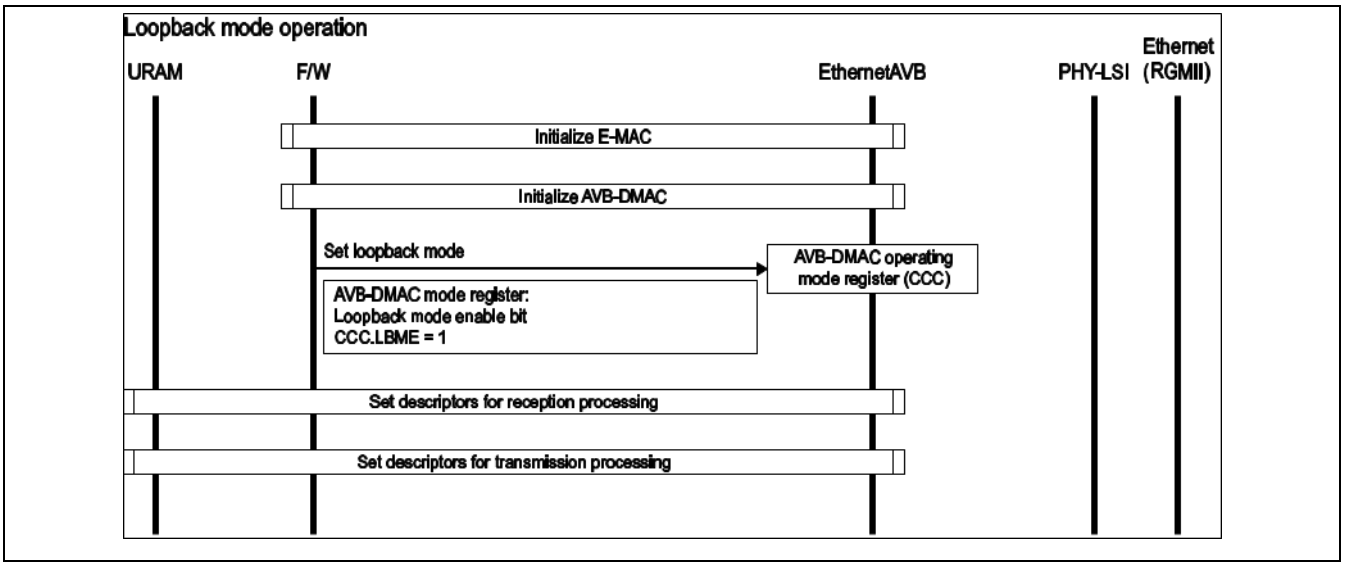



**Figure 100.1.3.12.14 Flow of Loopback Mode Operation**

### 100.1.3.11 Connection to PHY-LSI

#### (1) RGMII Frame Transmission/Reception Timing

Each RGMII frame transmission/reception timing is shown in Figure 100.1.3.13.1 to Figure 100.1.3.13.8.

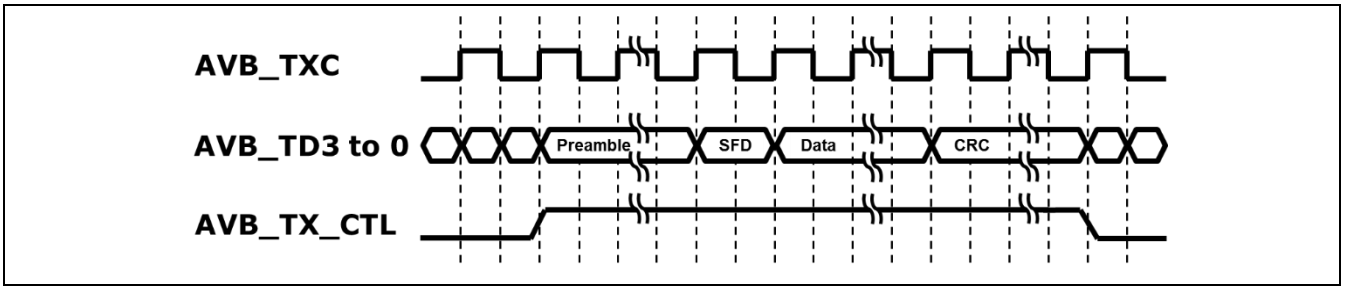



**Figure 100.1.3.13.1 RGMII Frame Transmit Timing (Normal Transmission)**

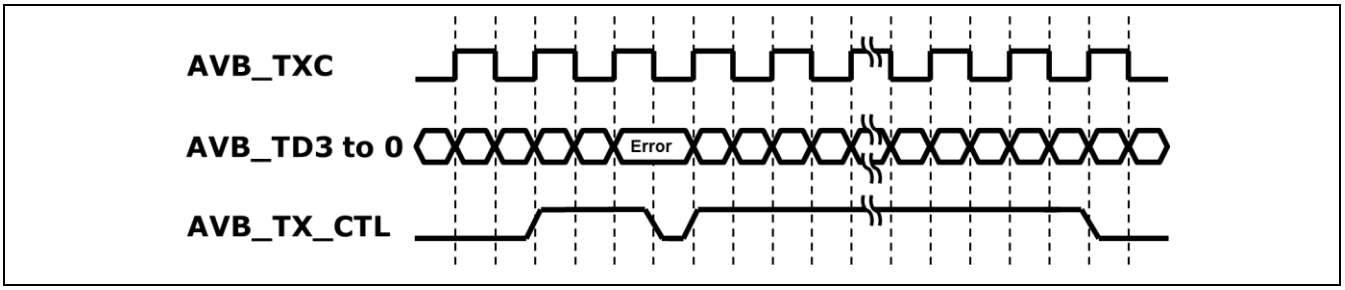



**Figure 100.1.3.13.2 RGMII Frame Transmit Timing (Transmit Error)**

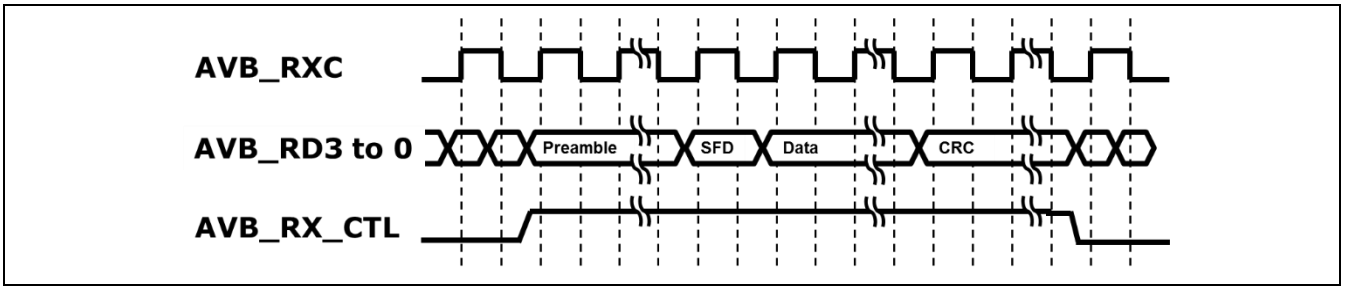



**Figure 100.1.3.13.3 RGMII Frame Receive Timing (Normal Reception)**

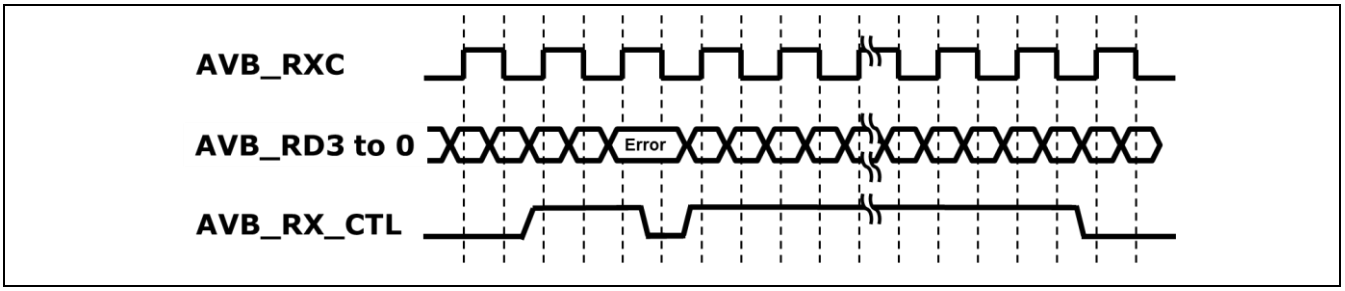



**Figure 100.1.3.13.4 RGMII Frame Receive Timing (Reception Error)**

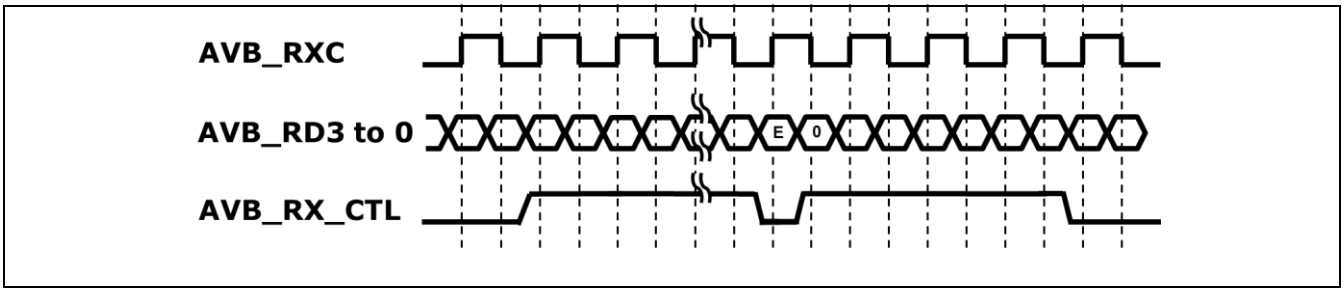



**Figure 100.1.3.13.5 RGMII Frame Receive Timing (False Carrier indication)**

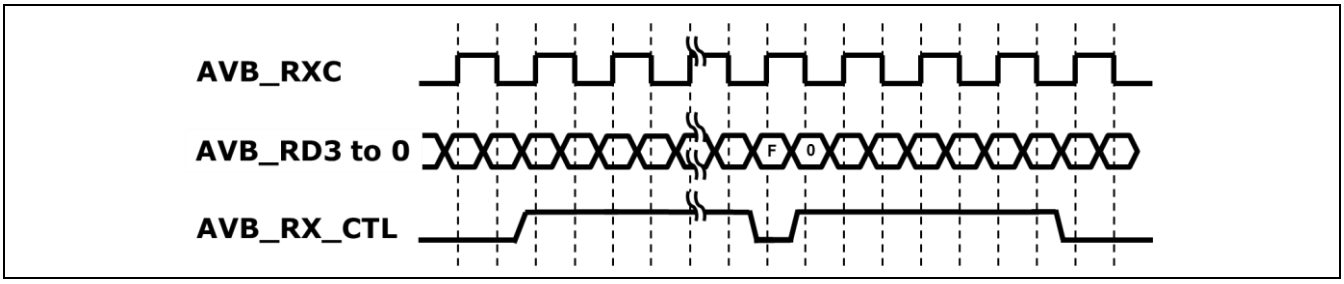



**Figure 100.1.3.13.6 RGMII Fame Receive Timing (Carrier Extend)**

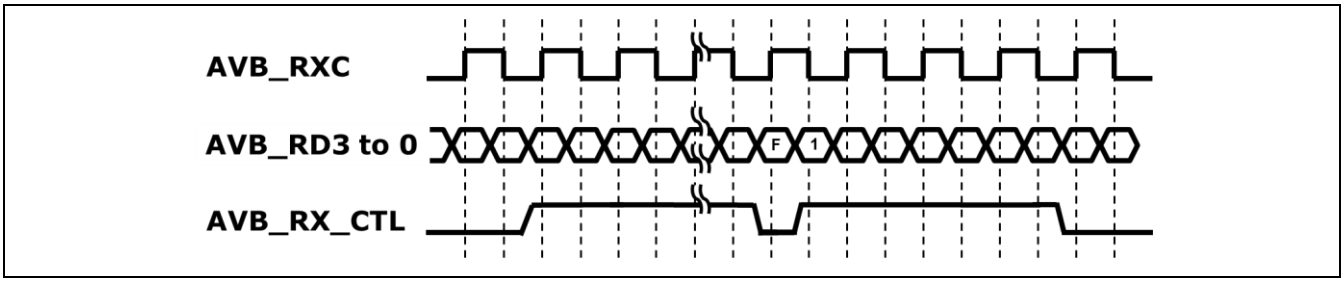



**Figure 100.1.3.13.7 RGMII Frame Receive Timing (Carrier Extend Error)**

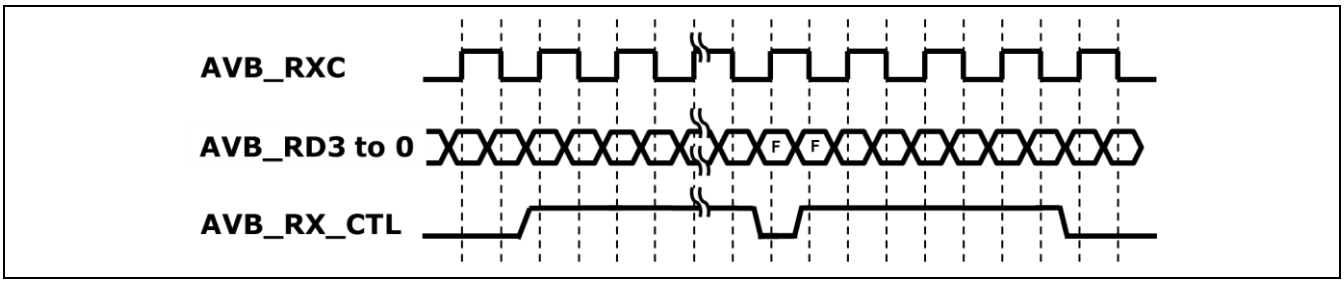



**Figure 100.1.3.13.8 RGMII Fame Receive Timing (Carrier Sense)**

### (2) Accessing MII Management Registers

MII management registers in the PHY-LSI are accessed via PIR in this LSI. PIR is used as a serial interface conforming to the MII frame format specified in IEEE802.3u.

#### (a) MII Management Frame Format

Figure 100.1.3.13.9 shows the format of an MII management frame. To access an MII management register, a management frame is implemented by the program in accordance with the procedures shown in MII Management Register Access Procedure.

| Access Type | MII Management Frame | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Item | PRE | ST | OP | PHYAD | REGAD | TA | DATA | IDLE |
| Number of bits | 32 | 2 | 2 | 5 | 5 | 2 | 16 | |
| Read | 1..1 | 01 | 10 | 00001 | RRRRR | Z0 | D...D | |
| Write | 1..1 | 01 | 01 | 00001 | RRRRR | 10 | D...D | X |

[Legend]
PRE: 32 consecutive 1 s
ST: Write of B'01 indicating start of frame
OP: Write of code indicating access type
PHYAD: Write of B'00001 if the PHY-LSI address is 1 (sequential write starting with the MSB).
This bit changes depending on the PHY-LSI address.
REGAD: Write of 000q if the register address is 1 (sequential write starting with the MSB).
This bit changes depending on the PHY-LSI register address.
TA: Time for switching data transmission source on MII interface
(a) Write: 10 written
(b) Read: Bus release (notation: Z0) performed
DATA: 16-bit data. Sequential write or read from MSB
(a) Write: 16-bit data write
(b) Read: 16-bit data read
IDLE: Wait time until next MII management format input
(a) Write: Independent bus release (notation: X) performed
(b) Read: Bus already released in TA: control unnecessary

**Figure 100.1.3.13.9 MII Management Frame Format**

#### (b) MII Management Register Access Procedure

The program accesses MII management registers via PIR. Access is implemented by a combination of 1-bit-unit data write, 1-bit-unit data read, bus release, and independent bus release. Figure 100.1.3.13.10 to Figure 100.1.3.13.13 show the MII management register timing. The timing will differ depending on the PHY-LSI type.

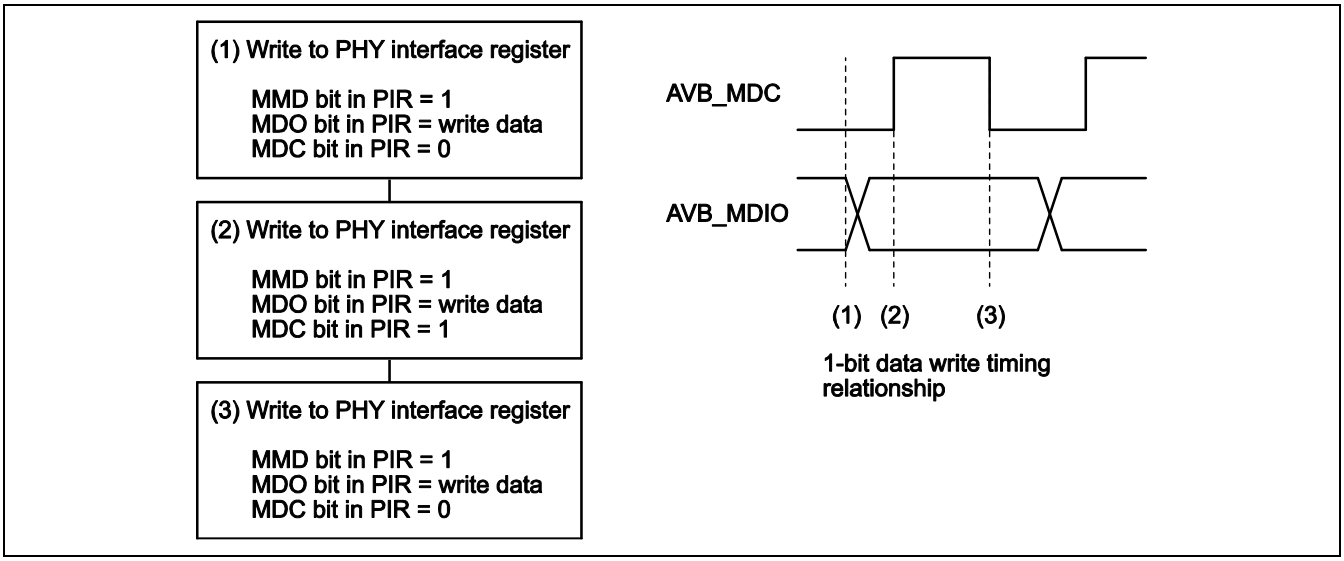
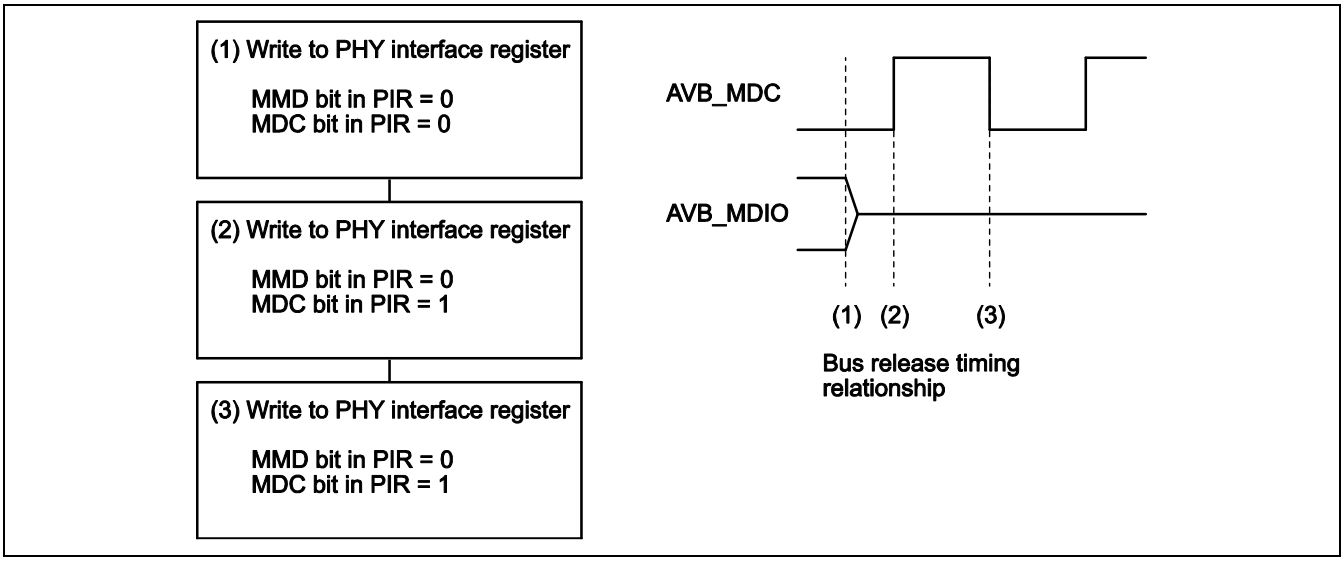



**Figure 100.1.3.13.10 1-Bit Data Write Flowchart**




Note: TA in Read in Figure 100.1.3.13.9

**Figure 100.1.3.13.11 Bus Release Flowchart**

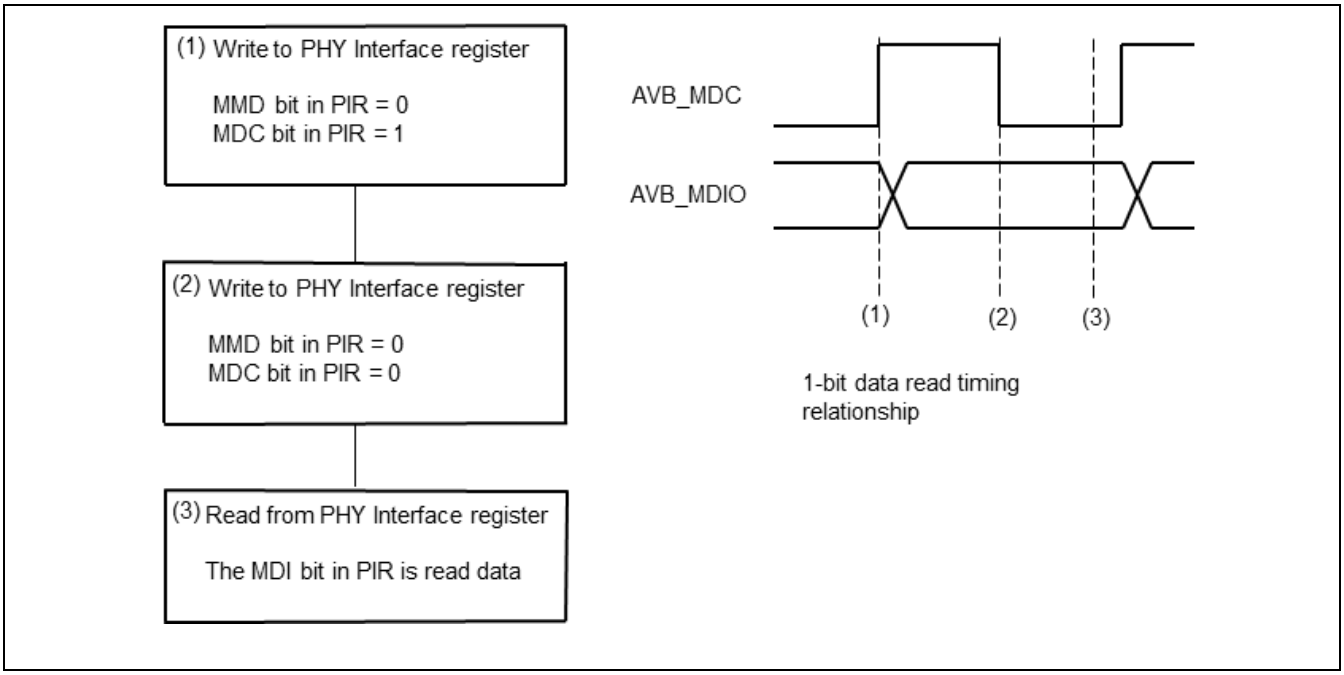
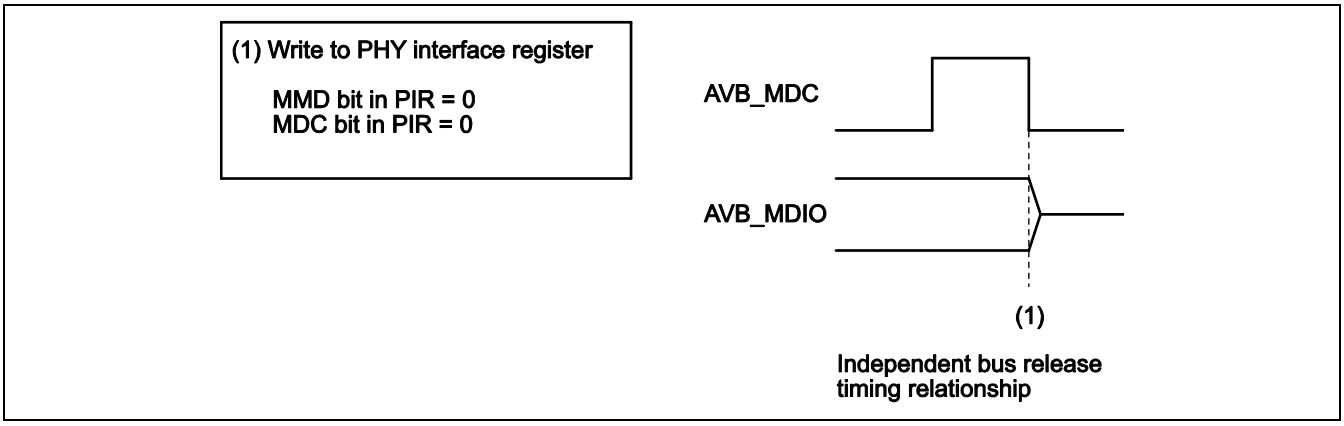



**Figure 100.1.3.13.12 1-Bit Data Read Flowchart**




Note: IDLE in Write in Figure 100.1.3.13.9

**Figure 100.1.3.13.13 Independent Bus Release Flowchart**

### 100.1.3.12 Usage Notes

**(a) Reception is Ignored when AVB_RX_CTL indicates an error from PHY is Flagged at 1st Data Byte**

The issue is limited to reception of frames with physical faults. Flagging of occurrence in SFR is missing. When EthernetAVB-IF is configured to store faulty frames in reception queue 0 (BE) by RCR.EFFS there is no storage.

**(b) Unexpected Data are Exported when AVB_RX_CTL indicates an error is Flagged at 8th Data Byte**

The issue is limited to reception of frames with physical faults.

Flagging of occurrence in SFR is invalid. When EthernetAVB-IF is configured to store faulty frames in reception queue 0 (BE) by RCR.EFFS corrupted data is stored and MAC status code is wrong.

**(c) Receive Frame Interrupt and Descriptor Interrupt may be Issued before Completion of Writing Data**

When receive frame interrupt and descriptor interrupt are issued, whether writing data is completed by DESCR.DT in a descriptor can be confirmed.

And CDARq register has the descriptor address which EthernetAVB-IF is processing currently or is going to process at the next time.

The descriptors before one indicated by CDARq register are scope for receive frame interrupt and descriptor interrupt scope.

If there is a descriptor which wait for HW processing (ex. FEMPTY), it's delayed. Confirm the descriptor again after waiting for 1ms.

### 100.1.3.13 Support for UDP/IP

#### (1) Reception Queues Selection

The AVB-DMAC applies its separation filtering mechanism to select the reception queue for storing a received frame. The AVB-DMAC stores all received frames in the URAM.

There are two conditions for the AVB-DMAC to discard a received frame.

- Detection of an error during reception by the E-MAC
  - Whether error frames are discarded or stored in reception queue 0 (best effort) depends on the setting of the error frame enable bit in the receive configuration register (RCR.EFFS). If error frames are to be stored (RCR.EFFS = 1), they are always stored in queue 0 (best effort). In this case, characteristics specific to the queue (e.g. truncation) will vary. If the storage of time stamps for reception queue 0 (best effort) is enabled (the time stamp enable bit in the receive configuration register RCR.ETS0 = 1), time stamps are stored even for error frames.
- Frame fails the separation filter
  - It depends on RCR.ESF if such frame is discarded or stored in receive queue 0 (best effort).

The flowchart in Figure 100.1.3.15.1 shows how the AVB-DMAC selects the reception queue in accord with the frame type, including judgment by the separation filter. Selection of the queue starts when the E-MAC completes frame reception. The result is storage of the frame in the proper queue or the frame being discarded.

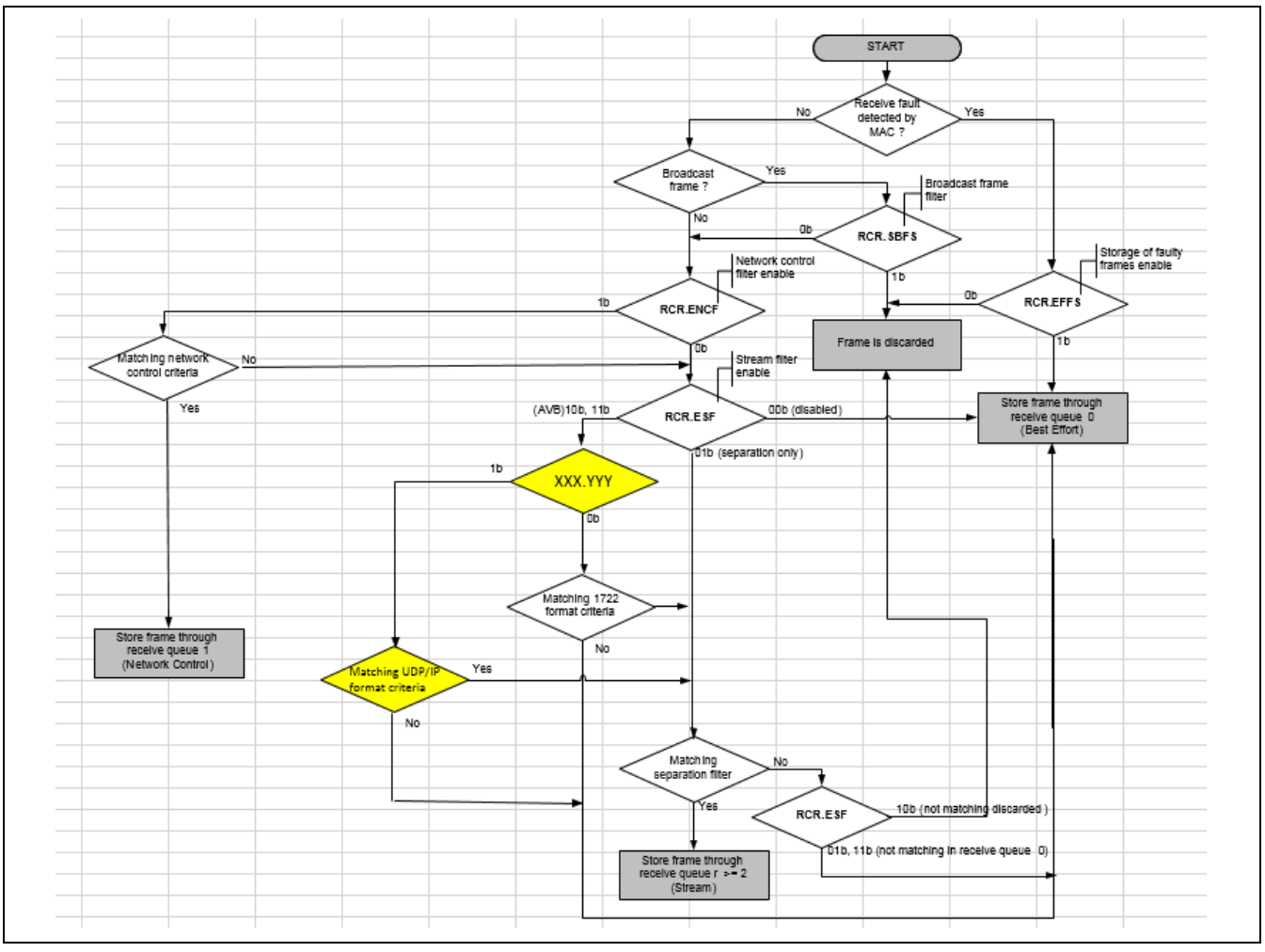



**Figure 100.1.3.15.1 Mechanism of Reception Queue Selection**

Notes on the meanings of entries in the flowchart

- "Condition for determining network control frames"
  The Ethernet destination address (DA) is 01:80:C2:00:00:0E.
  The Ethernet type (ET) is 88:F7.
- "Condition for determining IEEE 1722 frames"
  The Ethernet destination address (DA) is within the range from 91:E0:F0:00:00:00 to 91:E0:F0:00:FE:FF.
  The VLAN tagged TPID (tag protocol identifier) field (VL) is 81:00.
  The Ethernet type (ET) is 22:F0.
- "Condition for separation filter determination"
  See section 100.1.3.4(1)(a) Separation Filtering.

Figure 100.1.3.15.2 shows the allocation of bits related to the network and stream types in Ethernet frames. The preambles of Ethernet frames are not taken into account.

| Data bytes | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Network type | DA1 | DA2 | DA3 | DA4 | DA5 | DA6 | SA1 | SA2 | SA3 | SA4 | SA5 | SA6 | ET1 | ET2 | ... | ... | ... | ... | ... | ... |
| Stream type | DA1 | DA2 | DA3 | DA4 | DA5 | DA6 | SA1 | SA2 | SA3 | SA4 | SA5 | SA6 | VL1 | VL2 | - | - | ET1 | ET2 | ... | ... |

**Figure 100.1.3.15.2 Data Bytes of Ethernet Frames Used in Classification**

**(a) Separation Filtering**

Separation filtering involves the checking of up to 64 bits (eight successive bytes) in received Ethernet frames. The setting for the first byte (i.e. the setting of the separation filter offset configuration register (SFO.FBP)), selects the part of frames to be used in separation filtering. There is also a common filter mask (set in the separation filter mask configuration register (SFMi.CFM)) that can be freely set to reduce the number of bytes used in separation filtering or to mask particular bits.

**Examples**

To use one byte in separation, set separation filter mask configuration register 0 (SFM0.CFM) to H'0000 00FF and separation filter mask configuration register 1 (SFM1.CFM) to H'0000 0000.

To use seven bytes in separation, set separation filter mask configuration register 0 (SFM0.CFM) to H'FFFF FFFF and separation filter mask configuration register 1 (SFM1.CFM) to H'00FF FFFF.

Note: If bits at some positions are set to 0 in the separation mask, in order to match with the pattern, the bits at the corresponding positions of the pattern must also be set to 0. Only those bits in which the separation filter pattern configuration register (SFPi.FPs) setting is equal to the separation filter mask configuration register (SFMi.CFM) are sorted by matching with received data.

Figure 100.1.3.15.3 shows separation filtering. The selected data from a received frame (Rx_Frame[63:0]) are masked by the common filter mask. As a result, the selected frame data can be obtained. This value is compared with all filter patterns. The separation filter circuit in the AVB-DMAC selects the filter pattern that matches the queue having the lowest index s or selects a flag to indicate that there is no matching separation pattern.

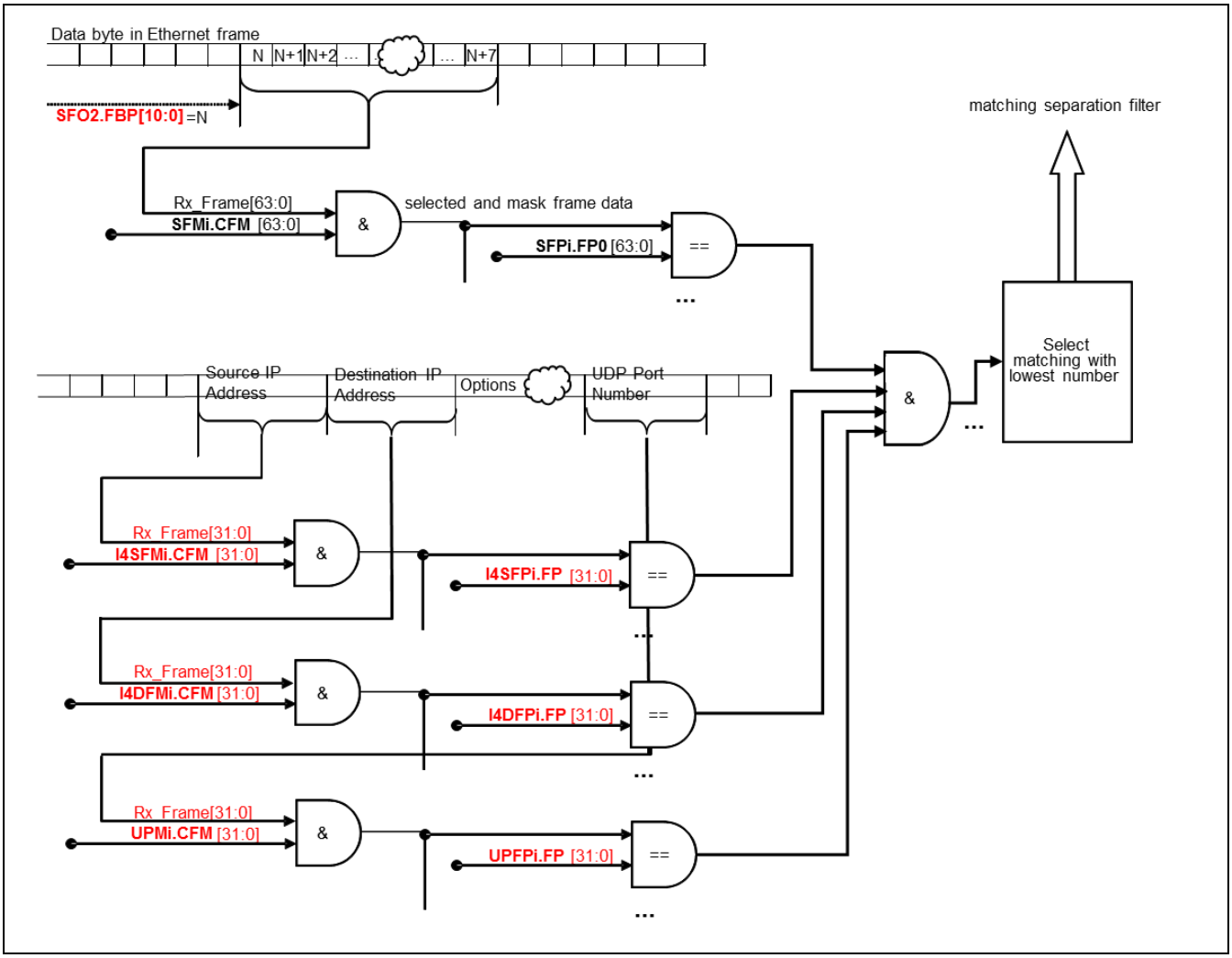



**Figure 100.1.3.15.3 Separation Filtering**

### (b) Separating Streams

The AVB-DMAC applies separation filtering to sort frames received in streams. An AVB network has a concept of “Talker” and “Listener”. A Talker is an end station that generates one or more streams. A Listener is an end station that has the role of being a sink for at least one stream. The various A/V streams are identified by 8-byte stream IDs.

The number of end stations within an AVB network and their roles differ with the application.

The stream ID is a general pattern of the AVB network for identifying one stream. Figure 100.1.3.4.5 shows the bit allocation of bits in IEEE1722 Ethernet frames and stream ID fields.

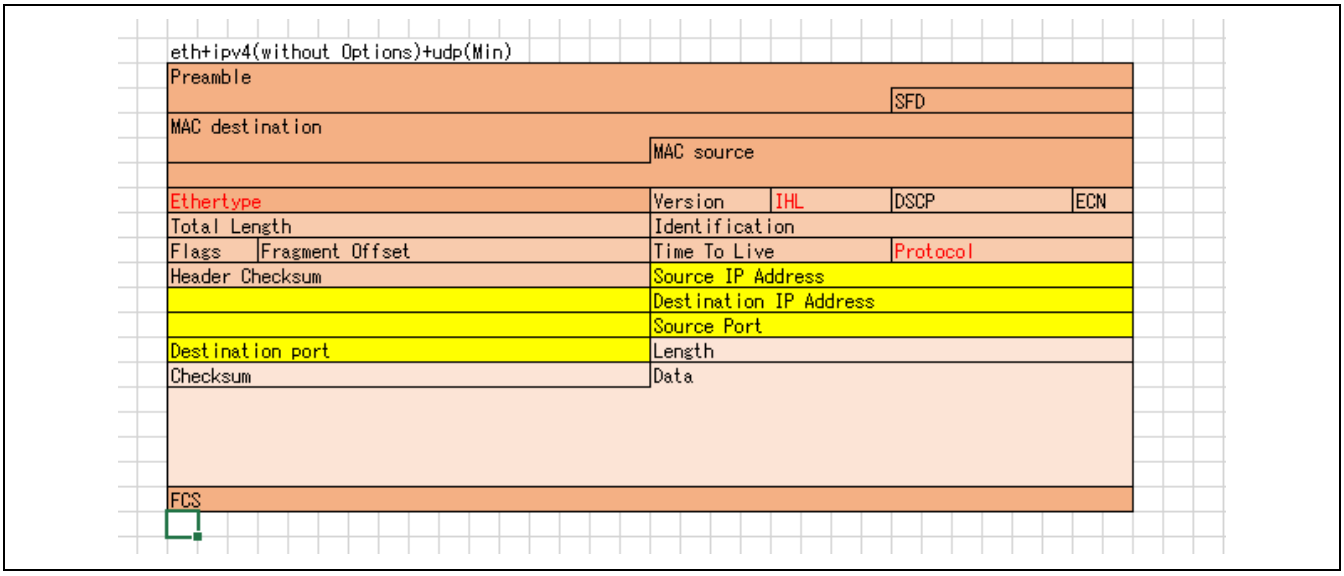



**Figure 100.1.3.15.4 With UDP/IPv4 Frame Layout**

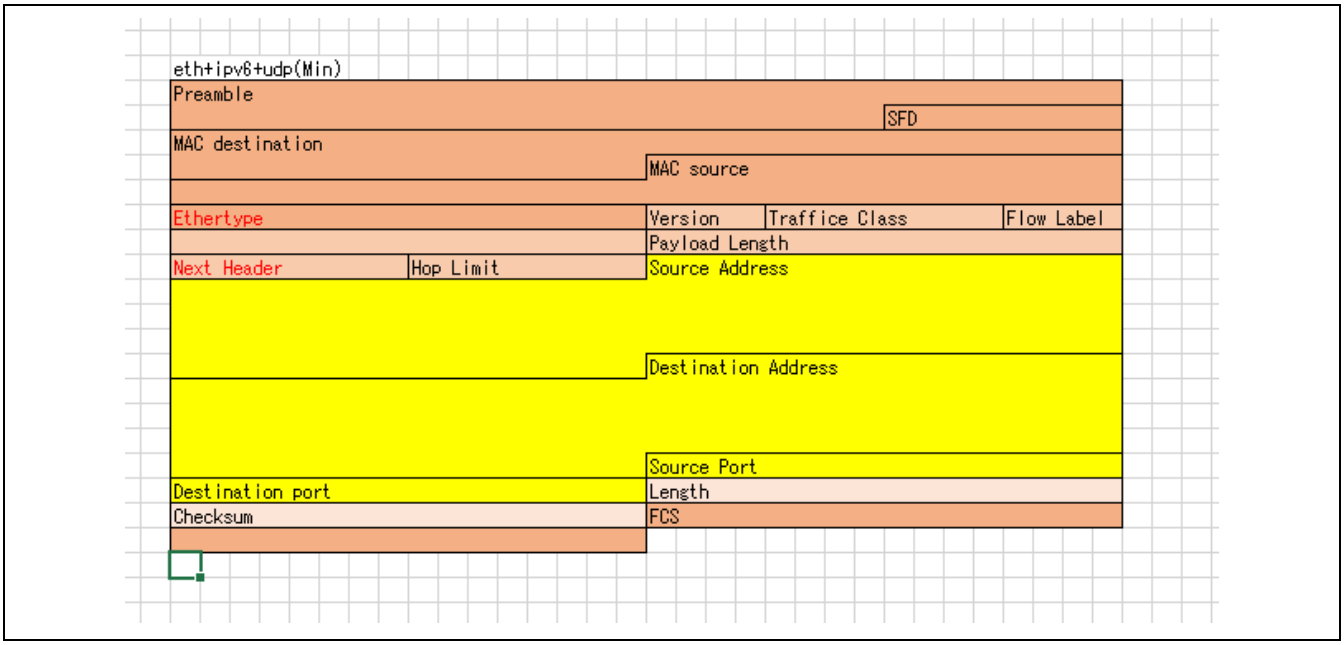



**Figure 100.1.3.15.5 With UDP/IPv6 Frame Layout**

The IEEE 1722 standard stipulates that the stream ID field starts from the 23rd byte (not counting the preamble). Accordingly, set the separation filter offset (SFO.FBP) to 22 in operations on IEEE 1722 streams. Set the separation filter mask (SFMi) and separation filter pattern (SFPi) in accord with the specification of the product in which the chip is being used.

Example: In the example of a stream ID shown in Figure 100.1.3.4.5, the current application divides the field into the talker source address and the unique stream ID. The unique ID is used to differentiate between multiple streams from the same talker. Based on this, there are two settings for separation filter masking:

- To divide various streams into individual queues, set SFM0.CFM to H'FFFF FFFF and SFM1.CFM to H'FFFF FFFF.
- To divide streams from various talkers into individual queues, set SFM0.CFM to H'FFFF FFFF and SFM1.CFM to H'0000 FFFF. This excludes the unique ID from the filter condition.

**(c) Reconfiguration of separation filter during OPERATION**

In CONFIG mode the initial separation filter configuration is defined. To prevent inconsistent filter results during OPERATION direct parameter change is not possible; instead update mechanism by **SFVi.LV** and **SFL.LC** should be used.

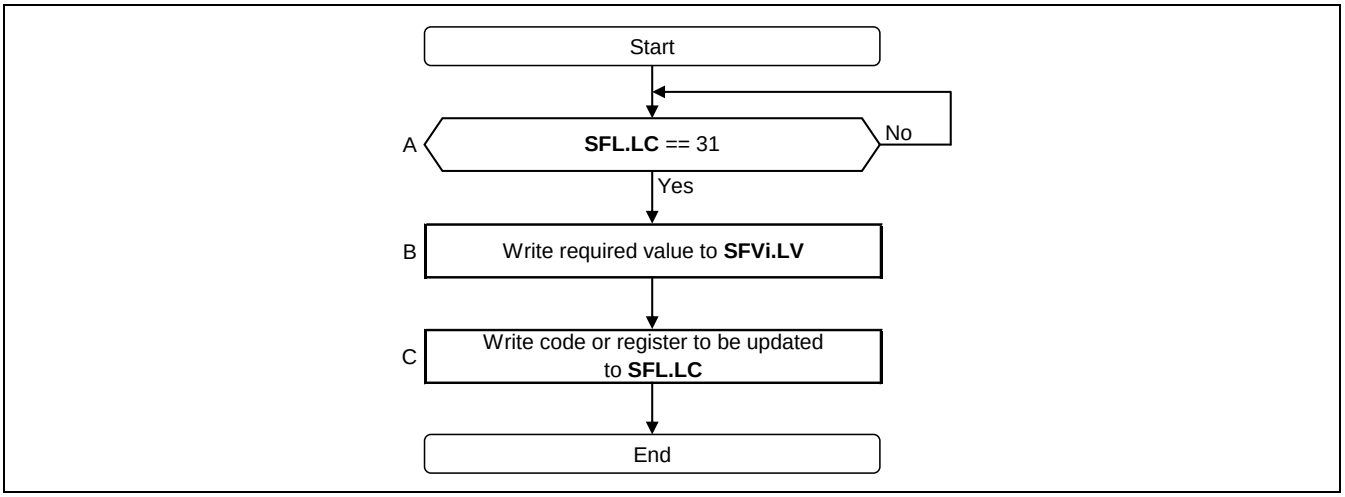



**Figure 100.1.3.15.6 SW flow to update separation filter during OPERATION**

When reconfigure separation filter, EthernetAVB-IF guarantees consistent classification for one received frame. If reconfiguration requires a sequence of parameter changes, CPU has to order reconfiguration based on application demand.
EthernetAVB-IF updates filter configurations when no frame reception is handled. In worse case the update is delayed by one frame reception. Exact duration depends on PHY speed and frame size. By reading **SFL.LC** CPU can observe when a previous update has been processed.
Note: Frame classification happens during frame reception and storage into internal Rx-FIFO. There may be frames with old classification pending in Rx-FIFO or under storage to URAM when filter reconfiguration is accepted.

**(d) Disable Manual Offset Mode**

Enable Manual Offset Mode

0: Manual Offset Mode disabled
1: Manual Offset Mode enabled

When this bit is 0, These bits set the position in Ethernet frames of the first byte of the bytes(see section (a) Disable Manual Offset Mode) to be used by the separation filter. A fixed value by rx_data[31:16] is selected as shown in the table below.

| mir_macrx_data[31:16] | IPv4 Source Address Filter Offset | IPv4 Destination Address Filter Offset | IPv6 Source Address Filter Offset | IPv6 Destination Address Filter Offset | UDP Port Number Filter Offset |
|---|---|---|---|---|---|
| 16'h8100 (802.1) | (ipv4_ihl << 2) + 10'd10 | (ipv4_ihl << 2) + 10'd14 | 10'd26 | 10'd42 | 10'd58 |
| 16'h0800 (ipv4) | (ipv4_ihl << 2) + 10'd6 | (ipv4_ihl << 2) + 10'd10 | 10'd22 | 10'd38 | (ipv4_ihl << 2) + 10'd14 |
| others | (ipv4_ihl << 2) + 10'd6 | (ipv4_ihl << 2) + 10'd10 | 10'd22 | 10'd38 | 10'd54 |

When this bit is 1, These bits set the position in Ethernet frames of the first byte of the bytes**(I4SFO.FBP, I4DFO.FBP, I6SFO.FBP, I6DFO.FBP and UPFO.FBP)** to be used by the separation filter.

| IPv4 Source Address Filter Offset | IPv4 Destination Address Filter Offset | IPv6 Source Address Filter Offset | IPv6 Destination Address Filter Offset | UDP Port Number Filter Offset |
|---|---|---|---|---|
| I4SFO.FBP Register | I4DFO.FBP Register | I6SFO.FBP Register | I6DFO.FBP Register | UPFO.FBP Register |

### (2) Checksum

#### (a) UDP/IPv4

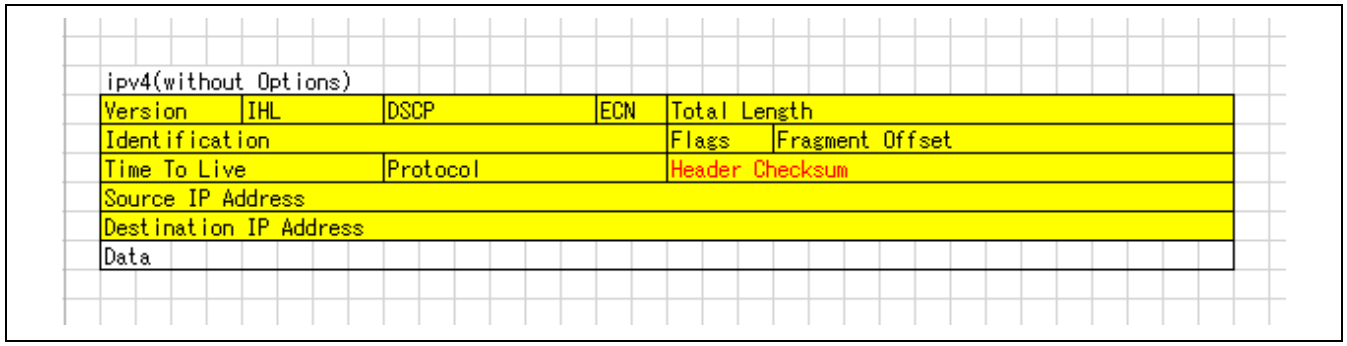



Figure 100.1.3.15.7 IPv4 checksum

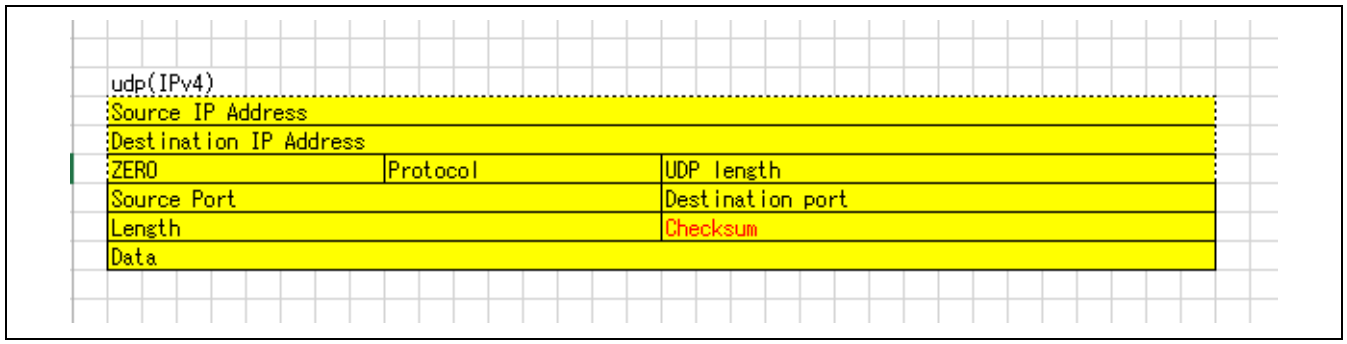



Figure 100.1.3.15.8 UDP/IPv4 checksum

#### (b) UDP/IPv6

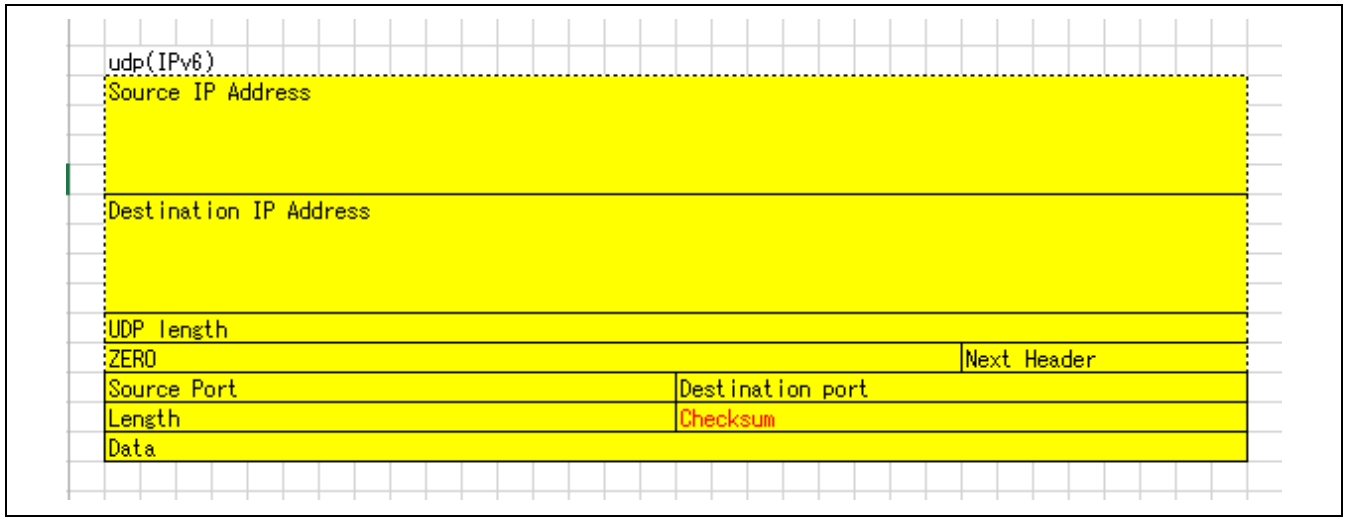



Figure 100.1.3.15.9 UDP/IPv6 checksum

#### (c) Configuration of Reception Frame Data Descriptors

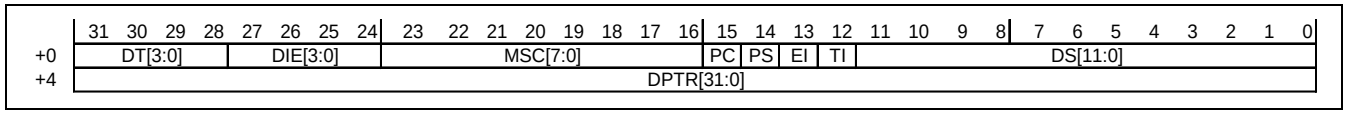



Figure 100.1.3.15.10 Configuration of Descriptor for a Received Frame

**Table 100.26 Configuration of a Received Descriptor**

| Bit Name | Function |
|---|---|
| MSC | E-MAC Status Code<br>These bits indicate errors in reception detected by the E-MAC.<br>In the case of a divided frame, these bits are set to the same value within all descriptors for the frame data. Details of the bits are as follows.<br>MSC[7]: Received frame has a multicast address.<br>MSC[6]: Check-sum error.<br>MSC[5]: Received Unsupported frame.<br>MSC[4]: Received frame has alignment error.<br>MSC[3]: Received frame is too long.<br>MSC[2]: Received frame is too short<br>MSC[1]: Error in frame reception (PHY detects an error)<br>MSC[0]: Received frame has a CRC error. |
| PC | Refer Table 100.16. |
| PS | Refer Table 100.16. |
| EI | Refer Table 100.16. |
| TI | Refer Table 100.16. |

## 100.1.4 Usage Notes

### 100.1.4.1 Checksum Calculation of Ethernet Frames

This LSI is capable of calculating the checksum data of the received frames. Only the data fields of the Ethernet frames are subject to checksum calculation. Specifically, a data field follows the length/type field and is followed by the CRC field. Figure 100.1.4.1.1 shows schematics indicating which parts of the Ethernet frames are calculated. Calculation involves 16-bit addition only; it does not involve bit inversion. Note that when the checksum data is valid, the CRC data (4 bytes) is not transferred as a receive frame, and the checksum data (sum data) is added automatically. Figure 100.1.4.1.2 shows schematics of Ethernet frames to which the checksum data has been added.

Note: Also for the frames with VLANtag inserted, the 15th byte from the top and the following bytes before the CRC field are subject to calculation.

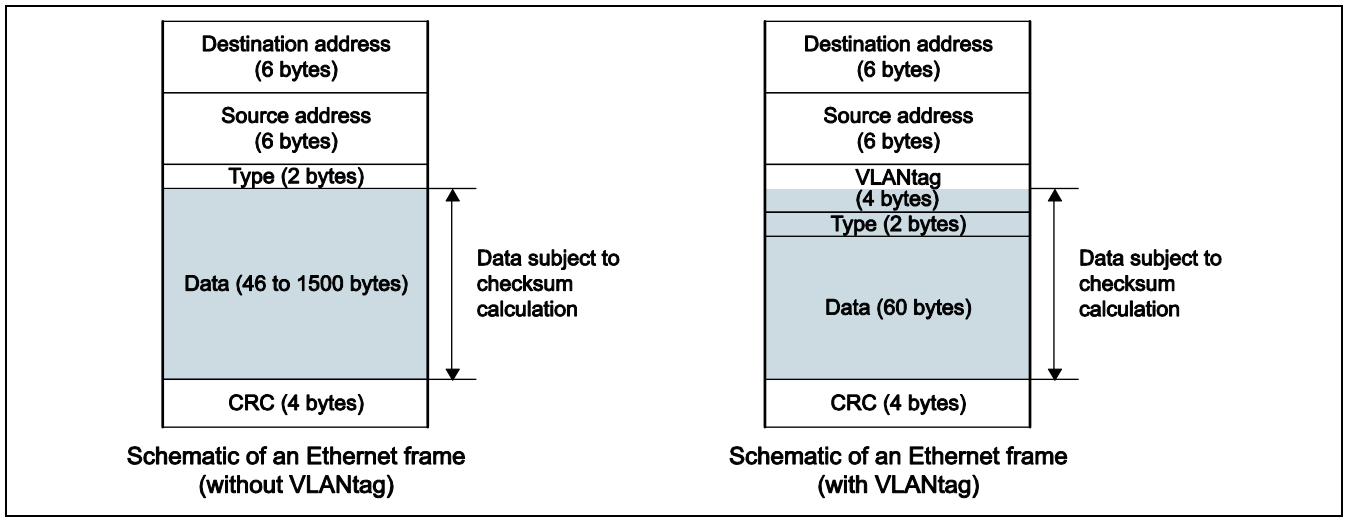



**Figure 100.1.4.1.1 Data Subject to Checksum Calculation**

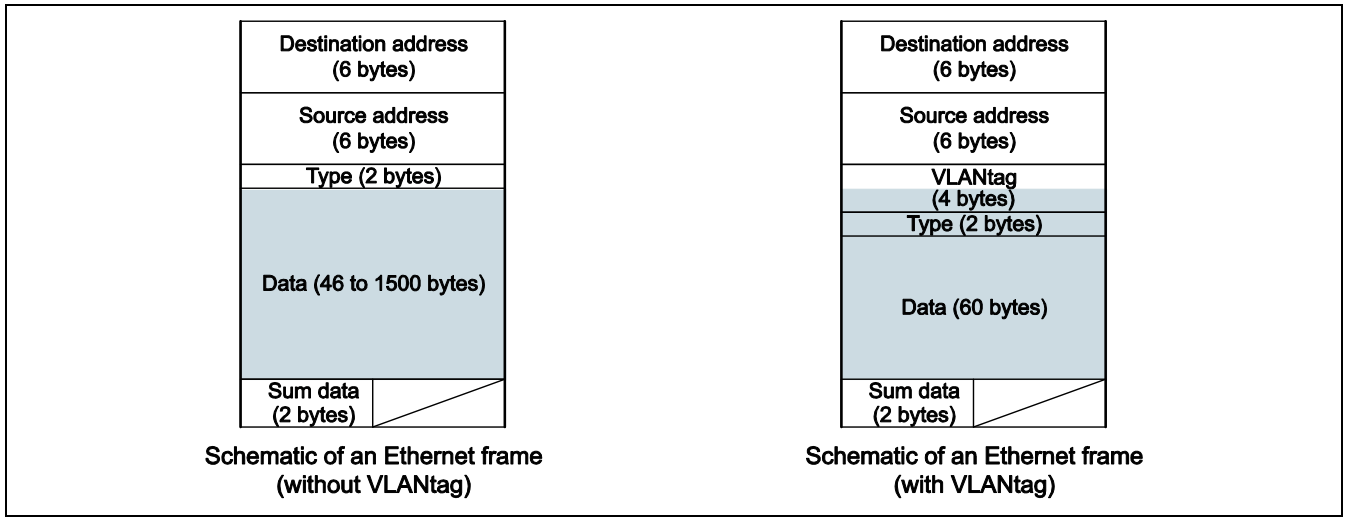



**Figure 100.1.4.1.2 Data after Checksum Data Addition**

### 100.1.4.2 Data Transfer Function Stops after Read Access Error

If EthernetAVB-IF gets an access error when reading from work RAM, it may happen that no further work RAM accesses are issued. So transfer of receive/transmit frames is halted. By using the data transfer suspend function SW can observe if data transfer function is available or not.

Perform one of given SW flows which are triggered by this start condition: when AXI read access error has been flagged by EIS.QEF is 1, and ESR.ET is B'0000 or B'0100; additionally when ESR.EIL is 1 (this is an indication that an work RAM read error may be lost).

There are two possible SW flows given. Figure 100.1.4.2.1 applies always the SW reset, independently if this is required or not. Figure 100.1.4.2.2 checks in addition if SW reset is required; checking CSR.DTS after wait gives an indication that issue has been occurred.

Note that the wait after write of CCC.DTSR is mandatory to prevent cases where issue has not been occurred.

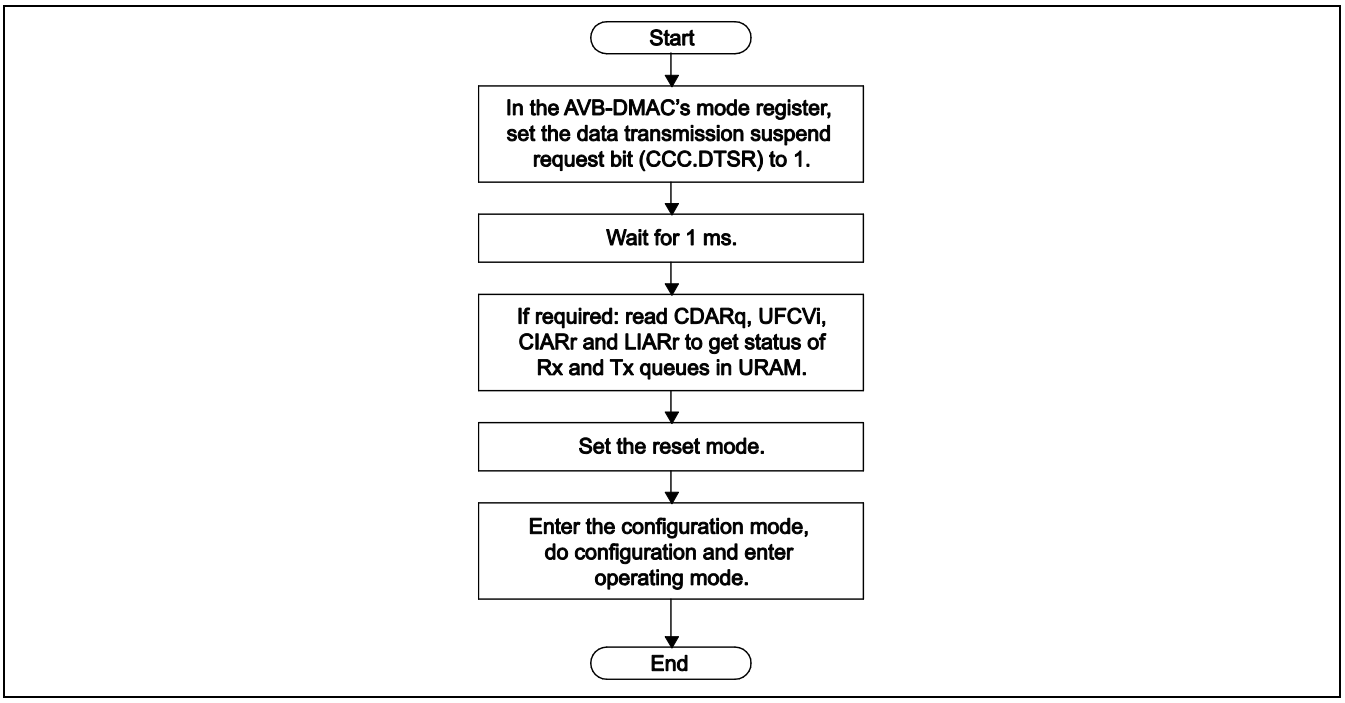



**Figure 100.1.4.2.1 Data Transfer Function Stop Flow1 after Read Access Error**

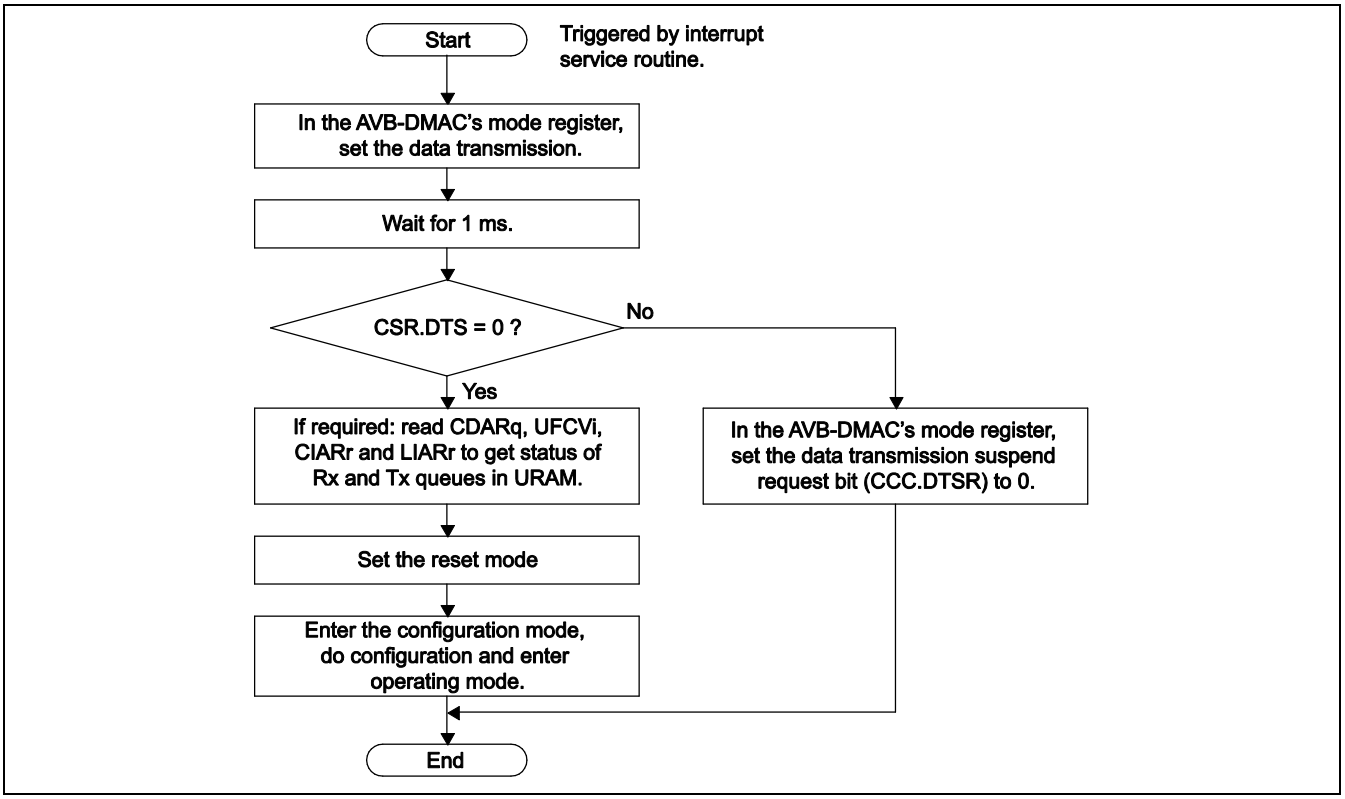



**Figure 100.1.4.2.2 Data Transfer Function Stop Flow2 after Read Access Error**

### 100.1.4.3 Power Down Request

If EthernetAVB-IF gets a power down request while writing to work RAM, it may happen that the request is not accepted even if the transfer is completed. Due to this the external power down module cannot complete the power down sequence.

A power down request is acceptable only when EthernetAVB-IF is in RESET state.

### 100.1.4.4 MII Interface

#### (1) Specifications of the MII in Outline

The media independent interface (MII) is a specification for use in connection between the MAC and PHY layers of an Ethernet communications interface. The MII is originally defined in IEEE802.3 as a standard interface to connect a Ethernet media access control (MAC) block to a PHY chip at 100Mbps.
MII supports 3.3V CMOS interface voltage for all IO pins and 25MHz reference clock for 100Mbps communication.

**(a)** Signal Line Specification in the MII

The MII uses fourteen signals to connect the MAC and PHY layers. With an MII, the contents of the data and control signals for transmission and reception change on rise edges of the respective clock signals.

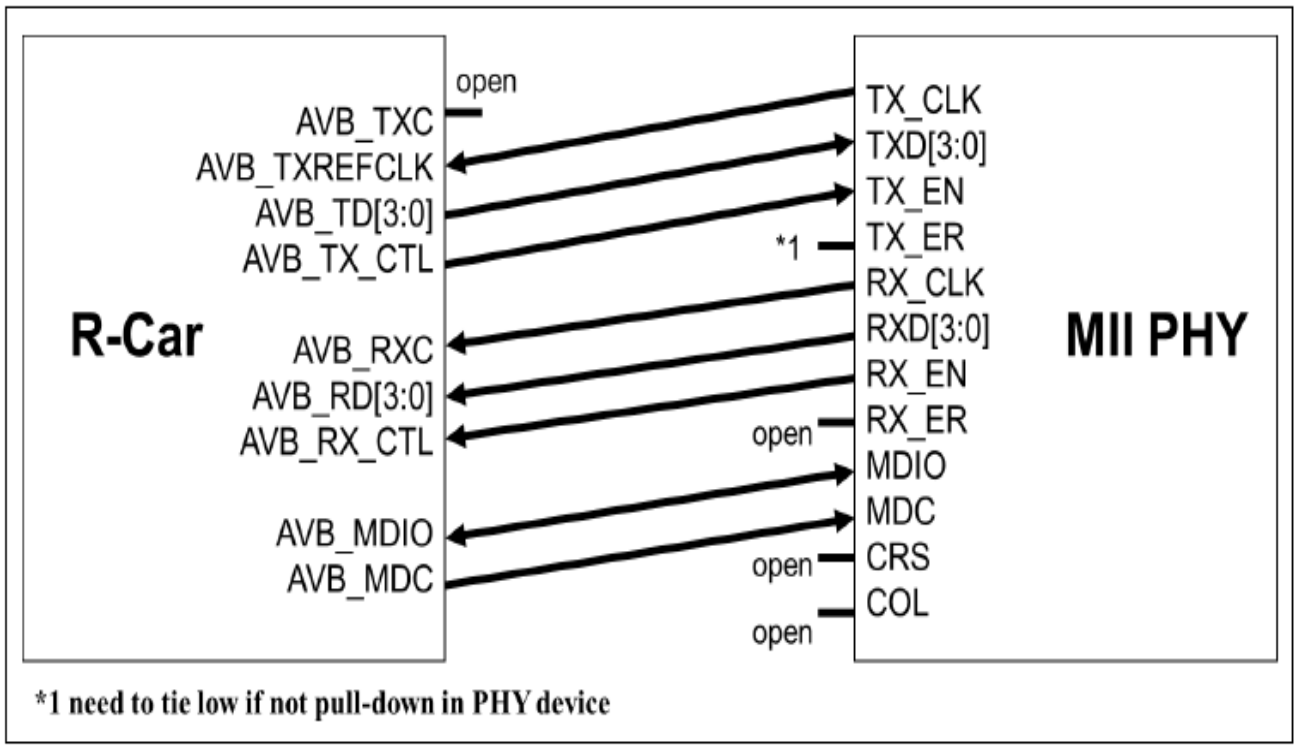
Definitions of Signals Used in an MII is shown below:

| MII Signal | Source | Sink | Description |
|---|---|---|---|
| TXC | PHY | MAC | Reference clock for use in transmission. It runs at 25MHz. |
| TD[3:0] | MAC | PHY | Data being transmitted. The meanings of the signals differ with the edge of the TXC signal. |
| TX_EN | MAC | PHY | Control signal for transmission. The meaning of the signal differs with the edge of the TXC signal. |
| RXC | PHY | MAC | Reference clock for use in reception. It runs at 25MHz. |
| RD[3:0] | PHY | MAC | Data being received. The meanings of the signals differ with the edge of the RXC signal. |
| RX_EN | PHY | MAC | Control signal for reception. The meaning of the signal differs with the edge of the RXC signal. |
| MDIO | MAC or PHY | MAC or PHY | IO signal for control over the PHY layer. |
| MDC | MAC | PHY | Clock signal for control over the PHY layer. |
| TX_ER | MAC | PHY | Transmit enable |
| RX_ER | PHY | MAC | Receive error |
| CLS | PHY | MAC | Carrier sense |
| COL | PHY | MAC | Collision detect |

**(b) External Devices**

We assume that two types of device, PHY devices and switches, may be externally connected to the MII of an R-Car device. Note that the MII connections to both types of device are handled in the same way.

• PHY device
This type of device is used for systems with a single port directly connected to an Ethernet cable.




### (2) Specifications for the MII in R-Car Series Devices

#### (a) Specifications for the MII in R-Car Series Devices

The correspondence between pins of R-Car devices and MII signals is given in below table. MII Signal “TXC” should be connected to “AVB_TXCREFCLK” in R-Car. “AVB_TXC” is not used in MII connection mode.

| MII Signal | R-Car Signal | Source | Sink |
|---|---|---|---|
| TX_CLK | AVB_TXCREFCLK | PHY | MAC |
| TXD[3:0] | AVB_TD[3:0] | MAC | PHY |
| TX_EN | AVB_TX_CTL | MAC | PHY |
| TX_ER | — | MAC | PHY |
| RX_CLK | AVB_RXC | PHY | MAC |
| RXD[3:0] | AVB_RD[3:0] | PHY | MAC |
| RX_DV | AVB_RX_CTL | PHY | MAC |
| RX_ER | — | PHY | MAC |
| MDIO | AVB_MDIO | MAC or PHY | MAC or PHY |
| MDC | AVB_MDC | MAC | PHY |
| CRS | — | PHY | MAC |
| COL | — | PHY | MAC |

#### (b) Clock setting

RGMII mode and MII mode can be switched by AVB-DMAC Product Specific Register (APSR)：H'E68n 008C (n=0-2).

Bit24 of APSR should be set “1” to change RGMII mode(default) to MII mode. The value should be fixed before starting communication.

**(c) Data rate setting**

Because MII is limited at 100Mbps, it should be set 100Mbps. Bit0 of E-MAC Mode Register 2 should be set “0” to change 1000Mbps to 100Mbps. The value should be fixed before starting communication.

### (3) Timing Specification: MII Timing

Timing specification between R-Car and PHY with MII communication is showed in below table. R-Car is not fully compliant with MII standard pin. Therefore, TX_ER, RX_ER, CLS and COL is not supported and undefined in timing specification.

Conditions: VDDQ18_25_AVB = 1.8/2.5V Tj= -40 to +125℃ GND = VSS = 0 V, CL = 30 pF

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| AVB_TXCREFCLK input frequency | fTx | — | 25 ±100ppm | — | MHz | — |
| AVB_TX_CTL output delay | tTEND | 0 | — | 25 | ns | Figure 100.1.4.4.1 |
| AVB_TD[3:0] output delay | tETDD | 0 | — | 25 | ns | |
| AVB_RXC input frequency | fRx | — | 25 ±100ppm | — | MHz | — |
| AVB_RX_CTL setup time | tRDVS | 10 | — | — | ns | Figure 100.1.4.4.2 |
| AVB_RX_CTL hold time | tRDVH | 10 | — | — | ns | |
| AVB_RD[3:0] setup time | tERDS | 10 | — | — | ns | |
| AVB_RD[3:0] hold time | tERDH | 10 | — | — | ns | |

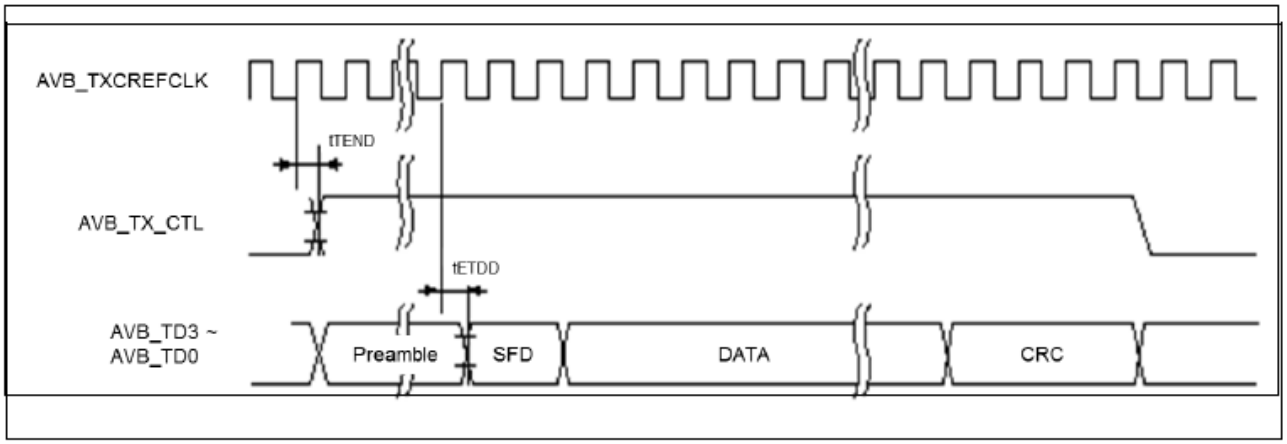



Figure 100.1.4.4.1 MII Connection Mode Transmission Timing

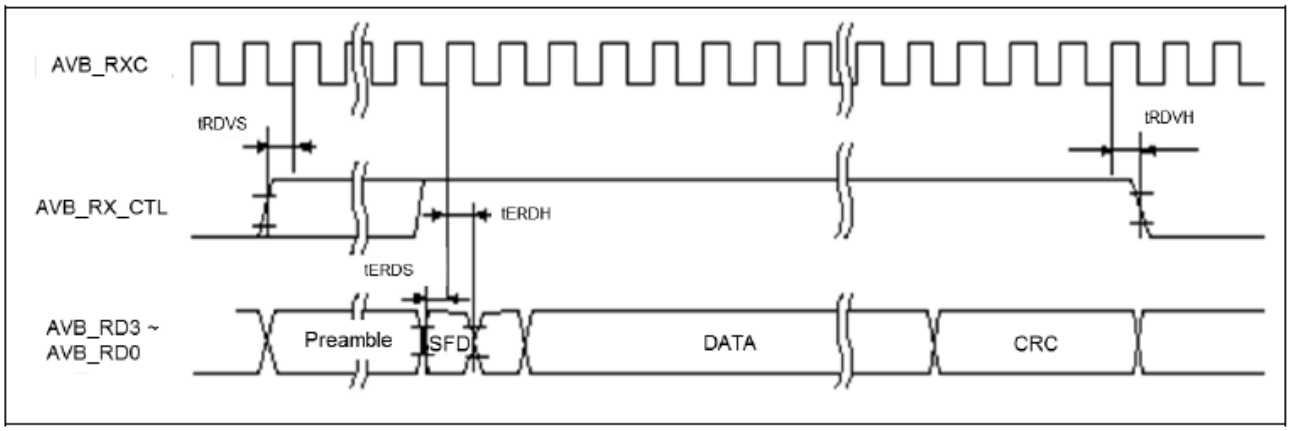



Figure 100.1.4.4.2 MII Connection Mode Reception Timing

# 100.2 gPTP

## 100.2.1 Overview

The gPTP module is described in the gPTP specification. Refer to "100_2_uciapgptp020(gPTP)_IPSpec_v1.01.pdf".

# 100.3 RGMII

## 100.3.1 Overview

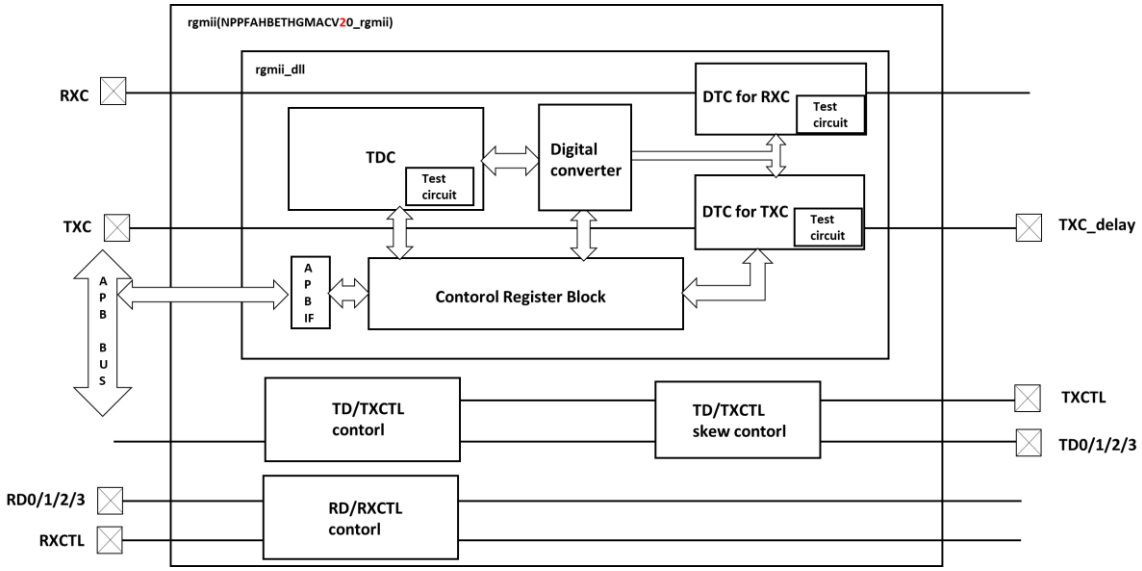
RGMII Interface logic.

Ethernet network using a 10 Mbps, 100 Mbps, or 1 Gbps full-duplex link via RGMII. Depending on the connected PHY.

AC spec covers RGMII Specification V2.0 and OPEN Alliance RGMII EPL v2.2

- Parallel loopback mode for AC measure and sort.
- Internal delay is adjusted with delay calibration circuit.
- Offset adjustment function for software workaround.
- Delay can be changed by software.

### 100.3.1.1 Block diagram



### 100.3.1.2 Register Configuration

| **Ch** | **Base Address** |
|---|---|
| AVB0 (EthernetAVB-IF 0) | H'E680 1000 |
| AVB1 (EthernetAVB-IF 1) | H'E681 1000 |
| AVB2 (EthernetAVB-IF 2) | H'E682 1000 |

| Name | Abbreviation | R/W | Offset Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| TDC enable register | TDCEN | R/W | H'0004 | H'0000 0000 | 32 |
| Offset value setting register | OFFSET | R/W | H'0008 | H'0000 0000 | 32 |
| Select output register | SEL_OUT | R/W | H'002C | H'0000 0020 | 32 |
| DTCDEC(TX) manual mode register | DTCDEC_TX_MANUAL_MODE | R/W | H'0040 | H'0000 0000 | 32 |
| DTCDEC(TX) config register | DTCDEC_TX_CONFIG | R/W | H'0044 | H'0010 4010 | 32 |
| DTCDEC(RX) manual mode register | DTCDEC_RX_MANUAL_MODE | R/W | H'0080 | H'0000 0000 | 32 |
| DTCDEC(RX) config register | DTCDEC_RX_CONFIG | R/W | H'0084 | H'0010 4010 | 32 |
| dTDC monitor register | DTDC | R/W | H'00A0 | H'0000 0000 | 32 |

## 100.3.2 Register Description

The registers of the RGMII are assigned to and located in the address space of the internal bus.

[Legend for Register Descriptions]

Initial value: Register value after a reset.

—: Undefined value

R/W: Readable/writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

R/WC1: Readable/writable. Writing 1 initializes the bit. Writing 0 is ignored.

R: Read-only. The write value should always be initial value.

−/WB: Write-only. The read value is undefined.

### 100.3.2.1 TDCEN Register (TDCEN)

TDCEN is a 32-bit register that controls the enable/disable of the TDC circuit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | tdcen |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 0 | tdcen | 0 | R/W | TDC circuit enable bit<br>0 : TDC circuit disable<br>1 : TDC circuit enable |

### 100.3.2.2 OFFSET Register (OFFSET)

OFFSET is a 32-bit register to specify the Offset value for TDC circuit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | offset [7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 7 to 0 | offset [7:0] | H'00 | R/W | TDC offset delay value setting<br>H'00 : Shortest offset delay<br>…<br>H'FF : Longest offset delay |

### 100.3.2.3 DTCDEC_TX_MANUAL_MODE Register (DTCDEC_TX_MANUAL_MODE)

DTCDEC_TX_MANUAL_MODE is a 32-bit register to specify delay value of DTC for TXC manually.

If Manual mode is selected, delay converter result is ignored.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | tx_manual_sel_mode | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | tx_manual_tap_num [6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | tx_manual_sel_mode | 0 | R/W | TXC DTC tap selection mode<br>0 : Auto mode<br>1 : Manual mode |
| 30 to 7 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 6 to 0 | tx_manual_tap_num [6:0] | H’00 | R/W | DTC offset delay value setting for TXC.<br>If TX_MAN_SEL=1, this value is used for DTC delay value.<br>H’00 : Shortest offset delay<br>…<br>H’7F : Longest offset delay |

### 100.3.2.4 DTCDEC_TX_CONFIG Register (DTCDEC_TX_CONFIG)

DTCDEC_TX_CONFIG is a 32-bit register to specify the calculation value for delay decoder circuit.

If “Auto mode” of DTCDEC_TX_MANUAL_MODE is selected, this result is used for DTC delay value.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | tx_offset2 [4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | tx_mult [7:0] | | | | | | | | tx_offset1 [7:0] | | | | | | | |
| Initial value: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 30 to 21 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 20 to 16 | tx_offset2 [4:0] | H’10 | R/W | Delay decoder offset2 value<br>H’00 : Shortest offset delay<br>…<br>H’1F : Longest offset delay |
| 15 to 8 | tx_mult [7:0] | H’40 | R/W | Delay decoder dividing value of TDC result<br>Set target delay value point of delay decorder.<br>Initial value is 64/256 = 1/4 period.<br>dDTC= (dTDC+offset1[7:0]-128)×TX_MULT[7:0]/256+offset2[4:0]-16<br>dDTC : DTC delay value.<br>dTDC : The result of TDC to measure 1 period. |
| 7 to 0 | tx_offset1 [7:0] | H’10 | R/W | Delay decoder offset1 value<br>H’00 : Shortest offset delay<br>…<br>H’FF : Longest offset delay |

Note : Recommended setting of DTCDEC_TX_CONFIG

| | | DTCDEC_TX_CONFIG |
|---|---|---|
| 1.8V | AVB0 | H’014880 |
| | AVB1 | H’0F4840 |
| | AVB2 | H’104840 |
| 2.5V | AVB0 | H’034880 |
| | AVB1 | H’114840 |
| | AVB2 | H’004880 |

### 100.3.2.5 DTCDEC_RX_MANUAL_MODE Register (DTCDEC_RX_MANUAL_MODE)

DTCDEC_RX_MANUAL_MODE is a 32-bit register to specify delay value of DTC for RXC manually.

If Manual mode is selected, delay converter result is ignored.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | rx_manual_sel_mode | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

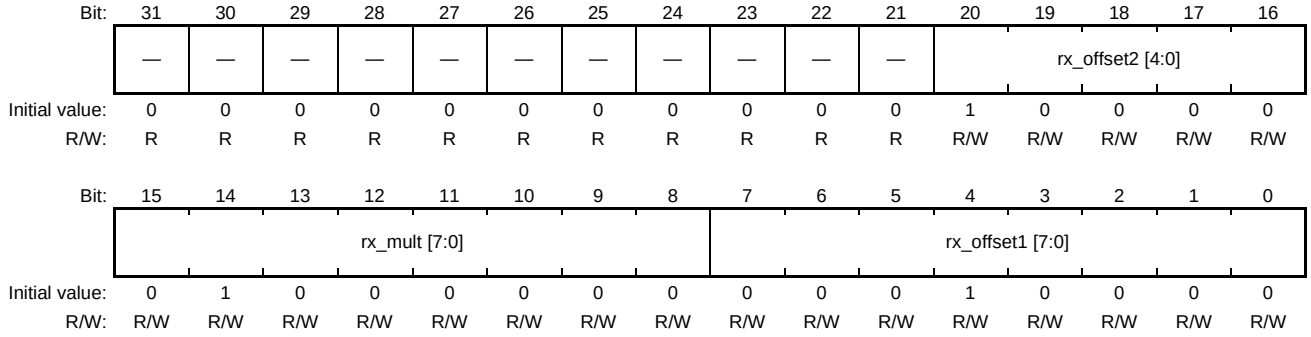
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | rx_manual_tap_num [6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | rx_manual_sel_mode | 0 | R/W | RXC DTC tap selection mode<br>0 : Auto mode<br>1 : Manual mode |
| 30 to 7 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 6 to 0 | rx_manual_tap_num [6:0] | H'00 | R/W | DTC offset delay value setting for RXC.<br>If RX_MAN_SEL=1, this value is used for DTC delay value.<br>H'00 : Shortest offset delay<br>…<br>H'7F : Longest offset delay |

### 100.3.2.6 DTCDEC_RX_CONFIG Register (DTCDEC_RX_CONFIG)

DTCDEC_RX_CONFIG is a 32-bit register to specify the calculation value for delay decoder circuit.

If “Auto mode” of DTCDEC_RX_MANUAL_MODE is selected, this result is used for DTC delay value.



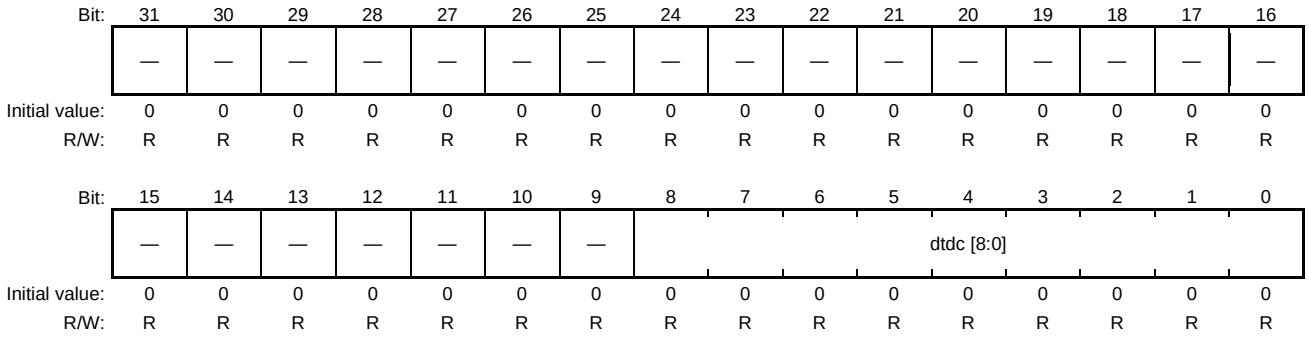

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 30 to 21 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 20 to 16 | rx_offset2 [4:0] | H’10 | R/W | Delay decoder offset2 value<br>H’00 : Shortest offset delay<br>…<br>H’1F : Longest offset delay |
| 15 to 8 | rx_mult [7:0] | H’40 | R/W | Delay decoder dividing value of TDC result<br>Set target delay value point of delay decorder.<br>Initial value is 64/256 = 1/4 period.<br>dDTC= (dTDC+offset1[7:0]-128)×RX_MULT[7:0]/256+offset2[4:0]-16<br>dDTC : DTC delay value.<br>dTDC : The result of TDC to measure 1 period. |
| 7 to 0 | rx_offset1 [7:0] | H’10 | R/W | Delay decoder offset1 value<br>H’00 : Shortest offset delay<br>…<br>H’FF : Longest offset delay |

Note : Recommended setting of DTCDEC_RX_CONFIG

| | | DTCDEC_RX_CONFIG |
|---|---|---|
| 1.8V | AVB0 | H’114840 |
| | AVB1 | H’014880 |
| | AVB2 | H’0C4840 |
| 2.5V | AVB0 | H’014880 |
| | AVB1 | H’044880 |
| | AVB2 | H’0F4840 |

### 100.3.2.7 DTDC Register (DTDC)

DTDC is a 32-bit register to monitor the measurement result of TDC circuit.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 8 to 0 | dtdc [8:0] | H'000 | R | TDC measurement result of 1 period.<br>H'000 : Shortest TDC delay is selected<br>…<br>H'1FF : Longest TDC delay is selected |

### 100.3.2.8 Select output signal register (SEL_OUT)

SEL_OUT is a 32-bit register that controls the signal of the test function.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

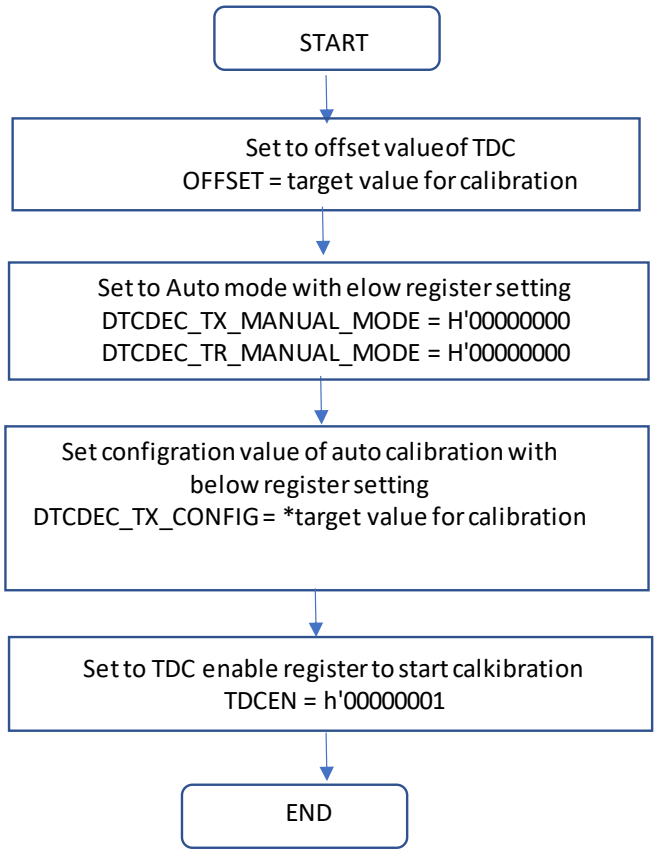
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | PLL mode | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 5 | — | 1 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 4 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |
| 3 | PLL mode | 0 | R/W | REFCLK selection<br>0 : TXC clock = Normal mode<br>1 : TXC clock = PLL mode |
| 2 to 0 | — | 0 | R | Reserved<br>These bits are always read as initial value. The write value should always be initial value. |

## 100.3.3 Operation

### 100.3.3.1 Initialize sequence with Auto calibration

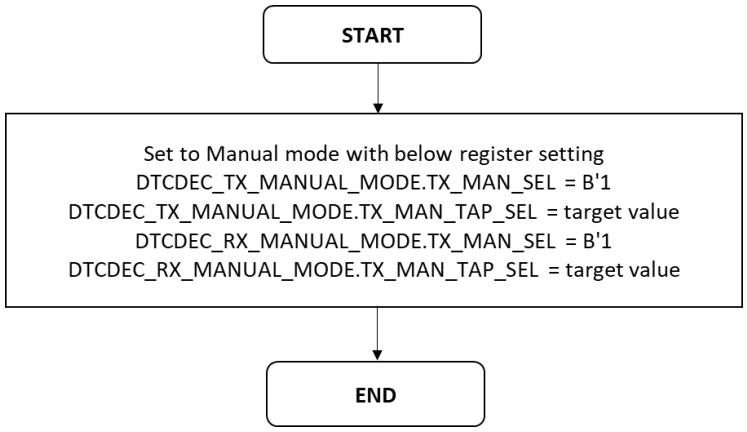
Refer following figure to initialize and activate sequence of this auto calibration function.




* TX/RX_OFFSET1 value shoukd be set ti same vlue with OFFSET register

### 100.3.3.2 Initialize sequence without Auto calibration

Refer following figure to initialize and activate sequence without auto calibration function.



### 100.3.3.3 PLL mode

[Block diagram of delay block with PLL mode circuit]
PLL mode use PLL7 for TXC clock.

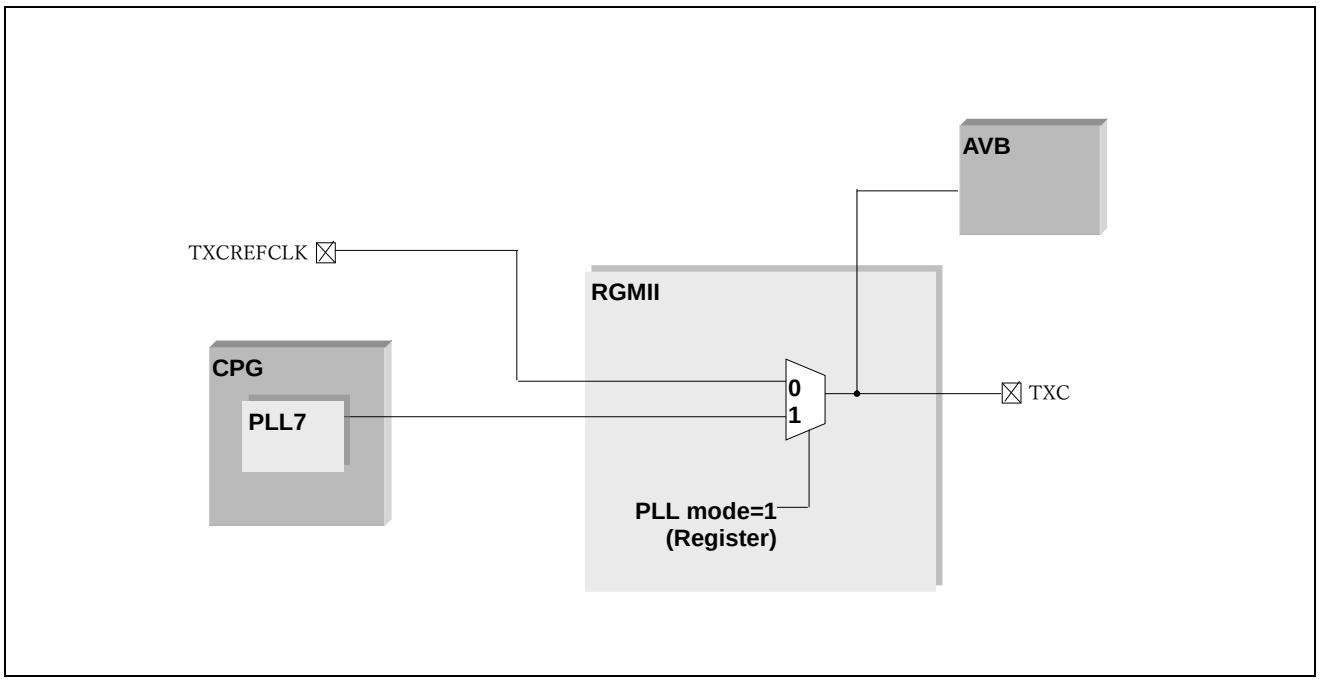



Figure 100.1 Block diagram of PLL mode

[Initialization procedure of PLL mode]
Enable PLL7 and set RGMII PLL mode=1 before RGMII initialization.

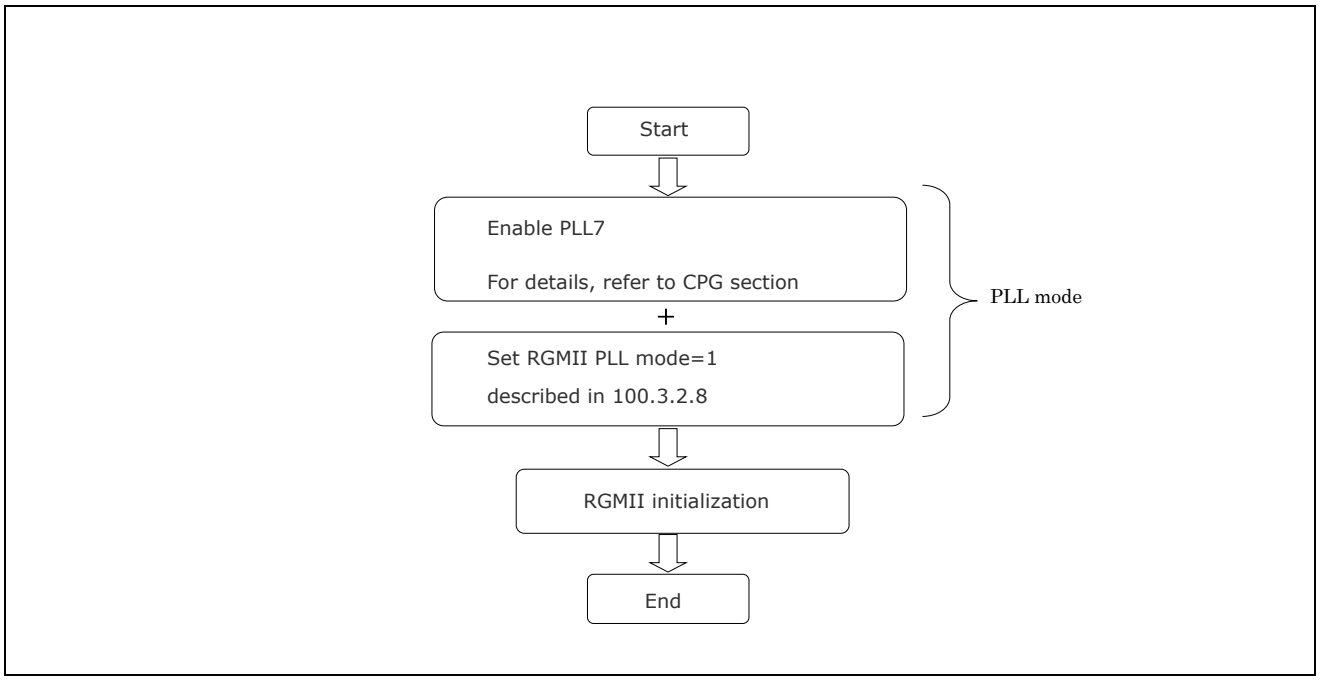



Figure 100.2 Initialization procedure of PLL mode

### 100.3.4 Usage Notes

When using RGMII interface, need to supply clock for EtherAVB module.

### 100.3.5 Safety Mechanisms

No description on this section.

# 102. CAN-FD

This section contains a generic description of the CANFD interface (RS-CANFD).
The first part of this section describes all specific properties, such as the number of units, register base addresses, etc. The remainder of the section describes the functions and registers of RS-CANFD.

## 102.1 Overview

### 102.1.1 Functional Overview

**Table 102.1 RS-CANFD Module Specifications**

| Item | Specification |
|---|---|
| Communication | CAN functionality conform to CAN-FD ISO 11898-1 (2015) |
| Protocol engine Version | RS-CANFD_PE V3.0 |
| Gateway Function | CAN 2.0 <> CAN 2.0<br>CAN 2.0 <> CAN-FD Gateway (Only 8 Byte Payload)<br>CAN-FD <> CAN-FD |
| Data transfer rate | up to 1Mbps for arbitration phase and up to 8Mbps for data phase , individually for each CAN channel |
| Proposed min. operation frequency<br>peripheral clock/APB clock | 133.33MHz |
| Data Link Layer clock (DLLC) | 40MHz(Max) |
| Input/Output pins | TX/RX |
| CAN channels | 4 |
| Selectable ID type | 11-bit Standard ID |
| | 11-bit Standard ID + 18-bit Extended ID |
| Selectable Frame Type | Data Frame (RTR = 0) (CAN and CAN-FD frames) |
| | Remote Frame (RTR = 1) (only CAN frames) |
| Variable Data Byte Count for Data Frames | DLC range: 0 to F |
| Message Buffer | 32 transmit message buffers per channel **1<br>4 transmission queue per channel<br>Automatic message transfer into transmission queues supported |
| | 64 shared buffers for RXMB and FIFO buffers per channel **1<br>- RXMB:<br>- Up to 16*(n+1) reception message buffers, shared among all the CAN channels<br>- FIFO buffers:<br>- 8 Reception FIFO Buffers<br>- Up to 3*(n+1) FIFOs individually configurable as<br>Reception FIFO / Transmission FIFO / CAN to CAN Gateway FIFO |
| Automatic delay interval timer for transmission | The delay timer can be applied to:<br>-Transmission FIFO<br>-CAN to CAN Gateway FIFO |
| Enhanced reception filtering | support of 11bit and 29bit CAN identifier |
| | programmable 29 bit CAN identifier acceptance filter mask for each entry |
| | programmable GW routing capability for each channel (up to 8 routing destinations) |

| | |
|---|---|
| | **RTR and IDE masking** |
| | **DLC filter** |
| | **Message buffer payload overload protection** |
| | **Payload filter** |
| | **Updating AFL entry during communication** |
| **General SW Support** | **Automatic label information added to receive message (for upper SW layer support)** |
| **Timer** | **TX and RX Time Stamp function** |
| **Power down function** | **Module start stop function for each CAN node (Channel & Global Sleep Mode)** |
| **RAM** | **RAM ECC protected** |
| **Bus traffic measurement** | **CAN bus traffic measurement of each Channel is possible** |
| **FFI mode** | **This product cannot use FFI mode.** |

### 102.1.2 Features of RS-CANFD

RS-CANFD has two interface modes (classical CAN mode and CAN FD mode), and uses different registers in each mode. The different register names RSCFD{unit}XXX and RSCFD{unit}CFDXXX are used in the different interface modes (XXX can be any character). In this document, the registers common to the two modes are indicated as RSCFD{unit}CFDXXX.

#### 102.1.2.1 Number of Units and Channels

This microcontroller has the following number of RS-CANFD units.

**Number of Units and Channels**

| Product Name | R-Car V4M |
|---|---|
| Number of Units | 1 |
| Number of Channels | 4 |
| Number of Channel per Unit | 4 |
| Transmission Message Buffers per Channel | 32 |
| Shared Buffers for per Channel | 64 |
| Total Message Buffer | 384 |

**Table 102.1 Index**

| Index | Description |
|---|---|
| unit | Throughout this section, the individual RS-CANFD units are generically indicated by the index “unit”.<br>For example, RSCFD[unit]CFDGCTR is the global control register of the RSCFDunit unit. |
| n | Throughout this section, the individual channels of RS-CANFD units are generically indicated by the index “n”.<br>For example, RSCFD[unit]CFDCnSTS is the channel n status register. |
| m | The individual channels of RSCFD unit is generically indicated by the index “m”.<br>For example, CANFDm_RX is receive data input. |
| d | The individual common FIFO buffers are generically indicated by the index “d”.<br>For example, RSCFD[unit]CFDCFCCd is the common FIFO buffer configuration/control register. |
| a | The individual receive FIFO buffers are generically identified by the index “a”.<br>For example, RSCFD[unit]CFDRFSTSa is the receive FIFO buffer status register. |
| p | Data field registers of common FIFO buffers and receive FIFO buffers are identified by “p”.<br>For example, RSCFD[unit]CFDCFDFp is the common FIFO access data field register. |
| i | The individual transmit control buffers are generically indicated by the index “i”.<br>For example, RSCFD[unit]CFDTMCi is transmit message buffer control register. |
| j | The individual transmit status buffers are generically indicated by the index “j”.<br>For example, RSCFD[unit]CFDTMSTSj is transmit message buffer status register. |
| k | The individual RAM tests for CAN are generically indicated by the index “k”.<br>For example, RSCFD[unit]CFDRPGACCk is the RAM test page access register. |
| f | The transmit message buffer transmission are generically indicated by index “f”.<br>For example, RSCFD[unit]CFDTMTRSTSf is the transmit message buffer transmission request status register. |
| w | A global register which setting for channels indicated by the index "w".<br>For example, RSCFD[unit]CFDGAFLCFGw is global Acceptance Filter List configuration register. |
| v | A global register which setting for 04 channels indicated by the index “v”.<br>For example, RSCFD[unit]CFDGTINTSTSv is the global transmit interrupt status register. |
| t | The receive message buffer new data are generically indicated by index “t”.<br>For example, RSCFD[unit]CFDRMNDt is a receive message buffer new data register. |

Note 1. The functions and descriptions of registers in this section are for the RS-CANFDs that has 4 channels (m = 0 to 3). When referring to information with indices, regard the index values as the ones corresponding to the target product.

Note 2. In some figure or table, this section use acronyms: CH, Ch, Ch instead of "channel".

**Table 102.2 Indices for Individual Products**

| Index Correspondence of Each Product | R-CarV4M |
|---|---|
| unit | 0 |
| n | 0 to 3 |
| m | 0 to 3 |
| d | 0 to 11 |
| a | 0 to 7 |
| p | 0 to 15 |
| i | 0 to 127 |
| j | 0 to 127 |
| k | 0 to 63 |
| f | 0 to 7 |
| w | 0 to 1 for RSCFD[unit]CFDGAFLCFGw<br>0 for RSCFD[unit]CFDGPFLCFGw |
| v | 0 |
| t | 0 to 1 |

### 102.1.3 Interface Mode

RS-CANFD has the following two interface modes.

- Classical CAN mode: Only classical CAN frames are handled.
- CAN FD mode: both the classical CAN frames and CAN FD frames are handled.

Two modes use different register maps with the same base address, and the register maps are switched by switching the modes.

Interface modes can be switched using the CLOE bit in the RSCFD{unit}CFDCnFDCFG register.

### 102.1.4 Block Diagram

The block diagram which shows connection with a peripheral module is shown.

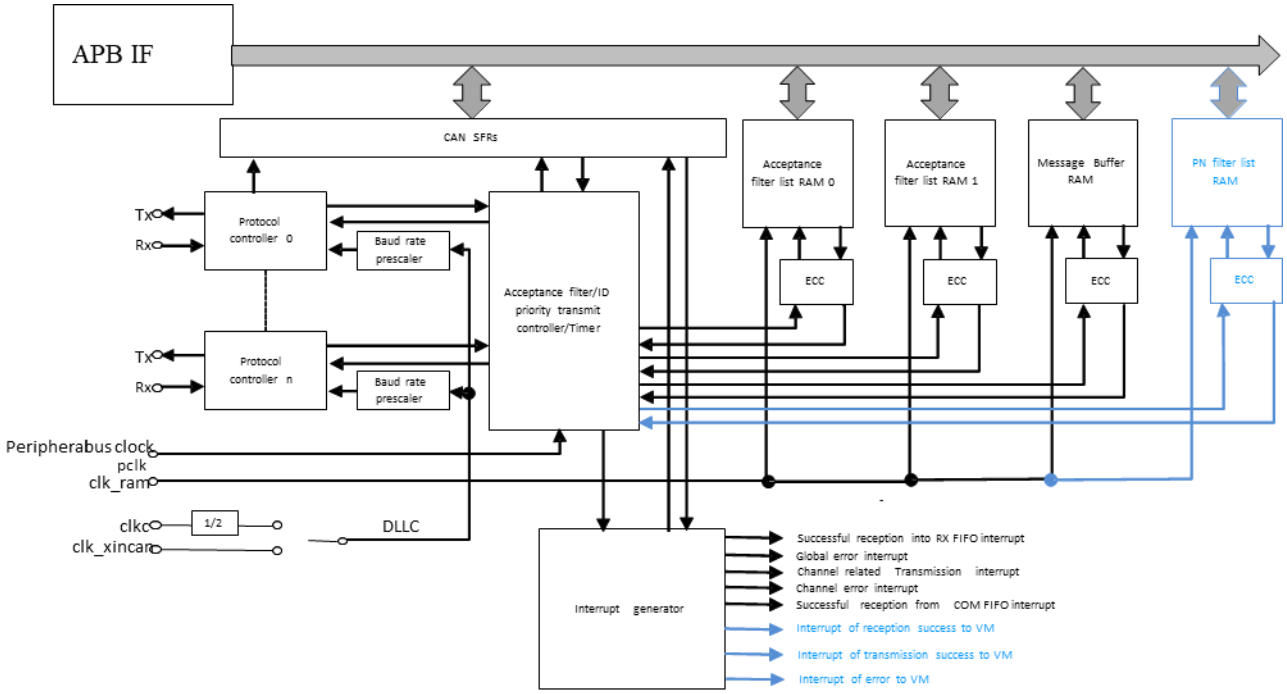



**Figure 102.1 RS-CANFD Module Block Diagram (in classical CAN mode)**

CANFDϕ must be set as follows.

R-Car V4M: 80(MHz)

Peripheral bus clock (pclk): SASYNCPERD2ϕ

clkc: CANFDϕ

clk_xincan: CAN_CLK

clk_ram: SASYNCPERD1ϕ

### 102.1.5 External Pins

Table 102.3 shows the CAN module pin.

Pin switching is required for pins which are multiplexed with other Function. For details, see section 7, Pin Function Controller (PFC).

**Table 102.3 Pin Configuration**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| CANFDm_RX | — | Input | Pins for receiving data |
| CANFDm_TX | — | Output | Pins for transmitting data |
| CAN_CLK | — | Input | Input pin used for external clock input. |

### 102.1.6 Connected module

**Table 102.4 Connected module**

| Module name | Connected module name | function of the related module |
|---|---|---|
| CANFD | AXI | Register access of CPU |
| | CPG | Clock output |
| | PFC | Selection of external pins |
| | Module Standby | Clock stop control |
| | Software Reset | Soft reset execution |
| | INTC | Interruption |
| | SYS-DMAC | DMA transmission |
| | CRC | CRC module |
| | MFIS | MFIS module |

#### 102.1.6.1 Register Base Address

RSCFD{unit} base addresses are listed in the following table.

RSCFD{unit} register addresses are given as offsets from the base addresses in general.

**Table 102.5 Register Base Address**

| Base Address Name | Base Address |
|---|---|
| <RSCFD0_base> | H'E666 0000 |

### 102.1.7 Register Configuration

This section describes all registers of RS-CANFD

#### 102.1.7.1 List of Registers

RSCFD registers are listed in the table below.

For details about Register Base Addresses.

| Register Name | Symbol RSCFD0+{registers of RS-CANFD} | R/W (*1) | Address | Access Size |
|---|---|---|---|---|
| Channel n Nominal Bitrate Configuration Register | RSCFD0CFDCnNCFG | - | <RSCFD0_base> + 0000h + n*0010h | 8, 16, 32 |
| Channel n Control Registers | RSCFD0CFDCnCTR | - | <RSCFD0_base> + 0004h + n*0010h | 8, 16, 32 |
| Channel n Status Registers | RSCFD0CFDCnSTS | - | <RSCFD0_base> + 0008h + n*0010h | 8, 16, 32 |
| Channel n Error Flag Registers | RSCFD0CFDCnERFL | - | <RSCFD0_base> + 000Ch + n*0010h | 8, 16, 32 |
| Global IP Version Register | RSCFD0CFDGIPV | - | <RSCFD0_base> + 0080h | 8, 16, 32 |
| Global Configuration Register | RSCFD0CFDGCFG | - | <RSCFD0_base> + 0084h | 8, 16, 32 |
| Global Control Register | RSCFD0CFDGCTR | - | <RSCFD0_base> + 0088h | 8, 16, 32 |
| Global Status Register | RSCFD0CFDGSTS | - | <RSCFD0_base> + 008Ch | 8, 16, 32 |
| Global Error Flag Register | RSCFD0CFDGERFL | - | <RSCFD0_base> + 0090h | 8, 16, 32 |
| Global Timestamp Counter Register | RSCFD0CFDGTSC | - | <RSCFD0_base> + 0094h | 16, 32 |
| Global Acceptance Filter List Entry Control Register | RSCFD0CFDGAFLECTR | - | <RSCFD0_base> + 0098h | 8, 16, 32 |
| Global Acceptance Filter List Configuration Register w | RSCFD0CFDGAFLCFGw | - | <RSCFD0_base> + 009Ch + w*0004h | 8, 16, 32 |
| RX Message Buffer Number Register | RSCFD0CFDRMNB | - | <RSCFD0_base> + 00Ach | 8, 16, 32 |
| RX Message Buffer New Data Register t | RSCFD0CFDRMNDt | - | <RSCFD0_base> + 00B0h + t*0004h | 8, 16, 32 |
| RX FIFO Configuration / Control Registers a | RSCFD0CFDRFCCa | - | <RSCFD0_base> + 00C0h + a*0004h | 8, 16, 32 |
| RX FIFO Status Registers a | RSCFD0CFDRFSTSa | - | <RSCFD0_base> + 00E0h + a*0004h | 8, 16, 32 |
| RX FIFO Pointer Control Registers a | RSCFD0CFDRFPCTRa | - | <RSCFD0_base> + 0100h + a*0004h | 8, 16, 32 |
| Common FIFO Configuration / Control Registers d | RSCFD0CFDCFCCd | - | <RSCFD0_base> + 0120h + d*0004h | 8, 16, 32 |
| Common FIFO Configuration / Control Enhancement Registers d | RSCFD0CFDCFCCEd | - | <RSCFD0_base> + 0180h + d*0004h | 8, 16, 32 |
| Common FIFO Status Registers d | RSCFD0CFDCFSTSd | - | <RSCFD0_base> + 01E0h + d*0004h | 8, 16, 32 |
| Common FIFO Pointer Control Registers d | RSCFD0CFDCFPCTRd | - | <RSCFD0_base> + 0240h + d*0004h | 8, 16, 32 |
| FIFO Empty Status Register | RSCFD0CFDFESTS | - | <RSCFD0_base> + 02A0h | 8, 16, 32 |
| FIFO Full Status Register | RSCFD0CFDFFSTS | - | <RSCFD0_base> + 02A4h | 8, 16, 32 |
| FIFO Message Lost Status Register | RSCFD0CFDFMSTS | - | <RSCFD0_base> + 02A8h | 8, 16, 32 |

| | | | | |
|---|---|---|---|---|
| RX FIFO Interrupt Flag Status Register | RSCFD0CFDRFISTS | - | <RSCFD0_base> + 02Ach | 8, 16, 32 |
| Common FIFO RX Interrupt Flag Status Register | RSCFD0CFDCFRISTS | - | <RSCFD0_base> + 02B0h | 8, 16, 32 |
| Common FIFO TX Interrupt Flag Status Register | RSCFD0CFDCFTISTS | - | <RSCFD0_base> + 02B4h | 8, 16, 32 |
| Common FIFO One Frame RX Interrupt Flag Status Register | RSCFD0CFDCFOFRISTS | - | <RSCFD0_base> + 02B8h | 8, 16, 32 |
| Common FIFO One Frame TX Interrupt Flag Status Register | RSCFD0CFDCFOFTISTS | - | <RSCFD0_base> + 02BCh | 8, 16, 32 |
| Common FIFO Message Over Write Status Register | RSCFD0CFDCFMOWSTS | - | <RSCFD0_base> + 02C0h | 8, 16, 32 |
| FIFO FDC Full Status Register | RSCFD0CFDFFFSTS | - | <RSCFD0_base> + 02C4h | 8, 16, 32 |
| TX Message Buffer Control Registers i | RSCFD0CFDTMCi | - | <RSCFD0_base> + 02D0h + i*0001h | 8 |
| TX Message Buffer Status Registers j | RSCFD0CFDTMSTSj | - | <RSCFD0_base> + 07D0h + j*0001h | 8 |
| TX Message Buffer Transmission Request Status Register f | RSCFD0CFDTMTRSTSf | - | <RSCFD0_base> + 0CD0h + f*0004h | 8, 16, 32 |
| TX Message Buffer Transmission Abort Request Status Register f | RSCFD0CFDTMTARSTSf | - | <RSCFD0_base> + 0D70h + f*0004h | 8, 16, 32 |
| TX Message Buffer Transmission Completion Status Register f | RSCFD0CFDTMTCSTSf | - | <RSCFD0_base> + 0E10h + f*0004h | 8, 16, 32 |
| TX Message Buffer Transmission Abort Status Register f | RSCFD0CFDTMTASTSf | - | <RSCFD0_base> + 0EB0h + f*0004h | 8, 16, 32 |
| TX Message Buffer Interrupt Enable Configuration Register f | RSCFD0CFDTMIECf | - | <RSCFD0_base> + 0F50h + f*0004h | 8, 16, 32 |
| TX Queue Configuration / Control Registers 0 [n] | RSCFD0CFDTXQCC0[n] | - | <RSCFD0_base> + 1000h + n*0004h | 8, 16, 32 |
| TX Queue Status Registers 0 [n] | RSCFD0CFDTXQSTS0[n] | - | <RSCFD0_base> + 1020h + n*0004h | 8, 16, 32 |
| TX Queue Pointer Control Registers 0 [n] | RSCFD0CFDTXQPCTR0[n] | - | <RSCFD0_base> + 1040h + n*0004h | 8, 16, 32 |
| TX Queue Configuration / Control Registers 1 [n] | RSCFD0CFDTXQCC1[n] | - | <RSCFD0_base> + 1060h + n*0004h | 8, 16, 32 |
| TX Queue Status Registers 1 [n] | RSCFD0CFDTXQSTS1[n] | - | <RSCFD0_base> + 1080h + n*0004h | 8, 16, 32 |
| TX Queue Pointer Control Registers 1 [n] | RSCFD0CFDTXQPCTR1[n] | - | <RSCFD0_base> + 10A0h + n*0004h | 8, 16, 32 |
| TX Queue Configuration / Control Registers 2 [n] | RSCFD0CFDTXQCC2[n] | - | <RSCFD0_base> + 10C0h + n*0004h | 8, 16, 32 |
| TX Queue Status Registers 2 [n] | RSCFD0CFDTXQSTS2[n] | - | <RSCFD0_base> + 10E0h + n*0004h | 8, 16, 32 |
| TX Queue Pointer Control Registers 2 [n] | RSCFD0CFDTXQPCTR2[n] | - | <RSCFD0_base> + 1100h + n*0004h | 8, 16, 32 |
| TX Queue Configuration / Control Registers 3 [n] | RSCFD0CFDTXQCC3[n] | - | <RSCFD0_base> + 1120h + n*0004h | 8, 16, 32 |
| TX Queue Status Registers 3 [n] | RSCFD0CFDTXQSTS3[n] | - | <RSCFD0_base> + 1140h + n*0004h | 8, 16, 32 |
| TX Queue Pointer Control Registers 3 [n] | RSCFD0CFDTXQPCTR3[n] | - | <RSCFD0_base> + 1160h + n*0004h | 8, 16, 32 |
| TX Queue Empty Status Register | RSCFD0CFDTXQESTS | - | <RSCFD0_base> + 1180h | 8, 16, 32 |
| TX Queue Full Interrupt Status Register | RSCFD0CFDTXQFISTS | - | <RSCFD0_base> + 1184h | 8, 16, 32 |
| TX Queue Message Lost Status Register | RSCFD0CFDTXQMSTS | - | <RSCFD0_base> + 1188h | 8, 16, 32 |
| TX Queue Message Overwrite Status Register | RSCFD0CFDTXQOWSTS | - | <RSCFD0_base> + 118Ch | 8, 16, 32 |
| TX Queue Interrupt Status Register | RSCFD0CFDTXQISTS | - | <RSCFD0_base> + 1190h | 8, 16, 32 |
| TX Queue One Frame TX Interrupt Status Register | RSCFD0CFDTXQOFTISTS | - | <RSCFD0_base> + 1194h | 8, 16, 32 |
| TX Queue One Frame RX Interrupt Status Register | RSCFD0CFDTXQOFRISTS | - | <RSCFD0_base> + 1198h | 8, 16, 32 |
| TX Queue Full Status Register | RSCFD0CFDTXQFSTS | - | <RSCFD0_base> + 119Ch | 8, 16, 32 |
| TX History List Configuration / Control Register n | RSCFD0CFDTHLCCn | - | <RSCFD0_base> + 1200h + n*0004h | 8, 16, 32 |
| TX History List Status Register n | RSCFD0CFDTHLSTSn | - | <RSCFD0_base> + 1220h + n*0004h | 8, 16, 32 |
| TX History List Pointer Control Registers n | RSCFD0CFDTHLPCTRn | - | <RSCFD0_base> + 1240h + n*0004h | 8, 16, 32 |
| Global TX Interrupt Status Register v | RSCFD0CFDGTINTSTSv | - | <RSCFD0_base> + 1300h + v*4 | 8, 16, 32 |
| Global Test Configuration Register | RSCFD0CFDGTSTCFG | - | <RSCFD0_base> + 1308h | 8, 16, 32 |
| Global Test Control Register | RSCFD0CFDGTSTCTR | - | <RSCFD0_base> + 130Ch | 8, 16, 32 |
| Global FD Configuration register | RSCFD0CFDGFDCFG | - | <RSCFD0_base> + 1314h | 8, 16, 32 |
| Global FD CRC Configuration register | RSCFD0CFDGCRCCFG | - | <RSCFD0_base> + 1318h | 8, 16, 32 |
| Global Lock Key Register | RSCFD0CFDGLOCKK | - | <RSCFD0_base> + 131Ch | 16, 32 |

| | | | | |
|---|---|---|---|---|
| Global AFL Ignore Entry Register | RSCFD0CFDGAFLIGNENT | - | <RSCFD0_base> + 1324h | 8, 16, 32 |
| Global AFL Ignore Control Register | RSCFD0CFDGAFLIGNCTR | - | <RSCFD0_base> + 1328h | 16, 32 |
| DMA Transfer Control Register | RSCFD0CFDCDTCT | - | <RSCFD0_base> + 1330h | 8, 16, 32 |
| DMA Transfer Status Register | RSCFD0CFDCDTSTS | - | <RSCFD0_base> + 1334h | 8, 16, 32 |
| DMA TX Transfer Control Register | RSCFD0CFDCDTTCT | - | <RSCFD0_base> + 1340h | 8, 16, 32 |
| DMA TX Transfer Status Register | RSCFD0CFDCDTTSTS | - | <RSCFD0_base> + 1344h | 8, 16, 32 |
| Global RX Interrupt Status Register n | RSCFD0CFDGRINTSTSn | - | <RSCFD0_base> + 1350h + n*0004h | 8, 16, 32 |
| Pretended Network Filter List Entry control Register | RSCFD0CFDGPFLECTR | - | <RSCFD0_base> + 1370h | 8, 16, 32 |
| Pretended Network Filter List Entry Configuration Register u | RSCFD0CFDGPFLCFGu | - | <RSCFD0_base> + 1374h + u * 0004h | 8, 16, 32 |
| Global SW reset Register | RSCFD0CFDGRSTC | - | <RSCFD0_base> + 1380h | 16, 32 |
| Global Flexible CAN mode Configuration Register | RSCFD0CFDGFCMC | - | <RSCFD0_base> + 1384h | 8, 16, 32 |
| Global Flexible transmission buffer assignment Configuration Register | RSCFD0CFDGFTBAC | - | <RSCFD0_base> + 138Ch | 8, 16, 32 |
| Channel n Data Bitrate Configuration Register | RSCFD0CFDCnDCFG | - | <RSCFD0_base> + 1400h + n*0020h | 8, 16, 32 |
| Channel n CAN-FD Configuration Register | RSCFD0CFDCnFDCFG | - | <RSCFD0_base> + 1404h + n*0020h | 8, 16, 32 |
| Channel n CAN-FD Control Register | RSCFD0CFDCnFDCTR | - | <RSCFD0_base> + 1408h + n*0020h | 8, 16, 32 |
| Channel n CAN-FD Status Register | RSCFD0CFDCnFDSTS | - | <RSCFD0_base> + 140Ch + n*0020h | 8, 16, 32 |
| Channel n CAN-FD CRC Register | RSCFD0CFDCnFDCRC | - | <RSCFD0_base> + 1410h + n*0020h | 8, 16, 32 |
| Channel n Bus load Control Register | RSCFD0CFDCnBLCT | - | <RSCFD0_base> + 1418h + n*0020h | 8, 16, 32 |
| Channel n Bus load Status Register | RSCFD0CFDCnBLSTS | - | <RSCFD0_base> + 141Ch + n*0020h | 8, 16, 32 |
| Global Acceptance Filter List ID Registers r = [1…10]h | RSCFD0CFDGAFLIDr | - | <RSCFD0_base> + 1800h + (r-1)*0010h | 8, 16, 32 |
| Global Acceptance Filter List Mask Registers r = [1…10]h | RSCFD0CFDGAFLMr | - | <RSCFD0_base> + 1804h + (r-1)*0010h | 8, 16, 32 |
| Global Acceptance Filter List Pointer 0 Registers r = [1…10]h | RSCFD0CFDGAFLP0r | - | <RSCFD0_base> + 1808h + (r-1)*0010h | 8, 16, 32 |
| Global Acceptance Filter List Pointer 1 Registers r = [1…10]h | RSCFD0CFDGAFLP1r | - | <RSCFD0_base> + 180Ch + (r-1)*0010h | 8, 16, 32 |
| Global Pretended Network Filter List ID Registers s = [1…4] | RSCFD0CFDGPFLIDs | - | <RSCFD0_base> + 1A00h + (s-1) * 0040h | 8, 16, 32 |
| Global Pretended Network Filter List Mask Registers s = [1...4] | RSCFD0CFDGPFLMs | - | <RSCFD0_base> + 1A04h + (s-1) * 0040h | 8, 16, 32 |
| Global Pretended Network Filter List Pointer 0 Registers s = [1...4] | RSCFD0CFDGPFLP0s | - | <RSCFD0_base> + 1A08h + (s-1) * 0040h | 8, 16, 32 |
| Global Pretended Network Filter List Pointer 1 Registers s = [1...4] | RSCFD0CFDGPFLP1s | - | <RSCFD0_base> + 1A0Ch + (s-1) * 0040h | 8, 16, 32 |
| Global Pretended Network Filter List Filter Payload Type Registers s = [1...4] | RSCFD0CFDGPFLPTs | - | <RSCFD0_base> + 1A10h + (s-1) * 0040h | 8, 16, 32 |
| Global Pretended Network Filter List Payload Data 0 Registers s = [1...4] | RSCFD0CFDGPFLPD0s | - | <RSCFD0_base> + 1A14h + (s-1) * 0040h | 8, 16, 32 |
| Global Pretended Network Filter List Payload Mask 0 Registers s = [1...4] | RSCFD0CFDGPFLPM0s | - | <RSCFD0_base> + 1A18h + (s-1) * 0040h | 8, 16, 32 |
| Global Pretended Network Filter List Payload Data 1 Registers s = [1...4] | RSCFD0CFDGPFLPD1s | - | <RSCFD0_base> + 1A1Ch + (s-1) * 0040h | 8, 16, 32 |
| Global Pretended Network Filter List Payload Mask 1 Registers s = [1...4] | RSCFD0CFDGPFLPM1s | - | <RSCFD0_base> + 1A20h + (s-1) * 0040h | 8, 16, 32 |
| Channel n TX History List Access Registers 0 | RSCFD0CFDTHLACC0[n] | - | <RSCFD0_base> + 8000h + n*0008h | 8, 16, 32 |
| Channel n TX History List Access Registers 1 | RSCFD0CFDTHLACC1[n] | - | <RSCFD0_base> + 8004h + n*0008h | 8, 16, 32 |
| RAM Test Page Access Registers k | RSCFD0CFDRPGACCk | - | <RSCFD0_base> + 8400h + k*0004h | 8, 16, 32 |
| RX Message Buffer ID Registers | RSCFD0CFDRMID | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| RX Message Buffer Pointer Registers | RSCFD0CFDRMPTR | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| RX Message Buffer CAN-FD Status Register | RSCFD0CFDRMFDSTS | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| RX Message Buffer Data Field p Registers | RSCFD0CFDRMDFp | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| RX FIFO Access ID Registers | RSCFD0CFDRFID | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| RX FIFO Access Pointer Register | RSCFD0CFDRFPTR | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| RX FIFO Access CAN-FD Status Register | RSCFD0CFDRFFDSTS | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| RX FIFO Access Data Field p Registers | RSCFD0CFDRFDFp | - | <RSCFD0_base> + Note2 | 8, 16, 32 |

| | | | | |
|---|---|---|---|---|
| Common FIFO Access ID Registers | RSCFD0CFDCFID | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| Common FIFO Access Pointer Registers | RSCFD0CFDCFPTR | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| Common FIFO Access CAN-FD Control/Status Register | RSCFD0CFDCFFDCSTS | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| Common FIFO Access Data Field p Registers | RSCFD0CFDCFDFp | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| TX Message Buffer ID Registers | RSCFD0CFDTMID | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| TX Message Buffer Pointer Registers | RSCFD0CFDTMPTR | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| TX Message Buffer CAN-FD Control Register | RSCFD0CFDTMFDCTR | - | <RSCFD0_base> + Note2 | 8, 16, 32 |
| TX Message Buffer Data Field p Registers | RSCFD0CFDTMDFp | - | <RSCFD0_base> + Note2 | 8, 16, 32 |

Notes: 1. Refer to the section 4.3 of "102_uciaprcn0140_IPSpec_v1p06_r2.pdf"

Notes: 2. Refer to the Figure 4.2 of "102_uciaprcn0140_IPSpec_v1p06_r2.pdf"

## 102.2 Register Description

### 102.2.1 Details of Interface Mode Related Registers

Refer to the section 4 of "102_uciaprcn0140_IPSpec_v1p06_r2.pdf"

## 102.3 Operation

Refer to the section 5 to 18 of "102_uciaprcn0140_IPSpec_v1p06_r2.pdf"

## 102.4 Usage Notes

No description on this section.

## 102.5 Safety Mechanisms

### 102.5.1 ECC for CAN-FD (IP uciapecc7w10)

#### 102.5.1.1 Overview

ECC IP – Function block diagram:

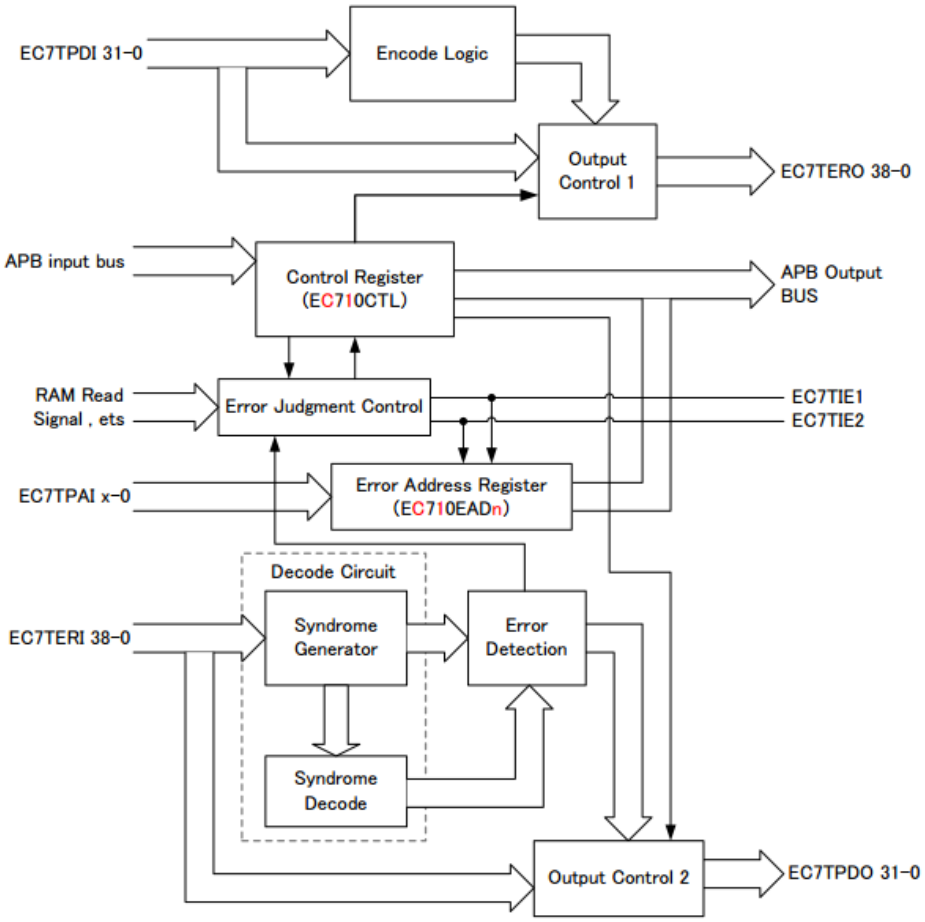


**Figure 102.2 ECC Block diagram**

EDC protection is applied to internal buffer (Acceptance filter list RAM, Message Buffer RAM) of CAN-FD.

#### 102.5.1.1.1 Error Detection and Correction

-Seven-bit ECC data is appended to the 32-bit RAM data.

-This ECC module provides 2-bit ECC error detection and 1-bit ECC error detection and correction.

*Note: This module is not capable of reliably detecting errors in three or more bits. If errors occur in three or more bits, the module may detect the errors as 1- or 2-bit ECC errors or not detect any errors. Depending on the settings, this may lead to the correction of a bit that was not actually inverted.

#### 102.5.1.1.2 Enabling or Disabling ECC Error Detection and Correction

-ECC error detection can be either enabled or disabled.

-One-bit ECC error correction can be either enabled or disabled.

-If all the bits of RAM output data are stuck to 0 or 1, it is detected as a 2-bit ECC error.

#### 102.5.1.1.3 Error Notification

-The ECM is notified when 2-bit ECC errors are detected (this can be enabled or disabled).

-The ECM is notified when 1-bit ECC errors are detected (this can be enabled or disabled).

-Once the ECM has been notified of an error, even if another ECC error is detected, the ECM is not notified until the error status bit corresponding to the initial error is cleared.

#### 102.5.1.1.4 Error Status

-Detection of 2-bit and 1-bit ECC errors can be monitored.

-Special registers are provided to clear error status.

#### 102.5.1.1.5 Address Capture

-Only one address at which an ECC error has occurred can be captured.

-When a 2-bit or 1-bit ECC error is detected, the error-causing address is captured. It is used to capture only when the first error is detected after the flag is cleared.

#### 102.5.1.1.6 Testing Function (Error Insertion)

-By setting the test mode, register values can be used as the data to be output to the RAM. When a peripheral module writes to the RAM, the ECEDB[31:0] register value can be written to the RAM data section, and the ECERDB[6:0] register value can be written to the ECC redundant bit section.

-By setting the test mode, the ECC redundant bit section can be latched when RAM data is read, and the value can be confirmed.

-By setting the test mode, the ECC redundant bit (encoding circuit) and syndrome code (decoding circuit), which are generated from the input data, can be confirmed.

### 102.5.1.2 Hardware Description

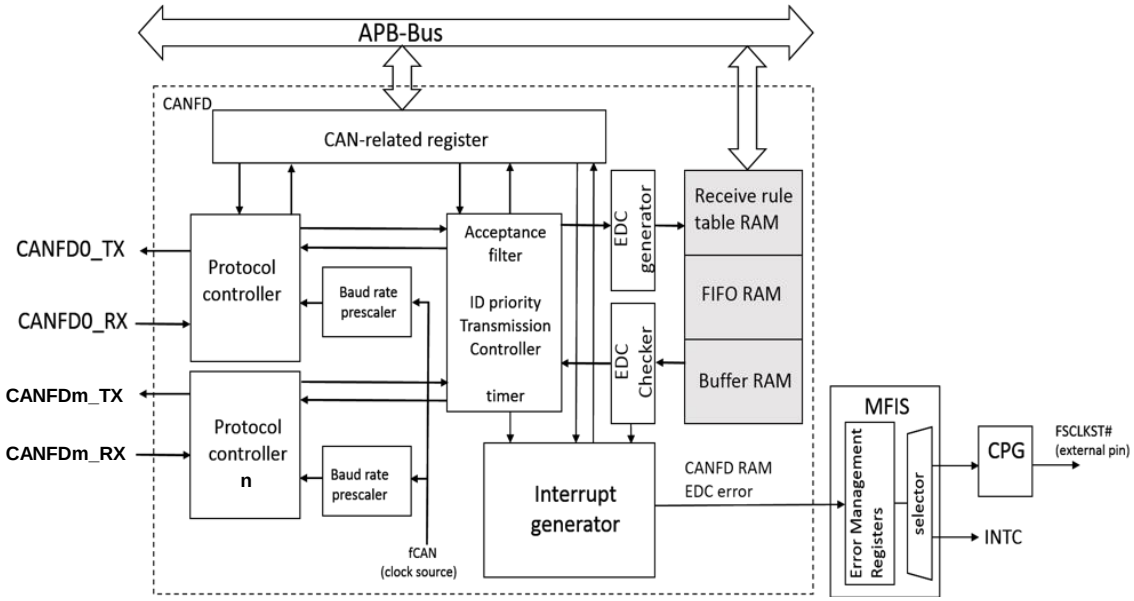



**Figure 102.3 EDC protection for CAN-FD**

**Table 102.6 Register Configuration**

| Register Name | Abbreviation | R/W | Address | Initial Value | Size |
|---|---|---|---|---|---|
| ECC control register | EC710CTL | R/W | H'E66C_8000 | H'0000 0000 | 32 |
| ECC test mode control register | EC710TMC | R/W | H'E66C_8004 | H'0000 0000 | 32 |
| ECC redundant bit data control test register | EC710TRC | R | H'E66C_8008 | H'0000 0000 | 32 |
| ECC encoder and decoder data test register | EC710TED | R/W | H'E66C_800C | H'0000 0000 | 32 |
| ECC error address register | EC710EAD | R | H'E66C_8010 | H'0000 0000 | 32 |

#### 102.5.1.2.1 Register Description

**(a) EC710CTL — ECC control register**

E710CTL controls the status and modes of the ECC module.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EMCA1 | EMCA0 | - | - | - | ECER2C | ECER1C | - | ECTHM | ECERVF | EC1ECP | EC2EDIC | EC1EDIC | ECER2F | ECER1F | ECEMF |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X |
| R/W: | R/W | R/W | R | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 14 | EMCA[1:0] | All 0 | R/W | These 2bits are the write trigger reserve bits to bit6. These read value are always B’0. When the value of these bits is 2’b01, then it is possible to write access to bit6 “ECERVF”. If these bits are not 2’b01, the write access to bit6 is ignored and the value is not written.<br>These bits reserve the write trigger to bit 7. |
| 13 to 11 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 10 | ECER2C | 0 | R/W | 2-Bit ECC Error Detection Flag Clear<br>Clears the bit 2 “ECER2F” status flag.<br>The read value is always B’0, and writing B’0 to ECER2C does not change the state.<br>If writing B’1 to ECER2C conflicts with the condition for setting bit 2, the former takes priority.<br>Writing B’1 to ECER2C while “ECER2F” is set clears “ECER2F” |
| 9 | ECER1C | 0 | R/W | 1-Bit ECC Error Detection Flag Clear<br>Clears the bit 1 “ECER1F” status flag.<br>The read value is always B’0, and writing B’0 to ECER1C does not change the state.<br>If writing B’1 to ECER1C conflicts with the condition for setting bit 1, the former takes priority.<br>Writing B’1 to ECER1C while “ECER1F” is set clears “ECER1F”.<br>Flag clear bit of accumulating ECC error detection and correctio |
| 8 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | ECTHM | 0 | R/W | ECC Function Disable Select<br>Selects the ECC decoding operation. Write access to ECTHM is enabled when the value of bits 15 and 14 is 2'b01. Therefore, only the 16-bit manipulation instruction is valid.<br>Setting ECTHM to B'1 disables error detection and bit correction. Here, if the data to be output to the peripheral module contains an error, the data is output without bit correction. Setting ECTHM to B'1 has no effect on the encoder side.<br>0: Enables ECC detection and correction.<br>1: Disables ECC detection and correction. |
| 6 | ECERVF | 0 | R/W | Flag bit to enable ECC error judgment<br>Setting this bit to B'1 enables to judge error. The correction of output data and the interrupt output depend on setting of bit 5 "EC1ECP", bit 4 "EC2EDIC" and bit 3 "EC1EDIC".<br>But the write access to this bit is valid when the value of upper bit15 and bit14 is 2'b01.<br>So only the 16bits operation command is valid in the case of the write access to this bit. |
| 5 | EC1ECP | 0 | R/W | Permission bit of ECC 1bit error correction<br>This bit sets enable or disable to correct the 1bit error when ECC error detection and correction is valid.<br>By setting B'1 to this bit, the non-corrected data is outputted if 1bit error is detected.<br>0: At 1bit error detection, the error correction is executed.<br>1: At 1bit error detection, the error correction is not executed. |
| 4 | EC2EDIC | 0 | R/W | Control bit of interrupt at ECC 2bits error detection<br>This bit controls the interrupt output at detecting 2bits error.<br>By setting B'1 to this bit, the 2bits error interrupt "EC7TIE2" is outputted when 2bits error detected.<br>0: When a 2-bit error is detected, the ECM is not notified of the error.<br>1: When a 2-bit error is detected, the ECM is notified of the error. |
| 3 | EC1EDIC | 0 | R/W | Control bit of interrupt at ECC 1bit error detection<br>This bit controls the interrupt output at detecting 1bit error.<br>By setting "1" to this bit, the 1bit error interrupt "EC7TIE1" is outputted when 1bit error detected.<br>0: Interrupt is not occurred at detecting 1bit error. (initial value)<br>1: EC7TIE1 is outputted at detecting 1bit error. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | ECER2F | 0 | R/W | 2-Bit ECC Error Detection Flag<br>Indicates that errors have been detected at two bits in bits 0 to 38 of the data read from the RAM during RAM read access while error detection is enabled. When the 2bits error interrupt output is enabled, 2bits error interrupt signal “E7TIE2” is outputted by setting this flag.<br>This bit is a read only flag, so writing “1” or “0” does not influence on inner condition.<br>When 2bits error is detected again under the condition that this bit is set, the interrupt signal is not occurred.<br>0: 2-bit error is not generated<br>1: 2-bit error is generated<br>Clearing conditions<br>(1) Reset is applied.<br>(2) 1 is written to ECER2C.<br>(3) ECC detection and correction being disabled (ECTHM = 1). |
| 1 | ECER1F | 0 | R/W | 1-Bit ECC Error Detection And Correction Flag<br>Indicates that an error has been detected at one bit in bits 0 to 38 of the data read from the RAM during RAM read access while error detection is enabled. When the 1bit error interrupt output is enabled, 1bit error interrupt signal “E7TIE1” is outputted by setting this flag.<br>This bit is a read only flag, so writing “1” or “0” does not influence on inner condition.<br>When 1bit error is detected again under the condition that this bit is set, the interrupt signal is not occurred.<br>0: 1-bit error is not generated<br>1: 1-bit error is generated<br>Clearing conditions<br>(1) Reset is applied.<br>(2) 1 is written to ECER1C.<br>(3) ECC detection and correction being disabled (ECTHM = 1). |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | ECEMF | X | R | ECC Error Message Flag<br>Indicates that the current read data bus contains an error.<br>ECEMF is updated every time RAM data is output. Since the RAM value after reset is undefined, the initial value of this module is determined to be an error, and thus ECEMF may be set. Therefore, the ECEMF value after reset is undefined.<br>0: The current read data bus contains no bit errors.<br>1: The current RAM output data contains bit errors.<br>ECEMF remains set as long as RAM data containing a bit error is output with error detection being enabled.<br>Clearing conditions<br>(1) ECC detection and correction being disabled (ECTHM = B'1).<br>(2) Decoding circuit input data contains no 1-bit errors. |

**(b) E710TMC — ECC Test Mode Control Register**

E710TMC is a register to switch the mode to test mode and controls the mode.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ETMA[1:0] | | - | - | - | - | - | - | ECTMCE | - | - | ECTRRS | ECREOS | ECENS | ECDCS | ECREIS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 14 | ETMA[1:0] | All 0 | R/W | Access Control 1 and 0 to ECC Test Mode Control Enable Bit<br>These bits reserve the write trigger to bit 7. The read value is always 0. |
| 13 to 8 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | ECTMCE | 0 | R/W | ECC Test Mode Control Enable<br>ECTMCE enables or disables access to the test registers and test control bits.<br>Write access to ETCMCE is enabled when the value of bits 15 and 14 is B'10.<br>0: Disables access to the test registers and test control bits.<br>1: Enables access to the test registers and test control bits. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 to 5 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | ECTRRS | 0 | R/W | ECC RAM Read Test Mode Control Enable<br>ECTRRS enables the read status of RAM to be generated by reading the E710TED register, and also allows the RAM output data to be read out when the E710TRC.ECERDB[7:0] bits and the E710TED register are read.<br>Write access to ECTRRS is enabled only when ECTMCE is 1 (can be set simultaneously).<br>ECTRRS is cleared by writing 0 to ECTMCE (can be cleared synchronously).<br>0: Disables generation of RAM read status for testing when E710TED is read.<br>1: Enables generation of RAM read status for testing when E710TED is read.<br>When E710TRC.ECERDB[7:0] and E710TED are read, the values of the RAM output data pin are read out. |
| 3 | ECREOS | 0 | R/W | ECC Redundant Bit Output Data Select<br>ECREOS selects either the ECC encoder output data or the ECERDB register value to be output as the ECC redundant bit output.<br>Write access to ECREOS is enabled only when ECTMCE is 1 (can be set simultaneously).<br>ECREOS is cleared by writing 0 to ECTMCE (can be cleared synchronously).<br>0: Allows encoding result to be output as the ECC redundant bit output.<br>1: Allows the E710TRC.ECERDB[6:0] value to be output as the ECC redundant bit output. |
| 2 | ECENS | 0 | R/W | ECC Encoder Input Select<br>ECENS selects either the data value from the peripheral module or the test register value (E710TED.ECEDB[31:0]) as the input signal to be encoded.<br>Write access to ECENS is enabled only when ECTMCE is 1 (can be set simultaneously).<br>ECENS is cleared by writing 0 to ECTMCE (can be cleared synchronously).<br>0: Allows the RAM write data from the peripheral module to be input as the ECC encoder input data.<br>1: Allows the E710TED.ECEDB[31:0] value to be input as the ECC encoder input data. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | ECDCS | 0 | R/W | ECC Decoder Input Select<br>ECDCS selects either the lower 32-bit data value from the RAM or the test register value (E710TED.ECEDB[31:0]) as the lower 32-bit data of the input signal to be decoded.<br>Write access to ECDCS is enabled only when ECTMCE is 1 (can be set simultaneously).<br>ECDCS is cleared by writing 0 to ECTMCE (can be cleared synchronously).<br>0: Allows the lower 32-bit RAM output data to be input to the data area (32 lower-order bits) to the decoder.<br>1: Allows the E710TED.ECEDB[31:0] value to be input to the data area to the decoder. |
| 0 | ECREIS | 0 | R/W | ECC Redundant Bit Input Data Select<br>ECREIS selects either the upper 7-bit data value from the RAM (redundant bit area) or the test register value (E710TRC.ECERDB[6:0]) as the upper 7-bit data of the input signal to be decoded.<br>Write access to ECREIS is enabled only when ECTMCE is 1 (can be set simultaneously).<br>ECREIS is cleared by writing 0 to ECTMCE (can be cleared synchronously).<br>0: Allows the upper 7-bit RAM output data to be input to the ECC redundant bit area to the decoder.<br>1: Allows the E710TRC.ECERDB[6:0] value to be input to the ECC redundant bit area to the decoder. |

**(c) E710TRC — ECC Redundant Bit Data Control Test Register**

E710TRC is a 32-bit test register consisting of four fields (ECSYND, ECHORD, ECECRD, and ECRODM) corresponding to the ECC redundant bit area. Each field can be accessed as the 8-bit register with the same name.

When ECTMCE = 0, E710TRC is always read as 0.

When ECTMCE = 1, E710TRC can be read using the 32-bit manipulation instruction.

Note: Changing ECTMCE from 1 to 0 resets E710TED synchronously

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | ECSYND[6:0] | | | | | | | - | ECHORD[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | ECECRD[6:0] | | | | | | | - | ECERDB[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R/W: R R R R R R R R R R R R R R R R

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 30 to 24 | ECSYND [6:0] | All 0 | R | ECC Decoder Syndrome Data Register<br>ECSYND is a read-only register to confirm the syndrome code generated by the decoding circuit in test mode (ECTMCE = 1).<br>Write access to ECSYND is ignored.<br>When read, the ECSYND bits return the value of the syndrome code (synd[6:0]) generated based on the input data to the decoding circuit.<br>The ECSYND bits are not holding circuits; the register value changes as the input signal changes.<br>ECSYND is valid only when ECTMCE = 1, and is always read as 00H when ECTMCE = 0. |
| 23 | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 22 to 16 | ECHORD [6:0] | All 0 | R | ECC 7-Bit Redundant Data Holding Test Register<br>ECHORD holds the 7-bit ECC redundant area (upper 7-bit RAM data), which cannot be confirmed by the peripheral module, when the peripheral module accesses the RAM for reading in test mode (ECTMCE = 1).<br>The ECHORD bits are loaded with the upper 7-bit RAM output data at the next rising edge of the operating clock signal after the peripheral module accesses the RAM for reading data in test mode (ECTMCE = 1).<br>The ECHORD bits are also loaded with the data on the input pins EC7TERI38 to EC7TERI32 at the next operating clock pulse when the ECEDB[15:0] register is read while E710TMC:ECTRRS = 1.<br>ECHORD is valid only when ECTMCE = 1, and is always read as 00H when ECTMCE = 0. |
| 15 | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 14 to 8 | ECECRD [6:0] | All 0 | R | ECC Encoder Test Register<br>ECECRD is a read-only register to read the 7-bit redundant section generated by the encoding circuit in test mode (ECTMCE = 1).<br>ECECRD is used to confirm the redundant bits generated by the input data from the peripheral module.<br>Here, the read-out data is the result of encoding (ecc[6:0]), not the output value.<br>ECECRD is valid only when ECTMCE = 1, and is always read as 00H when ECTMCE = 0. |
| 7 | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 to 0 | ECERDB [6:0] | All 0 | R | ECC Redundant Bit Input and Output Substitution Buffer Register<br>ECERDB is a buffer register for the data that substitutes for the input and output data for the 7-bit ECC redundant data area in test mode (ECTMCE = 1).<br>ECERDB can be read and written to in ECC test mode (ECTMCE = 1).<br>When ECREOSS = 1, the ECERDB value, instead of the seven redundant bits generated by the encoding circuit, is output to the pin to be supplied to the RAM.<br>When ECREIESS = 1, the ECERDB value, instead of the upper seven data bits to be input to the decoding circuit, is handled by the decoding circuit.<br>When ECTRRS = 1, reading ECERDB returns the signal value supplied to RAM instead of the data written to ECERDB. |

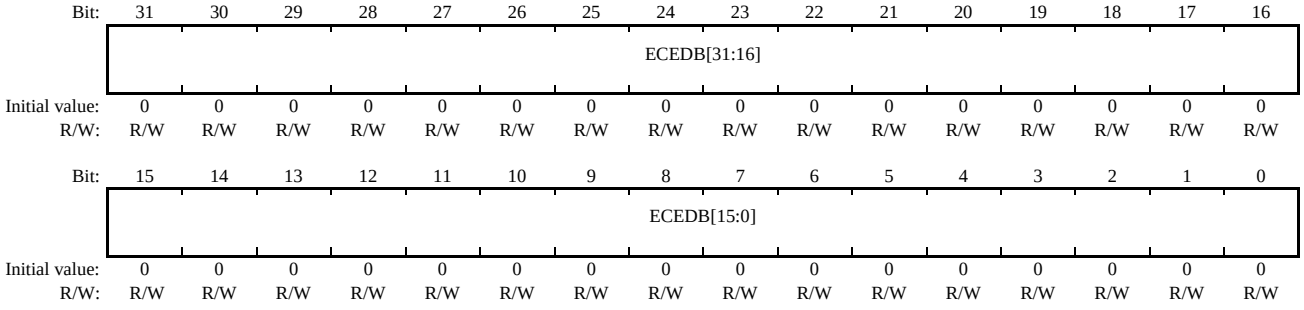
**(d) E710TED — ECC Encoder and Decoder Data Test Register**

E710TED is a 32-bit data test register for ECC encoding and decoding.

When ECTMCE = 1, E710TED can be read and written to using the 32-bit manipulation instruction.

When ECTMCE = 0, E710TED is always read as 0.

In test mode, the E710TED value can be used as the input data to the encoding circuit and decoding circuit.



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ECEDB | All 0 | R/W | ECC Encoder and Decoder Data Test Register<br>When E710TMC:ECENS = B'1, the ECEDB value is input to the encoding circuit and supplied to the RAM.<br>When E710TMC:ECDCS = B'1, the ECEDB value is input as the 31th to 0th bits of the input data to the decoding circuit.<br>When E710TMC:ECTRRS = B'1, reading ECEDB returns the RAM output data instead of the data written to ECEDB.<br>Note: Changing ECTMCE from B'1 to B'0 resets E710TED synchronously. |

**(e) E710EAD — ECC Error Address Register**

E710EAD is a read-only register to hold the address at which an ECC error has occurred.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEAD [31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECEAD [15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ECEAD | All 0 | R | ECC Error Address Register<br>If an ECC error is detected while ECC error detection is enabled, the RAM address is latched using the detection signal as a trigger, and the address is stored in E710EAD as the address at which the ECC error has occurred.<br>The address is stored upon detection of the first ECC error while no error status flag is set. However, if a 1-bit error is followed by a 2-bit error, the address of the latter is stored.<br>Only one address can be held in E710EAD. |

#### 102.5.1.2.2 Notification to ECM

Detection of errors in two bits can be set for this module and the ECM is notified of detected errors.

- 1-bit Error Notification

  While EC1EDIC is set to B'1, when an error is detected in one bit from among bits 0 to 38 of data read from RAM, the ECM is notified of the 1-bit error. When ECER1F or ECER2F is already set, however, the ECM will not be notified of the error.
- 2-bit Error Notification

While EC2EDIC is set to B'1, when an error is detected in two bits from among bits 0 to 38 of data read from RAM, the ECM is notified of the 2-bit error. When ECER2F is already set, however, the ECM will not be notified of the error.

#### 102.5.1.2.3 Test Function

(1) Writing to RAM Data

Write data to the peripheral RAM. However, the ECC corresponding to the written data will be written to the ECC bits simultaneously. In order to write a specified value to the ECC bits, use the ECC test mode described in (3) below.

(2) Reading RAM Data

(a) Set the ECTHM bit in the ECC control register to 1 to disable ECC error detection and correction.

(b) Read the peripheral RAM. Since neither error detection nor correction proceeds when the peripheral RAM is read, the RAM data is read unchanged.

How to exit this test mode:

(a) Set the ECTHM bit in the ECC control register to 0 to enable ECC error detection and correction.

(3) Writing to the ECC Bits

(a) Set the ECTMCE bit in the ECC test mode control register to 1 to set ECC test mode.

(b) Write the value for writing to the ECC bits to the E710TRC.ECERDB[6:0] bits.

(c) Set the ECREOS bit in the ECC test mode control register to 1 to select writing of the value of the E710TRC.ECERDB[6:0] bits to the ECC bits.

(d) When data are written to the peripheral RAM, the value in the E710TRC.ECERDB[6:0] bits will be written to the ECC bits.

How to exit this test mode:

(a) Set the ECTMCE bit in the ECC test mode control register to 0 to set normal mode.

(4) Reading the ECC Bits

(a) Set the ECTMCE bit in the ECC test mode control register to 1 to set ECC test mode.

(b) When data in the peripheral RAM is read, the ECC bits are stored in the E710TRC.ECHORD[6:0] bits.

How to exit this test mode:

(a) Set the ECTMCE bit in the ECC test mode control register to 0 to set normal mode.

### 102.5.1.3 Error detection operation

#### 102.5.1.3.1 Setting initial value

In this IP (ECC IP), it is possible to select whether the error in the initial value after reset is judged or not with the value of IP terminal “EC7TERVE”.

Followings show operation table with terminal condition and bit setting

| EC7TERVE terminal | ECTHM bit | ECERVF bit | Operation | |
|---|---|---|---|---|
| | | | Error judgment operation | ECERVF bit access |
| 0 | 0 | 0 | Error judgment is permitted. (reset value) | Writing access is invalid |
| | 0 | 1 | Error judgment is permitted. (reset value) | Writing access is invalid |
| | 1 | 0 | Error judgment is prohibited (through mode) | Writing access is invalid |
| | 1 | 1 | Error judgment is prohibited (through mode) | Writing access is invalid |
| 1 | 0 | 0 | Error judgment is prohibited (reset value) | Writing access is valid |
| | 0 | 1 | Error judgment is permitted. | Writing access is valid |
| | 1 | 0 | Error judgment is prohibited (through mode) | Writing access is valid |
| | 1 | 1 | Error judgment is prohibited (through mode) | Writing access is valid |

Example to use at EC7TERVE=1

If the peripheral IP using RAM needs the read access to RAM before completing the RAM initial setting after releasing reset (IP’s RAM access way is read-modify-write type, etc.), the error judgment is prohibited during the initial setting and permitted (ECERVF=1) after completing the RAM initial setting with software. (If some treatment is done to peripheral IP, these operations are not needed.)

#### 102.5.1.3.2 1bit error judgment

If the total number of “1” is odd in the syndrome code (7bits) generated from readout data (including ECC bit) “EC7TERO38-0”, then it is judged as 1bit error. (When 1bit error is occurred in data bit 31-0 and ECC bit 6-0, the total number of “1” set in syndrome code is necessarily odd. When there are plural errors, total number of “1” in syndrome code is odd. But this IP does not guarantee the function like it.)

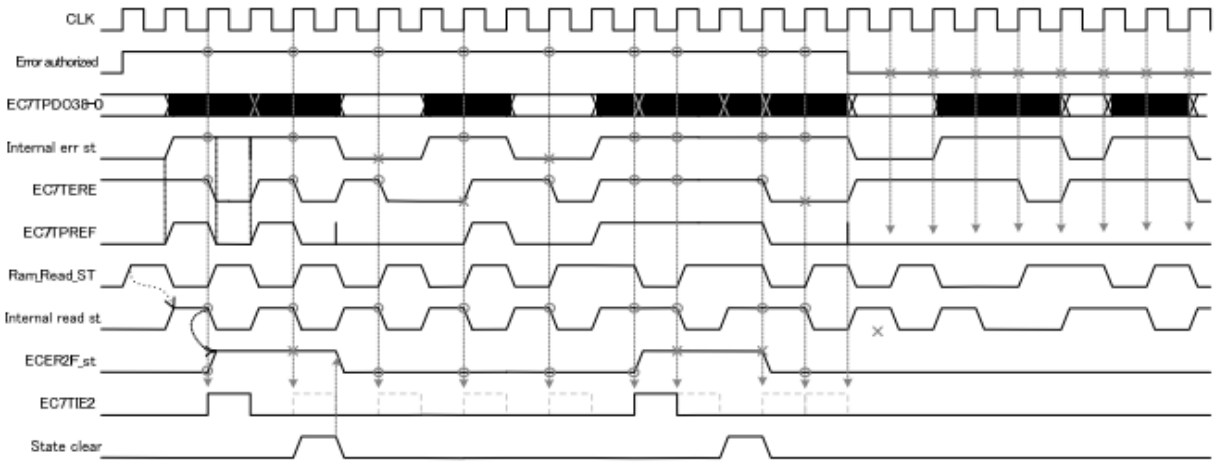
#### 102.5.1.3.3 2bit error judgment

If the syndrome code (7bits) generated from readout data (including ECC bit) “EC7TERO38－0” is not 7’h00 and the error bit placement detecting code (p38-0) (p38-32 are ECC bit parts (7bits redundant bit parts)) is 39’h0000000000, then it is judged 2bits error.

When “Synd[6:0] is not 7’h00” and “p[38:0] is 39’h0000000000”, then it is 2bits error.

Primly the 2bits error is judged when the total number of “1” set in syndrome code (7bits). But this IP judges with the syndrome code and the error bit placement detecting code to not detect the false error in the case that all data (including ECC bit part) are fixed to “0” or “1”. (So if there is more than 3bits error, the possibility is up to judge as 2bits error.)

#### 102.5.1.3.4 Timing chart error judgment (Close to user)




When the condition is not the error judgment enable (Error authorized =1 on above timing chart) and EC7TERE (error judgment enable signal) is not “High”, the 2bits error detect flag (ECER2F) and interrupt (EC7TIE2) are not occurred. (ECER1F: Flag bit of ECC 1bits error detection and correction and 1bit error interrupt is also. And ECEMF (ECC error detecting flag) is not outputted when the error judgment is enable and EC7TERE is not active.

#### 102.5.1.3.5 Error address capture overflow detection

When this IP has error address register (P_ERR_STAG=1-8), the overflow flag (ECOVFF) is set and the overflow interruption is outputted by detecting new error address under the condition that the register is full.
If all error address register is full of 1bit error address and 2bits error is detected, overflow flag (ECOVFF) and 2bits error detection flag (ECER2F) are set at the same time. And when 2bits error interruption output is permitted, so it occurs, too.

# 103. FlexRay

## 103.1 Overview

The FlexRay IP-module performs communication according to the FlexRay protocol specification v2.1. With maximum specified sample clock the bitrate is 10 MBit/s. Additional bus driver (BD) hardware is required for connection to the physical layer.

### 103.1.1 Features

For communication on a FlexRay network, individual message buffers with up to 254 data bytes are configurable. The message storage is a Message RAM that holds up to 128 message buffers. All functions concerning the handling of messages are implemented in the Message Handler. Those functions are the acceptance filtering, the transfer of messages between the two FlexRay Channel Protocol Controllers and the Message RAM, maintaining the transmission schedule as well as providing message status information.
The register set of the FlexRay IP-module can be accessed directly by an external Host via the module's Host interface. These registers are used to control/configure/monitor the FlexRay Channel Protocol Controllers, Message Handler, Global Time Unit, System Universal Control, Frame and Symbol Processing, Network Management, Interrupt Control, to access the Message RAM via Input / Output Buffer; and to control the data transfer between the Message RAM and the user RAM(URAM).

In this section, User RAM means to the local RAM (and external memory for the SoC if possible in your product).

The FlexRay IP-module supports the following features:

| Item | Specification |
|---|---|
| Communication | Comply with FlexRay protocol specification v2.1 |
| Data transfer rate | up to 10 Mbit/s on each channel |
| FlexRay channels | 2 (channel A and B) |
| Message buffer | Up to 128 message buffers configurable<br>Configuration of message buffers with different payload lengths possible<br>Each message buffer can be configured as receive buffer, as transmit buffer or as part of the receive FIFO<br>Filtering for slot counter, cycle counter, and channel |
| Message RAM | 8 Kbyte of Message RAM for storage of e.g.<br>128 message buffers with max. 48 byte data section<br>up to 30 message buffers with 254 byte data section |
| FIFO | One configurable receive FIFO |
| Message buffer access | by Host CPU via Input and Output Buffer<br>Input Buffer: Holds message to be transferred to the Message RAM<br>Output Buffer: Holds message read from the Message RAM |
| | by data transfer function<br>Input transfer: Message buffer content is transferred from URAM to Message RAM on CPU request<br>Output transfer: Message buffer content is transferred from Message RAM to URAM automatically |
| Network management | Automatic HW support |
| Interrupts | Maskable module interrupts |
| Timer | 2 absolute timers<br>1 relative timer |

### 103.1.2 Block Diagram

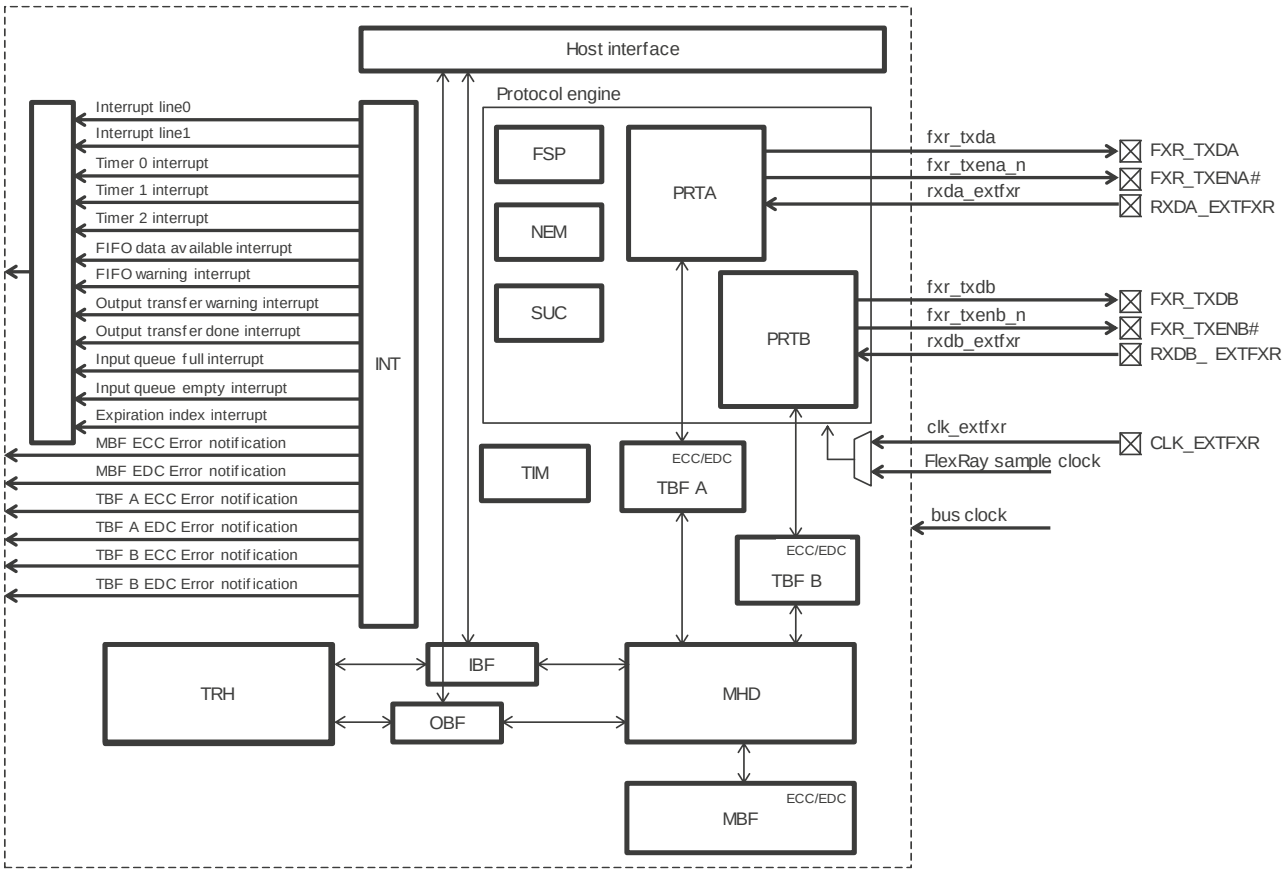



**Figure 103.1 FlexRay Block Diagram**

**Input Buffer (IBF)**
For write access to the message buffers configured in the Message RAM, the Host can write the header and data section for a specific message buffer to the Input Buffer. The Message Handler then transfers the data from the Input Buffer to the selected message buffer in the Message RAM.

**Output Buffer (OBF)**
For read access to a message buffer configured in the Message RAM the Message Handler transfers the selected message buffer to the Output Buffer. After the transfer has completed, the Host can read the header and data section of the transferred message buffer from the Output Buffer.

**Message Handler (MHD)**
The FlexRay Message Handler controls data transfers between the following components:

Input / Output Buffer and Message RAM

Transient Buffer RAMs of the two FlexRay Protocol Controllers and Message RAM

**Message RAM (MRAM, MBF)**
The Message RAM (MBF) consists of a single-ported RAM that stores up to 128 FlexRay message buffers together with the related configuration data (header and data partition).

**Transient Buffer RAM (TBF A/B)**
Stores the data section of two complete messages.

**FlexRay Channel Protocol Controllers (PRT A/B)**
The FlexRay Channel Protocol Controllers consist of shift register and FlexRay protocol FSM. They are connected to the Transient Buffer RAMs for intermediate message storage and to the physical layer via bus driver BD.

They perform the following functionality:
Control and check of bit timing
Reception / transmission of FlexRay frames and symbols
Check of header CRC
Generation / check of frame CRC
Interfacing to bus driver

**System Universal Control (SUC)**
The System Universal Control controls the following functions:
Configuration
Wakeup
Startup
Normal Operation
Passive Operation

**Frame and Symbol Processing (FSP)**
The Frame and Symbol Processing controls the following functions:
Checks the correct timing of frames and symbols
Tests the syntactical and semantical correctness of received frames
Sets the slot status flags

**Network Management (NEM)**
Handles the network management vector.

**Interrupt Control (INT)**
The Interrupt Controller performs the following functions:
Provides error and status interrupt flags
Enable / disable interrupt sources
Assignment of interrupt sources to one of the two general module interrupt lines
Enable / disable module interrupt lines

**Timer (TIM)**
The Timer module includes the following macrotick timer:
Two absolute timer
One relative timer
One stop watch timer

**Transfer Handler (TRH)**
Handles the data transfer between URAM and FlexRay module.
The Transfer Handler supports the following transfer types:
Transfer of buffer configuration data from the URAM to the Message RAM
Transfer of payload data for transmit buffer from the URAM to the Message RAM

Transfer of buffer configuration data and payload data for transmit buffer from the URAM to the Message RAM
Automatic transfer of payload data from receive buffer to the URAM upon frame reception
Automatic transfer of payload data, buffer configuration data and message buffer status data from receive buffer to the URAM upon frame reception
Automatic transfer of buffer configuration data and message buffer status data from dedicated transmit and receive buffer to the URAM upon slot status update
Manual transfer of payload data, buffer configuration data and message buffer status data from dedicated transmit and receive buffer to the URAM

### 103.1.3 External Pins

**Table 103.1 Pin Configuration**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| RXDA_EXTFXR | — | I | FlexRay receive data channel A |
| RXDB_EXTFXR | — | I | FlexRay receive data channel B |
| FXR_TXDA | — | O | FlexRay transmit data channel A |
| FXR_TXDB | — | O | FlexRay transmit data channel B |
| FXR_TXENA# | — | O | FlexRay transmit enable channel A |
| FXR_TXENB# | — | O | FlexRay transmit enable channel B |
| CLK_EXTFXR | — | I | Conformance test clock |

#### 103.1.3.1 RXDA_EXTFXR

FlexRay receive data channel A

If FlexRay communication is not used in systems, please input fixed value 1’b1 to this port.

#### 103.1.3.2 RXDB_EXTFXR

FlexRay receive data channel B

If FlexRay communication is not used in systems, please input fixed value 1’b1 to this port.

#### 103.1.3.3 FXR_TXDA

FlexRay transmit data signal channel A

Initial value is 1’b1. If FlexRay communication is not used in systems, please left fxr_txda open.

#### 103.1.3.4 FXR_TXDB

FlexRay transmit data signal channel B

Initial value is 1’b1. If FlexRay communication is not used in systems, please left fxr_txda open.

#### 103.1.3.5 FXR_TXENA#

FlexRay transmit enable signal channel A

When HIGH, channel A transmission is inactive. When LOW, channel A transmission is active.

#### 103.1.3.6 FXR_TXENB#

FlexRay transmit enable signal channel B

When HIGH, channel B transmission is inactive. When LOW, channel B transmission is active.

#### 103.1.3.7 CLK_EXTFXR

Conformance test clock

### 103.1.4 Register Configuration

The FlexRay module allocates an address space as shown in Table 103.2

Within this specification the "Values after reset" refers to the microcontroller's HW RESET. For registers in the address range h'0010 to h'0FFF the "Value after reset" are also applicable when the SW RESET (using **FROC.OE**) is applied.

The addresses given in this specification are relative addresses. The address offset ERAY needs to be added to all given addresses.

Offset Address : ERAY = h'E6B0

AccessSize : AS = 32/16/8

**Table 103.2 Register Configuration**

| Name | Abbreviation | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|---|
| FlexRay Operation Control register | FROC | R/W | <ERAY>+H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay Operation Status register | FROS | R/W | <ERAY>+H'000C | H'0000 0000 | 32,16,8 |
| FlexRay LoCK register | FRLCK | R/W | <ERAY>+H'001C | H'0000 0000 | 32,16,8 |
| FlexRay Error Interrupt Register | FREIR | R/W | <ERAY>+H'0020 | H'0000 0000 | 32,16,8 |
| FlexRay Status Interrupt Register | FRSIR | R/W | <ERAY>+H'0024 | H'0000 0000 | 32,16,8 |
| FlexRay Error Interrupt Line Select | FREILS | R/W | <ERAY>+H'0028 | H'0000 0000 | 32,16,8 |
| FlexRay Status Interrupt Line Select | FRSILS | R/W | <ERAY>+H'002C | H'0303 FFFF | 32,16,8 |
| FlexRay Error Interrupt Enable Set | FREIES | R/W | <ERAY>+H'0030 | H'0000 0000 | 32,16,8 |
| FlexRay Error Interrupt Enable Reset | FREIER | R/W | <ERAY>+H'0034 | H'0000 0000 | 32,16,8 |
| FlexRay Status Interrupt Enable Set | FRSIES | R/W | <ERAY>+H'0038 | H'0000 0000 | 32,16,8 |
| FlexRay Status Interrupt Enable Reset | FRSIER | R/W | <ERAY>+H'003C | H'0000 0000 | 32,16,8 |
| FlexRay Interrupt Line Enable | FRILE | R/W | <ERAY>+H'0040 | H'0000 0000 | 32,16,8 |
| FlexRay Timer 0 Configuration | FRT0C | R/W | <ERAY>+H'0044 | H'0000 0000 | 32,16,8 |
| FlexRay Timer 1 Configuration | FRT1C | R/W | <ERAY>+H'0048 | H'0002 0000 | 32,16,8 |
| FlexRay STop Watch register 1 | FRSTPW1 | R/W | <ERAY>+H'004C | H'0000 0000 | 32,16,8 |
| FlexRay STop Watch register 2 | FRSTPW2 | R/W | <ERAY>+H'0050 | H'0000 0000 | 32,16,8 |
| FlexRay SUC Configuration register 1 | FRSUCC1 | R/W | <ERAY>+H'0080 | H'0C40 1080 | 32,16,8 |
| FlexRay SUC Configuration register 2 | FRSUCC2 | R/W | <ERAY>+H'0084 | H'0100 0504 | 32,16,8 |
| FlexRay SUC Configuration register 3 | FRSUCC3 | R/W | <ERAY>+H'0088 | H'0000 0011 | 32,16,8 |
| FlexRay NEM Configuration register | FRNEMC | R/W | <ERAY>+H'008C | H'0000 0000 | 32,16,8 |
| FlexRay PRT Configuration register 1 | FRPRTC1 | R/W | <ERAY>+H'0090 | H'084C 0633 | 32,16,8 |
| FlexRay PRT Configuration register 2 | FRPRTC2 | R/W | <ERAY>+H'0094 | H'0F2D 0A0E | 32,16,8 |
| FlexRay MHD Configuration register | FRMHDC | R/W | <ERAY>+H'0098 | H'0000 0000 | 32,16,8 |
| FlexRay GTU Configuration register 1 | FRGTUC1 | R/W | <ERAY>+H'00A0 | H'0000 0280 | 32,16,8 |
| FlexRay GTU Configuration register 2 | FRGTUC2 | R/W | <ERAY>+H'00A4 | H'0002 000A | 32,16,8 |
| FlexRay GTU Configuration register 3 | FRGTUC3 | R/W | <ERAY>+H'00A8 | H'0202 0000 | 32,16,8 |

| Name | Abbreviation | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|---|
| FlexRay GTU Configuration register 4 | FRGTUC4 | R/W | <ERAY>+H'00AC | H'0008 0007 | 32,16,8 |
| FlexRay GTU Configuration register 5 | FRGTUC5 | R/W | <ERAY>+H'00B0 | H'0E00 0000 | 32,16,8 |
| FlexRay GTU Configuration register 6 | FRGTUC6 | R/W | <ERAY>+H'00B4 | H'0002 0000 | 32,16,8 |
| FlexRay GTU Configuration register 7 | FRGTUC7 | R/W | <ERAY>+H'00B8 | H'0002 0004 | 32,16,8 |
| FlexRay GTU Configuration register 8 | FRGTUC8 | R/W | <ERAY>+H'00BC | H'0000 0002 | 32,16,8 |
| FlexRay GTU Configuration register 9 | FRGTUC9 | R/W | <ERAY>+H'00C0 | H'0000 0101 | 32,16,8 |
| FlexRay GTU Configuration Register 10 | FRGTUC10 | R/W | <ERAY>+H'00C4 | H'0002 0005 | 32,16,8 |
| FlexRay GTU Configuration Register 11 | FRGTUC11 | R/W | <ERAY>+H'00C8 | H'0000 0000 | 32,16,8 |
| FlexRay CC Status Vector | FRCCSV | R | <ERAY>+H'0100 | H'0010 4000 | 32,16,8 |
| FlexRay CC Error Vector | FRCCEV | R | <ERAY>+H'0104 | H'0000 0000 | 32,16,8 |
| FlexRay Slot Counter Value | FRSCV | R | <ERAY>+H'0110 | H'0000 0000 | 32,16,8 |
| FlexRay Macrotick and Cycle Counter Value | FRMTCCV | R | <ERAY>+H'0114 | H'0000 0000 | 32,16,8 |
| FlexRay Rate Correction Value | FRRCV | R | <ERAY>+H'0118 | H'0000 0000 | 32,16,8 |
| FlexRay Offset Correction Value | FROCV | R | <ERAY>+H'011C | H'0000 0000 | 32,16,8 |
| FlexRay Sync Frame Status | FRSFS | R | <ERAY>+H'0120 | H'0000 0000 | 32,16,8 |
| FlexRay Symbol Window and NIT status | FRSWNIT | R | <ERAY>+H'0124 | H'0000 0000 | 32,16,8 |
| FlexRay Aggregated Channel Status | FRACS | R/W | <ERAY>+H'0128 | H'0000 0000 | 32,16,8 |
| FlexRay Even Sync ID [n](n=1-15) | FRESID[n] | R | <ERAY>+H'0130+([n]-1)*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay Odd Sync ID [n](n=1-15) | FROSID[n] | R | <ERAY>+H'0170+([n]-1)*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay Network Management Vector [n](n=1-3) | FRNMV[n] | R | <ERAY>+H'01B0+([n]-1)*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay Message Ram Configuration | FRMRC | R/W | <ERAY>+H'0300 | H'0180 0000 | 32,16,8 |
| FlexRay FIFO Rejection Filter | FRFRF | R/W | <ERAY>+H'0304 | H'0180 0000 | 32,16,8 |
| FlexRay FIFO Rejection Filter Mask | FRFRFM | R/W | <ERAY>+H'0308 | H'0000 0000 | 32,16,8 |
| FlexRay FIFO Critical Level | FRFCL | R/W | <ERAY>+H'030C | H'0000 0080 | 32,16,8 |
| FlexRay Message HanDler Status | FRMHDS | R/W | <ERAY>+H'0310 | H'0000 0080 | 32,16,8 |
| FlexRay Last Dynamic Transmit Slot | FRLDTS | R | <ERAY>+H'0314 | H'0000 0000 | 32,16,8 |
| FlexRay FIFO Status Register | FRFSR | R | <ERAY>+H'0318 | H'0000 0000 | 32,16,8 |
| FlexRay Message HanDler constraints Flags | FRMHDF | R/W | <ERAY>+H'031C | H'0000 0000 | 32,16,8 |
| FlexRay Transmission ReQuest [i](i=1-4) | FRTXRQ[i] | R | <ERAY>+H'0320+([i]-1)*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay New DATa [i](i=1-4) | FRNDAT[i] | R | <ERAY>+H'0330+([i]-1)*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay Message Buffer Status Changed [i](i=1-4) | FRMBSC[i] | R | <ERAY>+H'0340+([i]-1)*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay WRite Data Section [n](n=1-64) | FRWRDS[n] | R/W | <ERAY>+H'0400+([n]-1)*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay WRite Header Section 1 | FRWRHS1 | R/W | <ERAY>+H'0500 | H'0000 0000 | 32,16,8 |
| FlexRay WRite Header Section 2 | FRWRHS2 | R/W | <ERAY>+H'0504 | H'0000 0000 | 32,16,8 |

| Name | Abbreviation | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|---|
| FlexRay WRite Header Section 3 | FRWRHS3 | R/W | <ERAY>+H'0508 | H'0000 0000 | 32,16,8 |
| FlexRay Input Buffer Command Mask | FRIBCM | R/W | <ERAY>+H'0510 | H'0000 0000 | 32,16,8 |
| FlexRay Input Buffer Command Request | FRIBCR | R/W | <ERAY>+H'0514 | H'0000 0000 | 32,16,8 |
| FlexRay ReaD Data Section [n](n=1-64) | FRRDDS[n] | R | <ERAY>+H'0600+([n]-1)*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay ReaD Header Section 1 | FRRDHS1 | R | <ERAY>+H'0700 | H'0000 0000 | 32,16,8 |
| FlexRay ReaD Header Section 2 | FRRDHS2 | R | <ERAY>+H'0704 | H'0000 0000 | 32,16,8 |
| FlexRay ReaD Header Section 3 | FRRDHS3 | R | <ERAY>+H'0708 | H'0000 0000 | 32,16,8 |
| FlexRay Message Buffer Status | FRMBS | R | <ERAY>+H'070C | H'0000 0000 | 32,16,8 |
| FlexRay Output Buffer Command Mask | FROBCM | R/W | <ERAY>+H'0710 | H'0000 0000 | 32,16,8 |
| FlexRay Output Buffer Command Request | FROBCR | R/W | <ERAY>+H'0714 | H'0000 0000 | 32,16,8 |
| FlexRay Input Transfer Configuration register | FRITC | R/W | <ERAY>+H'0800 | H'0000 0000 | 32,16,8 |
| FlexRay Output Transfer Configuration register | FROTC | R/W | <ERAY>+H'0804 | H'0000 0000 | 32,16,8 |
| FlexRay Input pointer table Base Address register | FRIBA | R/W | <ERAY>+H'0808 | H'0000 0000 | 32,16,8 |
| FlexRay FIFO pointer table Base Address register | FRFBA | R/W | <ERAY>+H'080C | H'0000 0000 | 32,16,8 |
| FlexRay Output pointer table Base Address register | FROBA | R/W | <ERAY>+H'0810 | H'0000 0000 | 32,16,8 |
| FlexRay Input Queue Control register | FRIQC | R/W | <ERAY>+H'0814 | H'0000 0000 | 32,16,8 |
| FlexRay User Input transfer Request register | FRUIR | R/W | <ERAY>+H'0818 | H'0000 0000 | 32,16,8 |
| FlexRay User Output transfer Request register | FRUOR | R/W | <ERAY>+H'081C | H'0000 0000 | 32,16,8 |
| FlexRay Input Transfer Status register | FRITS | R/W | <ERAY>+H'0820 | H'0000 0000 | 32,16,8 |
| FlexRay Output Transfer Status register | FROTS | R/W | <ERAY>+H'0824 | H'0000 0000 | 32,16,8 |
| FlexRay Access Error Status register | FRAES | R/W | <ERAY>+H'0828 | H'0000 0000 | 32,16,8 |
| FlexRay Access Error Address register | FRAEA | R | <ERAY>+H'082C | H'0000 0000 | 32,16,8 |
| FlexRay message Data Available register [i](i=0-3) | FRDA[i] | R/W | <ERAY>+H'0830+[i:*H'0004 | H'0000 0000 | 32,16,8 |
| FlexRay Timer 2 Configuration | FRT2C | R/W | <ERAY>+H'0844 | H'0000 0000 | 32,16,8 |

### 103.1.5 Connected Module

**Table 103.3 Connected Modules**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| FlexRay | CPG | Clock output |
| | RST | Reset output |
| | INTC and ECM | Interrupt input |
| | PAD logic | I/O terminal connection |
| | | Bus connection<br>(register access and data transfer) |

## 103.2 Register Description

[Initial value Symbol Description]

- : Undefined value

[R/W Symbol Description]

R/W : read and write allowed. You can read the writing value.

R/WC0 : read and write allowed. 0 is written, the bit is initialized, but one of the writing will be ignored.

R : Read only variable. Please value is always set to 0 to write.

-/WB : write only variable. Read value is undefined.

### 103.2.1 Register Operation register

#### 103.2.1.1 FROC [h'0004]

FlexRay Operation Control register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | T2IE | T1IE | T0IE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | OEP | - | - | - | - | - | BEC | OE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18] | T2IE | 0 | R/W | Timer 2 interrupt enable. 0: Disabled , 1: Enabled |
| [17] | T1IE | 0 | R/W | Timer 1 interrupt enable. 0: Disabled , 1: Enabled |
| [16] | T0IE | 0 | R/W | Timer 0 interrupt enable. 0: Disabled , 1: Enabled |
| [15:8] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [7] | OEP | 0 | R/W | Operation Enable bit Protection.<br>0: OE is unprotected , 1: OE is protected |
| [6:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1] | BEC | 0 | R/W | Byte Endian Control. 0: Little endian , 1: Big endian |
| [0] | OE | 0 | R/W | Operation Enable.<br>0: Operation disabled, SW reset , 1: Operation Enabled |

### (1) FROC.T2IE

Timer 2 interrupt enable

This bit controls the timer 2 interrupt.

0: Disabled
No interrupt will be requested and the timer 2 interrupt line will be released if pending.

1: Enabled
Timer 2 interrupt will be asserted when **FROS.T2IS** is '1'.

### (2) FROC.T1IE

Timer 1 interrupt enable

The user should only set this bit to '1' when timer 1 interrupt is not enabled in the FlexRay Status interrupt enable register (**FRSIES.TI1E** should be '0').

This bit controls the timer 1 interrupt.

0: Disabled
No interrupt will be requested and the timer 1 interrupt line will be released if pending.

1: Enabled
Timer 1 interrupt will be asserted when **FROS.T1IS** is '1'.

### (3) FROC.T0IE

Timer 0 interrupt enable

The user should only set this bit to '1' when timer 0 interrupt is not enabled in the FlexRay Status interrupt enable register (**FRSIES.TI0E** should be '0').

This bit controls the timer 0 interrupt.

0: Disabled

No interrupt will be requested and the timer 0 interrupt line will be released if pending.

1: Enabled
Timer 0 interrupt will be asserted when **FROS.T0IS** is '1'.

### (4) FROC.OEP

Operation Enable bit Protection

This bit protects against unintended write access to the **FROC.OE** bit.

0: **OE** is unprotected
Write access to the **OE** bit is enabled

1: **OE** is protected
Write access to the **OE** bit is disabled

### (5) FROC.BEC

Byte Endian Control

The user should only change this bit when **FROS.OS** is '1'.

This bit controls the byte order on reading and writing the FlexRay Network Management Vector register (**FRNMVn**), FlexRay Write Data Section (**FRWRDSn**) and FlexRay Read Data Section (**FRRDDSn**). This bit also controls the byte order when reading or writing FlexRay payload data using the data transfer function.

For details about the byte alignment please refer to section 103.3.17.

**0**: Little endian
Byte alignment in **FRNMVn**, **FRWRDSn** and **FRRDDSn** is in little endian style.

**1**: Big endian
Byte alignment in **FRNMVn**, **FRWRDSn** and **FRRDDSn** is in big endian style.

### (6) FROC.OE

Operation Enable

The user can only write to this bit when **FROC.OEP** is '0'.

The user should only write this bit with '0' when **FROS.OS** is '1'.

The user should only write this bit with '1' when **FROS.OS** is '0' and the FlexRay sample clock is enabled.

This bit controls the operation state and serves the software reset of the FlexRay module. The operation status bit (**FROS.OS**) indicates whether the FlexRay module is in reset state or not.

0: Operation disabled, SW reset
Forcibly moves the FlexRay module to its reset state, whatever the state of the FlexRay module is.

1: Operation Enabled
Reset state of the FlexRay module is released.

### 103.2.1.2 FROS [h'000C]

FlexRay operation status register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | T2IS | T1IS | T0IS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | OS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18] | T2IS | 0 | R/W | Timer 2 Interrupt Status.<br>0: Timer 2 has not matched the conditions configured in the FRT2C register.<br>1: Timer 2 matched the conditions configured in the FRT2C register. |
| [17] | T1IS | 0 | R/W | Timer 1 Interrupt Status.<br>0: Timer 1 has not matched the conditions configured in the FRT1C register.<br>1: Timer 1 matched the conditions configured in the FRT1C register. |
| [16] | T0IS | 0 | R/W | Timer 0 Interrupt Status.<br>0: Timer 0 has not matched the conditions configured in the FRT0C register.<br>1: Timer 0 matched the conditions configured in the FRT0C register. |
| [15:1] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [0] | OS | 0 | R | Operation Status.<br>0: Operation disabled, reset state , 1: Operation enabled |

#### (1) FROS.T2IS

Timer 2 Interrupt Status

Writing '0' has no effect on the bit value.

This bit represents that the expiration criteria configured in **FRT2C** register has matched the FlexRay local time.

If enabled in **FROC.T2IE** the timer 2 interrupt is generated when **FROS.T2IS** is '1'.

[Clearing condition]
This bit is cleared by writing '1' to **FROS.T2IS**.
This bit is cleared when **FROS.OS** changes from '1' to '0'.

[Setting condition]
This bit is set when the expiration criteria configured in **FRT2C** matches the FlexRay local time.

### (2) FROS.T1IS

Timer 1 Interrupt Status

Writing '0' has no effect on the bit value.

This bit represents that the expiration criteria configured in **FRT1C** register has matched the FlexRay local time.

If enabled in **FROC.T1IE** the timer 1 interrupt is generated when **FROS.T1IS** is '1'.

[Clearing condition]
This bit is cleared by writing '1' to **FROS.T1IS**.
This bit is cleared when **FROS.OS** changes from '1' to '0'.

[Setting condition]
This bit is set when the expiration criteria configured in **FRT1C** matches the FlexRay local time.

### (3) FROS.T0IS

Timer 0 Interrupt Status

Writing '0' has no effect on the bit value.

This bit represents that the expiration criteria configured in **FRT0C** register has matched the FlexRay local time.

If enabled in **FROC.T0IE** the timer 0 interrupt is generated when **FROS.T0IS** is '1'.

[Clearing condition]
This bit is cleared by writing '1' to **FROS.T0IS**.
This bit is cleared when **FROS.OS** changes from '1' to '0'.

[Setting condition]
This bit is set when the expiration criteria configured in **FRT0C** matches the FlexRay local time.

### (4) FROS.OS

Operation Status

This bit represents if the FlexRay module is in the reset or the operation state.

When **FROS.OS** is '0' the FlexRay module gets initialized and registers mapped to the address area ERAY+h'0010 to ERAY+h'0FFF cannot be accessed; read access from these registers will return undefined data.

When **FROS.OS** is '1' it is possible to access to the address area ERAY+h'0010 to ERAY+h'0FFF and to perform FlexRay communication.

When **FROS.OS** changes from '0' to '1' all registers in the address range ERAY+h'0010 to ERAY+h'0FFF are set to the "Values after reset".

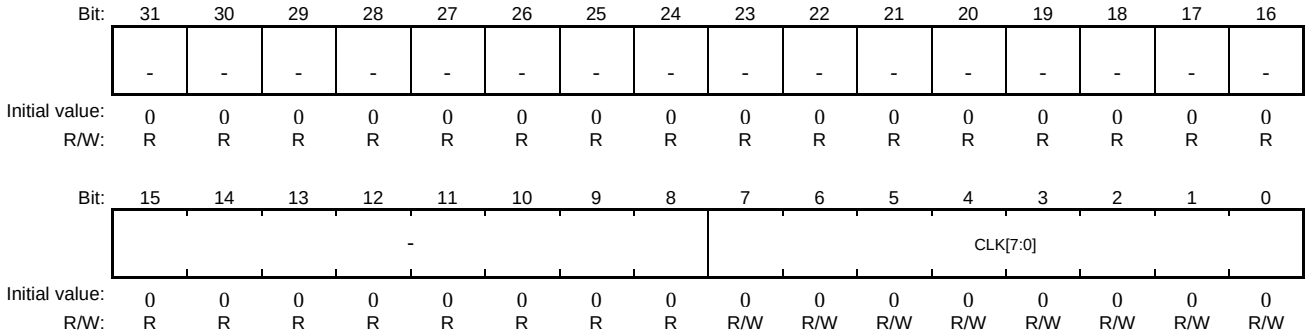
[Clearing condition]
When **FROC.OE** is set to ‘0’. It takes up to two peripheral bus clock cycles until **FROS.OS** is set to '0'.

[Setting condition]
When **FROC.OE** is set to ‘1’ it takes up to four peripheral clock cycles of the clock with the lower frequency out of the FlexRay sample clock and peripheral bus clock until **FROS.OS** is set to ‘1’.

### 103.2.2 Special Registers

#### 103.2.2.1 FRLCK [h'001C]

FlexRay Lock Register




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:16] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [15:8] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [7:0] | CLK | 0 | R/W | Configuration Lock Key |

**(1) FRLCK.CLK**

Configuration Lock Key

The Lock Register is write-only. Reading the register will return 0x0000 0000.

To leave CONFIG state by writing **FRSUCC1.CMD** (command READY), the write operation has to be directly preceded by two write accesses to the Configuration Lock Key (unlock sequence). If the write sequence below is interrupted by other write accesses between the second write to the Configuration Lock Key and the write access to the **FRSUCC1** register, the CC remains in CONFIG state and the sequence has to be repeated.

First write: **FRLCK.CLK** = "1100 1110" (0xCE)
Second write: **FRLCK.CLK** = "0011 0001" (0x31)
Third write: **FRSUCC1.CMD**

**Note:** In case that the Host uses 8/16-bit accesses to write the listed bit fields, the programmer has to ensure that no "dummy accesses" e.g. to the remaining register bytes / words are inserted by the compiler.

### 103.2.3 Interrupt Registers

#### 103.2.3.1 FREIR [h'0020]

FlexRay Error Interrupt Register

The flags are set when the CC detects one of the listed error conditions. The flags remain set until the Host clears them.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | TABB | LTVB | EDB | - | - | - | - | - | TABA | LTVA | EDA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | MHF | IOBA | IIBA | EFA | RFO | AERR | CCL | CCF | SFO | SFBM | CNA | PEMC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26] | TABB | 0 | R/W | Transmission Across Boundary Channel B<br>0 = No transmission across slot boundary detected on channel B<br>1 = Transmission across slot boundary detected on channel B |
| [25] | LTVB | 0 | R/W | Latest Transmit Violation Channel B<br>0 = No latest transmit violation detected on channel B<br>1 = Latest transmit violation detected on channel B |
| [24] | EDB | 0 | R/W | Error Detected on Channel B<br>0 = No error detected on channel B<br>1 = Error detected on channel B |
| [23:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18] | TABA | 0 | R/W | Transmission Across Boundary Channel A<br>0 = No transmission across slot boundary detected on channel A<br>1 = Transmission across slot boundary detected on channel A |
| [17] | LTVA | 0 | R/W | Latest Transmit Violation Channel A<br>0 = No latest transmit violation detected on channel A<br>1 = Latest transmit violation detected on channel A |
| [16] | EDA | 0 | R/W | Error Detected on Channel A<br>0 = No error detected on channel A<br>1 = Error detected on channel A |
| [15:12] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [11] | MHF | 0 | R/W | Message Handler Constraints Flag<br>0 = No Message Handler failure detected<br>1 = Message Handler failure detected |
| [10] | IOBA | 0 | R/W | Illegal Output buffer Access<br>0 = No illegal Host access to Output Buffer occurred<br>1 = Illegal Host access to Output Buffer occurred |
| [9] | IIBA | 0 | R/W | Illegal Input Buffer Access<br>0 = No illegal Host access to Input Buffer occurred<br>1 = Illegal Host access to Input Buffer occurred |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [8] | EFA | 0 | R/W | Empty FIFO Access<br>0 = No Host access to empty FIFO occurred<br>1 = Host access to empty FIFO occurred |
| [7] | RFO | 0 | R/W | Receive FIFO Overrun<br>0 = No receive FIFO overrun detected<br>1 = A receive FIFO overrun has been detected |
| [6] | AERR | 0 | R/W | Access Error<br>0 = No access error detected<br>1 = Access error detected |
| [5] | CCL | 0 | R/W | CHI Command Locked<br>0 = CHI command accepted<br>1 = CHI command not accepted |
| [4] | CCF | 0 | R/W | Clock Correction Failure<br>0 = No clock correction error<br>1 = Clock correction failed |
| [3] | SFO | 0 | R/W | Sync Frame Overflow<br>0 = Number of received sync frames <= FRGTUC2.SNM<br>1 = More sync frames received than configured by FRGTUC2.SNM |
| [2] | SFBM | 0 | R/W | Sync Frames Below Minimum<br>0 = Sync node: 1 or more sync frames received,<br>non-sync node: 2 or more sync frames received<br>1 = Less than the required minimum of sync frames received |
| [1] | CNA | 0 | R/W | Command Not Accepted<br>0 = CHI command accepted<br>1 = CHI command not accepted |
| [0] | PEMC | 0 | R/W | POC Error Mode Changed<br>0 = Error mode has not changed<br>1 = Error mode has changed |

(1) **FREIR.TABB**

Transmission Across Boundary Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag signals to the Host that a transmission across a slot boundary occurred for channel B.

(2) **FREIR.LTVB**

Latest Transmit Violation Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag signals a latest transmit violation on channel B to the Host.

(3) **FREIR.EDB**

Error Detected on Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This bit is set whenever one of the flags **FRACS.SEDB**, **FRACS.CEDB**, **FRACS.CIB**, **FRACS.SBVB** changes from ’0’ to ’1’.

(4) **FREIR.TABA**

Transmission Across Boundary Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag signals to the Host that a transmission across a slot boundary occurred for channel A.

(5) **FREIR.LTVA**

Latest Transmit Violation Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag signals a latest transmit violation on channel A to the Host.

(6) **FREIR.EDA**

Error Detected on Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This bit is set whenever one of the flags **FRACS.SEDA**, **FRACS.CEDA**, **FRACS.CIA**, **FRACS.SBVA** changes from ’0’ to ’1’.

(7) **FREIR.MHF**

Message Handler Constraints Flag

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag signals a Message Handler constraints violation condition. It is set whenever one of the flags **FRMHDF.SNUA**, **FRMHDF.SNUB**, **FRMHDF.FNFA**, **FRMHDF.FNFB**, **FRMHDF.TBFA**, **FRMHDF.TBFB**, **FRMHDF.WAHP** changes from ’0’ to ’1’.

(8) **FREIR.IOBA**

Illegal Output buffer Access

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Host requests the transfer of a message buffer from the Message RAM to the Output Buffer while **FROBCR.OBSYS** is set to ’1’.

(9) **FREIR.IIBA**

Illegal Input Buffer Access

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Host wants to modify a message buffer via Input Buffer and one of the following conditions applies:

1) The CC is not in CONFIG or DEFAULT_CONFIG state and the Host writes to the Input Buffer Command Request register to modify the
Header section of message buffer 0, 1 if configured for transmission in key slot
Header section of static message buffers with buffer number < **FRMRC.FDB** while **FRMRC.SEC** = "01"
Header section of any static or dynamic message buffer while **FRMRC.SEC** = "1x"
Header and / or data section of any message buffer belonging to the receive FIFO
2) The Host writes to any register of the Input Buffer while **FRIBCR.IBSYH** is set to ’1’.

(10) **FREIR.EFA**

Empty FIFO Access

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Host requests the transfer of a message from the receive FIFO via Output Buffer while the receive FIFO is empty.

(11) **FREIR.RFO**

Receive FIFO Overrun

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag is set by the CC when a receive FIFO overrun is detected. When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. The actual state of the FIFO is monitored in register **FRFSR**.

(12) **FREIR.AERR**

Access Error

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag signals an access error to the Host.

It is set when of the flags **FRMHDS.AMR**, **FRMHDS.ATBF1** or **FRMHDS.ATBF2** changes from ’0’ to ’1’.

(13) **FREIR.CCL**

CHI Command Locked

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag signals that the write access to the CHI command vector **FRSUCC1.CMD** was not successful because the execution of the previous CHI command has not yet completed. In this case bit **FREIR.CNA** is also set to ’1’.

(14) **FREIR.CCF**

Clock Correction Failure

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set at the end of the cycle whenever one of the following errors occurred:

Missing offset and / or rate correction

Clock correction limit reached

The clock correction status is monitored in registers **FRCCEV** and **FRSFS**. A failure may occur during startup, therefore bit **FREIR.CCF** should be cleared by the Host after the CC entered NORMAL_ACTIVE state.

(15) **FREIR.SFO**

Sync Frame Overflow

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

Set when either the number of sync frames received during the last communication cycle or the total number of different sync frame IDs received during the last double cycle exceeds the maximum number of sync frames as defined by **FRGTUC2.SNM**.

(16) **FREIR.SFBM**

Sync Frames Below Minimum

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set at the end of a cycle if the number of sync frames received during the last communication cycle was below the limit required for rate or offset correction term calculation (i.e. missing offset and / or missing rate correction). The clock correction status is monitored in **FRCCEV** and **FRSFS**.

The required limit may be dropped during startup. Therefore **FREIR.SFBM** should be cleared by the Host after the CC entered NORMAL_ACTIVE state.

(17) **FREIR .CNA**

Command Not Accepted

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

The flag signals that the write access to the CHI command vector **FRSUCC1.CMD** was not successful because the requested command was not valid in the actual POC state, or because the CHI command was locked (**FREIR.CCL** = ’1’).

(18) **FREIR .PEMC**

POC Error Mode Changed

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set whenever the error mode signaled by **FRCCEV.ERRM** has changed.

### 103.2.3.2 FRSIR [h'0024]

FlexRay Status Interrupt Register

The flags are set when the CC detects one of the listed events. The flags remain set until the Host clears them.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | MTSB | WUPB | - | - | - | - | - | - | MTSA | WUPA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SDS | MBSI | SUCS | SWE | TOBC | TIBC | TI1 | TI0 | NMVC | RFCL | RFNE | RXI | TXI | CYCS | CAS | WST |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:26] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [25] | MTSB | 0 | R/W | MTS Received on Channel B (vSS!ValidMTSB)<br>0 = No MTS symbol received on channel B<br>1 = MTS symbol received on channel B |
| [24] | WUPB | 0 | R/W | Wakeup Pattern Channel B<br>0 = No wakeup pattern received on channel B<br>1 = Wakeup pattern received on channel B |
| [23:18] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [17] | MTSA | 0 | R/W | MTS Received on Channel A (vSS!ValidMTSA)<br>0 = No MTS symbol received on channel A<br>1 = MTS symbol received on channel A |
| [16] | WUPA | 0 | R/W | Wakeup Pattern Channel A<br>0 = No wakeup pattern received on channel A<br>1 = Wakeup pattern received on channel A |
| [15] | SDS | 0 | R/W | Start of Dynamic Segment<br>0 = Dynamic segment not yet started<br>1 = Dynamic segment started |
| [14] | MBSI | 0 | R/W | Message Buffer Status Interrupt<br>0 = No message buffer status change of message buffer with MBI = '1'<br>1 = Message buffer status of at least one message buffer with MBI = '1' has changed |
| [13] | SUCS | 0 | R/W | Startup Completed Successfully<br>0 = No startup completed successfully<br>1 = Startup completed successfully |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [12] | SWE | 0 | R/W | Stop Watch Event<br>0 = No Stop Watch Event<br>1 = Stop Watch Event occurred |
| [11] | TOBC | 0 | R/W | Transfer Output Buffer Completed<br>0 = No transfer completed<br>1 = Transfer between Message RAM and Output Buffer completed |
| [10] | TIBC | 0 | R/W | Transfer Input Buffer Completed<br>0 = No transfer completed<br>1 = Transfer between Input Buffer and Message RAM completed |
| [9] | TI1 | 0 | R/W | Timer Interrupt 1<br>0 = No timer interrupt 1<br>1 = Timer interrupt 1 occurred |
| [8] | TI0 | 0 | R/W | Timer Interrupt 0<br>0 = No timer interrupt 0<br>1 = Timer interrupt 0 occurred |
| [7] | NMVC | 0 | R/W | Network Management Vector Changed<br>0 = No change in the network management vector<br>1 = Network management vector changed |
| [6] | RFCL | 0 | R/W | Receive FIFO Critical Level<br>0 = Receive FIFO below critical level<br>1 = Receive FIFO critical level reached |
| [5] | RFNE | 0 | R/W | Receive FIFO Not Empty<br>0 = Receive FIFO is empty<br>1 = Receive FIFO is not empty |
| [4] | RXI | 0 | R/W | Receive Interrupt<br>0 = No ND flag of a receive buffer with MBI = ’1’ has been set to ’1’<br>1 = At least one ND flag of a receive buffer with MBI = ’1’ has been set to ’1’ |
| [3] | TXI | 0 | R/W | Transmit Interrupt<br>0 = No frame transmitted from a transmit buffer with MBI = ’1’<br>1 = At least one frame was transmitted from a transmit buffer with MBI = ’1’ |
| [2] | CYCS | 0 | R/W | Cycle Start Interrupt<br>0 = No communication cycle started<br>1 = Communication cycle started |
| [1] | CAS | 0 | R/W | Collision Avoidance Symbol<br>0 = No bit pattern matching the CAS symbol received<br>1 = Bit pattern matching the CAS symbol received |
| [0] | WST | 0 | R/W | Wakeup Status<br>0 = Wakeup status unchanged<br>1 = Wakeup status changed |

(1) **FRSIR.MTSB**

MTS Received on Channel B (vSS!ValidMTSB)

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

Media Access Test symbol received on channel B during the preceding symbol window.

Updated by the CC for each channel at the end of the symbol window.

(2) **FRSIR.WUPB**

Wakeup Pattern Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when a wakeup pattern was received on channel B.
Only set when the CC is in one of these states:
WAKEUP
READY
STARTUP

(3) **FRSIR.MTSA**

MTS Received on Channel A (vSS!ValidMTSA)

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

Media Access Test symbol received on channel A during the preceding symbol window.

Updated by the CC for each channel at the end of the symbol window.

(4) **FRSIR.WUPA**

Wakeup Pattern Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when a wakeup pattern was received on channel A.
Only set when the CC is in one of these states:
WAKEUP
READY
STARTUP

(5) **FRSIR.SDS**

Start of Dynamic Segment

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the dynamic segment starts.

(6) **FRSIR.MBSI**

Message Buffer Status Interrupt

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the message buffer status **FRMBS** has changed and if bit **MBI** of that message buffer is set (see received valid data frames only. Table 103.3).

(7) **FRSIR.SUCS**

Startup Completed Successfully

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set whenever a startup completed successfully and the CC entered NORMAL_ACTIVE state.

(8) **FRSIR.SWE**

Stop Watch Event

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set after a stop watch activation when the actual cycle counter and macrotick value are stored in the Stop Watch register (see section 103.2.4.4).

(9) **FRSIR.TOBC**

Transfer Output Buffer Completed

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set whenever a transfer from the Message RAM to the Output Buffer has completed and **FROBCR.OBSYS** has been reset by the Message Handler.

(10) **FRSIR.TIBC**

Transfer Input Buffer Completed

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set whenever a transfer from Input Buffer to the Message RAM has completed and **FRIBCR.IBSYS** has been reset by the Message Handler.

(11) **FRSIR.TI1**

Timer Interrupt 1

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set whenever timer 1 matches the conditions configured in register **FRT1C**.

A FlexRay timer interrupt 1 is generated if enabled in **FROC.T1IE**.

(12) **FRSIR.TI0**

Timer Interrupt 0

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set whenever timer 0 matches the conditions configured in register **FRT0C**.

A FlexRay timer interrupt 0 is generated if enabled in **FROC.T0IE**.

(13) **FRSIR.NMVC**

Network Management Vector Changed

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This interrupt flag signals a change in the Network Management Vector visible to the Host.

(14) **FRSIR.RFCL**

Receive FIFO Critical Level

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set when the receive FIFO fill level **FRFSR.RFFL** is equal or greater than the critical level as configured by **FRFCL.CL**.

(15) **FRSIR.RFNE**

Receive FIFO Not Empty

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when a received valid frame was stored into the empty receive FIFO. The actual state of the receive FIFO is monitored in register **FRFSR**.

(16) **FRSIR.RXI**

Receive Interrupt

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC whenever the set condition of a message buffers **ND** flag is fulfilled (see section 103.2.8.6), and if bit **MBI** of that message buffer is set to ’1’ (see received valid data frames only. Table 103.3)

(17) **FRSIR.TXI**

Transmit Interrupt

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC at the end of frame transmission if bit **MBI** in the respective message buffer is set to ’1’ (see received valid data frames only. Table 103.3).

(18) **FRSIR.CYCS**

Cycle Start Interrupt

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when a communication cycle starts.

(19) **FRSIR.CAS**

Collision Avoidance Symbol

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC during STARTUP state when a CAS or a potential CAS was received.

(20) **FRSIR.WST**

Wakeup Status

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set when **FRCCSV.WSV** changes to a value other than UNDEFINED.

### 103.2.3.3 FREILS [h'0028]

FlexRay Error Interrupt Line Select

The FlexRay Error Interrupt Line Select register assigns an interrupt generated by a specific error interrupt flag from register **FREIR** to one of the FlexRay IP interrupt lines (FlexRay interrupt 0, FlexRay interrupt 1).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | TABBL | LTVBL | EDBL | - | - | - | - | - | TABAL | LTVAL | EDAL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | MHFL | IOBAL | IIBAL | EFAL | RFOL | AERRL | CCLL | CCFL | SFOL | SFBML | CNAL | PEMCL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26] | TABBL | 0 | R/W | Transmission Across Boundary Channel B Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [25] | LTVBL | 0 | R/W | Latest Transmit Violation Channel B Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [24] | EDBL | 0 | R/W | Error Detected on Channel B Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [23:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18] | TABAL | 0 | R/W | Transmission Across Boundary Channel A Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [17] | LTVAL | 0 | R/W | Latest Transmit Violation Channel A Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [16] | EDAL | 0 | R/W | Error Detected on Channel A Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [15:12] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [11] | MHFL | 0 | R/W | Message Handler Constraints Flag Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [10] | IOBAL | 0 | R/W | Illegal Output Buffer Access Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [9] | IIBAL | 0 | R/W | Illegal Input Buffer Access Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [8] | EFAL | 0 | R/W | Empty FIFO Access Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [7] | RFOL | 0 | R/W | Receive FIFO Overrun Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [6] | AERRL | 0 | R/W | Access Error Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [5] | CCLL | 0 | R/W | CHI Command Locked Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [4] | CCFL | 0 | R/W | Clock Correction Failure Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [3] | SFOL | 0 | R/W | Sync Frame Overflow Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [2] | SFBML | 0 | R/W | Sync Frames Below Minimum Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [1] | CNAL | 0 | R/W | Command Not Accepted Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [0] | PEMCL | 0 | R/W | POC Error Mode Changed Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |

### 103.2.3.4 FRSILS [h'002C]

FlexRay Status Interrupt Line Select

The FlexRay Status Interrupt Line Select register assign an interrupt generated by a specific status interrupt flag from register **FRSIR** to one of the two module interrupt lines (FlexRay interrupt 0, FlexRay interrupt 1).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | MTSBL | WUPBL | - | - | - | - | - | - | MTSAL | WUPAL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SDSL | MBSIL | SUCSL | SWEL | TOBCL | TIBCL | TI1L | TI0L | NMVCL | RFCLL | RFNEL | RXIL | TXIL | CYCSL | CASL | WSTL |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:26] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [25] | MTSBL | 1 | R/W | Media Access Test Symbol Channel B Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [24] | WUPBL | 1 | R/W | Wakeup Pattern Channel B Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [23:18] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [17] | MTSAL | 1 | R/W | Media Access Test Symbol Channel A Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [16] | WUPAL | 1 | R/W | Wakeup Pattern Channel A Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [15] | SDSL | 1 | R/W | Start of Dynamic Segment Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [14] | MBSIL | 1 | R/W | Message Buffer Status Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [13] | SUCSL | 1 | R/W | Startup Completed Successfully Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [12] | SWEL | 1 | R/W | Stop Watch Event Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [11] | TOBCL | 1 | R/W | Transfer Output Buffer Completed Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [10] | TIBCL | 1 | R/W | Transfer Input Buffer Completed Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [9] | TI1L | 1 | R/W | Timer Interrupt 1 Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [8] | TI0L | 1 | R/W | Timer Interrupt 0 Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [7] | NMVCL | 1 | R/W | Network Management Vector Changed Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [6] | RFCLL | 1 | R/W | Receive FIFO Critical Level Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [5] | RFNEL | 1 | R/W | Receive FIFO Not Empty Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [4] | RXIL | 1 | R/W | Receive Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [3] | TXIL | 1 | R/W | Transmit Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [2] | CYCSL | 1 | R/W | Cycle Start Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [1] | CASL | 1 | R/W | Collision Avoidance Symbol Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |
| [0] | WSTL | 1 | R/W | Wakeup Status Interrupt Line<br>0 = Interrupt assigned to FlexRay interrupt 0<br>1 = Interrupt assigned to FlexRay interrupt 1 |

### 103.2.3.5 FREIES [h'0030]

FlexRay Error Interrupt Enable Set

The settings in the FlexRay Error Interrupt Enable Set and FlexRay Error Interrupt Enable Reset registers determine which status changes in the FlexRay Error Interrupt Register will result in an interrupt.

The enable bits are set by writing to **FREIES** and reset by writing to **FREIER**. Reading from both addresses will result in the same value.

Writing '0' has no effect on the bit value.

Writing a '1' sets the interrupt enable bit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | TABBE | LTVBE | EDBE | - | - | - | - | - | TABAE | LTVAE | EDAE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | MHFE | IOBAE | IIBAE | EFAE | RFOE | AERRE | CCLE | CCFE | SFOE | SFBME | CNAE | PEMCE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26] | TABBE | 0 | R/W | Transmission Across Boundary Channel B Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [25] | LTVBE | 0 | R/W | Latest Transmit Violation Channel B Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [24] | EDBE | 0 | R/W | Error Detected on Channel B Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [23:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18] | TABAE | 0 | R/W | Transmission Across Boundary Channel A Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [17] | LTVAE | 0 | R/W | Latest Transmit Violation Channel A Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [16] | EDAE | 0 | R/W | Error Detected on Channel A Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [15:12] | - | 0 | R/W | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [11] | MHFE | 0 | R/W | Message Handler Constraints Flag Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [10] | IOBAE | 0 | R/W | Illegal Output Buffer Access Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [9] | IIBAE | 0 | R/W | Illegal Input Buffer Access Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [8] | EFAE | 0 | R/W | Empty FIFO Access Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [7] | RFOE | 0 | R/W | Receive FIFO Overrun Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [6] | AERRE | 0 | R/W | Access Error Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [5] | CCLE | 0 | R/W | CHI Command Locked Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [4] | CCFE | 0 | R/W | Clock Correction Failure Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [3] | SFOE | 0 | R/W | Sync Frame Overflow Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [2] | SFBME | 0 | R/W | Sync Frames Below Minimum Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [1] | CNAE | 0 | R/W | Command Not Accepted Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [0] | PEMCE | 0 | R/W | POC Error Mode Changed Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |

### 103.2.3.6 FREIER [h'0034]

FlexRay Error Interrupt Enable Reset

The settings in the FlexRay Error Interrupt Enable Set and FlexRay Error Interrupt Enable Reset registers determine which status changes in the FlexRay Error Interrupt Register will result in an interrupt.

The enable bits are set by writing to **FREIES** and reset by writing to **FREIER**. Reading from both addresses will result in the same value.

Writing '0' has no effect on the bit value.

Writing a '1' clears the interrupt enable bit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | TABBD | LTVBD | EDBD | - | - | - | - | - | TABAD | LTVAD | EDAD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | MHFD | IOBAD | IIBAD | EFAD | RFOD | AERRD | CCLD | CCFD | SFOD | SFBMD | CNAD | PEMCD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26] | TABBD | 0 | R/W | Transmission Across Boundary Channel B Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [25] | LTVBD | 0 | R/W | Latest Transmit Violation Channel B Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [24] | EDBD | 0 | R/W | Error Detected on Channel B Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [23:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18] | TABAD | 0 | R/W | Transmission Across Boundary Channel A Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [17] | LTVAD | 0 | R/W | Latest Transmit Violation Channel A Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [16] | EDAD | 0 | R/W | Error Detected on Channel A Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [15:12] | - | 0 | R/W | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [11] | MHFD | 0 | R/W | Message Handler Constraints Flag Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [10] | IOBAD | 0 | R/W | Illegal Output Buffer Access Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [9] | IIBAD | 0 | R/W | Illegal Input Buffer Access Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [8] | EFAD | 0 | R/W | Empty FIFO Access Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [7] | RFOD | 0 | R/W | Receive FIFO Overrun Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [6] | AERRD | 0 | R/W | Access Error Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [5] | CCLD | 0 | R/W | CHI Command Locked Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [4] | CCFD | 0 | R/W | Clock Correction Failure Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [3] | SFOD | 0 | R/W | Sync Frame Overflow Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [2] | SFBMD | 0 | R/W | Sync Frames Below Minimum Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [1] | CNAD | 0 | R/W | Command Not Accepted Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [0] | PEMCD | 0 | R/W | POC Error Mode Changed Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |

### 103.2.3.7 FRSIES [h'0038]

FlexRay Status Interrupt Enable Set

The settings in the FlexRay Status Interrupt Enable Set and FlexRay Status Interrupt Enable Reset registers determine which status changes in the FlexRay Status Interrupt Register will result in an interrupt.

The enable bits are set by writing to **FRSIES** and reset by writing to **FRSIER**. Reading from both addresses will result in the same value.

Writing '0' has no effect on the bit value.

Writing a '1' sets the interrupt enable bit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | MTSBE | WUPBE | - | - | - | - | - | - | MTSAE | WUPAE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SDSE | MBSIE | SUCSE | SWEE | TOBCE | TIBCE | TI1E | TI0E | NMVCE | RFCLE | RFNEE | RXIE | TXIE | CYCSE | CASE | WSTE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:26] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [25] | MTSBE | 0 | R/W | MTS Received on Channel B Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [24] | WUPBE | 0 | R/W | Wakeup Pattern Channel B Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [23:18] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [17] | MTSAE | 0 | R/W | MTS Received on Channel A Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [16] | WUPAE | 0 | R/W | Wakeup Pattern Channel A Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [15] | SDSE | 0 | R/W | Start of Dynamic Segment Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [14] | MBSIE | 0 | R/W | Message Buffer Status Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [13] | SUCSE | 0 | R/W | Startup Completed Successfully Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [12] | SWEE | 0 | R/W | Stop Watch Event Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [11] | TOBCE | 0 | R/W | Transfer Output Buffer Completed Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [10] | TIBCE | 0 | R/W | Transfer Input Buffer Completed Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [9] | TI1E | 0 | R/W | Timer Interrupt 1 Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [8] | TI0E | 0 | R/W | Timer Interrupt 0 Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [7] | NMVCE | 0 | R/W | Network Management Vector Changed Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [6] | RFCLE | 0 | R/W | Receive FIFO Critical Level Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [5] | RFNEE | 0 | R/W | Receive FIFO Not Empty Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [4] | RXIE | 0 | R/W | Receive Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [3] | TXIE | 0 | R/W | Transmit Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [2] | CYCSE | 0 | R/W | Cycle Start Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [1] | CASE | 0 | R/W | Collision Avoidance Symbol Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [0] | WSTE | 0 | R/W | Wakeup Status Interrupt Enable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |

### 103.2.3.8 FRSIER [h'003C]

FlexRay Status Interrupt Enable Reset

The settings in the FlexRay Status Interrupt Enable Set and FlexRay Status Interrupt Enable Reset registers determine which status changes in the FlexRay Status Interrupt Register will result in an interrupt.

The enable bits are set by writing to **FRSIES** and reset by writing to **FRSIER**. Reading from both addresses will result in the same value.

Writing '0' has no effect on the bit value.

Writing a '1' clears the interrupt enable bit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | MTSBD | WUPBD | - | - | - | - | - | - | MTSAD | WUPAD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SDSD | MBSID | SUCSD | SWED | TOBCD | TIBCD | TI1D | TI0D | NMVCD | RFCLD | RFNED | RXID | TXID | CYCSD | CASD | WSTD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:26] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [25] | MTSBD | 0 | R/W | MTS Received on Channel B Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [24] | WUPBD | 0 | R/W | Wakeup Pattern Channel B Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [23:18] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [17] | MTSAD | 0 | R/W | MTS Received on Channel A Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [16] | WUPAD | 0 | R/W | Wakeup Pattern Channel A Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [15] | SDSD | 0 | R/W | Start of Dynamic Segment Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [14] | MBSID | 0 | R/W | Message Buffer Status Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [13] | SUCSD | 0 | R/W | Startup Completed Successfully Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [12] | SWED | 0 | R/W | Stop Watch Event Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [11] | TOBCD | 0 | R/W | Transfer Output Buffer Completed Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [10] | TIBCD | 0 | R/W | Transfer Input Buffer Completed Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [9] | TI1D | 0 | R/W | Timer Interrupt 1 Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [8] | TI0D | 0 | R/W | Timer Interrupt 0 Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [7] | NMVCD | 0 | R/W | Network Management Vector Changed Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [6] | RFCLD | 0 | R/W | Receive FIFO Critical Level Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [5] | RFNED | 0 | R/W | Receive FIFO Not Empty Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [4] | RXID | 0 | R/W | Receive Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [3] | TXID | 0 | R/W | Transmit Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [2] | CYCSD | 0 | R/W | Cycle Start Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [1] | CASD | 0 | R/W | Collision Avoidance Symbol Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |
| [0] | WSTD | 0 | R/W | Wakeup Status Interrupt Disable<br>0 = Interrupt disabled<br>1 = Interrupt enabled |

### 103.2.3.9 FRILE [h'0040]

FlexRay Interrupt Line Enable

Each of the two module interrupt lines (FlexRay interrupt 0, FlexRay interrupt 1) can be enabled / disabled separately by programming bit **FRILE.EINT0** and **FRILE.EINT1**.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | EINT1 | EINT0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1] | EINT1 | 0 | R/W | Enable FlexRay Interrupt Line 1<br>0 = FlexRay interrupt 1 disabled<br>1 = FlexRay interrupt 1 enabled |
| [0] | EINT0 | 0 | R/W | Enable FlexRay Interrupt Line 0<br>0 = FlexRay interrupt 0 disabled<br>1 = FlexRay interrupt 0 enabled |

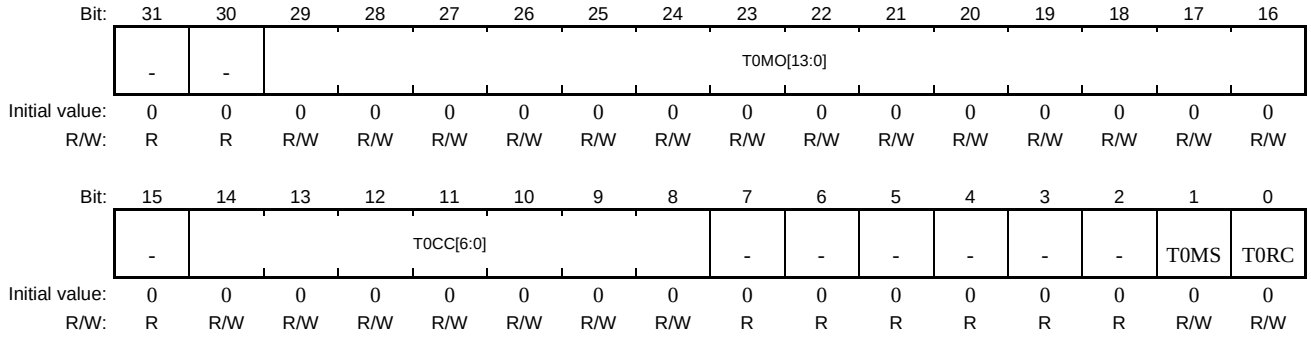
## 103.2.4 FlexRay Timer

### 103.2.4.1 FRT0C [h'0044]

FlexRay Timer 0 Configuration

Absolute timer.
Specifies in terms of cycle count and macrotick the point in time when the FlexRay timer 0 interrupt occurs.
When the FlexRay timer 0 elapses, **FRSIR.TI0** and **FROS.T0IS** are set to '1' and, if enabled in **FROC.T0IE** the FlexRay timer interrupt 0 is asserted.

**Note:** The configuration of timer 0 is compared against the macrotick counter value, there is no separate counter for timer 0.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29:16] | T0MO | 0 | R/W | Timer 0 Macrotick Offset |
| [15] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [14:8] | T0CC | 0 | R/W | Timer 0 Cycle Code |
| [7:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1] | T0MS | 0 | R/W | Timer 0 Mode Select<br>0 = Single-shot mode<br>1 = Continuous mode |
| [0] | T0RC | 0 | R/W | Timer 0 Run Control<br>0 = Timer 0 halted<br>1 = Timer 0 running |

(1) **FRT0C.T0MO**

Timer 0 Macrotick Offset

Before reconfiguration of the timer, the timer has to be halted first by writing bit **FRT0C.T0RC** to ’0’.

Configures the macrotick offset from the beginning of the cycle where the interrupt is to occur. The FlexRay timer 0 interrupt occurs at this offset for each cycle of the cycle set.

(2) **FRT0C.T0CC**

Timer 0 Cycle Code

Before reconfiguration of the timer, the timer has to be halted first by writing bit **FRT0C.T0RC** to ’0’.

The 7-bit timer 0 cycle code determines the cycle set used for generation of the FlexRay timer 0 interrupt. For details about the configuration of the cycle code see section 103.3.8.2.

(3) **FRT0C.T0MS**

Timer 0 Mode Select

Before reconfiguration of the timer, the timer has to be halted first by writing bit **FRT0C.T0RC** to ’0’.

Configures the timer run mode. In Single-shot mode the timer is deactivated when the timer configuration matches the configured cycle counter and macrotick value.

(4) **FRT0C.T0RC**

Timer 0 Run Control

Timer 0 can be only activated (set **FRT0C.T0RC** to '1') when the POC is either in NORMAL_ACTIVE state or in NORMAL_PASSIVE state.

Timer 0 is deactivated when leaving NORMAL_ACTIVE state or NORMAL_PASSIVE state except for transitions between the two states.

### 103.2.4.2 FRT1C [h'0048]

FlexRay Timer 1 Configuration

Relative timer.
After the specified number of macroticks has expired, the FlexRay timer 1 interrupt is asserted.
When the FlexRay timer 1 elapses, **FRSIR.TI1** and **FROS.T1IS** are set to '1' and, if enabled in **FROC.T1IE** the FlexRay timer interrupt 1 is asserted.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | T1MC[13:0] | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | T1MS | T1RC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29:16] | T1MC | 2h | R/W | Timer 1 Macrotick Count |
| [15:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1] | T1MS | 0 | R/W | Timer 1 Mode Select<br>0 = Single-shot mode<br>1 = Continuous mode |
| [0] | T1RC | 0 | R/W | Timer 1 Run Control<br>0 = Timer 1 halted<br>1 = Timer 1 running |

(1) **FRT1C.T1MC**

Timer 1 Macrotick Count

Before reconfiguration of the timer, the timer has to be halted first by writing bit **FRT1C.T1RC** to '0'.

Valid values are 2 to 16383 MT in continuous mode
Valid values are 1 to 16383 MT in single-shot mode

When the configured macrotick count is reached the FlexRay timer 1 interrupt is generated.

(2) **FRT1C.T1MS**

Timer 1 Mode Select

Before reconfiguration of the timer, the timer has to be halted first by writing bit **FRT1C.T1RC** to ’0’.

Configures the timer run mode. In Single-shot mode the timer is deactivated when the timer configuration matches the configured cycle counter and macrotick value.

(3) **FRT1C.T1RC**

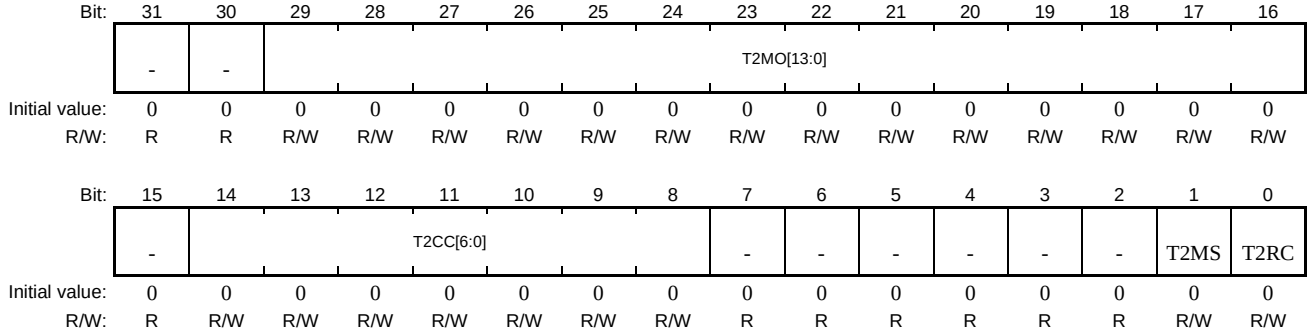
Timer 1 Run Control

Timer 1 can be activated (set **FRT1C.T1RC** to '1') as long as the POC is either in NORMAL_ACTIVE state or in NORMAL_PASSIVE state.

Timer 1 is deactivated when leaving NORMAL_ACTIVE state or NORMAL_PASSIVE state except for transitions between the two states.

### 103.2.4.3 FRT2C [h'0844]

FlexRay Timer 2 Configuration




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29:16] | T2MO | 0 | R/W | Timer 2 Macrotick Offset |
| [15] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [14:8] | T2CC | 0 | R/W | Timer 2 Cycle Code |
| [7:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1] | T2MS | 0 | R/W | Timer 2 Mode Select<br>0: Single-shot mode<br>1: Continuous mode |
| [0] | T2RC | 0 | R/W | Timer 2 Run Control<br>0: Timer halted<br>1: Timer running |

(1) **FRT2C.T2MO**

Timer 2 Macrotick Offset

The user can only write to these bits when **FRT2C.T2RC** is '0'.

Allowed range is 0 to **FRGTUC2.MPC**.

These bits represent the timer macrotick value defining the timer expiration condition.

(2) **FRT2C.T2CC**

Timer 2 Cycle Code

The user can only write to these bits when **FRT2C.T2RC** is '0'.

These bits represent the cycle counter filter code defining the timer expiration condition. See section 103.3.8.2 for cycle filtering.

(3) **FRT2C.T2MS**

Timer 2 Mode Select

The user can only write to these bits when **FRT2C.T2RC** is '0'.

This bit represents the operation mode of the Timer 2.

0: Single-shot mode

The timer is operating in the non-repetitive (single shot) mode. Once the configured expiration criteria are matching the timer will be automatically halted.

1: Continuous mode

The timer is operating in the repetitive (continuous) mode. The timer will expire every time the configured expiration criteria are matching. The timer is not halted.

(4) **FRT2C.T2RC**

Timer 2 Run Control

The user can only set this bit to '1' when the POC is in NORMAL_ACTIVE or NORMAL_PASSIVE state.

This bit represents the activation state of the Timer 2.

When the expiration criteria are matching for a single shot timer, then this bit is cleared automatically and the Timer 2 is halted.

[Setting condition]
This bit is set by writing '1' to it when the POC is in NORMAL_ACTIVE or NORMAL_PASSIVE state.

[Clearing condition]
This bit is cleared when the POC state leaves the NORMAL_ACTIVE or NORMAL_PASSIVE state except for transitions between the two states.

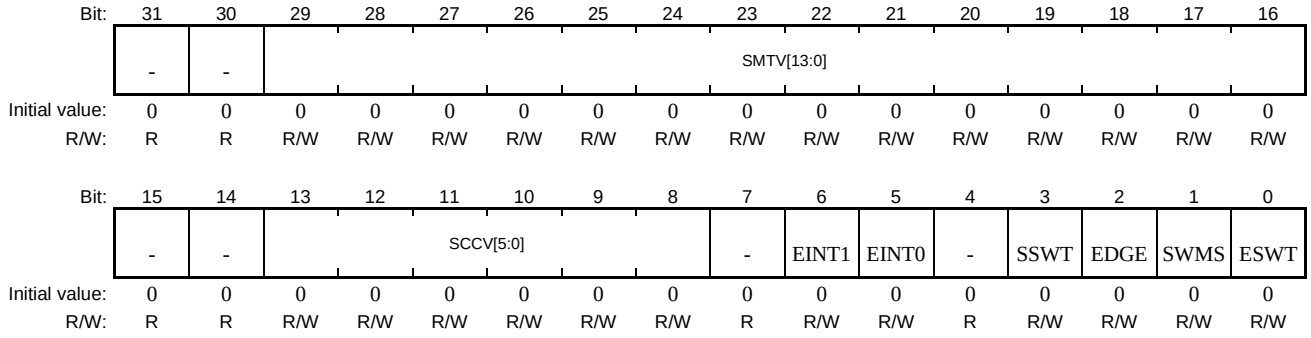
This bit is cleared when the timer is operating in the non-repetitive (single shot) mode (**FRT2C.T2MS** is '0') and the expiration criteria are matching.

This bit is cleared by writing '0' to it.

### 103.2.4.4 FRSTPW1 [h'004C]

FlexRay Stop Watch Register 1

The stop watch is activated by
a FlexRay interrupt 0 event, FlexRay interrupt 1 event or
by the Host by writing bit **FRSTPW1.SSWT** to '1'.
With the macrotick counter increment following next to the stop watch activation the actual cycle counter and macrotick values are captured in register **FRSTPW1** while the slot counter values for channel A and B are captured in register **FRSTPW2**.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29:16] | SMTV | 0 | R/W | Stop Watch Captured Macrotick Value |
| [15:14] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [13:8] | SCCV | 0 | R/W | Stop Watch Captured Cycle Counter Value |
| [7] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [6] | EINT1 | 0 | R/W | Enable FlexRay Interrupt 1 Trigger<br>0 = Stop watch trigger by FlexRay interrupt 1 disabled<br>1 = FlexRay interrupt 1 event triggers stop watch |
| [5] | EINT0 | 0 | R/W | Enable FlexRay Interrupt 0 Trigger<br>0 = Stop watch trigger by FlexRay interrupt 0 disabled<br>1 = FlexRay interrupt 0 event triggers stop watch |
| [4] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [3] | SSWT | 0 | R/W | Software Stop Watch Trigger<br>0 = Software trigger reset<br>1 = Stop watch activated by software trigger |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [2] | EDGE | 0 | R/W | Stop Watch Trigger Edge Select<br>0 = Falling edge<br>1 = Rising edge |
| [1] | SWMS | 0 | R/W | Stop Watch Mode Select<br>0 = Single-shot mode<br>1 = Continuous mode |
| [0] | ESWT | 0 | R/W | Enable Stop Watch Trigger<br>0 = Stop watch trigger disabled<br>1 = Stop watch trigger enabled |

(1) **FRSTPW1.SMTV**

Stop Watch Captured Macrotick Value

State of the macrotick counter when the stop watch event occurred.

(2) **FRSTPW1.SCCV**

Stop Watch Captured Cycle Counter Value

State of the cycle counter when the stop watch event occurred.

(3) **FRSTPW1.EINT1**

Enable FlexRay Interrupt 1 Trigger

Enables stop watch trigger by FlexRay interrupt 1 event if **FRSTPW1.ESWT** = ’1’.

(4) **FRSTPW1.EINT0**

Enable FlexRay Interrupt 0 Trigger

Enables stop watch trigger by FlexRay interrupt 0 event if **FRSTPW1.ESWT** = ’1’.

(5) **Reserved**

(6) **FRSTPW1.SSWT**

Software Stop Watch Trigger

Bits **FRSTPW1.ESWT** and **FRSTPW1.SSWT** cannot be set to ’1’ simultaneously. In this case the write access to the register is ignored, and both bits keep their previous values. Either the external stop watch trigger or the software stop watch trigger may be used.

When the Host writes this bit to ’1’ the stop watch is activated. After the actual cycle counter, slot counter and macrotick value are stored in the FlexRay Stop Watch registers and this bit is reset to ’0’.

(7) **FRSTPW1.EDGE**

Stop Watch Trigger Edge Select

(8) **FRSTPW1.SWMS**

Stop Watch Mode Select

(9) **FRSTPW1.ESWT**

Enable Stop Watch Trigger

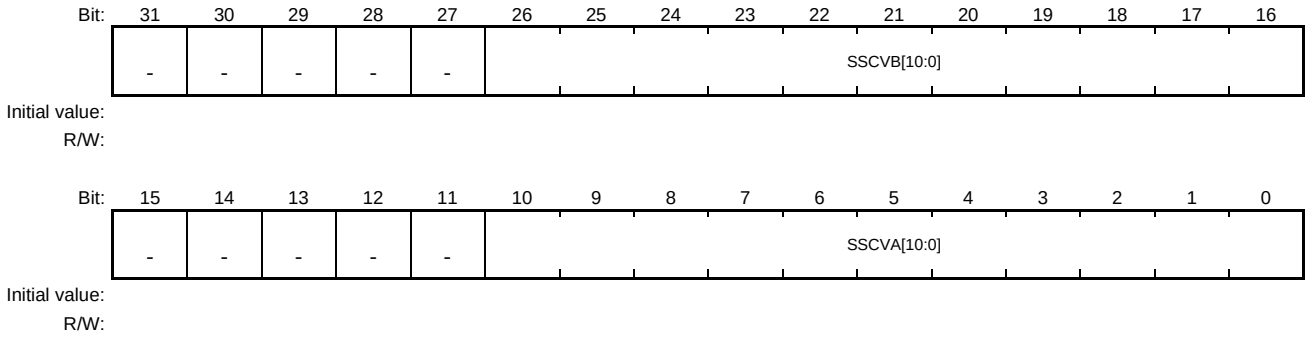
Bits **FRSTPW1.ESWT** and **FRSTPW1.SSWT** cannot be set to ’1’ simultaneously. In this case the write access to the register is ignored, and both bits keep their previous values. Either the external stop watch trigger or the software stop watch trigger may be used.

If enabled an edge on input STPWT_FLEXRAY or a FlexRay interrupt 0 event or a FlexRay interrupt 1 event activates the stop watch.

In single-shot mode this bit is reset to ’0’ after the actual cycle counter, slot counter and macrotick value are stored in the FlexRay Stop Watch registers.

### 103.2.4.5 FRSTPW2 [h'0050]

FlexRay Stop Watch Register 2




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26:16] | SSCVB | 0 | R/W | Stop Watch Captured Slot Counter Value Channel B |
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | SSCVA | 0 | R/W | Stop Watch Captured Slot Counter Value Channel A |

(1) **FRSTPW2.SSCVB**

Stop Watch Captured Slot Counter Value Channel B

State of the slot counter for channel B when the stop watch event occurred.

(2) **FRSTPW2.SSCVA**

Stop Watch Captured Slot Counter Value Channel A

State of the slot counter for channel A when the stop watch event occurred.

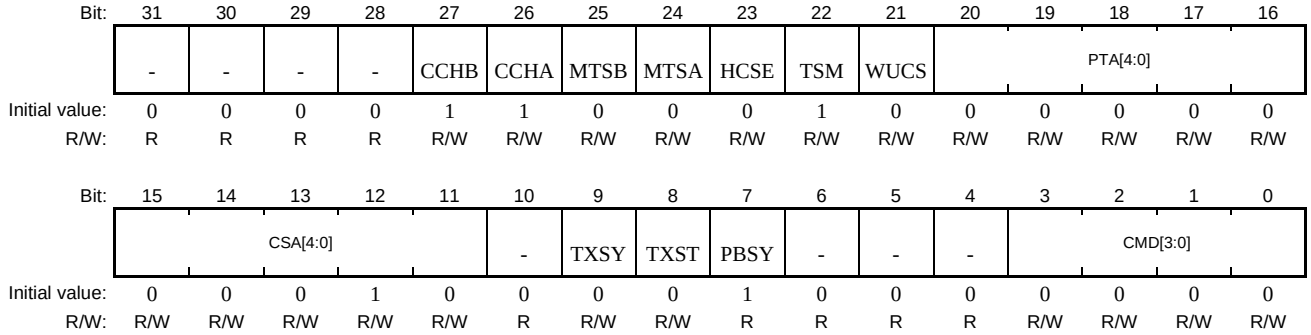
## 103.2.5 CC Control Registers

This section describes the registers provided by the CC to allow the Host to control the operation of the CC. The FlexRay protocol specification requires the Host to write application configuration data in CONFIG state only. Please consider that the configuration registers are not locked for writing in DEFAULT_CONFIG state.

The configuration data is reset when DEFAULT_CONFIG state is entered from reset. To change POC state from DEFAULT_CONFIG to CONFIG state the Host has to apply CHI command CONFIG. If the Host wants the CC to leave CONFIG state, the Host has to proceed an unlock sequence as described in section 103.2.2.1.

### 103.2.5.1 FRSUCC1 [h'0080]

FlexRay SUC Configuration Register 1




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:28] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [27] | CCHB | 1 | R/W | Connected to Channel B<br>Configures pChannels<br>0 = Not connected to channel B<br>1 = Node connected to channel B |
| [26] | CCHA | 1 | R/W | Connected to Channel A<br>Configures pChannels<br>0 = Not connected to channel A<br>1 = Node connected to channel A |
| [25] | MTSB | 0 | R/W | Select Channel B for MTS Transmission<br>0 = Channel B disabled for MTS transmission<br>1 = Channel B selected for MTS transmission |
| [24] | MTSA | 0 | R/W | Select Channel A for MTS Transmission<br>0 = Channel A disabled for MTS transmission<br>1 = Channel A selected for MTS transmission |
| [23] | HCSE | 0 | R/W | Halt due to Clock Sync Error<br>Configures pAllowHaltDueToClock<br>0 = CC will enter / remain in NORMAL_PASSIVE<br>1 = CC will enter HALT state |
| [22] | TSM | 1 | R/W | Transmission Slot Mode<br>Configures pSingleSlotEnabled<br>0 = ALL Slot Mode<br>1 = SINGLE Slot Mode |
| [21] | WUCS | 0 | R/W | Wakeup Channel Select<br>Configures pWakeupChannel<br>0 = Send wakeup pattern on channel A<br>1 = Send wakeup pattern on channel B |
| [20:16] | PTA | 0 | R/W | Passive to Active<br>Configures pAllowPassiveToActive |
| [15:11] | CSA | 2 | R/W | Cold Start Attempts<br>Configures gColdStartAttempts |
| [10] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [9] | TXSY | 0 | R/W | Transmit Sync Frame in Key Slot<br>Configures pKeySlotUsedForSync<br>0 = No sync frame transmission in key slot, node is neither sync nor coldstart node<br>1 = Key slot used to transmit sync frame, node is sync node |
| [8] | TXST | 0 | R/W | Transmit Startup Frame in Key Slot<br>Configures pKeySlotUsedForStartup<br>0 = No startup frame transmission in key slot, node is non-coldstarter<br>1 = Key slot used to transmit startup frame, node is leading or following coldstarter |
| [7] | PBSY | 1 | R | POC Busy<br>0 = POC not busy, FRSUCC1.CMD writeable<br>1 = POC is busy, FRSUCC1.CMD locked |
| [6:4] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [3:0] | CMD | 0 | R/W | CHI Command Vector<br>0000 = command_not_accepted<br>0001 = CONFIG<br>0010 = READY<br>0011 = WAKEUP<br>0100 = RUN<br>0101 = ALL_SLOTS<br>0110 = HALT<br>0111 = FREEZE<br>1000 = SEND_MTS<br>1001 = ALLOW_COLDSTART<br>1010 = RESET_STATUS_INDICATORS<br>1100 = CLEAR_RAMS<br>others = reserved |

(1) **FRSUCC1.CCHB**

Connected to Channel B

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Configures whether the node is connected to channel B (pChannels).

(2) **FRSUCC1.CCHA**

Connected to Channel A

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Configures whether the node is connected to channel A (pChannels).

(3) **FRSUCC1.MTSB**

Select Channel B for MTS Transmission

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

**FRSUCC1.MTSB** may also be changed outside DEFAULT_CONFIG or CONFIG state when the write to **FRSUCC1** register is directly preceded by the unlock sequence for the Configuration Lock Key as described in section 103.2.2.1. This may be combined with CHI command SEND_MTS.

If both bits **FRSUCC1.MTSA** and **FRSUCC1.MTSB** are set to '1' an MTS symbol will be transmitted on both channels when requested by writing **FRSUCC1.CMD** = "1000".

The bit selects channel B for MTS symbol transmission.

(4) **FRSUCC1.MTSA**

Select Channel A for MTS Transmission

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

**FRSUCC1.MTSA** may also be changed outside DEFAULT_CONFIG or CONFIG state when the write to **FRSUCC1** register is directly preceded by the unlock sequence for the Configuration Lock Key as described in section 103.2.2.1. This may be combined with CHI command SEND_MTS.

If both bits **FRSUCC1.MTSA** and **FRSUCC1.MTSB** are set to '1' an MTS symbol will be transmitted on both channels when requested by writing **FRSUCC1.CMD** = "1000".

The bit selects channel A for MTS symbol transmission.

(5) **FRSUCC1.HCSE**

Halt due to Clock Sync Error

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Controls the transition to HALT state due to a clock synchronization error (pAllowHaltDueToClock).

(6) **FRSUCC1.TSM**

Transmission Slot Mode

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Selects the initial transmission slot mode (pSingleSlotEnabled).

In SINGLE slot mode the CC may only transmit in the preconfigured key slot. The key slot ID is configured in the header section of message buffer 0 respectively message buffers 0 and 1 depending on bit **FRMRC.SPLM**.

In case **FRSUCC1.TSM** = '1', message buffer 0 respectively message buffers 0,1 can be (re)configured in DEFAULT_CONFIG or CONFIG state only. In ALL slot mode the CC may transmit in all slots.

**FRSUCC1.TSM** is a configuration bit which can only be set / reset by the Host.

The CC changes to ALL slot mode when the Host successfully applied the ALL_SLOTS command by writing **FRSUCC1.CMD** = "0101" in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. The actual slot mode is monitored by **FRCCSV.SLM**.

(7) **FRSUCC1.WUCS**

Wakeup Channel Select

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

With this bit the Host selects the channel on which the CC sends the Wakeup pattern (pWakeupChannel).

(8) **FRSUCC1.PTA**

Passive to Active

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 31 even / odd cycle pairs.

Defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the CC is allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state (pAllowPassiveToActive).

If set to "00000" the CC is not allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state.

(9) **FRSUCC1.CSA**

Cold Start Attempts

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Must be identical in all nodes of a cluster.

Valid values are 2 to 31.

Configures the maximum number of attempts that a cold starting node is permitted to try to start up the network without receiving any valid response from another node (gColdStartAttempts).

(10) **FRSUCC1.TXSY**

Transmit Sync Frame in Key Slot

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Defines whether the key slot is used to transmit sync frames (pKeySlotUsedForSync).

(11) **FRSUCC1.TXST**

Transmit Startup Frame in Key Slot

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Defines whether the key slot is used to transmit startup frames (pKeySlotUsedForStartup).

**Note:** The protocol requires that both bits **FRSUCC1.TXST** and **FRSUCC1.TXSY** are set for coldstart nodes.

(12) **FRSUCC1.PBSY**

POC Busy

Signals that the POC is busy and cannot accept a command from the Host. **FRSUCC1.CMD** is locked against write accesses.

Set to ’1’ after reset during initialization of internal RAM blocks.

(13) **FRSUCC1.CMD**

CHI Command Vector

The Host may write any CHI command at any time, but certain commands are enabled only in certain POC states. If a command is not enabled, it will not be executed, the CHI command vector **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted, and flag **FREIR.CNA** will be set to '1'.

In general the Host must check **FRSUCC1.PBSY** before writing a new CHI command.

In case the previous CHI command has not yet completed, **FREIR.CCL** is set to '1' together with **FREIR.CNA**; the CHI command needs to be repeated.

Except for HALT state, a POC state change command applied while the CC is already in the requested POC state neither causes a state change nor will **FREIR.CNA** be set.

Reading **FRSUCC1.CMD** shows whether the last CHI command was accepted. The actual POC state is monitored by **FRCCSV.POCS**.

**command_not_accepted**
**FRSUCC1.CMD** is reset to "0000" due to one of the following conditions:
Illegal command applied by the Host
Host applied command to leave CONFIG state without preceding config lock key
Host applied new command while execution of the previous Host command has not completed
Host writes command_not_accepted
When **FRSUCC1.CMD** is reset to "0000", **FREIR.CNA** is set, and, if enabled, an interrupt is generated.

Commands which are not accepted are not executed.

**CONFIG**
Go to POC state CONFIG when called in POC states DEFAULT_CONFIG, or READY. When called in HALT state the CC transits to POC state DEFAULT_CONFIG. When called in any other state, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**READY**
Go to POC state READY when called in POC states CONFIG, NORMAL_ACTIVE, NORMAL_PASSIVE, STARTUP, or WAKEUP. When called in any other state, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**WAKEUP**
Go to POC state WAKEUP when called in POC state READY. When called in any other state, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**RUN**
Go to POC state STARTUP when called in POC state READY. When called in any other state, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**ALL_SLOTS**
Leave SINGLE slot mode after successful startup / integration at the next end of cycle when called in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. When called in any other state, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**HALT**
Set halt request **FRCCSV.HRQ** and go to POC state HALT at the next end of cycle when called in POC states NORMAL_ACTIVE or NORMAL_PASSIVE. When called in any other state, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**FREEZE**
Set the freeze status indicator **FRCCSV.FSI** and go to POC state HALT immediately. Can be called from any state.

**SEND_MTS**

Send single MTS symbol during the next following symbol window on the channel configured by **FRSUCC1.MTSA**, **FRSUCC1.MTSB**, when called in POC state NORMAL_ACTIVE after CC entered ALL slot mode (**FRCCSV.SLM** = "11"). When called in any other state, or when called while a previously requested MTS has not yet been transmitted, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**ALLOW_COLDSTART**
The command resets **FRCCSV.CSI** to enable the node to become leading coldstarter. When called in states DEFAULT_CONFIG, CONFIG, or HALT, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted. To become leading coldstarter it is also required that both **FRSUCC1.TXST** and **FRSUCC1.TXSY** are set.

**RESET_STATUS_INDICATORS**
Resets status flags **FRCCSV.CSNI**, **FRCCSV.CSAI**, and **FRCCSV.WSV** to their default values. May be called in POC states READY and STARTUP. When called in any other state, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**CLEAR_RAMS**
Sets **FRMHDS.CRAM** when called in DEFAULT_CONFIG or CONFIG state. When called in any other state, **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted.

**FRMHDS.CRAM** is also set when the CC leaves reset. By setting **FRMHDS.CRAM** all internal RAM blocks are initialized to zero. During the initialization of the RAMs, **FRSUCC1.PBSY** will show POC busy. Access to the configuration and status registers is possible during execution of CHI command CLEAR_RAMS.

The initialization of the internal message RAM requires 2048 bus clock cycles. There should be no Host access to IBF or OBF during initialization of the internal RAM blocks after reset or after assertion of CHI command CLEAR_RAMS.

Before asserting CHI command CLEAR_RAMS the Host should make sure that no transfer between Message RAM and IBF / OBF or the Transient Buffer RAMs is ongoing and the data transfer handlers are disabled (**FRITS.ITS** = '0' and **FROTS.OTS** = '0'). This command also resets the Message Buffer Status registers **FRMHDS**, **FRLDTS**, **FRFSR**, **FRMHDF**, **FRTXRQ1/2/3/4.TXR**, **FRNDAT1/2/3/4**, and **FRMBSC1/2/3/4**.

**Note:** All accepted commands with exception of CLEAR_RAMS and SEND_MTS will cause a change of the POC state in the FlexRay sample clock domain after at most 8 cycles of the slower of the two clocks 'bus clock' and 'FlexRay sample clock', assumed that POC was not busy when the command was applied and that no POC state change was forced by bus activity in that time frame. Reading register **FRCCSV** will show data that is additionally delayed by synchronization from the FlexRay sample clock domain to the bus clock domain. The maximum additional delay is 12 cycles of the slower of the two clocks 'bus clock' and 'FlexRay sample clock'.

Table 103.2a below references the CHI commands from the FlexRay Protocol Specification [FR_PROT] (section 2.2.1.1, Table 2-2) to the FlexRay CHI command vector **FRSUCC1.CMD**.

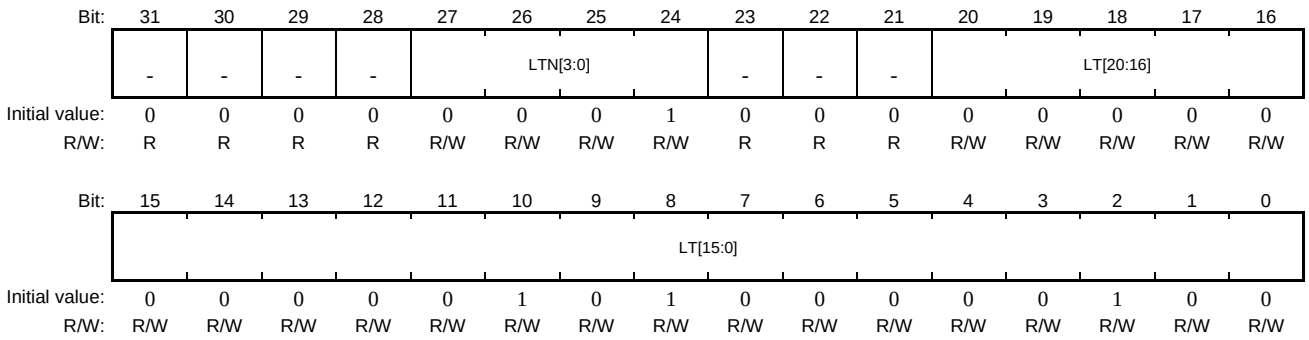
**Table 103.2a Reference to CHI Host command summary from FlexRay protocol specification**

| CHI command | Where processed (POC States) | CHI Command Vector CMD |
|---|---|---|
| ALL_SLOTS | POC:normal active, POC:normal passive | ALL_SLOTS |
| ALLOW_COLDSTART | All except POC:default config, POC:config, POC:halt | ALLOW_COLDSTART |
| CONFIG | POC:default config, POC:ready | CONFIG |
| CONFIG_COMPLETE | POC:config | Unlock sequence & READY |
| DEFAULT_CONFIG | POC:halt | CONFIG |
| FREEZE | All | FREEZE |
| HALT | POC:normal active, POC:normal passive | HALT |
| READY | All except POC:default config, POC:config, | READY |

| | | |
|---|---|---|
| | POC:ready, POC:halt | |
| RUN | POC:ready | RUN |
| WAKEUP | POC:ready | WAKEUP |

### 103.2.5.2 FRSUCC2 [h'0084]

FlexRay SUC Configuration Register 2




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:28] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [27:24] | LTN | 1 | R/W | Listen Timeout Noise<br>Configures (gListenNoise - 1) |
| [23:21] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [20:0] | LT | 504h | R/W | Listen Timeout<br>Configures pdListenTimeout |

(1) **FRSUCC2.LTN**

Listen Timeout Noise

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

The range for gListenNoise is 2 to 16.

**FRSUCC2.LTN** must be configured identical in all nodes of a cluster.

Configures the upper limit for startup and wakeup listen timeout in the presence of noise expressed as a multiple of pdListenTimeout.

**Note:** The wakeup / startup noise timeout is calculated as follows:

pdListenTimeout * gListenNoise = **FRSUCC2.LT** * (**FRSUCC2.LTN** + 1)

(2) **FRSUCC2.LT**

Listen Timeout

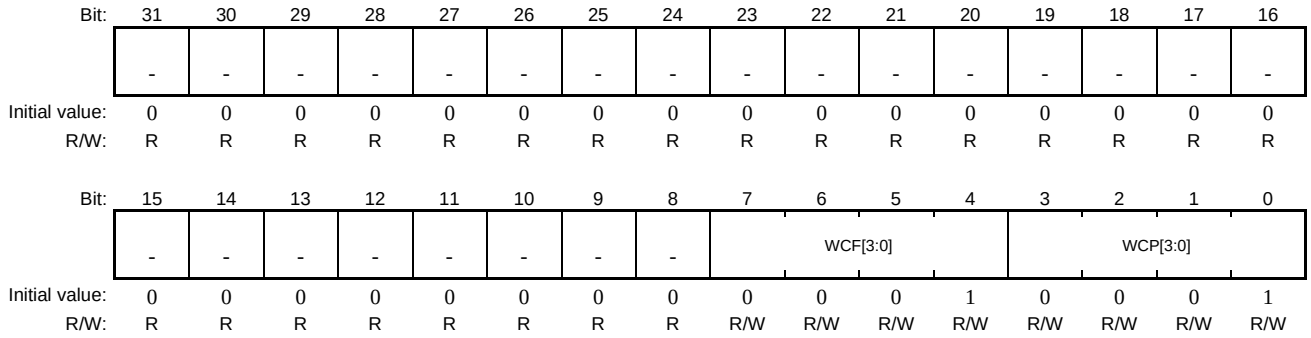
The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

The range for pdListenTimeout is 1284 to 1283846 µT.

Configures wakeup / startup listen timeout in µT.

### 103.2.5.3 FRSUCC3 [h'0088]

FlexRay SUC Configuration Register 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:8] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [7:4] | WCF | 1 | R/W | Maximum Without Clock Correction Fatal<br>Configures gMaxWithoutClockCorrectionFatal |
| [3:0] | WCP | 1 | R/W | Maximum Without Clock Correction Passive<br>Configures gMaxWithoutClockCorrectionPassive |

(1) **FRSUCC3.WCF**

Maximum Without Clock Correction Fatal

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 1 to 15 cycle pairs.

**FRSUCC3.WCF** must be identical in all nodes of a cluster.

Defines the number of consecutive even / odd cycle pairs with missing clock correction terms that will cause a transition from NORMAL_ACTIVE or NORMAL_PASSIVE to HALT state.

**Note:** The transition to HALT state is prevented if **FRSUCC1.HCSE** is not set.

(2) **FRSUCC3.WCP**

Maximum Without Clock Correction Passive

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 1 to 15 cycle pairs.

**FRSUCC3.WCP** must be identical in all nodes of a cluster.

Defines the number of consecutive even / odd cycle pairs with missing clock correction terms that will cause a transition from NORMAL_ACTIVE to NORMAL_PASSIVE state.

### 103.2.5.4 FRNEMC [h'008C]

FlexRay NEM Configuration Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | NML[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:4] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [3:0] | NML | 0 | R/W | Network Management Vector Length<br>Configures gNetworkManagementVectorLength |

(1) **FRNEMC.NML**

Network Management Vector Length

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

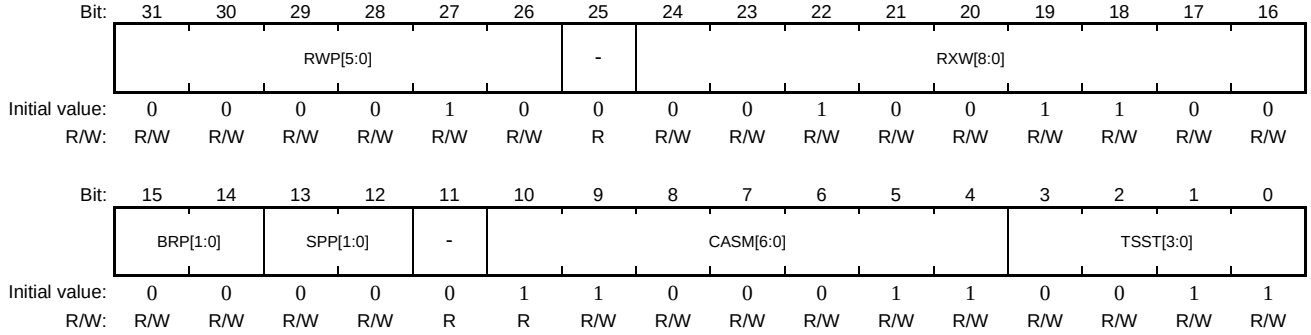
Valid values are 0 to 12 bytes.

**FRNEMC.NML** must be identical in all nodes of a cluster.

These bits configure the length of the NM vector.

### 103.2.5.5 FRPRTC1 [h'0090]

FlexRay PRT Configuration Register 1




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:26] | RWP | 2h | R/W | Repetitions of Tx Wakeup Pattern<br>Configures pWakeupPattern |
| [25] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [24:16] | RXW | 4Ch | R/W | Wakeup Symbol Receive Window Length<br>Configures gdWakeupSymbolRxWindow |
| [15:14] | BRP | 0 | R/W | Baud Rate Prescaler<br>Configures gdSampleClockPeriod and pSamplesPerMicrotick<br>00 = 10 MBit/s<br>01 = 5 MBit/s<br>10 = 2.5 MBit/s<br>11 = 2.5 MBit/s |
| [13:12] | SPP | 0 | R/W | Strobe Point Position<br>Configures Strobe point position<br>00 = Sample 5<br>01 = Sample 4<br>10 = Sample 6<br>11 = Sample 5 |
| [11] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [10:4] | CASM | 63h | R/W | Collision Avoidance Symbol Max<br>Configures gdCASRxLowMax |
| [3:0] | TSST | 3h | R/W | Transmission Start Sequence Transmitter<br>Configures gdTSSTransmitter |

Notes: Bit[10]=CASM[6] is fixed to 1, therefore bit[10] is "Read Only" bit: user cannot write this bit.

(1) **FRPRTC1.RWP**

Repetitions of Tx Wakeup Pattern

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 63.

Configures the number of repetitions (sequences) of the Tx wakeup symbol.

(2) **FRPRTC1.RXW**

Wakeup Symbol Receive Window Length

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 76 to 301 bit times.

**FRPRTC1.RXW** must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to test the duration of the received wakeup pattern.

(3) **FRPRTC1.BRP**

Baud Rate Prescaler

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

The Baud Rate Prescaler configures the baud rate on the FlexRay bus. The baud rates listed below are valid with a sample clock set to 80 MHz. One bit time always consists of 8 samples independent of the configured baud rate.

00 = 10 MBit/s
gdSampleClockPeriod = 12.5 ns = 1 * 'sample clock'
pSamplesPerMicrotick = 2 (1 µT = 25 ns)

01 = 5 MBit/s
gdSampleClockPeriod = 25 ns = 2 * 'sample clock'
pSamplesPerMicrotick = 1 (1 µT = 25 ns)

10, 11 = 2.5 MBit/s
gdSampleClockPeriod = 50 ns = 4 * 'sample clock'
pSamplesPerMicrotick = 1 (1 µT = 50 ns)

(4) **FRPRTC1.SPP**

Strobe Point Position

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Defines the sample count value for strobing. The strobed bit value is set to the voted value when the sample count is incremented to the value configured by **FRPRTC1.SPP**.

**Note:** The current revision 2.1 of the FlexRay protocol requires that **FRPRTC1.SPP** = "00". The alternate strobe point positions could be used to compensate for asymmetries in the physical layer.

(5) **FRPRTC1.CASM**

Collision Avoidance Symbol Max

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

CASM6 is fixed to ’1’.

Valid values are 67 to 99 bit times.

Configures the upper limit of the acceptance window for a collision avoidance symbol (CAS).

(6) **FRPRTC1.TSST**

Transmission Start Sequence Transmitter

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

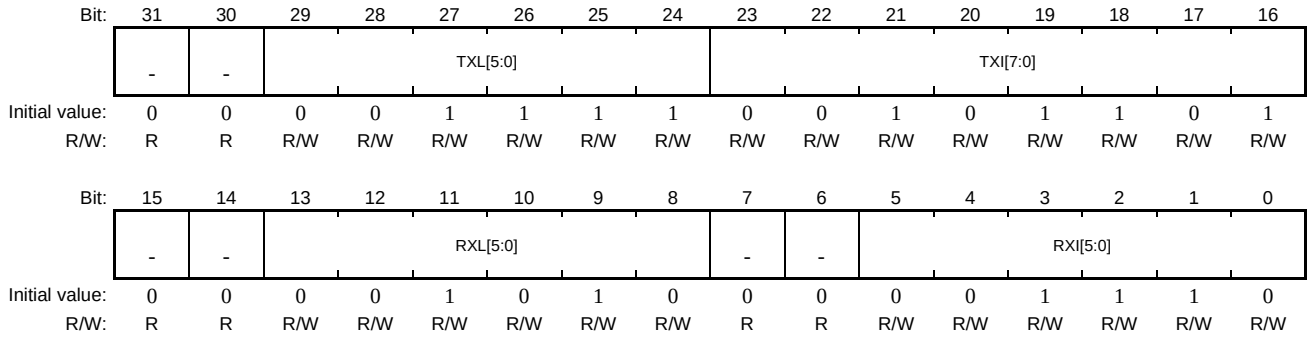
Valid values are 3 to 15 bit times.

**FRPRTC1.TSST** must be identical in all nodes of a cluster.

Configures the duration of the Transmission Start Sequence (TSS) in terms of bit times (1 bit time = 4 µT = 100ns @ 10Mbps).

### 103.2.5.6 FRPRTC2 [h'0094]

FlexRay PRT Configuration Register 2




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29:24] | TXL | Fh | R/W | Wakeup Symbol Transmit Low<br>Configures gdWakeupSymbolTxLow |
| [23:16] | TXI | 2Dh | R/W | Wakeup Symbol Transmit Idle<br>Configures gdWakeupSymbolTxIdle |
| [15:14] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [13:8] | RXL | Ah | R/W | Wakeup Symbol Receive Low<br>Configures gdWakeupSymbolRxLow |
| [7:6] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [5:0] | RXI | Eh | R/W | Wakeup Symbol Rx Idle<br>Configures gdWakeupSymbolRxIdle |

(1) **FRPRTC2.TXL**

Wakeup Symbol Transmit Low

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 15 to 60 bit times.

**FRPRTC2.TXL** must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to transmit the low phase of the wakeup symbol.

(2) **FRPRTC2.TXI**

Wakeup Symbol Transmit Idle

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 45 to 180 bit times.

**FRPRTC2.TXI** must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to transmit the idle phase of the wakeup symbol.

(3) **FRPRTC2.RXL**

Wakeup Symbol Receive Low

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 10 to 55 bit times.

**FRPRTC2.RXL** must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to test the duration of the low phase of the received wakeup symbol.

(4) **FRPRTC2.RXI**

Wakeup Symbol Rx Idle

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

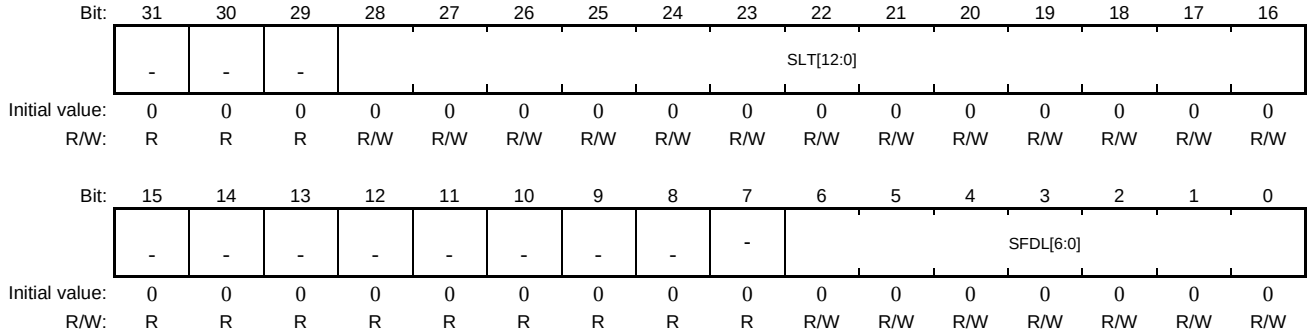
Valid values are 14 to 59 bit times.

**FRPRTC2.RXI** must be identical in all nodes of a cluster.

Configures the number of bit times used by the node to test the duration of the idle phase of the received wakeup symbol.

### 103.2.5.7 FRMHDC [h'0098]

FlexRay MHD Configuration Register




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:29] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [28:16] | SLT | 0 | R/W | Start of Latest Transmit<br>Configures pLatestTx |
| [15:7] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [6:0] | SFDL | 0 | R/W | Static Frame Data Length<br>Configures gPayloadLengthStatic |

(1) **FRMHDC.SLT**

Start of Latest Transmit

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 7981 minislots.

Configures the maximum minislot value allowed before inhibiting frame transmission in the dynamic segment of the cycle. There is no transmission in dynamic segment if **FRMHDC.SLT** is set to zero.

(2) **FRMHDC.SFDL**

Static Frame Data Length

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

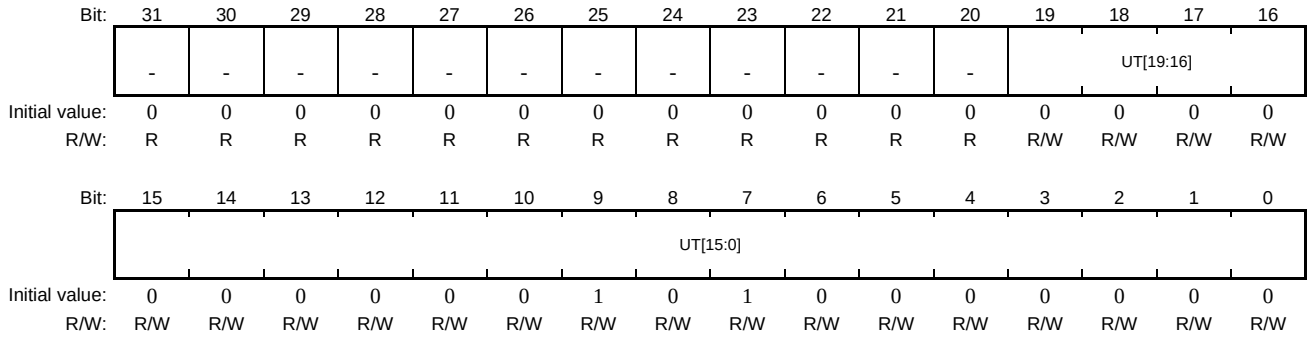
Valid values are 0 to 127.

**FRMHDC.SLT** must be identical in all nodes of a cluster.

Configures the cluster-wide payload length for all frames sent in the static segment in double bytes.

### 103.2.5.8 FRGTUC1 [h'00A0]

FlexRay GTU Configuration Register 1




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:20] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [19:0] | UT | 280h | R/W | Microtick per Cycle<br>Configures pMicroPerCycle |

(1) **FRGTUC1.UT**

Microtick per Cycle

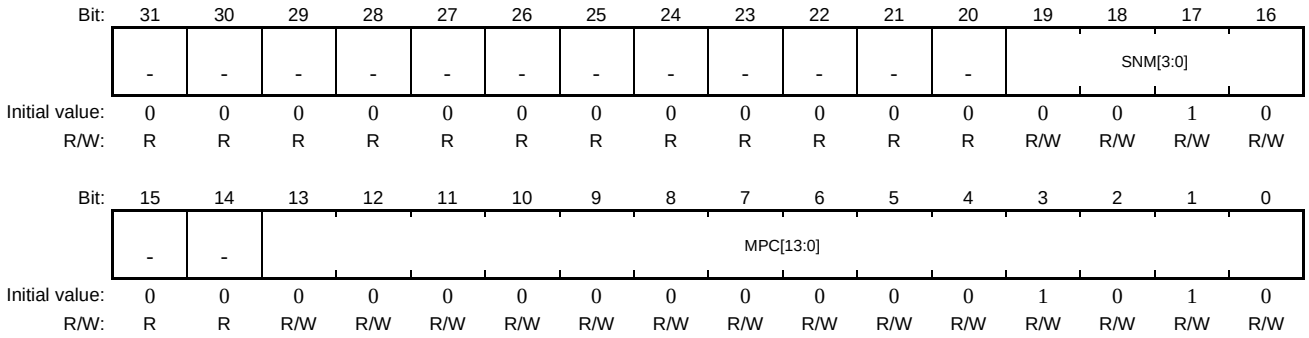
The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 640 to 640000 µT.

Configures the duration of the communication cycle in microticks.

### 103.2.5.9 FRGTUC2 [h'00A4]

FlexRay GTU Configuration Register 2




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:20] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [19:16] | SNM | 2h | R/W | Sync Node Max<br>Sync Node Max |
| [15:14] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [13:0] | MPC | Ah | R/W | Macrotick Per Cycle<br>Configures gMacroPerCycle |

(1) **FRGTUC2.SNM**

Sync Node Max

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 15.

**FRGTUC2.SNM** must be identical in all nodes of a cluster.

Maximum number of frames within a cluster with sync frame indicator bit SYN set to '1'.

(2) **FRGTUC2.MPC**

Macrotick Per Cycle

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

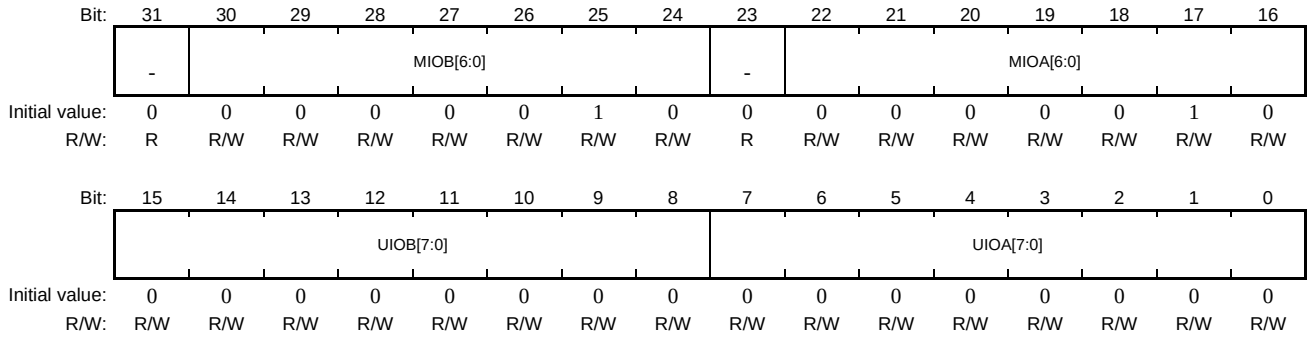
Valid values are 10 to 16000 MT.

**FRGTUC2.MPC** must be identical in all nodes of a cluster.

Configures the duration of one communication cycle in macroticks.

### 103.2.5.10 FRGTUC3 [h'00A8]

FlexRay GTU Configuration Register 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [30:24] | MIOB | 2h | R/W | Macrotick Initial Offset Channel B<br>Configures pMacroInitialOffset[B] |
| [23] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [22:16] | MIOA | 2h | R/W | Macrotick Initial Offset Channel A<br>Configures pMacroInitialOffset[A] |
| [15:8] | UIOB | 0 | R/W | Microtick Initial Offset Channel B<br>Configures pMicroInitialOffset[B] |
| [7:0] | UIOA | 0 | R/W | Microtick Initial Offset Channel A<br>Configures pMicroInitialOffset[A] |

(1) **FRGTUC3.MIOB**

Macrotick Initial Offset Channel B

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 72 MT.

**FRGTUC3.MIOB** must be identical in all nodes of a cluster.

Configures the number of macroticks between the static slot boundary and the subsequent macrotick boundary of the secondary time reference point based on the nominal macrotick duration.

(2) **FRGTUC3.MIOA**

Macrotick Initial Offset Channel A

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 72 MT.

**FRGTUC3.MIOA** must be identical in all nodes of a cluster.

Configures the number of macroticks between the static slot boundary and the subsequent macrotick boundary of the secondary time reference point based on the nominal macrotick duration.

(3) **FRGTUC3.UIOB**

Microtick Initial Offset Channel B

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 240 µT.

Configures the number of microticks between the actual time reference point on channel B and the subsequent macrotick boundary of the secondary time reference point. The parameter depends on pDelayCompensation[B] and therefore has to be set for each channel independently.

(4) **FRGTUC3.UIOA**

Microtick Initial Offset Channel A

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

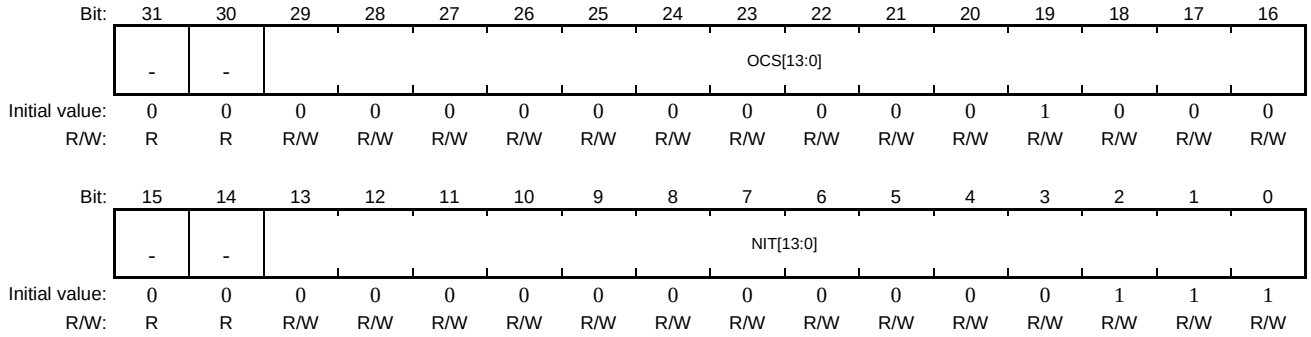
Valid values are 0 to 240 µT.

Configures the number of microticks between the actual time reference point on channel A and the subsequent macrotick boundary of the secondary time reference point. The parameter depends on pDelayCompensation[A] and therefore has to be set for each channel independently.

### 103.2.5.11 FRGTUC4 [h'00AC]

FlexRay GTU Configuration Register 4

For details about configuration of **FRGTUC4.NIT** and **FRGTUC4.OCS** see section 103.3.2.5.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29:16] | OCS | 8h | R/W | Offset Correction Start<br>Configures (gOffsetCorrectionStart - 1) |
| [15:14] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [13:0] | NIT | 7h | R/W | Network Idle Time Start<br>Configures (gMacroPerCycle - gdNIT - 1) |

(1) **FRGTUC4.OCS**

Offset Correction Start

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 8 to 15998 MT.

For cluster consisting of ERAY implementations only, it is sufficient to program **FRGTUC4.OCS** = **FRGTUC4.NIT** + 1.

**FRGTUC4.OCS** must be identical in all nodes of a cluster.

Determines the start of the offset correction within the NIT phase, calculated from start of cycle.

(2) **FRGTUC4.NIT**

Network Idle Time Start

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

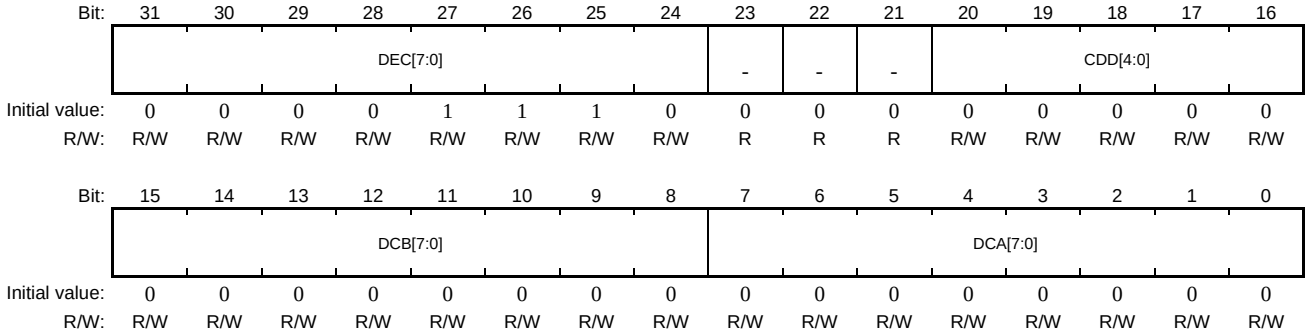
Valid values are 7 to 15997 MT.

**FRGTUC4.NIT** must be identical in all nodes of a cluster.

Configures the starting point of the Network Idle Time (NIT) at the end of the communication cycle expressed in terms of macroticks from the beginning of the cycle.

### 103.2.5.12 FRGTUC5 [h'00B0]

FlexRay GTU Configuration Register 5




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:24] | DEC | Eh | R/W | Decoding Correction<br>Configures pDecodingCorrection |
| [23:21] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [20:16] | CDD | 0 | R/W | Cluster Drift Damping<br>Configures pClusterDriftDamping |
| [15:8] | DCB | 0 | R/W | Delay Compensation Channel B<br>Configures pDelayCompensation[B] |
| [7:0] | DCA | 0 | R/W | Delay Compensation Channel A<br>Configures pDelayCompensation[A] |

(1) **FRGTUC5.DEC**

Decoding Correction

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 14 to 143 µT.

Configures the decoding correction value used to determine the primary time reference point.

(2) **FRGTUC5.CDD**

Cluster Drift Damping

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 20 µT.

Configures the cluster drift damping value used in clock synchronization to minimize accumulation of rounding errors.

(3) **FRGTUC5.DCB**

Delay Compensation Channel B

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 200 µT.

Used to compensate for reception delays on the indicated channel. This covers assumed propagation delay up to cPropagationDelayMax for microticks in the range of 0.0125 to 0.05µs. In practice, the minimum of the propagation delays of all sync nodes should be applied.

(4) **FRGTUC5.DCA**

Delay Compensation Channel A

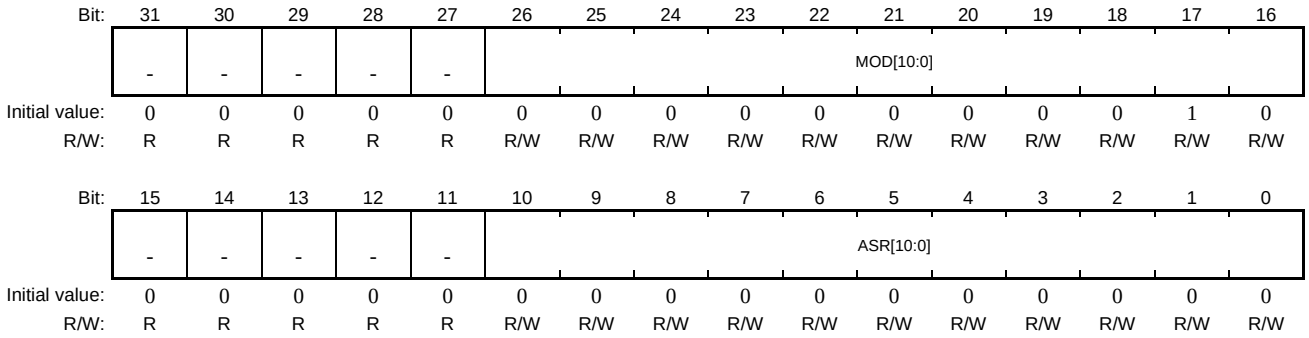
The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 200 µT.

Used to compensate for reception delays on the indicated channel. This covers assumed propagation delay up to cPropagationDelayMax for microticks in the range of 0.0125 to 0.05µs. In practice, the minimum of the propagation delays of all sync nodes should be applied.

### 103.2.5.13 FRGTUC6 [h'00B4]

FlexRay GTU Configuration Register 6




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26:16] | MOD | 2h | R/W | Maximum Oscillator Drift<br>Configures pdMaxDrift |
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | ASR | 0 | R/W | Accepted Startup Range<br>Configures pdAcceptedStartupRange |

(1) **FRGTUC6.MOD**

Maximum Oscillator Drift

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 1923 µT.

Maximum drift offset between two nodes that operate with unsynchronized clocks over one communication cycle in µT.

(2) **FRGTUC6.ASR**

Accepted Startup Range

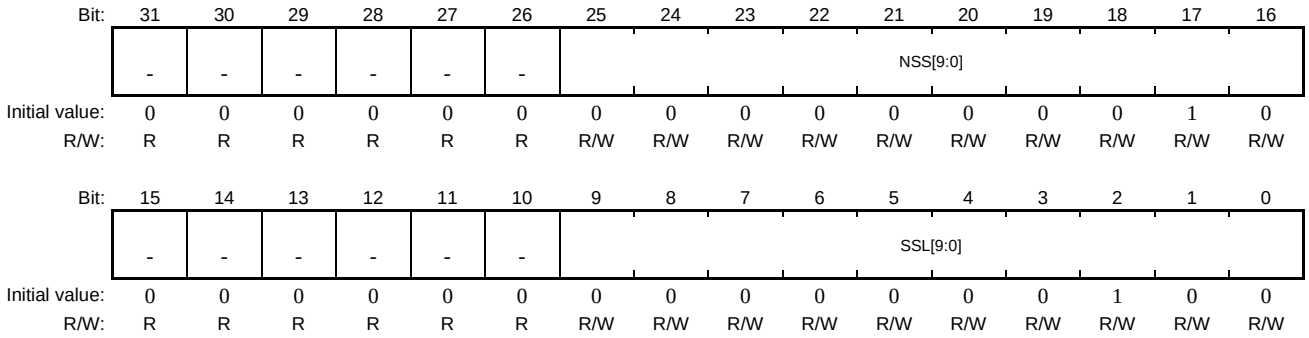
The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 1875 µT.

Number of microticks constituting the expanded range of measured deviation for startup frames during integration.

### 103.2.5.14 FRGTUC7 [h'00B8]

FlexRay GTU Configuration Register 7




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:26] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [25:16] | NSS | 2h | R/W | Number of Static Slots<br>Configures gNumberOfStaticSlots |
| [15:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [9:0] | SSL | 4h | R/W | Static Slot Length<br>Configures gdStaticSlot |

(1) **FRGTUC7.NSS**

Number of Static Slots

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 1023.

**FRGTUC7.NSS** must be identical in all nodes of a cluster.

Configures the number of static slots in a cycle.

(2) **FRGTUC7.SSL**

Static Slot Length

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

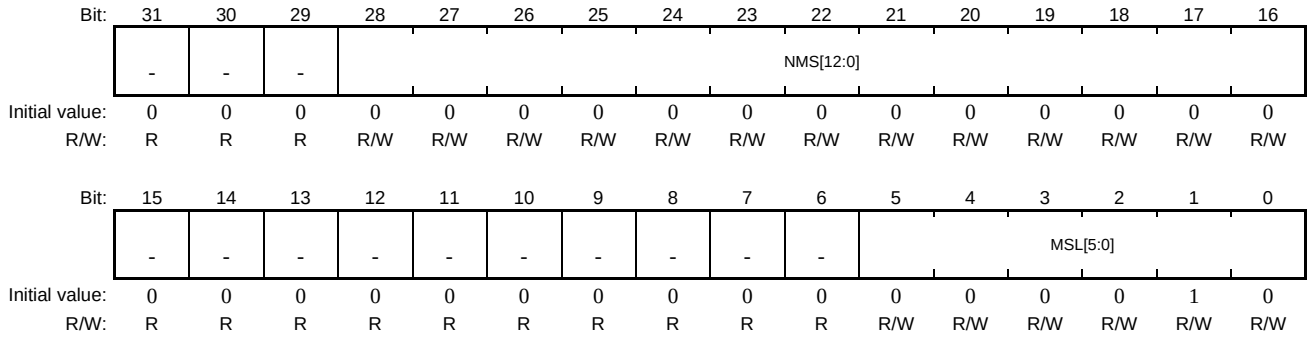
Valid values are 4 to 659 MT.

**FRGTUC7.SSL** must be identical in all nodes of a cluster.

Configures the duration of a static slot in macroticks.

### 103.2.5.15 FRGTUC8 [h'00BC]

FlexRay GTU Configuration Register 8




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:29] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [28:16] | NMS | 0 | R/W | Number of Minislots<br>Configures gNumberOfMinislots |
| [15:6] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [5:0] | MSL | 2h | R/W | Minislot Length<br>Configures gdMinislot |

(1) **FRGTUC8.NMS**

Number of Minislots

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 7986.

**FRGTUC8.NMS** must be identical in all nodes of a cluster.

Configures the number of minislots within the dynamic segment of a cycle.

(2) **FRGTUC8.MSL**

Minislot Length

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

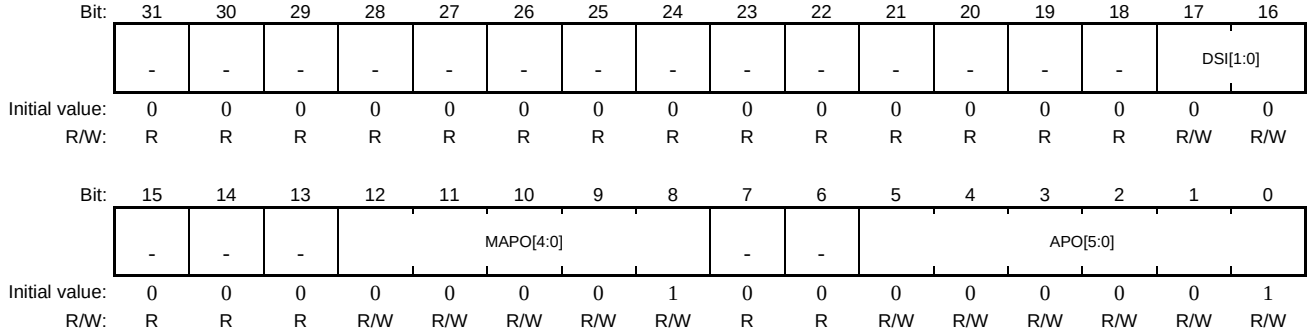
Valid values are 2 to 63 MT.

**FRGTUC8.MSL** must be identical in all nodes of a cluster.

Configures the duration of a minislot in macroticks.

### 103.2.5.16 FRGTUC9 [h'00C0]

FlexRay GTU Configuration Register 9




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:18] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [17:16] | DSI | 0 | R/W | Dynamic Slot Idle Phase<br>Configures gdDynamicSlotIdlePhase |
| [15:13] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [12:8] | MAPO | 1h | R/W | Minislot Action Point Offset<br>Configures gdMinislotActionPointOffset |
| [7:6] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [5:0] | APO | 1h | R/W | Action Point Offset<br>Configures gdActionPointOffset |

(1) **FRGTUC9.DSI**

Dynamic Slot Idle Phase

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 2 Minislot.

**FRGTUC9.DSI** must be identical in all nodes of a cluster.

Configures the duration of the dynamic slot idle phase and has to be greater or equal than the idle detection time.

(2) **FRGTUC9.MAPO**

Minislot Action Point Offset

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 1 to 31 MT.

**FRGTUC9.MAPO** must be identical in all nodes of a cluster.

Configures the action point offset in macroticks within the minislots of the dynamic segment.

(3) **FRGTUC9.APO**

Action Point Offset

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

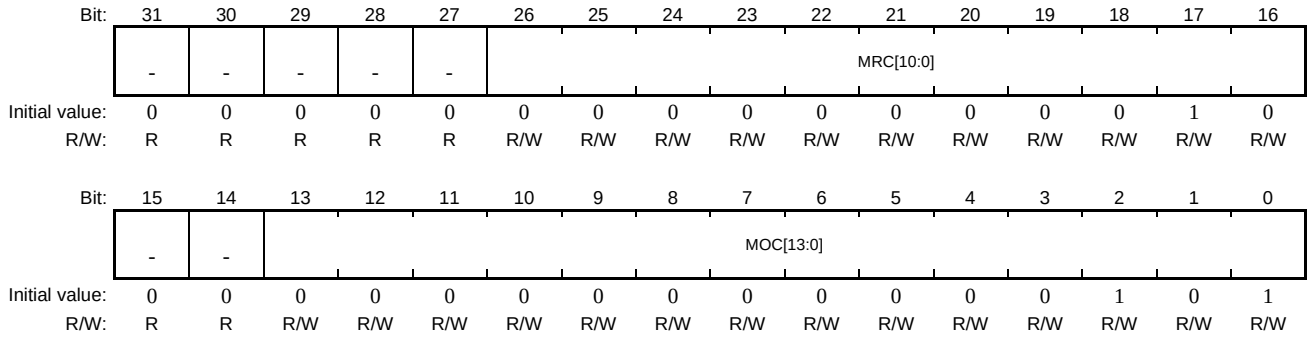
Valid values are 1 to 63 MT.

**FRGTUC9.APO** must be identical in all nodes of a cluster.

Configures the action point offset in macroticks within static slots and symbol window.

### 103.2.5.17 FRGTUC10 [h'00C4]

FlexRay GTU Configuration Register 10




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26:16] | MRC | 2h | R/W | Maximum Rate Correction<br>Configures pRateCorrectionOut |
| [15:14] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [13:0] | MOC | 5h | R/W | Maximum Offset Correction<br>Configures pOffsetCorrectionOut |

(1) **FRGTUC10.MRC**

Maximum Rate Correction

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 2 to 1923 µT.

Holds the maximum permitted rate correction value to be applied by the internal clock synchronization algorithm. The CC checks only the internal rate correction value against the maximum rate correction value (absolute value).

(2) **FRGTUC10.MOC**

Maximum Offset Correction

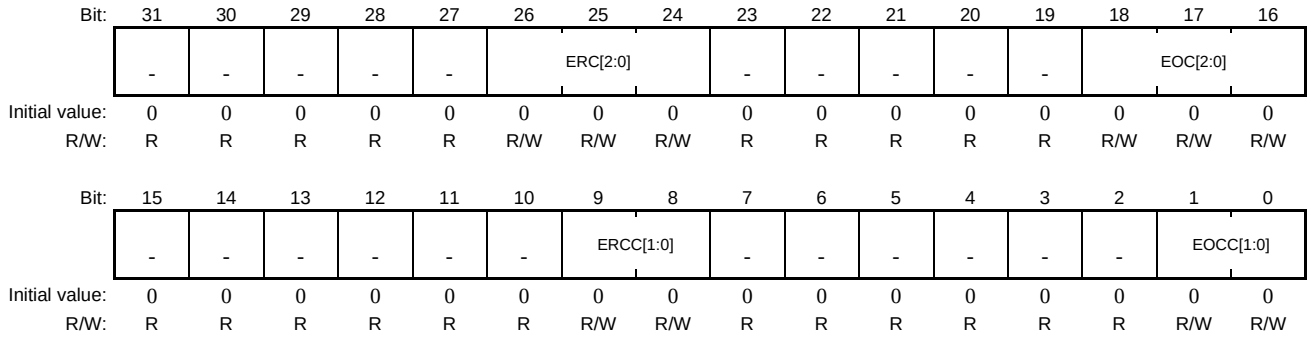
The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 5 to 15266 µT.

Holds the maximum permitted offset correction value to be applied by the internal clock synchronization algorithm (absolute value). The CC checks only the internal offset correction value against the maximum offset correction value.

### 103.2.5.18 FRGTUC11 [h'00C8]

FlexRay GTU Configuration Register 11




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26:24] | ERC | 0 | R/W | External Rate Correction<br>Configures pExternRateCorrection |
| [23:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18:16] | EOC | 0 | R/W | External Offset Correction<br>Configures pExternOffsetCorrection |
| [15:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [9:8] | ERCC | 0 | R/W | External Rate Correction Control<br>Configures vExternRateControl<br>00 = No external rate correction<br>01 = No external rate correction<br>10 = Subtract<br>11 = Add |
| [7:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1:0] | EOCC | 0 | R/W | External Offset Correction Control<br>Configures vExternOffsetControl<br>00 = No external offset correction<br>01 = No external offset correction<br>10 = Subtract<br>11 = Add |

(1) **FRGTUC11.ERC**

External Rate Correction

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 7 µT.

Holds the external rate correction value in microticks to be applied by the internal clock synchronization algorithm. The value is subtracted / added from / to the calculated rate correction value. The value is applied during NIT.

(2) **FRGTUC11.EOC**

External Offset Correction

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Valid values are 0 to 7 µT.

Holds the external offset correction value in microticks to be applied by the internal clock synchronization algorithm. The value is subtracted / added from / to the calculated offset correction value. The value is applied during NIT.

(3) **FRGTUC11.ERCC**

External Rate Correction Control

Should be modified only outside NIT.

By writing to **FRGTUC11.ERCC** the external rate correction is enabled as specified below.

00 = No external rate correction

01 = No external rate correction

10 = Subtract
External rate correction value subtracted from calculated rate correction value

11 = Add
External rate correction value added to calculated rate correction value

(4) **FRGTUC11.EOCC**

External Offset Correction Control

Should be modified only outside NIT.

By writing to **FRGTUC11.EOCC** the external offset correction is enabled as specified below.

00 = No external offset correction

01 = No external offset correction

10 = Subtract
External offset correction value subtracted from calculated offset correction value

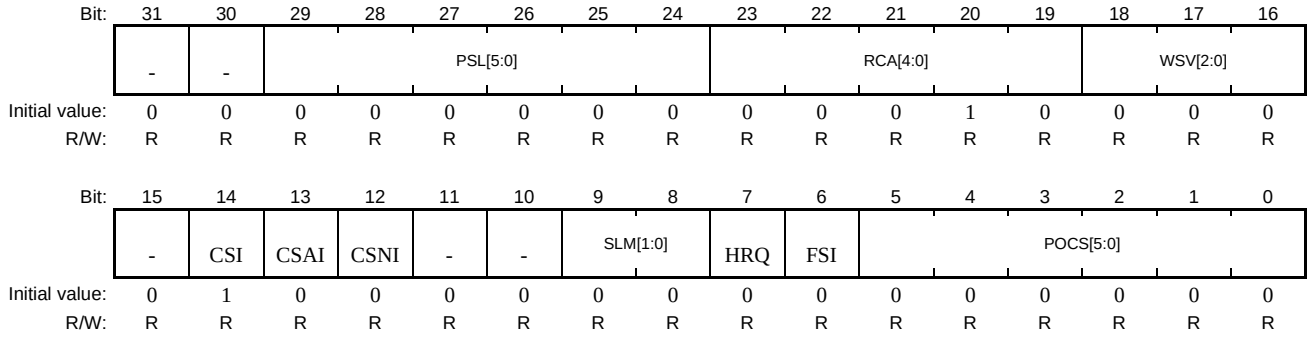
11 = Add
External offset correction value added to calculated offset correction value

### 103.2.6 CC Status Registers

During 8/16-bit accesses to status variables coded with more than 8/16-bit, the variable might be updated by the CC between two accesses (non-atomic read accesses).

#### 103.2.6.1 FRCCSV [h'0100]

FlexRay CC Status Vector




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29:24] | PSL | 0 | R | POC Status Log<br>Status of FRCCSV.POCS immediately before entering HALT state. |
| [23:19] | RCA | 2h | R | Remaining Coldstart Attempts<br>Indicates vRemainingColdstartAttempts |
| [18:16] | WSV | 0 | R | Wakeup Status<br>Indicates vPOC!WakeupStatus |
| [15] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [14] | CSI | 1 | R | Cold Start Inhibit<br>Indicates vColdStartInhibit<br>0 = Cold starting of node enabled<br>1 = Cold starting of node disabled |
| [13] | CSAI | 0 | R | Coldstart Abort Indicator<br>Coldstart Abort Indicator |
| [12] | CSNI | 0 | R | Coldstart Noise Indicator<br>Indicates vPOC!ColdstartNoise |
| [11:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [9:8] | SLM | 0 | R | Slot Mode<br>Indicates vPOC!SlotMode<br>00 = SINGLE<br>01 = reserved<br>10 = ALL_PENDING<br>11 = ALL |
| [7] | HRQ | 0 | R | Halt Request<br>Indicates vPOC!CHIHaltRequest |
| [6] | FSI | 0 | R | Freeze Status Indicator<br>Indicates vPOC!Freeze |
| [5:0] | POCS | 0 | R | Protocol Operation Control Status<br>Protocol Operation Control Status |

(1) **FRCCSV.PSL**

POC Status Log

Status of **FRCCSV.POCS** immediately before entering HALT state.

Set when entering HALT state. Set to HALT when FREEZE command is applied during HALT state and **FRCCSV.FSI** is not already set i.e. the HALT state was not reached by FREEZE command.

Reset to "00 0000" when leaving HALT state.

(2) **FRCCSV.RCA**

Remaining Coldstart Attempts

Indicates the number of remaining coldstart attempts (vRemainingColdstartAttempts).

The initial value of **FRCCSV.RCA** during CONFIG and DEFAULT_CONFIG state is also **FRSUCC1.CSA**.

The RUN command resets this counter to the maximum number of coldstart attempts as configured by **FRSUCC1.CSA**.

(3) **FRCCSV.WSV**

Wakeup Status

Indicates the status of the current wakeup attempt (vPOC!WakeupStatus).

Reset when entering Wakeup state, by CHI command RESET_STATUS_INDICATORS, or by transition from DEFAULT_CONFIG to CONFIG state

000 = UNDEFINED
Wakeup not yet executed by the CC.

001 = RECEIVED_HEADER
Set when the CC finishes wakeup due to the reception of a frame header without coding violation on either channel in WAKEUP_LISTEN state.

010 = RECEIVED_WUP
Set when the CC finishes wakeup due to the reception of a valid wakeup pattern on the configured wakeup channel in WAKEUP_LISTEN state.

011 = COLLISION_HEADER

Set when the CC stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid header on either channel.

100 = COLLISION_WUP
Set when the CC stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid wakeup pattern on the configured wakeup channel.

101 = COLLISION_UNKNOWN
Set when the CC stops wakeup by leaving WAKEUP_DETECT state after expiration of the wakeup timer without receiving a valid wakeup pattern or a valid frame header.

110 = TRANSMITTED
Set when the CC has successfully completed the transmission of the wakeup pattern.

111 = reserved

(4) **FRCCSV.CSI**

Cold Start Inhibit

Indicates that the node is disabled from cold starting (vColdStartInhibit).

The flag is set whenever the POC enters READY state due to CHI command READY.

The flag has to be reset under control of the Host by CHI command ALLOW_COLDSTART (**FRSUCC1.CMD** = "1001").

(5) **FRCCSV.CSAI**

Coldstart Abort Indicator

Indicates that a coldstart attempt was aborted.

Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state or from READY to STARTUP state.

(6) **FRCCSV.CSNI**

Coldstart Noise Indicator

Indicates that the cold start procedure occurred under noisy conditions (vPOC!ColdstartNoise).

Reset by CHI command RESET_STATUS_INDICATORS or by transition from HALT to DEFAULT_CONFIG state or from READY to STARTUP state.

(7) **FRCCSV.SLM**

Slot Mode

Indicates the actual slot mode of the POC (vPOC!SlotMode) in states READY, WAKEUP, STARTUP, NORMAL_ACTIVE, and NORMAL_PASSIVE.

Default value is SINGLE. Changes to ALL, depending on **FRSUCC1.TSM**.

In NORMAL_ACTIVE or NORMAL_PASSIVE state the CHI command ALL_SLOTS will change the slot mode from SINGLE over ALL_PENDING to ALL.

Set to SINGLE in all other states.

(8) **FRCCSV.HRQ**

Halt Request

Indicates that a request from the Host has been received to halt the POC at the end of the communication cycle (vPOC!CHIHaltRequest).

Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

(9) **FRCCSV.FSI**

Freeze Status Indicator

Indicates that the POC has entered the HALT state due to CHI command FREEZE or due to an error condition requiring an immediate POC halt (vPOC!Freeze).

Reset by transition from HALT to DEFAULT_CONFIG state.

(10) **FRCCSV.POCS**

Protocol Operation Control Status

Indicates the actual state of operation of the CC Protocol Operation Control
00 0000 = DEFAULT_CONFIG state
00 0001 = READY state
00 0010 = NORMAL_ACTIVE state
00 0011 = NORMAL_PASSIVE state
00 0100 = HALT state
00 1111 = CONFIG state

Indicates the actual state of operation of the POC in the wakeup path
01 0000 = WAKEUP_STANDBY state
01 0001 = WAKEUP_LISTEN state
01 0010 = WAKEUP_SEND state
01 0011 = WAKEUP_DETECT state

Indicates the actual state of operation of the POC in the startup path
10 0000 = STARTUP_PREPARE state
10 0001 = COLDSTART_LISTEN state
10 0010 = COLDSTART_COLLISION_RESOLUTION state
10 0011 = COLDSTART_CONSISTENCY_CHECK state
10 0100 = COLDSTART_GAP state
10 0101 = COLDSTART_JOIN State
10 0110 = INTEGRATION_COLDSTART_CHECK state
10 0111 = INTEGRATION_LISTEN state
10 1000 = INTEGRATION_CONSISTENCY_CHECK state
10 1001 = INITIALIZE_SCHEDULE state
10 1010 = ABORT_STARTUP state
10 1011 = STARTUP_SUCCESS state

others = reserved

### 103.2.6.2 FRCCEV [h'0104]

FlexRay CC Error Vector

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | PTAC[4:0] | | | | | ERRM[1:0] | | - | - | CCFC[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:13] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [12:8] | PTAC | 0 | R | Passive to Active Count<br>Indicates vAllowPassiveToActive |
| [7:6] | ERRM | 0 | R | Error Mode<br>Indicates vPOC!ErrorMode<br>00 = ACTIVE<br>01 = PASSIVE<br>10 = COMM_HALT<br>11 = reserved |
| [5:4] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [3:0] | CCFC | 0 | R | Clock Correction Failed Counter<br>Indicates vClockCorrectionFailed |

(1) **FRCCEV.PTAC**

Passive to Active Count

Indicates the number of consecutive even / odd cycle pairs that have passed with valid rate and offset correction terms, while the node is waiting to transit from NORMAL_PASSIVE state to NORMAL_ACTIVE state. The transition takes place when **FRCCEV.PTAC** equals **FRSUCC1.PTA** -1.

Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

(2) **FRCCEV.ERRM**

Error Mode

Indicates the actual error mode of the POC (vPOC!ErrorMode).

Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

(3) **FRCCEV.CCFC**

Clock Correction Failed Counter

Indicates the clock correction failed counter of the POC (vClockCorrectionFailed).

The Clock Correction Failed Counter is incremented by one at the end of any odd communication cycle where either the missing offset correction error or missing rate correction error are active.

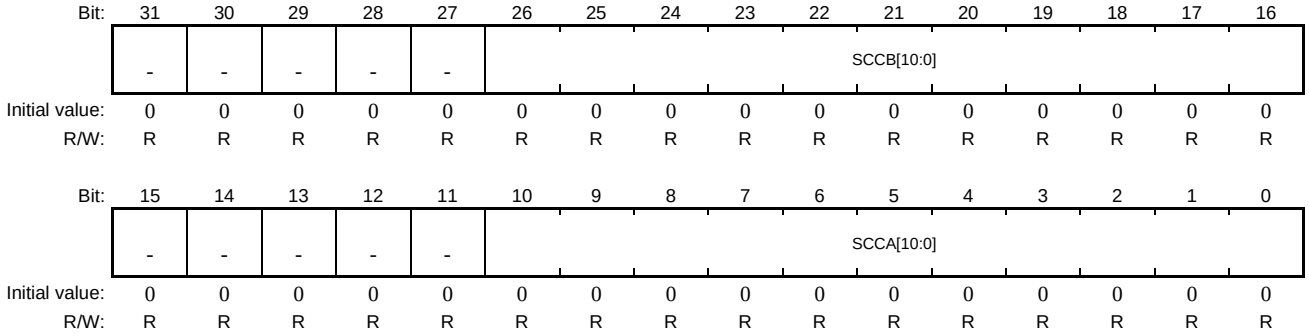
The Clock Correction Failed Counter is reset to ’0’ at the end of an odd communication cycle if neither the offset correction failed nor the rate correction failed errors are active.

The Clock Correction Failed Counter stops incrementing at 15.

Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state.

### 103.2.6.3 FRSCV [h'0110]

FlexRay Slot Counter Value




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26:16] | SCCB | 0 | R | Slot Counter Channel B<br>Indicates vSlotCounter[B] |
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | SCCA | 0 | R | Slot Counter Channel A<br>Indicates vSlotCounter[A] |

(1) **FRSCV.SCCB**

Slot Counter Channel B

Current slot counter value on channel B (vSlotCounter[B]). The value is incremented by the CC and reset at the start of a communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

(2) **FRSCV.SCCA**

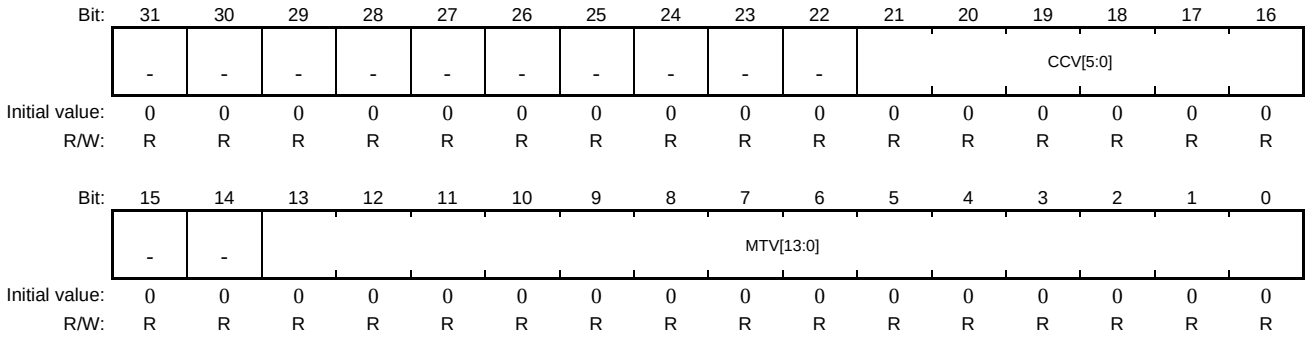
Slot Counter Channel A

Current slot counter value on channel A (vSlotCounter[A]). The value is incremented by the CC and reset at the start of a communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

### 103.2.6.4 FRMTCCV [h'0114]

FlexRay Macrotick and Cycle Counter Value




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:22] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [21:16] | CCV | 0 | R | Cycle Counter Value<br>Indicates vCycleCounter |
| [15:14] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [13:0] | MTV | 0 | R | Macrotick Value<br>Indicates vMacrotick |

(1) **FRMTCCV.CCV**

Cycle Counter Value

Current cycle counter value (vCycleCounter). The value is incremented by the CC at the start of a communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

(2) **FRMTCCV.MTV**

Macrotick Value

Current macrotick value (vMacrotick). The value is incremented by the CC and reset at the start of a communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

### 103.2.6.5 FRRCV [h'0118]

FlexRay Rate Correction Value

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | RCV[11:0] | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:12] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [11:0] | RCV | 0 | R | Rate Correction Value<br>Indicates vRateCorrection |

(1) **FRRCV.RCV**

Rate Correction Value

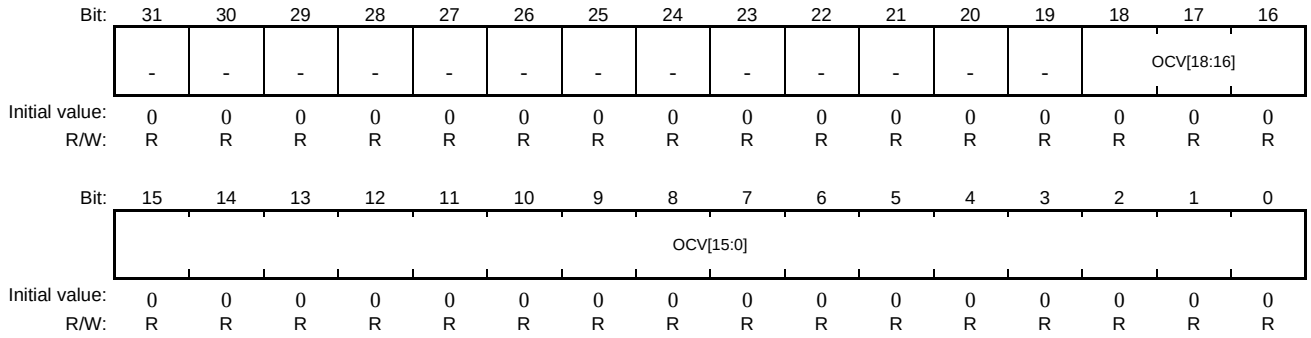
Rate correction value (vRateCorrection/ two's complement). Calculated internal rate correction value before limitation. If the **FRRCV.RCV** value exceeds the limits defined by **FRGTUC10.MRC**, flag **FRSFS.RCLR** is set to '1'.

**Note:** The external rate correction value is added to the limited rate correction value.

Reset when leaving CONFIG state or when entering STARTUP state.

### 103.2.6.6 FROCV [h'011C]

FlexRay Offset Correction Value




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18:0] | OCV | 0 | R | Offset Correction Value<br>Indicates vOffsetCorrection |

(1) **FROCV.OCV**

Offset Correction Value

Offset correction value (vOffsetCorrection/ two's complement). Calculated internal offset correction value before limitation. If the **FROCV.OCV** value exceeds the limits defined by **FRGTUC10.MOC**, flag **FRSFS.OCLR** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

**Note:** The external offset correction value is added to the limited offset correction value.

### 103.2.6.7 FRSFS [h'0120]

FlexRay Sync Frame Status

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | RCLR | MRCS | OCLR | MOCS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VSBO[3:0] | | | | VSBE[3:0] | | | | VSAO[3:0] | | | | VSAE[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:20] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [19] | RCLR | 0 | R | Rate Correction Limit Reached<br>0 = Rate correction below limit<br>1 = Rate correction limit reached |
| [18] | MRCS | 0 | R | Missing Rate Correction Signal<br>0 = Rate correction signal valid<br>1 = Missing rate correction signal |
| [17] | OCLR | 0 | R | Offset Correction Limit Reached<br>0 = Offset correction below limit<br>1 = Offset correction limit reached |
| [16] | MOCS | 0 | R | Missing Offset Correction Signal<br>0 = Offset correction signal valid<br>1 = Missing offset correction signal |
| [15:12] | VSBO | 0 | R | Valid Sync Frames Channel B, odd communication cycle |
| [11:8] | VSBE | 0 | R | Valid Sync Frames Channel B, even communication cycle |
| [7:4] | VSAO | 0 | R | Valid Sync Frames Channel A, odd communication cycle |
| [3:0] | VSAE | 0 | R | Valid Sync Frames Channel A, even communication cycle |

(1) **FRSFS.RCLR**

Rate Correction Limit Reached

The Rate Correction Limit Reached flag signals to the Host, that the rate correction value has exceeded its limit as defined by **FRGTUC10.MRC**. The flag is updated by the CC at start of offset correction phase.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

(2) **FRSFS.MRCS**

Missing Rate Correction Signal

The Missing Rate Correction flag signals to the Host, that no rate correction calculation can be performed because no pairs of even / odd sync frames were received. The flag is updated by the CC at start of offset correction phase.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

(3) **FRSFS.OCLR**

Offset Correction Limit Reached

The Offset Correction Limit Reached flag signals to the Host, that the offset correction value has exceeded its limit as defined by **FRGTUC10.MOC**. The flag is updated by the CC at start of offset correction phase.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

(4) **FRSFS.MOCS**

Missing Offset Correction Signal

The Missing Offset Correction flag signals to the Host, that no offset correction calculation can be performed because no sync frames were received. The flag is updated by the CC at start of offset correction phase.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

(5) **FRSFS.VSBO**

Valid Sync Frames Channel B, odd communication cycle

These bits are only valid when **FRSUCC1.CCHB** is '1'.

Holds the number of valid sync frames received on channel B in the odd communication cycle. If transmission of sync frames is enabled by **FRSUCC1.TXSY** the value is incremented by one. The value is updated during the NIT of each odd communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

(6) **FRSFS.VSBE**

Valid Sync Frames Channel B, even communication cycle

These bits are only valid when **FRSUCC1.CCHB** is '1'.

Holds the number of valid sync frames received on channel B in the even communication cycle. If transmission of sync frames is enabled by **FRSUCC1.TXSY** the value is incremented by one. The value is updated during the NIT of each even communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

(7) **FRSFS.VSAO**

Valid Sync Frames Channel A, odd communication cycle

These bits are only valid when **FRSUCC1.CCHA** is '1'.

Holds the number of valid sync frames received on channel A in the odd communication cycle. If transmission of sync frames is enabled by **FRSUCC1.TXSY** the value is incremented by one. The value is updated during the NIT of each odd communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

(8) **FRSFS.VSAE**

Valid Sync Frames Channel A, even communication cycle

These bits are only valid when **FRSUCC1.CCHA** is '1'.

Holds the number of valid sync frames received on channel A in the even communication cycle. If transmission of sync frames is enabled by **FRSUCC1.TXSY** the value is incremented by one. The value is updated during the NIT of each even communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state, INTEGRATION_COLDSTART_CHECK state or INTEGRATION_CONSISTENCY_CHECK state.

### 103.2.6.8 FRSWNIT [h'0124]

FlexRay Symbol Window and NIT Status

Symbol window related status information is updated by the CC at the end of the symbol window for each channel. NIT related status information is updated by the CC at the end of the NIT for each channel.

During startup the status data is not updated.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | SBNB | SENB | SBNA | SENA | MTSB | MTSA | TCSB | SBSB | SESB | TCSA | SBSA | SESA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:12] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [11] | SBNB | 0 | R | Slot Boundary Violation during NIT Channel B<br>0 = No slot boundary violation detected on channel B<br>1 = Slot boundary violation during NIT detected on channel B |
| [10] | SENB | 0 | R | Syntax Error during NIT Channel B<br>0 = No syntax error detected on channel B<br>1 = Syntax error during NIT detected on channel B |
| [9] | SBNA | 0 | R | Slot Boundary Violation during NIT Channel A<br>0 = No slot boundary violation detected on channel A<br>1 = Slot boundary violation during NIT detected on channel A |
| [8] | SENA | 0 | R | Syntax Error during NIT Channel A<br>0 = No syntax error detected on channel A<br>1 = Syntax error during NIT detected on channel A |
| [7] | MTSB | 0 | R | MTS Received on Channel B<br>0 = No MTS symbol received on channel B<br>1 = MTS symbol received on channel B |
| [6] | MTSA | 0 | R | MTS Received on Channel A<br>0 = No MTS symbol received on channel A<br>1 = MTS symbol received on channel A |
| [5] | TCSB | 0 | R | Transmission Conflict in Symbol Window Channel B<br>0 = No transmission conflict detected on channel B<br>1 = Transmission conflict in symbol window detected on channel B |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [4] | SBSB | 0 | R | Slot Boundary Violation in Symbol Window Channel B<br>0 = No slot boundary violation detected on channel B<br>1 = Slot boundary violation during symbol window detected on channel B |
| [3] | SESB | 0 | R | Syntax Error in Symbol Window Channel B<br>0 = No syntax error detected on channel B<br>1 = Syntax error during symbol window detected on channel B |
| [2] | TCSA | 0 | R | Transmission Conflict in Symbol Window Channel A<br>0 = No transmission conflict detected on channel A<br>1 = Transmission conflict in symbol window detected on channel A |
| [1] | SBSA | 0 | R | Slot Boundary Violation in Symbol Window Channel A<br>0 = No slot boundary violation detected on channel A<br>1 = Slot boundary violation during symbol window detected on channel A |
| [0] | SESA | 0 | R | Syntax Error in Symbol Window Channel A<br>0 = No syntax error detected on channel A<br>1 = Syntax error during symbol window detected on channel A |

(1) **FRSWNIT.SBNB**

Indicates a Slot Boundary Violation during NIT Channel B (vSS!BViolationB).

Reset when leaving CONFIG state or when entering STARTUP state.

(2) **FRSWNIT.SENB**

Indicates a Syntax Error during NIT Channel B (vSS!SyntaxErrorB).

Reset when leaving CONFIG state or when entering STARTUP state.

(3) **FRSWNIT.SBNA**

Indicates a Slot Boundary Violation during NIT Channel A (vSS!BViolationA).

Reset when leaving CONFIG state or when entering STARTUP state.

(4) **FRSWNIT.SENA**

Indicates a Syntax Error during NIT Channel A (vSS!SyntaxErrorA).

Reset when leaving CONFIG state or when entering STARTUP state.

(5) **FRSWNIT.MTSB**

Indicates a MTS Received on Channel B (vSS!ValidMTSB).

Media Access Test symbol received on channel B during the preceding symbol window. Updated by the CC for each channel at the end of the symbol window.

When this bit is set to ’1’, also interrupt flag **FRSIR.MTSB** is set to ’1’.

Reset when leaving CONFIG state or when entering STARTUP state.

(6) **FRSWNIT.MTSA**

Indicates a MTS Received on Channel A (vSS!ValidMTSA).

Media Access Test symbol received on channel A during the preceding symbol window. Updated by the CC for each channel at the end of the symbol window.

When this bit is set to ’1’, also interrupt flag **FRSIR.MTSA** is set to ’1’.

Reset when leaving CONFIG state or when entering STARTUP state.

(7) **FRSWNIT.TCSB**

Indicates a Transmission Conflict in Symbol Window Channel B (vSS!TxConflictB).

Reset when leaving CONFIG state or when entering STARTUP state.

(8) **FRSWNIT.SBSB**

Indicates a Slot Boundary Violation in Symbol Window Channel B (vSS!BViolationB).

Reset when leaving CONFIG state or when entering STARTUP state.

(9) **FRSWNIT.SESB**

Indicates a Syntax Error in Symbol Window Channel B (vSS!SyntaxErrorB).

Reset when leaving CONFIG state or when entering STARTUP state.

(10) **FRSWNIT.TCSA**

Indicates a Transmission Conflict in Symbol Window Channel A (vSS!TxConflictA).

Reset when leaving CONFIG state or when entering STARTUP state.

(11) **FRSWNIT.SBSA**

Indicates a Slot Boundary Violation in Symbol Window Channel A (vSS!BViolationA).

Reset when leaving CONFIG state or when entering STARTUP state.

(12) **FRSWNIT.SESA**

Indicates a Syntax Error in Symbol Window Channel A (vSS!SyntaxErrorA).

Reset when leaving CONFIG state or when entering STARTUP state.

### 103.2.6.9 FRACS [h'0128]

FlexRay Aggregated Channel Status

The aggregated channel status provides the Host with an accrued status of channel activity for all communication slots regardless of whether they are assigned for transmission or subscribed for reception.

The aggregated channel status also includes status data from the symbol window and the network idle time.

The status data is updated (set) after each slot and aggregated until it is reset by the Host.

During startup the status data is not updated.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | SBVB | CIB | CEDB | SEDB | VFRB | - | - | - | SBVA | CIA | CEDA | SEDA | VFRA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:13] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [12] | SBVB | 0 | R/W | Slot Boundary Violation on Channel B<br>0 = No slot boundary violation observed on channel B<br>1 = Slot boundary violation(s) observed on channel B |
| [11] | CIB | 0 | R/W | Communication Indicator Channel B<br>0 = No valid frame(s) received on channel B in slots containing any additional communication<br>1 = Valid frame(s) received on channel B in slots containing any additional communication |
| [10] | CEDB | 0 | R/W | Content Error Detected on Channel B<br>0 = No frame with content error received on channel B<br>1 = Frame(s) with content error received on channel B |
| [9] | SEDB | 0 | R/W | Syntax Error Detected on Channel B<br>0 = No syntax error observed on channel B<br>1 = Syntax error(s) observed on channel B |
| [8] | VFRB | 0 | R/W | Valid Frame Received on Channel B<br>0 = No valid frame received on channel B<br>1 = Valid frame(s) received on channel B |
| [7:5] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [4] | SBVA | 0 | R/W | Slot Boundary Violation on Channel A<br>0 = No slot boundary violation observed on channel A<br>1 = Slot boundary violation(s) observed on channel A |
| [3] | CIA | 0 | R/W | Communication Indicator Channel A<br>0 = No valid frame(s) received on channel A in slots containing any additional communication<br>1 = Valid frame(s) received on channel A in slots containing any additional communication |
| [2] | CEDA | 0 | R/W | Content Error Detected on Channel A<br>0 = No frame with content error received on channel A<br>1 = Frame(s) with content error received on channel A |
| [1] | SEDA | 0 | R/W | Syntax Error Detected on Channel A<br>0 = No syntax error observed on channel A<br>1 = Syntax error(s) observed on channel A |
| [0] | VFRA | 0 | R/W | Valid Frame Received on Channel A<br>0 = No valid frame received on channel A<br>1 = Valid frame(s) received on channel A |

(1) **FRACS.SBVB**

Slot Boundary Violation on Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more slot boundary violations were observed on channel B at any time during the observation period (static or dynamic slots, symbol window, and NIT).

When this flag changes from '0' to '1', interrupt flag **FREIR.EDB** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

(2) **FRACS.CIB**

Communication Indicator Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more valid frames were received on channel B in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid frame AND had any combination of either syntax error OR content error OR slot boundary violation.

**Note:** The set condition of the flag **FRACS.CIB** is also fulfilled if there is only one single frame in the slot and the slot boundary at the end of the slot is reached during the frames channel idle recognition phase.

When this flag changes from '0' to '1', interrupt flag **FREIR.EDB** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

(3) **FRACS.CEDB**

Content Error Detected on Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more frames with a content error were received on channel B in any static or dynamic slot during the observation period.

When this flag changes from '0' to '1', interrupt flag **FREIR.EDB** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

(4) **FRACS.SEDB**

Syntax Error Detected on Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more syntax errors in static or dynamic slots, symbol window, and NIT were observed on channel B.

When this flag changes from '0' to '1', interrupt flag **FREIR.EDB** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

(5) **FRACS.VFRB**

Valid Frame Received on Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more valid frames were received on channel B in any static or dynamic slot during the observation period.

Reset when leaving CONFIG state or when entering STARTUP state.

(6) **FRACS.SBVA**

Slot Boundary Violation on Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more slot boundary violations were observed on channel A at any time during the observation period (static or dynamic slots, symbol window, and NIT).

When this flag changes from '0' to '1', interrupt flag **FREIR.EDA** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

(7) **FRACS.CIA**

Communication Indicator Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more valid frames were received on channel A in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid frame AND had any combination of either syntax error OR content error OR slot boundary violation.

**Note:** The set condition of the flag **FRACS.CIA** is also fulfilled if there is only one single frame in the slot and the slot boundary at the end of the slot is reached during the frames channel idle recognition phase.

When this flag changes from '0' to '1', interrupt flag **FREIR.EDA** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

(8) **FRACS.CEDA**

Content Error Detected on Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more frames with a content error were received on channel A in any static or dynamic slot during the observation period.

When this flag changes from '0' to '1', interrupt flag **FREIR.EDA** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

(9) **FRACS.SEDA**

Syntax Error Detected on Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more syntax errors in static or dynamic slots, symbol window, and NIT were observed on channel A.

When this flag changes from '0' to '1', interrupt flag **FREIR.EDA** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

(10) **FRACS.VFRA**

Valid Frame Received on Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

One or more valid frames were received on channel A in any static or dynamic slot during the observation period.

Reset when leaving CONFIG state or when entering STARTUP state.

### 103.2.6.10 FRESIDn (n=1 to 15) [h'0130 – h'0168]

FlexRay Even Sync ID n (n=1 to 15)

Registers **FRESID1** to **FRESID15** hold the frame IDs of the sync frames received in even communication cycles used for clock synchronisation up to the limit of gSyncNodeMax. The values are sorted in ascending order, with register **FRESID1** holding the lowest received sync frame ID. If the node itself transmits a sync frame in an even communication cycle, register **FRESID1** holds the respective sync frame ID as configured in message buffer 0 and flags **FRESID1.RXEA**, **FRESID1.RXEB** are set. The value is updated during the NIT of each even communication cycle.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RXEB | RXEA | - | - | - | - | EID[9:0] | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:16] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [15] | RXEB | 0 | R | Received / Configured Even Sync ID on Channel B<br>0 = No sync frame received on channel B / node not configured to transmit sync frames<br>1 = Sync frame received on channel B / node configured to transmit sync frames |
| [14] | RXEA | 0 | R | Received / Configured Even Sync ID on Channel A<br>0 = No sync frame received on channel A / node not configured to transmit sync frames<br>1 = Sync frame received on channel A / node configured to transmit sync frames |
| [13:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [9:0] | EID | 0 | R | Even Sync ID (vsSyncIDListA,B even) |

(1) **FRESIDn.RXEB**

Received / Configured Even Sync ID on Channel B

Signals that a sync frame corresponding to the stored even sync ID was received on channel B or that the node is configured to be a sync node with key slot = **FRESID1.EID**.

Reset when leaving CONFIG state or when entering STARTUP state.

(2) **FRESIDn.RXEA**

Received / Configured Even Sync ID on Channel A

Signals that a sync frame corresponding to the stored even sync ID was received on channel A or that the node is configured to be a sync node with key slot = **FRESID1.EID**.

Reset when leaving CONFIG state or when entering STARTUP state.

(3) **FRESIDn.EID**

Even Sync ID (vsSyncIDListA,B even)

Sync frame ID even communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

### 103.2.6.11 FROSIDn (n=1 to 15) [h'0170 – h'01A8]

FlexRay Odd Sync ID n (n=1 to 15)

Registers **FROSID1** to **FROSID15** hold the frame IDs of the sync frames received in odd communication cycles used for clock synchronisation up to the limit of gSyncNodeMax. The values are sorted in ascending order, with register **FROSID1** holding the lowest received sync frame ID. If the node itself transmits a sync frame in an odd communication cycle, register **FROSID1** holds the respective sync frame ID as configured in message buffer 0 and flags **FROSID1.RXOA**, **FROSID1.RXOB** are set. The value is updated during the NIT of each odd communication cycle.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RXOB | RXOA | - | - | - | - | OID[9:0] | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:16] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [15] | RXOB | 0 | R | Received / Configured Odd Sync ID on Channel B<br>0 = No sync frame received on channel B / node not configured to transmit sync frames<br>1 = Sync frame received on channel B / node configured to transmit sync frames |
| [14] | RXOA | 0 | R | Received / Configured Odd Sync ID on Channel A<br>0 = No sync frame received on channel A / node not configured to transmit sync frames<br>1 = Sync frame received on channel A / node configured to transmit sync frames |
| [13:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [9:0] | OID | 0 | R | Odd Sync ID (vsSyncIDListA,B odd) |

(1) **FROSIDn.RXOB**

Received / Configured Odd Sync ID on Channel B

Signals that a sync frame corresponding to the stored odd sync ID was received on channel B or that the node is configured to be a sync node with key slot = **FROSID1.OID**.

Reset when leaving CONFIG state or when entering STARTUP state.

(2) **FROSIDn.RXOA**

Received / Configured Odd Sync ID on Channel A

Signals that a sync frame corresponding to the stored odd sync ID was received on channel A or that the node is configured to be a sync node with key slot = **FRISID1.OID**.

Reset when leaving CONFIG state or when entering STARTUP state.

(3) **FROSIDn.OID**

Odd Sync ID (vsSyncIDListA,B odd)

Sync frame ID odd communication cycle.

Reset when leaving CONFIG state or when entering STARTUP state.

### 103.2.6.12 FRNMVn (n=1 to 3) [h'01B0 – h'01B8]

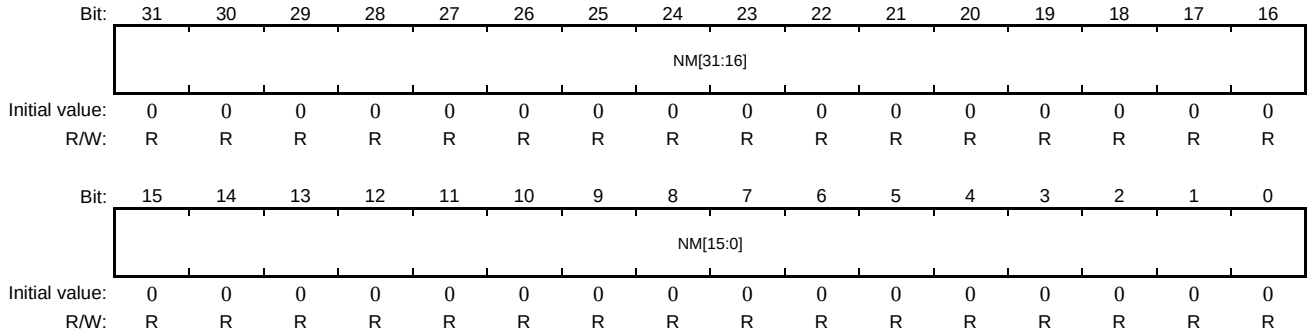
FlexRay Network Management Vector n (n=1 to 3)

The three network management registers hold the accrued NM vector (see section 103.3.7).

The CC updates the NM vector at the end of each communication cycle as long as the CC is either in NORMAL_ACTIVE or NORMAL_PASSIVE state.

NMVn-bytes exceeding the configured NM vector length are not valid.

For information about the byte alignment of the received NM vector in this register see section 103.3.17.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:0] | NM | 0 | R | NM vector |

(1) **FRNMVn.NM**

NM vector

The three network management registers hold the accrued NM vector (configurable 0 to 12 bytes).

The accrued NM vector is generated by the CC by bit-wise ORing each NM vector received (valid static frames with PPI = '1') on each channel (see section 103.3.7).

For information about the byte alignment of the received NM vector in this register see section 103.3.17.

**FRNMVn**-bytes exceeding the configured NM vector length are not valid.

The CC updates the NM vector at the end of each communication cycle as long as the CC is either in NORMAL_ACTIVE or NORMAL_PASSIVE state.

Reset when leaving CONFIG state or when entering STARTUP state.

### 103.2.7 Message Buffer Control Registers

#### 103.2.7.1 FRMRC [h'0300]

FlexRay Message RAM Configuration

The Message RAM Configuration register defines the number of message buffers assigned to the static segment, dynamic segment, and FIFO.

The Message RAM can be divided into up three different areas; Static Buffer area, Static and Dynamic Buffer area, FIFO area. If present, the Static Buffer area is starting at Message Buffer 0.

The start of the Static and Dynamic Buffer area is configured by **FRMRC.FDB**. **FRMRC.FDB** defines the end of the Static Buffer area. If no Static Buffer area is present, the Static and Dynamic Buffer area starts at Message Buffer 0.

The start of the FIFO area is configured by **FRMRC.FFB**. **FRMRC.FFB** defines the end of the previous area, which can be either the Static Buffer area or the Static and Dynamic Buffer area. If no Static Buffer area and no Static and Dynamic Buffer area is present, the FIFO area starts at Message Buffer 0.

With **FRMRC.LCB** the end of the last configured area is configured which can be the Static Buffer area, the Static and Dynamic Buffer area or the FIFO area.

Figure 103.2 shows an example configuration of the Message RAM where all there area are configured.

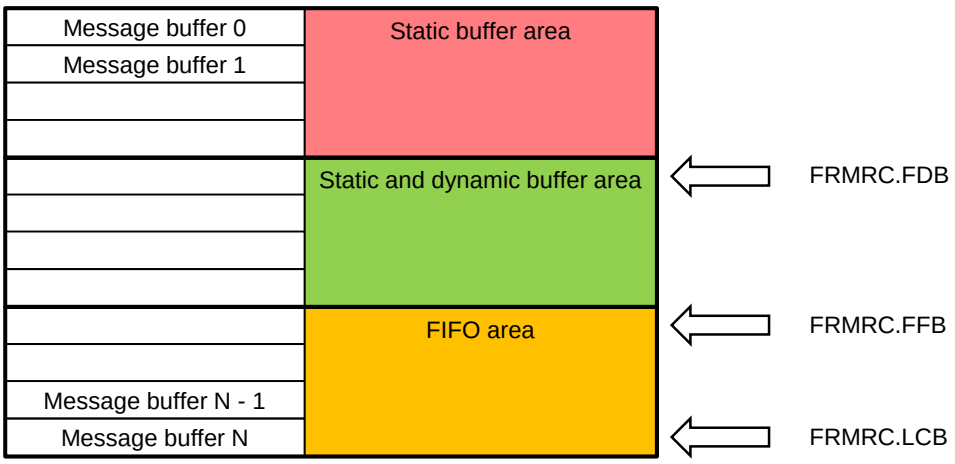



**Figure 103.2 Message RAM organisation**

**Note 1:** In case the node is configured as sync node (**FRSUCC1.TXSY** = '1') or for single slot mode operation (**FRSUCC1.TSM** = '1'), message buffer 0 resp. 1 is reserved for sync frames or single slot frames and have to be configured with the node-specific key slot ID. In case the node is neither configured as sync node nor for single slot operation message buffer 0 resp. 1 is treated like all other message buffers.

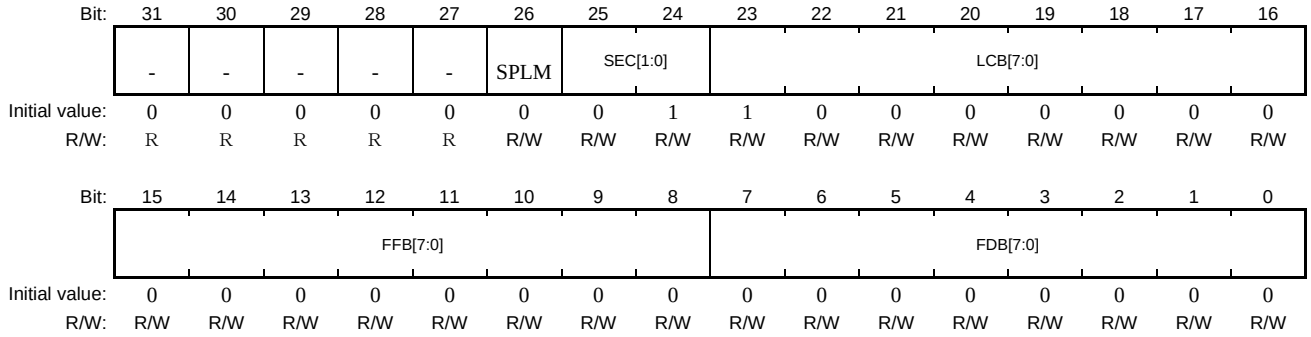
**Note 2:** The maximum number of header sections is 128. This means a maximum of 128 message buffers can be configured. The maximum length of a data section is 254 bytes. The length of the data section may be configured differently for each message buffer. For details see section 103.3.13.

**Note 3:** In case two or more message buffers are assigned to slot 1 by use of cycle filtering, all of them must be located either in the "Static Buffer area" or at the beginning of the "Static and Dynamic Buffer area" section.

**Note 4:** The FlexRay protocol specification requires that each node has to send a frame in its key slot. Therefore at least message buffer 0 is reserved for transmission in the key slot. Due to this requirement a maximum number of 127 message

buffers can be assigned to the FIFO area. Nevertheless, a non protocol conform configuration without a transmission slot in the static segment would still be operational.

**Note 5:** The payload length configured and the length of the data section need to be configured identical for all message buffers belonging to the FIFO area via **FRWRHS2.PLC** and **FRWRHS3.DP**. When the CC is not in DEFAULT_CONFIG or CONFIG state reconfiguration of message buffers belonging to the FIFO area is locked.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26] | SPLM | 0 | R/W | Sync Frame Payload Multiplex<br>0 = Only message buffer 0 locked against reconfiguration<br>1 = Both message buffers 0 and 1 are locked against reconfiguration |
| [25:24] | SEC | 1h | R/W | Secure Buffers<br>00 = all buffers unlocked<br>01 = static buffers locked, FIFO locked, limited transmission<br>10 = all buffers locked<br>11 = all buffers locked, limited transmission |
| [23:16] | LCB | 80h | R/W | Last Configured Buffer<br>0…127 = Number of message buffers is FRMRF.LCB + 1<br>128 = No message buffer configured |
| [15:8] | FFB | 0 | R/W | First Buffer of FIFO<br>0 = All message buffers assigned to the FIFO<br>1…127 = Message buffers from FRMRC.FFB to FRMRC.LCB assigned to the FIFO<br>128 = No message buffer assigned to the FIFO |
| [7:0] | FDB | 0 | R/W | First Dynamic Buffer<br>0 = No group of message buffers exclusively for the static segment configured<br>1…127 = Message buffers 0 to FRMRC.FDB - 1 reserved for static segment<br>128 = No dynamic message buffers configured |

(1) **FRMRC.SPLM**

Sync Frame Payload Multiplex

The user can only write to this bit when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

This bit is only evaluated if the node is configured as sync node (**FRSUCC1.TXSY** = ’1’) or for single slot mode operation (**FRSUCC1.TSM** = ’1’).

When this bit is set to ’1’ message buffers 0 and 1 are dedicated for sync frame transmission with different payload data on channel A and B.

When this bit is set to ’0’, sync frames are transmitted from message buffer 0 with the same payload data on all configured channels.

Note that the channel filter configuration for message buffer 0 resp. message buffer 1 has to be chosen accordingly.

(2) **FRMRC.SEC**

Secure Buffers

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Not evaluated when the CC is in DEFAULT_CONFIG or CONFIG state. For temporary unlocking see section 6.13.4.3.

00 = all buffers unlocked
Reconfiguration of message buffers enabled with numbers < **FRMRC.FFB** enabled
Exception: In nodes configured for sync frame transmission or for single slot mode operation message buffer 0 (and if **FRMRC.SPLM** = ’1’, also message buffer 1) is always locked

01 = static buffers locked, FIFO locked, limited transmission
Reconfiguration of message buffers with numbers < **FRMRC.FDB** and with numbers => **FRMRC.FFB** locked and transmission of message buffers for static segment with numbers => **FRMRC.FDB** disabled

10 = all buffers locked
Reconfiguration of all message buffers locked

11 = all buffers locked, limited transmission
Reconfiguration of all message buffers locked and transmission of message buffers for static segment with numbers >= **FRMRC.FDB** disabled

(3) **FRMRC.LCB**

Last Configured Buffer

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

When a Static and Dynamic Buffer area is configured (**FRMRC.FDB** < 128), the user should configure **FRMRC.LCB** >= **FRMRC.FDB**.

When a FIFO area is configured (**FRMRC.FFB** < 128), the user should configure **FRMRC.LCB** >= **FRMRC.FFB**.

(4) **FRMRC.FFB**

First Buffer of FIFO

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

When a Static and Dynamic Buffer area is configured (**FRMRC.FDB** < 128), the user should configure **FRMRC.FFB** > **FRMRC.FDB**.

(5) **FRMRC.FDB**

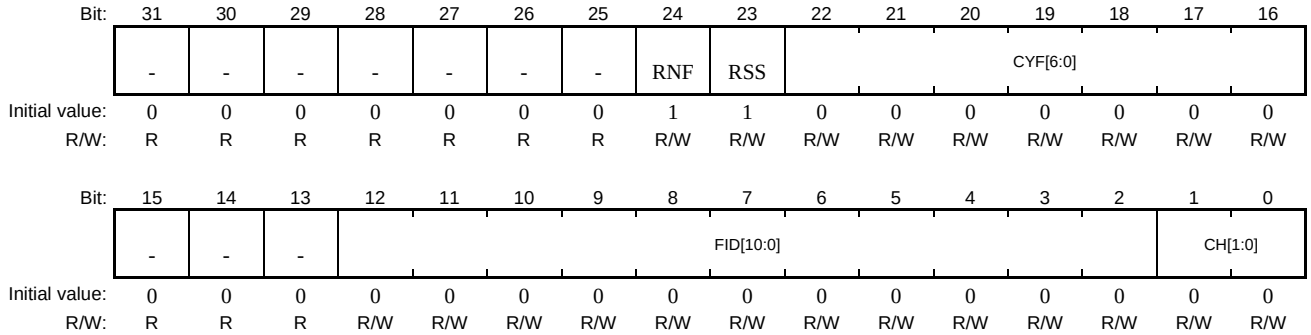
First Dynamic Buffer

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

### 103.2.7.2 FRFRF [h'0304]

FlexRay FIFO Rejection Filter

The FIFO Rejection Filter defines a user specified sequence of bits to which channel, frame ID, and cycle count of the incoming frames are compared. Together with the FIFO Rejection Filter Mask this register determines whether a message is rejected by the FIFO.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:25] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [24] | RNF | 1 | R/W | Reject Null Frames<br>0 = Null frames are stored in the FIFO<br>1 = Reject all null frames |
| [23] | RSS | 1 | R/W | Reject in Static Segment<br>0 = FIFO also used for static segment<br>1 = Reject messages in static segment |
| [22:16] | CYF | 0 | R/W | Cycle Counter Filter |
| [15:13] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [12:2] | FID | 0 | R/W | Frame ID Filter<br>0 ... 2047 = Frame ID filter values |
| [1:0] | CH | 0 | R/W | Channel Filter<br>00 = receive on both channels<br>01 = receive only on channel B<br>10 = receive only on channel A<br>11 = no reception |

(1) **FRFRF.RNF**

Reject Null Frames

The user can only write to this bit when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

If this bit is set, received null frames are not stored in the FIFO.

(2) **FRFRF.RSS**

Reject in Static Segment

The user can only write to this bit when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

If this bit is set, the FIFO is used only for the dynamic segment.

(3) **FRFRF.CYF**

Cycle Counter Filter

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

The 7-bit cycle counter filter determines the cycle set to which frame ID and channel rejection filter are applied. In cycles not belonging to the cycle set specified by **FRFRF.CYF**, all frames are rejected. For details about the configuration of the cycle counter filter see section 103.3.8.2.

(4) **FRFRF.FID**

Frame ID Filter

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

Determines the frame ID to be rejected by the FIFO. With the additional configuration of register **FRFRFM**, the corresponding frame ID filter bits are ignored, which results in further rejected frame IDs. When **FRFRFM.MFID** is zero, a frame ID filter value of zero means that no frame ID is rejected.

(5) **FRFRF.CH**

Channel Filter

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

If reception on both channels is configured, also in static segment always both frames (from channel A and B) are stored in the FIFO, even if they are identical.

### 103.2.7.3 FRFRFM [h'0308]

FlexRay FIFO Rejection Filter Mask

The FlexRay FIFO Rejection Filter Mask specifies which of the corresponding frame ID filter bits are relevant for rejection filtering. If a bit is set, it indicates that the corresponding bit in the **FRFRF** register will not be considered for rejection filtering.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | MFID[10:0] | | | | | | | | | | | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:13] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [12:2] | MFID | 0 | R/W | Mask Frame ID Filter<br>0 = Corresponding frame ID filter bit is used for rejection filtering<br>1 = Ignore corresponding frame ID filter bit. |
| [1:0] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |

(1) **FRFRFM.MFID**

Mask Frame ID Filter

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

### 103.2.7.4 FRFCL [h'030C]

FlexRay FIFO Critical Level

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | CL[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:8] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [7:0] | CL | 80h | R/W | Critical Level |

(1) **FRFCL.CL**

Critical Level

The user can only write to these bits when **FRCCSV.POCS** is DEFAULT_CONFIG or CONFIG.

When the receive FIFO fill level **FRFSR.RFFL** is equal or greater than the critical level configured by **FRFCL.CL**, the receive FIFO critical level flag **FRFSR.RFCL** is set.

If **FRFCL.CL** is programmed to values > 128, bit **FRFSR.RFCL** is never set.

## 103.2.8 Message Buffer Status Registers

### 103.2.8.1 FRMHDS [h'0310]

FlexRay Message Handler Status

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | MBU[6:0] | | | | | | | - | MBT[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | FMB[6:0] | | | | | | | CRAM | MFMB | FMBD | ATBF2 | ATBF1 | AMR | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [30:24] | MBU | 0 | R | Message Buffer Updated |
| [23] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [22:16] | MBT | 0 | R | Message Buffer Transmitted |
| [15] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [14:8] | FMB | 0 | R | Faulty Message Buffer |
| [7] | CRAM | 1 | R | Clear all internal RAM's<br>0 = No execution of the CHI command CLEAR_RAMS<br>1 = Execution of the CHI command CLEAR_RAMS ongoing |
| [6] | MFMB | 0 | R/W | Multiple Faulty Message Buffers detected<br>0 = No additional faulty message buffer<br>1 = Another faulty message buffer was detected while flag FRMHDS.FMBD is set |
| [5] | FMBD | 0 | R/W | Faulty Message Buffer Detected<br>0 = No faulty message buffer<br>1 = Message buffer referenced by FRMHDS.FMB holds faulty data due to an access error |
| [4] | ATBF2 | 0 | R/W | Access Error Transient buffer RAM B<br>0 = No access error<br>1 = Access error occurred when reading the Transient buffer RAM B |
| [3] | ATBF1 | 0 | R/W | Access Error Transient buffer RAM A<br>0 = No access error<br>1 = Access error occurred when reading the Transient buffer RAM A |
| [2] | AMR | 0 | R/W | Access Error Message RAM<br>0 = No access error<br>1 = Access error occurred when reading the Message RAM |
| [1:0] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |

(1) **FRMHDS.MBU**

Message Buffer Updated

Number of message buffer that was updated last by the CC. For this message buffer the respective **ND** and / or **MBC** flag in the **FRNDAT1/2/3/4** registers and the **FRMBSC1/2/3/4** registers are also set.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(2) **FRMHDS.MBT**

Message Buffer Transmitted

Number of last successfully transmitted message buffer. If the message buffer is configured for single-shot mode, the respective **FRTXRQn.TXR** flag in the **FRTXRQ1/2/3/4** registers was reset.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(3) **FRMHDS.FMB**

Faulty Message Buffer

Access error occurred when reading from the message buffer referenced by **FRMHDS.FMB**.

Value only valid when one of the flags **FRMHDS.AMR**, **FRMHDS.ATBF1**, **FRMHDS.ATBF2** or **FRMHDS.FMBD** is set.

Flag is not updated while flag **FRMHDS.FMBD** is set.

Reset by the CHI command CLEAR_RAMS.

(4) **FRMHDS.CRAM**

Clear all internal RAM’s

Signals that execution of the CHI command CLEAR_RAMS is ongoing (all bits of all message RAM, input buffer, output buffer and transient buffer are written to ’0’).

This bit is set by the CHI command CLEAR_RAMS.

(5) **FRMHDS.MFMB**

Multiple Faulty Message Buffers detected

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This bit indicates that another faulty message buffer was detected while flag **FRMHDS.FMBD** is set

Reset by the CHI command CLEAR_RAMS.

(6) **FRMHDS.FMBD**

Faulty Message Buffer Detected

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This bit indicates that the Message buffer referenced by **FRMHDS.FMB** holds faulty data due to an access error

Reset by the CHI command CLEAR_RAMS.

(7) **FRMHDS.ATBF2**

Access Error Transient buffer RAM B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This bit indicates that an access error occurred when reading the Transient buffer RAM B

**Note:** When **FRMHDS.ATBF2** changes from ’0’ to ’1’ **FREIR.AERR** is set to ’1’.

Reset by the CHI command CLEAR_RAMS.

(8) **FRMHDS.ATBF1**

Access Error Transient buffer RAM A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This bit indicates that an access error occurred when reading the Transient buffer RAM A

**Note:** When **FRMHDS.ATBF1** changes from ’0’ to ’1’ **FREIR.AERR** is set to ’1’.

Reset by the CHI command CLEAR_RAMS.

(9) **FRMHDS.AMR**

Access Error Message RAM

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This bit indicates that an access error occurred when reading the Message RAM

**Note:** When **FRMHDS.AMR** changes from ’0’ to ’1’ **FREIR.AERR** is set to ’1’.

Reset by the CHI command CLEAR_RAMS.

### 103.2.8.2 FRLDTS [h’0314]

FlexRay Last Dynamic Transmit Slot

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | LDTB[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | LDTA[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:27] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [26:16] | LDTB | 0 | R | Last Dynamic Transmission Channel B |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | LDTA | 0 | R | Last Dynamic Transmission Channel A |

(1) **FRLDTS.LDTB**

Last Dynamic Transmission Channel B

Value of vSlotCounter[B] at the time of the last frame transmission on channel B in the dynamic segment of this node.

It is updated at the end of the dynamic segment and is reset to zero if no frame was transmitted during the dynamic segment on channel B.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(2) **FRLDTS.LDTA**

Last Dynamic Transmission Channel A

Value of vSlotCounter[A] at the time of the last frame transmission on channel A in the dynamic segment of this node.

It is updated at the end of the dynamic segment and is reset to zero if no frame was transmitted during the dynamic segment on channel A.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

### 103.2.8.3 FRFSR [h'0318]

FlexRay FIFO Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFFL[7:0] | | | | | | | | - | - | - | - | - | RFO | RFCL | RFNE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:16] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [15:8] | RFFL | 0 | R | Receive FIFO Fill Level |
| [7:3] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [2] | RFO | 0 | R | Receive FIFO Overrun<br>0 = No receive FIFO overrun detected<br>1 = A receive FIFO overrun has been detected |
| [1] | RFCL | 0 | R | Receive FIFO Critical Level<br>0 = Receive FIFO below critical level<br>1 = Receive FIFO critical level reached |
| [0] | RFNE | 0 | R | Receive FIFO Not Empty<br>0 = Receive FIFO is empty<br>1 = Receive FIFO is not empty |

(1) **FRFSR.RFFL**

Receive FIFO Fill Level

Number of FIFO buffers filled with new data not yet read by the Host. Maximum value is 128.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(2) **FRFSR.RFO**

Receive FIFO Overrun

The flag is set by the CC when a receive FIFO overrun is detected.

When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. In addition, interrupt flag **FREIR.RFO** is set.

The flag is cleared by the next FIFO read access issued by the Host.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(3) **FRFSR.RFCL**

Receive FIFO Critical Level

This flag is set when the receive FIFO fill level **FRFSR.RFFL** is equal or greater than the critical level as configured by **FRFCL.CL**.

When **FRFSR.RFCL** changes from ’0’ to ’1’ bit **FRSIR.RFCL** is set to ’1’, and if enabled, an interrupt is generated.

The flag is cleared by the CC as soon as **FRFSR.RFFL** drops below **FRFCL.CL**.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(4) **FRFSR.RFNE**

Receive FIFO Not Empty

This flag is set by the CC when a received valid frame (data or null frame depending on rejection mask) was stored in the FIFO. In addition, interrupt flag **FRSIR.RFNE** is set.

The bit is reset after the Host has read all messages from the FIFO.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

### 103.2.8.4 FRMHDF [h'031C]

FlexRay Message Handler Constraints Flags

Some constraints exist for the Message Handler regarding bus clock frequency, Message RAM configuration, and FlexRay bus traffic (see Addendum to ERAY FlexRay IP-Module Specification). To simplify software development, constraints violations are reported by setting flags in the **FRMHDF**.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | WAHP | TNSB | TNSA | TBFB | TBFA | FNFB | FNFA | SNUB | SNUA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:9] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [8] | WAHP | 0 | R/W | Write Attempt to Header Partition<br>0 = No write attempt to header partition<br>1 = Write attempt to header partition |
| [7] | TNSB | 0 | R/W | Transmission Not Started Channel B<br>0 = No transmission not started on channel B<br>1 = Transmission not started on channel B |
| [6] | TNSA | 0 | R/W | Transmission Not Started Channel A<br>0 = No transmission not started on channel A<br>1 = Transmission not started on channel A |
| [5] | TBFB | 0 | R/W | Transient Buffer Access Failure B<br>0 = No TBF B access failure<br>1 = TBF B access failure |
| [4] | TBFA | 0 | R/W | Transient Buffer Access Failure A<br>0 = No TBF A access failure<br>1 = TBF A access failure |
| [3] | FNFB | 0 | R/W | Find Sequence Not Finished Channel B<br>0 = No find sequence not finished for channel B<br>1 = Find sequence not finished for channel B |
| [2] | FNFA | 0 | R/W | Find Sequence Not Finished Channel A<br>0 = No find sequence not finished for channel A<br>1 = Find sequence not finished for channel A |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [1] | SNUB | 0 | R/W | Status Not Updated Channel B<br>0 = No overload condition occurred when updating MBS for channel B<br>1 = MBS for channel B not updated |
| [0] | SNUA | 0 | R/W | Status Not Updated Channel A<br>0 = No overload condition occurred when updating MBS for channel A<br>1 = MBS for channel A not updated |

(1) **FRMHDF.WAHP**

Write Attempt to Header Partition

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

Outside DEFAULT_CONFIG and CONFIG state this flag is set by the CC when the message handler tries to write message data into the header partition of the Message RAM due to faulty configuration of a message buffer. The write attempt is not executed, to protect the header partition from unintended write accesses.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(2) **FRMHDF.TNSB**

Transmission Not Started Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel B at the action point of the configured slot.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(3) **FRMHDF.TNSA**

Transmission Not Started Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel A at the action point of the configured slot.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(4) **FRMHDF.TBFB**

Transient Buffer Access Failure B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when a read or write access to TBF B requested by PRT B could not complete within the available time.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(5) **FRMHDF.TBFA**

Transient Buffer Access Failure A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when a read or write access to TBF A requested by PRT A could not complete within the available time.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(6) **FRMHDF.FNFB**

Find Sequence Not Finished Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching message buffer) with respect to channel B.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(7) **FRMHDF.FNFA**

Find Sequence Not Finished Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching message buffer) with respect to channel A.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(8) **FRMHDF.SNUB**

Status Not Updated Channel B

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Message Handler, due to overload condition, was not able to update a message buffer’s status MBS with respect to channel B.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

(9) **FRMHDF.SNUA**

Status Not Updated Channel A

Writing '0' has no effect on the bit value.

This bit is cleared when writing a '1' to it.

This flag is set by the CC when the Message Handler, due to overload condition, was not able to update a message buffer’s status MBS with respect to channel A.

When this flag changes from '0' to '1', interrupt flag **FREIR.MHF** is set to '1'.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

### 103.2.8.5 FRTXRQi (i=1 to 4) [h'0320 – h'032C]

FlexRay Transmission Request i (i=1 to 4)

The four registers reflect the state of the **TXR** flags of all configured message buffers. The flags are evaluated for transmit buffers only. If the number of configured message buffers is less than 128, the remaining **TXR** flags have no meaning.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TXRo[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TXRo[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:0] | TXRo | 0 | R | Transmission Request o |

(1) **FRTXRQi.TXRo (o=(i-1)*32 to (i*32-1))**

Transmission Request o

If the flag is set, the respective message buffer is ready for transmission respectively transmission of this message buffer is in progress.

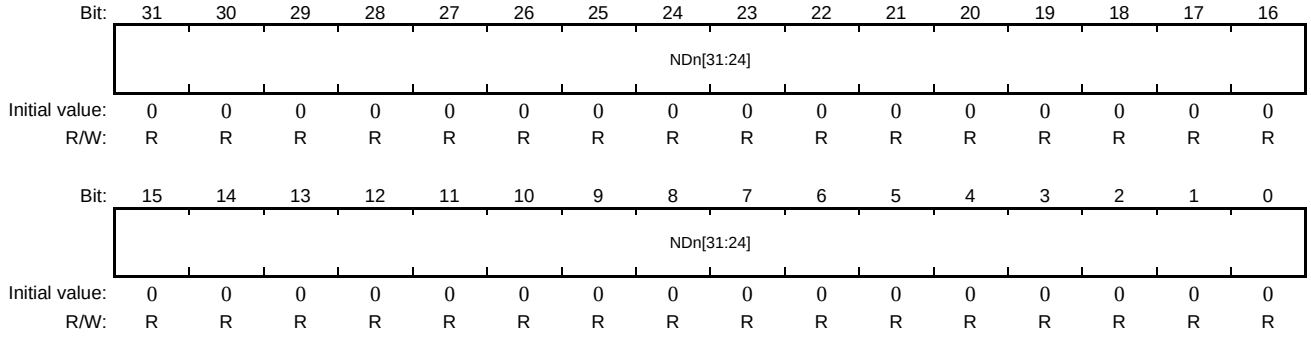
In single-shot mode the flags are reset after transmission has completed.

Reset by the CHI command CLEAR_RAMS.

### 103.2.8.6 FRNDATi (i=1 to 4) [h'0330 – h'033C]

FlexRay New Data i (i=1 to 4)

The four registers reflect the state of the **ND** flags of all configured message buffers. **ND** flags belonging to transmit buffers have no meaning. If the number of configured message buffers is less than 128, the remaining **ND** flags have no meaning.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:0] | NDn | 0 | R | New Data n |

(1) **FRNDATi.NDn (n=(i-1)*32 to (i*32-1))**

New Data n

The flags are set when a valid received data frame matches the message buffer's filter configuration, independent of the payload length received or the payload length configured for that message buffer.

The flags are not set after reception of null frames except for message buffers belonging to the receive FIFO.

An **ND** flag is reset when the header section of the corresponding message buffer is reconfigured or when the data section has been transferred to the Output Buffer.

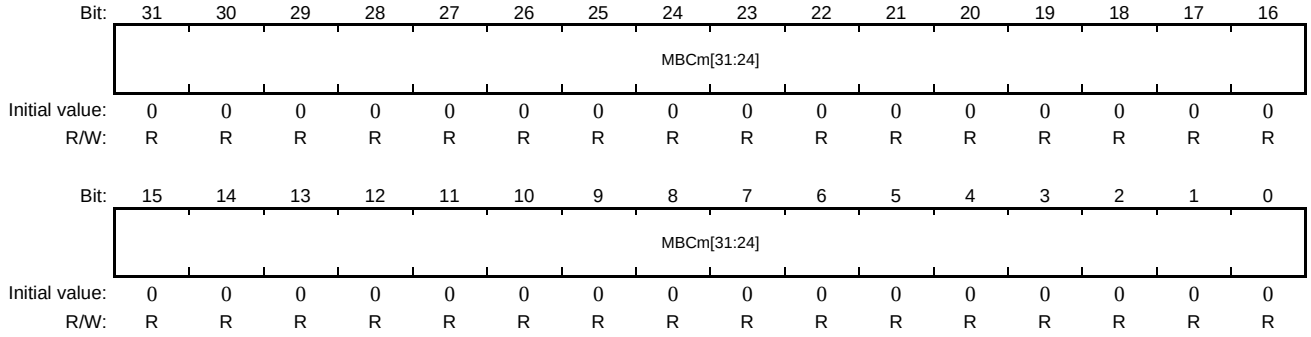
Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

### 103.2.8.7 FRMBSCi (i=1 to 4) [h'0340 – h'034C]

FlexRay Message Buffer Status Changed i (i=1 to 4)

The four registers reflect the state of the **MBC** flags of all configured message buffers. If the number of configured message buffers is less than 128, the remaining **MBC** flags have no meaning.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:0] | MBCm | 0 | R | Message Buffer Status Changed m |

(1) **FRMBSCi.MBCm (m=(i-1)*32 to (i*32-1))**

Message Buffer Status Changed m

An **MBC** flag is set whenever the Message Handler changes one of the status flags **VFRA**, **VFRB**, **SEOA**, **SEOB**, **CEOA**, **CEOB**, **SVOA**, **SVOB**, **TCIA**, **TCIB**, **ESA**, **ESB**, **MLST**, **FTA**, **FTB** in the header section (see section 103.2.11.5 and section 103.3.13.1 of the respective message buffer.

An **MBC** flag is reset when the header section of the corresponding message buffer is reconfigured or when it has been transferred to the Output Buffer.

Reset when leaving CONFIG state or when entering STARTUP state.

Reset by the CHI command CLEAR_RAMS.

### 103.2.9 Reserved

### 103.2.10 Input Buffer

Double buffer structure consisting of Input Buffer Host and Input Buffer Shadow. While the Host can write to Input Buffer Host, the transfer to the Message RAM is done from Input Buffer Shadow. The Input Buffer holds the header and data sections to be transferred to the selected message buffer in the Message RAM. It is used to configure the message buffers in the Message RAM and to update the data sections of transmit buffers.

When updating the header section of a message buffer in the Message RAM from the Input Buffer, the Message Buffer Status as described in section 103.2.11.5 is automatically reset to zero.

The header sections of message buffers belonging to the receive FIFO can only be (re)configured when the CC is in DEFAULT_CONFIG or CONFIG state. For those message buffers only the payload length configured and the data pointer need to be configured via **FRWRHS2.PLC** and **FRWRHS3.DP**. All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask.

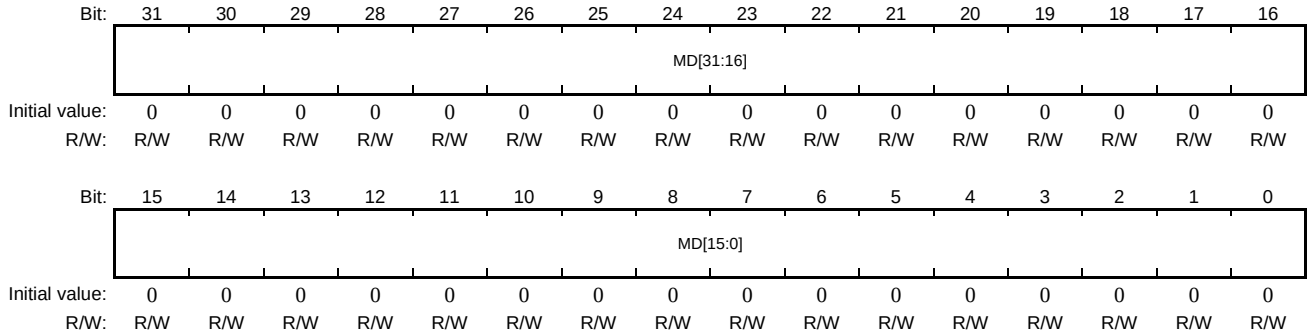
The data transfer between Input Buffer (IBF) and Message RAM is described in detail in section 103.3.12.2.(1).

When using the input data transfer function as described in section 103.3.16.1 and **FRITS.ITS** is '1', the user should not write to these registers.

#### 103.2.10.1 FRWRDSn (n=1 to 64) [h'0400 – h'04FC]

FlexRay Write Data Section n (n=1 to 64)

Holds the data words to be transferred to the data section of the addressed message buffer. The number of data words written to the Message RAM is defined by the payload length configured in **FRWRHS2.PLC**.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:0] | MD | 0 | R/W | Message Data |

(1) **FRWRDSn.MD**

Message Data

For information about the byte alignment of the Message Data in this register see section 103.3.17.

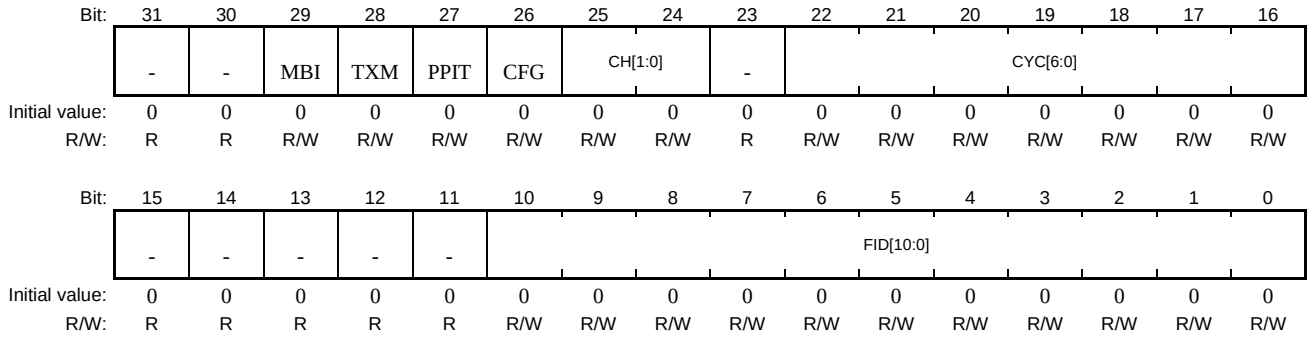
**Note 1:** In case **FRWRHS2.PLC** specifies an odd payload length, the remaining message data bytes are unused.

**Note 2:** When writing to **FRWRDSn**, each 32-bit word has to be filled up by one 32-bit access OR two consecutive 16-bit accesses OR four consecutive 8- bit accesses before the transfer from the Input Buffer to the Message RAM is started. If not all bytes of a 32- bit word have been written by the Host (8/16-bit access only), **FRWRDSn** holds partly undefined data.

Reset by the CHI command CLEAR_RAMS.

### 103.2.10.2 FRWRHS1 [h'0500]

FlexRay Write Header Section 1




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29] | MBI | 0 | R/W | Message Buffer Interrupt<br>0 = Message buffer interrupt is disabled<br>1 = Message buffer interrupt is enabled |
| [28] | TXM | 0 | R/W | Transmission Mode<br>0 = Continuous mode<br>1 = Single-shot mode |
| [27] | PPIT | 0 | R/W | Payload Preamble Indicator Transmit<br>0 = Payload Preamble Indicator not set<br>1 = Payload Preamble Indicator set |
| [26] | CFG | 0 | R/W | Message Buffer Direction Configuration Bit<br>0 = Message buffer is configured as Receive Buffer<br>1 = Message buffer is configured as Transmit Buffer |
| [25:24] | CH | 0 | R/W | Channel Filter Control |
| [23] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [22:16] | CYC | 0 | R/W | Cycle Code |
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | FID | 0 | R/W | Frame ID |

(1) **FRWRHS1.MBI**

Message Buffer Interrupt

This bit enables the receive / transmit interrupt for the corresponding message buffer.

After a dedicated receive buffer has been updated by the Message Handler, flag **FRSIR.RXI** and /or **FRSIR.MBSI** are set. After a transmission has completed flag **FRSIR.TXI** is set.

(2) **FRWRHS1.TXM**

Transmission Mode

This bit is used to select the transmission mode (see section 103.3.9.3) for the corresponding message buffer.

(3) **FRWRHS1.PPIT**

Payload Preamble Indicator Transmit

This bit is used to control the state of the Payload Preamble Indicator in transmit frames for the corresponding message buffer.

If the bit is set in a static message buffer, the respective message buffer holds network management information.

If the bit is set in a dynamic message buffer the first two bytes of the payload segment may be used for message ID filtering by the receiver. Message ID filtering of received FlexRay frames is not supported by the FlexRay module, but can be done by the Host.

(4) **FRWRHS1.CFG**

Message Buffer Direction Configuration Bit

This bit is used to configure the corresponding buffer as transmit buffer or as receive buffer. For message buffers belonging to the receive FIFO the bit is not evaluated.

(5) **FRWRHS1.CH**

Channel Filter Control

The 2-bit channel filtering field associated with each buffer serves as a filter for receive buffers, and as a control field for transmit buffers.

| CH[1:0] | Transmit Buffer<br>transmit frame on | Receive Buffer<br>store frame received from |
|---|---|---|
| 00 | no transmission | ignore frame |
| 01 | channel A | channel A |
| 10 | channel B | channel B |
| 11 | both channels (static segment only) | channel A or B<br>(store first semantically valid frame; static segment only) |

**Note:** If a message buffer is configured for the dynamic segment and both bits of the channel filtering field are set to ’1’, no frames are transmitted resp. received frames are ignored (same function as **CH** = "00")

(6) **FRWRHS1.CYC**

Cycle Code

The 7-bit cycle code determines the cycle set used for cycle counter filtering.

For details about the configuration of the cycle code section 103.3.8.2.

(7) **FRWRHS1.FID**

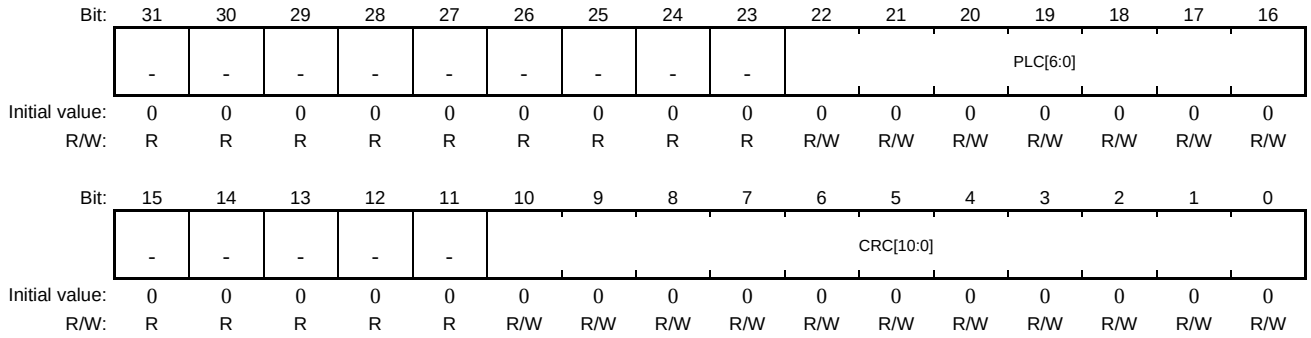
Frame ID

Frame ID of the selected message buffer. The frame ID defines the slot number for transmission / reception of the respective message.

Message buffers with frame ID = ’0’ are considered as not valid.

### 103.2.10.3 FRWRHS2 [h'0504]

FlexRay Write Header Section 2




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:23] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [22:16] | PLC | 0 | R/W | Payload Length Configured |
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | CRC | 0 | R/W | Header CRC<br>Receive Buffer: Configuration not required<br>Transmit Buffer: Header CRC calculated and configured by the Host |

(1) **FRWRHS2.PLC**

Payload Length Configured

Length of data section (number of 2-byte words) as configured by the Host.

During static segment the static frame payload length as configured by **FRMHDC.SFDL** defines the payload length for all static frames. If the payload length configured by **FRWRHS2.PLC** is shorter than this value padding bytes are inserted to ensure that frames have proper physical length. The padding pattern is logical zero (see section 103.3.9.3).

(2) **FRWRHS2.CRC**

Header CRC

For receive message buffer configuration not required

For transmit message buffer the header CRC needs to be calculated and configured by the Host
For calculation of the header CRC the payload length of the frame send on the bus has to be considered. In static segment the payload length of all frames is configured by **FRMHDC.SFDL**.

### 103.2.10.4 FRWRHS3 [h'0508]

FlexRay Write Header Section 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | DP[10:0] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | DP | 0 | R/W | Data Pointer |

(1) **FRWRHS3.DP**

Data Pointer

Pointer to the first 32-bit word of the data section of the addressed message buffer in the Message RAM.

### 103.2.10.5 FRIBCM [h'0510]

FlexRay Input Buffer Command Mask

Configures how the message buffer in the Message RAM selected by register **FRIBCR** is updated. When IBF Host and IBF Shadow are swapped, also mask bits **FRIBCM.LHSH**, **FRIBCM.LDSH**, and **FRIBCM.STXRH** are swapped with bits **FRIBCM.LHSS**, **FRIBCM.LDSS**, and **FRIBCM.STXRS** to keep them attached to the respective Input Buffer transfer.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | STXRS | LDSS | LHSS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | STXRH | LDSH | LHSH |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:19] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [18] | STXRS | 0 | R/W | Set Transmission Request Shadow<br>0 = Reset TXR flag<br>1 = Set TXR flag, transmit buffer released for transmission (operation ongoing or finished) |
| [17] | LDSS | 0 | R/W | Load Data Section Shadow<br>0 = Data section is not updated<br>1 = Data section selected for transfer from Input Buffer to the Message RAM (transfer ongoing or finished) |
| [16] | LHSS | 0 | R/W | Load Header Section Shadow<br>0 = Header section is not updated<br>1 = Header section selected for transfer from Input Buffer to the Message RAM (transfer ongoing or finished) |
| [15:3] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [2] | STXRH | 0 | R/W | Set Transmission Request Host<br>0 = Reset TXR flag<br>1 = Set TXR flag, transmit buffer released for transmission |
| [1] | LDSH | 0 | R/W | Load Data Section Host<br>0 = Data section is not updated<br>1 = Data section selected for transfer from Input Buffer to the Message RAM |
| [0] | LHSH | 0 | R/W | Load Header Section Host<br>0 = Header section is not updated<br>1 = Header section selected for transfer from Input Buffer to the Message RAM |

(1) **FRIBCM.STXRS**

Set Transmission Request Shadow

(2) **FRIBCM.LDSS**

Load Data Section Shadow

(3) **FRIBCM.LHSS**

Load Header Section Shadow

(4) **FRIBCM.STXRH**

Set Transmission Request Host

If this bit is set to ’1’, the **TXR** flag for the selected message buffer is set in the **FRTXRQ1/2/3/4** registers to release the message buffer for transmission. In single-shot mode the flag is cleared by the CC after transmission has completed.

**TXR** is evaluated for transmit buffers only.

(5) **FRIBCM.LDSH**

Load Data Section Host

(6) **FRIBCM.LHSH**

Load Header Section Host

### 103.2.10.6 FRIBCR [h'0514]

FlexRay Input Buffer Command Request

When the Host writes the number of the target message buffer in the Message RAM to **FRIBCR.IBRH**,

IBF Host and IBF Shadow are swapped. In addition the message buffer numbers stored under **FRIBCR.IBRH** and **FRIBCR.IBRS** are also swapped (see section 103.3.12.2.(1)).

With this write operation the **FRIBCR.IBSYS** is set to '1'. The Message Handler then starts to transfer the contents of IBF Shadow to the message buffer in the Message RAM selected by **FRIBCR.IBRS**.

While the Message Handler transfers the data from IBF Shadow to the target message buffer in the Message RAM, the Host may write the next message into the IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, **FRIBCR.IBSYS** is set back to '0' and the next transfer to the Message RAM may be started by the Host by writing the respective target message buffer number to **FRIBCR.IBRH**.

If a write access to **FRIBCR.IBRH** occurs while **FRIBCR.IBSYS** is '1', **FRIBCR.IBSYH** is set to '1'. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, **FRIBCR.IBSYH** is reset to '0'. **FRIBCR.IBSYS** remains set to '1', and the next transfer to the Message RAM is started. In addition the message buffer numbers stored under **FRIBCR.IBRH** and **FRIBCR.IBRS** are also swapped.

Any write access to an Input Buffer register while both **FRIBCR.IBSYS** and **FRIBCR.IBSYH** are set will cause the error flag **FREIR.IIBA** to be set. In this case the write access has no effect and the Input Buffer will not be changed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IBSYS | - | - | - | - | - | - | - | - | IBRS[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IBSYH | - | - | - | - | - | - | - | - | IBRH[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31] | IBSYS | 0 | R | Input Buffer Busy Shadow<br>0 = Transfer between IBF Shadow and Message RAM completed<br>1 = Transfer between IBF Shadow and Message RAM in progress |
| [30:23] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [22:16] | IBRS | 0 | R | Input Buffer Request Shadow |
| [15] | IBSYH | 0 | R | Input Buffer Busy Host<br>0 = No request pending<br>1 = Request while transfer between IBF Shadow and Message RAM in progress |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [14:7] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [6:0] | IBRH | 0 | R/W | Input Buffer Request Host |

(1) **FRIBCR.IBSYS**

Input Buffer Busy Shadow

Set to ’1’ after writing **FRIBCR.IBRH**.

This bit indicates that a transfer between IBF Shadow and Message RAM is in progress.

When the transfer between IBF Shadow and the Message RAM has completed, **FRIBCR.IBSYS** is set back to ’0’.

(2) **FRIBCR.IBRS**

Input Buffer Request Shadow

Number of the target message buffer actually updated / lately updated.

(3) **FRIBCR.IBSYH**

Input Buffer Busy Host

Set to ’1’ by writing **FRIBCR.IBRH** while **FRIBCR.IBSYS** is still ’1’.

This bit indicates that a transfer between IBF Shadow and Message RAM is in progress

After the ongoing transfer between IBF Shadow and the Message RAM has completed, the **FRIBCR.IBSYH** is set back to ’0’.

(4) **FRIBCR.IBRH**

Input Buffer Request Host

Selects the target message buffer in the Message RAM for data transfer from Input Buffer.

### 103.2.11 Output Buffer

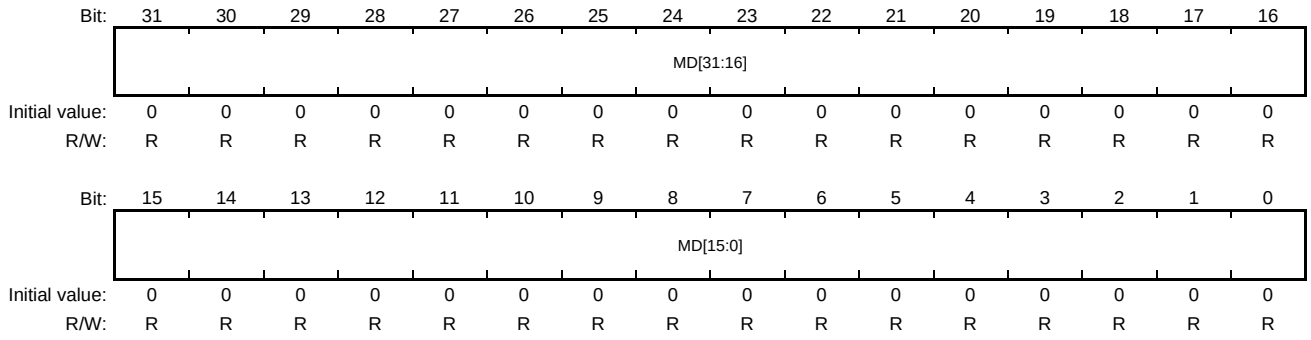
Double buffer structure consisting of Output Buffer Host and Output Buffer Shadow. Used to read out message buffers from the Message RAM. While the Host can read from Output Buffer Host, the Message Handler transfers the selected message buffer from Message RAM to Output Buffer Shadow. The data transfer between Message RAM and Output Buffer (OBF) is described in section 103.3.12.2.(2).

When using the output data transfer function as described in section 103.3.16.2 and **FROTS.OTS** is '1', the user should not write to these registers.

#### 103.2.11.1 FRRDDSn (n=1 to 64) [h'0600 – h'06FC]

FlexRay Read Data Section n (n=1 to 64)

Holds the data words read from the data section of the addressed message buffer. The number of data words read from the Message RAM is defined by the payload length configured in **FRRDHS2.PLC**.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:0] | MD | 0 | R | Message Data |

(1) **FRRDDSn.MD**

Message Data

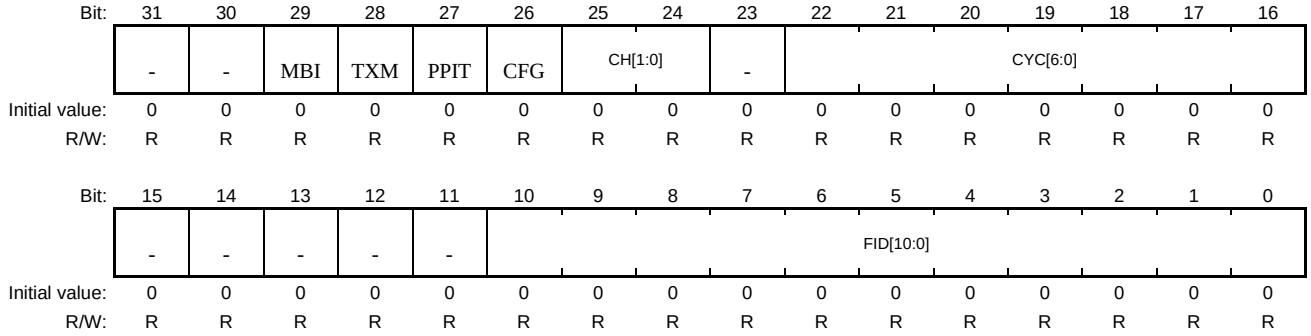
For information about the byte alignment of the Message Data in this register see section 103.3.17.

**Note:** In case **FRRDHS2.PLC** specifies an odd payload length, the remaining message data bytes are undefined.

Reset by the CHI command CLEAR_RAMS.

### 103.2.11.2 FRRDHS1 [h'0700]

FlexRay Read Header Section 1




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29] | MBI | 0 | R | Message Buffer Interrupt |
| [28] | TXM | 0 | R | Transmission Mode |
| [27] | PPIT | 0 | R | Payload Preamble Indicator Transmit |
| [26] | CFG | 0 | R | Message Buffer Direction Configuration Bit |
| [25:24] | CH | 0 | R | Channel Filter Control |
| [23] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [22:16] | CYC | 0 | R | Cycle Code |
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | FID | 0 | R | Frame ID |

(1) **FRRDHS1.MBI**

Message Buffer Interrupt

Value as configured by the Host via **FRWRHS1.MBI**.

In case that the message buffer read from the Message RAM belongs to the receive FIFO this bit is set to '0'.

(2) **FRRDHS1.TXM**

Transmission Mode

Value as configured by the Host via **FRWRHS1.TXM**.

In case that the message buffer read from the Message RAM belongs to the receive FIFO this bit is set to '0'.

(3) **FRRDHS1.PPIT**

Payload Preamble Indicator Transmit

Value as configured by the Host via **FRWRHS1.PPIT**.

In case that the message buffer read from the Message RAM belongs to the receive FIFO this bit is set to '0'.

(4) **FRRDHS1.CFG**

Message Buffer Direction Configuration Bit

Value as configured by the Host via **FRWRHS1.CFG**.

In case that the message buffer read from the Message RAM belongs to the receive FIFO this bit is set to '0'.

(5) **FRRDHS1.CH**

Channel Filter Control

Value as configured by the Host via **FRWRHS1.CH**.

In case that the message buffer read from the Message RAM belongs to the receive FIFO these bits are set to '0'.

(6) **FRRDHS1.CYC**

Cycle Code

Value as configured by the Host via **FRWRHS1.CYC**.

In case that the message buffer read from the Message RAM belongs to the receive FIFO these bits are set to '0'.

(7) **FRRDHS1.FID**

Frame ID

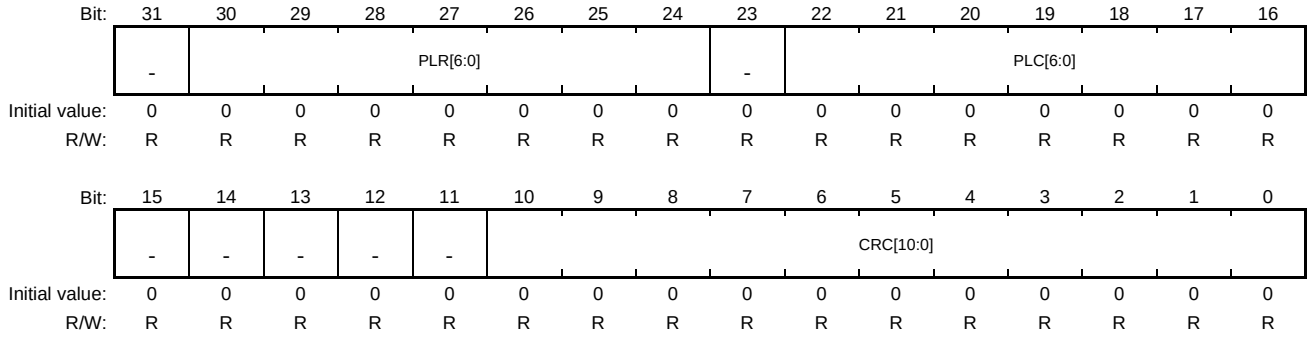
Value as configured by the Host via **FRWRHS1.FID**

In case that the message buffer read from the Message RAM belongs to the receive FIFO these bits are holding the received frame ID.

### 103.2.11.3 FRRDHS2 [h'0704]

FlexRay Read Header Section 2

**Note:** For message buffers belonging to the Static Buffer area or Static and Dynamic Buffer area **FRWRHS2** is updated from data frames only.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [30:24] | PLR | 0 | R | Payload Length Received<br>vRF!Header!Length |
| [23] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [22:16] | PLC | 0 | R | Payload Length Configured |
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | CRC | 0 | R | Header CRC<br>vRF!Header!HeaderCRC |

(1) **FRRDHS2.PLR**

Payload Length Received

Payload length value updated from received data frames (vRF!Header!Length) (exception: if message buffer belongs to the receive FIFO **PLR** is also updated from received null frames).

(2) **FRRDHS2.PLC**

Payload Length Configured

Length of data section (number of 2-byte words) as configured by the Host.

(3) **FRRDHS2.CRC**

Header CRC

Receive Buffer: Header CRC (vRF!Header!HeaderCRC) updated from received data frames

Transmit Buffer: Header CRC calculated and configured by the Host

(4) **Data storage**

When a message is stored into a message buffer the following behaviour with respect to payload length received and payload length configured is implemented:

**FRRDHS2.PLR** > **FRRDHS2.PLC**:
The payload data stored in the message buffer is truncated to the payload length configured if **FRRDHS2.PLC** even or else truncated to **FRRDHS2.PLC** + 1.

**FRRDHS2.PLR** <= **FRRDHS2.PLC**:
The received payload data is stored into the message buffers data section. The remaining data bytes of the data section as configured by **FRRDHS2.PLC** are filled with undefined data.

**FRRDHS2.PLR** = zero:
The message buffer's data section is filled with undefined data

**FRRDHS2.PLC** = zero:
Message buffer has no data section configured. No data is stored into the message buffer's data section.

**Note 1:** The Message RAM is organized in 4-byte words. When received data is stored into a message buffer's data section, the number of 2-byte data words written into the message buffer is **FRRDHS2.PLC** rounded to the next even value.

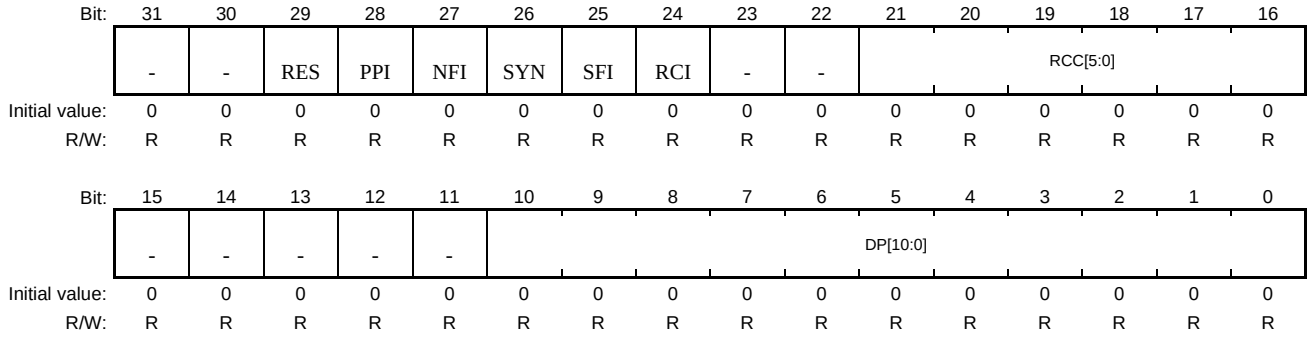
**Note 2:** **FRRDHS2.PLC** should be configured identical for all message buffers belonging to the receive FIFO.

For message buffers belonging to the Static Buffer area or Static and Dynamic Buffer area **FRWRHS2** is updated from data frames only.

### 103.2.11.4 FRRDHS3 [h'0708]

FlexRay Read Header Section 3

**Note:** For message buffers belonging to the Static Buffer area or Static and Dynamic Buffer area **FRWRHS3** is updated from data frames only.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29] | RES | 0 | R | Reserved Bit<br>vRF!Header!Reserved |
| [28] | PPI | 0 | R | Payload Preamble Indicator<br>vRF!Header!PPIndicator |
| [27] | NFI | 0 | R | Null Frame Indicator<br>vRF!Header!NFIndicator<br>0 = Up to now no data frame has been stored into the respective message buffer<br>1 = At least one data frame has been stored into the respective message buffer |
| [26] | SYN | 0 | R | Sync Frame Indicator<br>vRF!Header!SyFIndicator<br>0 = The received frame is not a sync frame<br>1 = The received frame is a sync frame |
| [25] | SFI | 0 | R | Startup Frame Indicator<br>vRF!Header!SuFIndicator<br>0 = The received frame is not a startup frame<br>1 = The received frame is a startup frame |
| [24] | RCI | 0 | R | Received on Channel Indicator<br>vSS!Channel<br>0 = Frame received on channel B<br>1 = Frame received on channel A |
| [23:22] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [21:16] | RCC | 0 | R | Receive Cycle Count<br>vRF!Header!CycleCount |
| [15:11] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [10:0] | DP | 0 | R | Data Pointer |

(1) **FRRDHS3.RES**

Reserved Bit

Reflects the state of the received reserved bit (vRF!Header!Reserved). The reserved bit is transmitted as ’0’.

(2) **FRRDHS3.PPI**

Payload Preamble Indicator

The payload preamble indicator (vRF!Header!PPIndicator) defines whether a network management vector or message ID is contained within the payload segment of the received frame.

0 = PPI indicator set to '0'
The payload segment of the received frame does not contain a network management vector nor a message ID

1 = PPI indicator set to '1'
Static segment: Network management vector in the first part of the payload
Dynamic segment: Message ID in the first part of the payload

(3) **FRRDHS3.NFI**

Null Frame Indicator

The Null Frame Indicator (vRF!Header!NFIndicator) is set to ’1’ after storage of the first received data frame.

(4) **FRRDHS3.SYN**

Sync Frame Indicator

A sync frame is marked by the sync frame indicator (vRF!Header!SyFIndicator).

(5) **FRRDHS3.SFI**

Startup Frame Indicator

A startup frame is marked by the startup frame indicator (vRF!Header!SuFIndicator).

(6) **FRRDHS3.RCI**

Received on Channel Indicator

Indicates the channel from which the received data frame was taken to update the respective receive buffer (vSS!Channel).

(7) **FRRDHS3.RCC**

Receive Cycle Count

Cycle counter value updated from received data frame (vRF!Header!CycleCount).

(8) **FRRDHS3.DP**

Data Pointer

Pointer to the first 32-bit word of the data section of the addressed message buffer in the Message RAM.

Value as configured by the Host via **FRWRHS3.DP**.

### 103.2.11.5 FRMBS [h'070C]

FlexRay Message Buffer Status

The message buffer status is updated by the CC with respect to the assigned channel(s) latest at the end of the slot following the slot assigned to the message buffer.

The flags are updated only when the CC is in NORMAL_ACTIVE or NORMAL_PASSIVE state.

If only one channel (A or B) is assigned to a message buffer, the channel-specific status flags of the other channel are written to zero. If both channels are assigned to a message buffer, the channel-specific status flags of both channels are updated.

The message buffer status is updated only when the slot counter reached the configured frame ID and when the cycle counter filter matched. When the Host updates a message buffer via Input Buffer, all **FRMBS** flags are reset to zero independent of which **FRIBCM** bits are set or not.

For details about receive / transmit filtering see section 103.3.8, section 103.3.9 and section 103.3.10.

Whenever the Message Handler changes one of the flags **FRMBS.VFRA**, **FRMBS.VFRB**, **FRMBS.SEOA**, **FRMBS.SEOB**, **FRMBS.CEOA**, **FRMBS.CEOB**, **FRMBS.SVOA**, **FRMBS.SVOB**, **FRMBS.TCIA**, **FRMBS.TCIB**, **FRMBS.ESA**, **FRMBS.ESB**, **FRMBS.MLST**, **FRMBS.FTA**, **FRMBS.FTB** the respective message buffer's **MBC** flag in registers **FRMBSC1/2/3/4** is set.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | RESS | PPIS | NFIS | SYNS | SFIS | RCIS | - | - | CCS[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FTB | FTA | - | MLST | ESB | ESA | TCIB | TCIA | SVOB | SVOA | CEOB | CEOA | SEOB | SEOA | VFRB | VFRA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29] | RESS | 0 | R | Reserved Bit Status<br>vRF!Header!Reserved |
| [28] | PPIS | 0 | R | Payload Preamble Indicator Status<br>0 = PPI indicator set to '0'<br>1 = PPI indicator set to '1' |
| [27] | NFIS | 0 | R | Null Frame Indicator Status<br>0 = Received frame is a null frame<br>1 = Received frame is not a null frame |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [26] | SYNS | 0 | R | Sync Frame Indicator Status<br>0 = No sync frame received<br>1 = The received frame is a sync frame |
| [25] | SFIS | 0 | R | Startup Frame Indicator Status<br>0 = No startup frame received<br>1 = The received frame is a startup frame |
| [24] | RCIS | 0 | R | Received on Channel Indicator Status<br>0 = Frame received on channel B<br>1 = Frame received on channel A |
| [23:22] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [21:16] | CCS | 0 | R | Cycle Count Status |
| [15] | FTB | 0 | R | Frame Transmitted on Channel B<br>0 = No data frame transmitted on channel B<br>1 = Data frame transmitted on channel B |
| [14] | FTA | 0 | R | Frame Transmitted on Channel A<br>0 = No data frame transmitted on channel A<br>1 = Data frame transmitted on channel A |
| [13] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [12] | MLST | 0 | R | Message Lost<br>0 = No message lost<br>1 = Unprocessed message was overwritten |
| [11] | ESB | 0 | R | Empty Slot Channel B<br>0 = Bus activity detected in the assigned slot on channel B<br>1 = No bus activity detected in the assigned slot on channel B |
| [10] | ESA | 0 | R | Empty Slot Channel A<br>0 = Bus activity detected in the assigned slot on channel A<br>1 = No bus activity detected in the assigned slot on channel A |
| [9] | TCIB | 0 | R | Transmission Conflict Indication Channel B<br>0 = No transmission conflict occurred on channel B<br>1 = Transmission conflict occurred on channel B |
| [8] | TCIA | 0 | R | Transmission Conflict Indication Channel A<br>0 = No transmission conflict occurred on channel A<br>1 = Transmission conflict occurred on channel A |
| [7] | SVOB | 0 | R | Slot Boundary Violation Observed on Channel B<br>0 = No slot boundary violation observed on channel B<br>1 = Slot boundary violation observed on channel B |
| [6] | SVOA | 0 | R | Slot Boundary Violation Observed on Channel A<br>0 = No slot boundary violation observed on channel A<br>1 = Slot boundary violation observed on channel A |
| [5] | CEOB | 0 | R | Content Error Observed on Channel B<br>0 = No content error observed on channel B<br>1 = Content error observed on channel B |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [4] | CEOA | 0 | R | Content Error Observed on Channel A<br>0 = No content error observed on channel A<br>1 = Content error observed on channel A |
| [3] | SEOB | 0 | R | Syntax Error Observed on Channel B<br>0 = No syntax error observed on channel B<br>1 = Syntax error observed on channel B |
| [2] | SEOA | 0 | R | Syntax Error Observed on Channel A<br>0 = No syntax error observed on channel A<br>1 = Syntax error observed on channel A |
| [1] | VFRB | 0 | R | Valid Frame Received on Channel B<br>0 = No valid frame received on channel B<br>1 = Valid frame received on channel B |
| [0] | VFRA | 0 | R | Valid Frame Received on Channel A<br>0 = No valid frame received on channel A<br>1 = Valid frame received on channel A |

(1) **FRMBS.RESS**

Reserved Bit Status

Reflects the state of the received reserved bit (vRF!Header!Reserved). The reserved bit is transmitted as ’0’.

For receive buffers (**FRWRHS1.CFG** = ’0’) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

(2) **FRMBS.PPIS**

Payload Preamble Indicator Status

The payload preamble indicator (vRF!Header!PPIndicator) defines whether a network management vector or message ID is contained within the payload segment of the received frame.

For receive buffers (**FRWRHS1.CFG** = ’0’) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

0 = PPI indicator set to '0'
The payload segment of the received frame does not contain a network management vector or a message ID

1 = PPI indicator set to '1'
Dynamic segment: Message ID at the beginning of the payload
Static segment: Network management vector at the beginning of the payload

(3) **FRMBS.NFIS**

Null Frame Indicator Status

If set to ’0’ the PPI bit (vRF!Header!NFIndicator) was set to '0' and the payload segment of the received frame contains no usable data.

For receive buffers (**FRWRHS1.CFG** = ’0’) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

(4) **FRMBS.SYNS**

Sync Frame Indicator Status

A sync frame is marked by the sync frame indicator (vRF!Header!SyFIndicator).

For receive buffers (**FRWRHS1.CFG** = ’0’) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

(5) **FRMBS.SFIS**

Startup Frame Indicator Status

A startup frame is marked by the startup frame indicator (vRF!Header!SuFIndicator).

For receive buffers (**FRWRHS1.CFG** = ’0’) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

(6) **FRMBS.RCIS**

Received on Channel Indicator Status

Indicates the channel on which the frame was received (vSS!Channel).

For receive buffers (**FRWRHS1.CFG** = ’0’) this bit is updated from both valid data and null frames. If no valid frame was received, the previous value is maintained. For transmit buffers the flag has no meaning and should be ignored.

(7) **FRMBS.CCS**

Cycle Count Status

Actual cycle count when status was updated.

(8) **FRMBS.FTB**

Frame Transmitted on Channel B

Indicates that this node has transmitted a data frame in the configured slot on channel B.

**Note:** The FlexRay protocol specification requires that **FRMBS.FTB** can only be reset by the Host. Therefore the Cycle Count Status **FRMBS.CCS** for this bit is only valid for the cycle where the bit is set to ’1’.

(9) **FRMBS.FTA**

Frame Transmitted on Channel A

Indicates that this node has transmitted a data frame in the configured slot on channel A.

**Note:** The FlexRay protocol specification requires that **FRMBS.FTA** can only be reset by the Host. Therefore the Cycle Count Status **FRMBS.CCS** for this bit is only valid for the cycle where this bit is set to ’1’.

0 = No data frame transmitted on channel A

(10) **FRMBS.MLST**

Message Lost

The flag is set in case the Host did not read the message before the message buffer was updated from a received data frame.

Not affected by reception of null frames except for message buffers belonging to the receive FIFO. The flag is reset by a Host write to the message buffer via IBF or when a new message is stored into the message buffer after the message buffers **ND** flag was reset by reading out the message buffer via OBF.

(11) **FRMBS.ESB**

Empty Slot Channel B

In an empty slot there is no activity detected on the bus. The condition is checked in static and dynamic slots.

(12) **FRMBS.ESA**

Empty Slot Channel A

In an empty slot there is no activity detected on the bus. The condition is checked in static and dynamic slots.

(13) **FRMBS.TCIB**

Transmission Conflict Indication Channel B

A transmission conflict indication (vSS!TxConflictB) is set if a transmission conflict has occurred on channel B.

(14) **FRMBS.TCIA**

Transmission Conflict Indication Channel A

A transmission conflict indication (vSS!TxConflictA) is set if a transmission conflict has occurred on channel A.

(15) **FRMBS.SVOB**

Slot Boundary Violation Observed on Channel B

A slot boundary violation (channel active at the start or at the end of the assigned slot) (vSS!BViolationB) was observed on channel B.

(16) **FRMBS.SVOA**

Slot Boundary Violation Observed on Channel A

A slot boundary violation (channel active at the start or at the end of the assigned slot) (vSS!BViolationA) was observed on channel A.

(17) **FRMBS.CEOB**

Content Error Observed on Channel B

A content error (vSS!ContentErrorB) was observed in the assigned slot on channel B.

(18) **FRMBS.CEOA**

Content Error Observed on Channel A

A content error (vSS!ContentErrorA) was observed in the assigned slot on channel A.

(19) **FRMBS.SEOB**

Syntax Error Observed on Channel B

A syntax error (vSS!SyntaxErrorB) was observed in the assigned slot on channel B.

(20) **FRMBS.SEOA**

Syntax Error Observed on Channel A

A syntax error was (vSS!SyntaxErrorA) observed in the assigned slot on channel A.

(21) **FRMBS.VFRB**

Valid Frame Received on Channel B

A valid frame indication (vSS!ValidFrameB) is set if a valid frame was received on channel B.

(22) **FRMBS.VFRA**

Valid Frame Received on Channel A

A valid frame indication (vSS!ValidFrameA) is set if a valid frame was received on channel A.

### 103.2.11.6 FROBCM [h'0710]

FlexRay Output Buffer Command Mask

Configures how the Output Buffer is updated from the message buffer in the Message RAM selected by **FROBCR.OBRS**.

Mask bits **FROBCM.RDSS** and **FROBCM.RHSS** are copied to the register internal storage when a Message RAM transfer is requested by **FROBCR.REQ**.

When OBF Host and OBF Shadow are swapped, mask bits **FROBCM.RDSH** and **FROBCM.RHSH** are swapped with the register internal storage to keep them attached to the respective Output Buffer transfer.

The data transfer between Output Buffer and Message RAM is described in detail in section 103.3.12.2.(2).

**Note:** After the transfer of the header section from the Message RAM to OBF Shadow has completed, the message buffer status changed flag **MBC** of the selected message buffer in the **FRMBSC1/2/3/4** registers is cleared. After the transfer of the data section from the Message RAM to OBF Shadow has completed, the new data flag **ND** of the selected message buffer in the **FRNDAT1/2/3/4** registers is cleared.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | RDSH | RHSH |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | RDSS | RHSS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:18] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [17] | RDSH | 0 | R | Read Data Section Host<br>0 = Data section is not read<br>1 = Data section selected for transfer from Message RAM to Output Buffer |
| [16] | RHSH | 0 | R | Read Header Section Host<br>0 = Header section is not read<br>1 = Header section selected for transfer from Message RAM to Output Buffer |
| [15:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1] | RDSS | 0 | R/W | Read Data Section Shadow<br>0 = Data section is not read<br>1 = Data section selected for transfer from Message RAM to Output Buffer |
| [0] | RHSS | 0 | R/W | Read Header Section Shadow<br>0 = Header section is not read<br>1 = Header section selected for transfer from Message RAM to Output Buffer |

(1) **FROBCM.RDSH**

Read Data Section Host

(2) **FROBCM.RHSH**

Read Header Section Host

(3) **FROBCM.RDSS**

Read Data Section Shadow

(4) **FROBCM.RHSS**

Read Header Section Shadow

### 103.2.11.7 FROBCR [h'0714]

FlexRay Output Buffer Command Request

After setting bit **FROBCR.REQ** to '1' while **FROBCR.OBSYS** is '0', **FROBCR.OBSYS** is automatically set to '1', **FROBCR.OBRS** is copied to the register internal storage, mask bits **FROBCM.RDSS** and **FROBCM.RHSS** are copied to register **FROBCM** internal storage, and the transfer of the message buffer selected by **FROBCR.OBRS** from the Message RAM to OBF Shadow is started. When the transfer between the Message RAM and OBF Shadow has completed, this is signaled by setting **FROBCR.OBSYS** back to '0'.

By setting bit **FROBCR.VIEW** to '1' while **FROBCR.OBSYS** is '0', OBF Host and OBF Shadow are swapped. Additionally mask bits **FROBCM.RDSH** and **FROBCM.RHSH** are swapped with the register **FROBCM** internal storage to keep them attached to the respective Output Buffer transfer. **FROBCR.OBRH** signals the number of the message buffer currently accessible by the Host.

If bits **FROBCR.REQ** and **FROBCR.VIEW** are set to '1' with the same write access while **FROBCR.OBSYS** is '0', **FROBCR.OBSYS** is automatically set to '1' and OBF Shadow and OBF Host are swapped. Additionally mask bits **FROBCM.RDSH** and **FROBCM.RHSH** are swapped with the registers internal storage to keep them attached to the respective Output Buffer transfer. Afterwards **FROBCR.OBRS** is copied to the register internal storage, and the transfer of the selected message buffer from the Message RAM to OBF Shadow is started. While the transfer is ongoing the Host can read the message buffer transferred by the previous transfer from OBF Host. When the current transfer between Message RAM and OBF Shadow has completed, this is signaled by setting **FROBCR.OBSYS** back to '0'.

Any write access to **FROBCR[15:8]** while **FROBCR.OBSYS** is set will cause the error flag **FREIR.IOBA** to be set. In this case the write access has no effect and the Output buffer will not be changed.

The data transfer between Output Buffer and Message RAM is described in detail in section 103.3.12.2.(2).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | OBRH[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OBSYS | - | - | - | - | - | REQ | VIEW | - | OBRS[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:23] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [22:16] | OBRH | 0 | R | Output Buffer Request Host |
| [15] | OBSYS | 0 | R | Output Buffer Busy Shadow<br>0 = No transfer in progress<br>1 = Transfer between Message RAM and OBF Shadow in progress |
| [14:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [9] | REQ | 0 | R/W | Request Message RAM Transfer<br>0 = No request<br>1 = Transfer to OBF Shadow requested |
| [8] | VIEW | 0 | R/W | View Shadow Buffer<br>0 = No action<br>1 = Swap OBF Shadow and OBF Host |
| [7] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [6:0] | OBRS | 0 | R/W | Output Buffer Request Shadow |

(1) **FROBCR.OBRH**

Output Buffer Request Host

Number of message buffer currently accessible by the Host via **FRRDHS1/2/3**, **FRMBS**, and **FRRDDSn (n=1 to 64)**.

By writing **FROBCR.VIEW** to ’1’ OBF Shadow and OBF Host are swapped and the transferred message buffer is accessible by the Host.

(2) **FROBCR.OBSYS**

Output Buffer Busy Shadow

Set to ’1’ after setting bit **FROBCR.REQ**. When the transfer between the Message RAM and OBF Shadow has completed, **FROBCR.OBSYS** is set back to ’0’.

(3) **FROBCR.REQ**

Request Message RAM Transfer

Only writeable while **FROBCR.OBSYS** = ’0’.

Requests transfer of message buffer addressed by **FROBCR.OBRS** from Message RAM to OBF Shadow.

(4) **FROBCR.VIEW**

View Shadow Buffer

Only writeable while **FROBCR.OBSYS** = ’0’.

Toggles between OBF Shadow and OBF Host.

(5) **FROBCR.OBRS**

Output Buffer Request Shadow

Only writeable while **FROBCR.OBSYS** = ’0’.

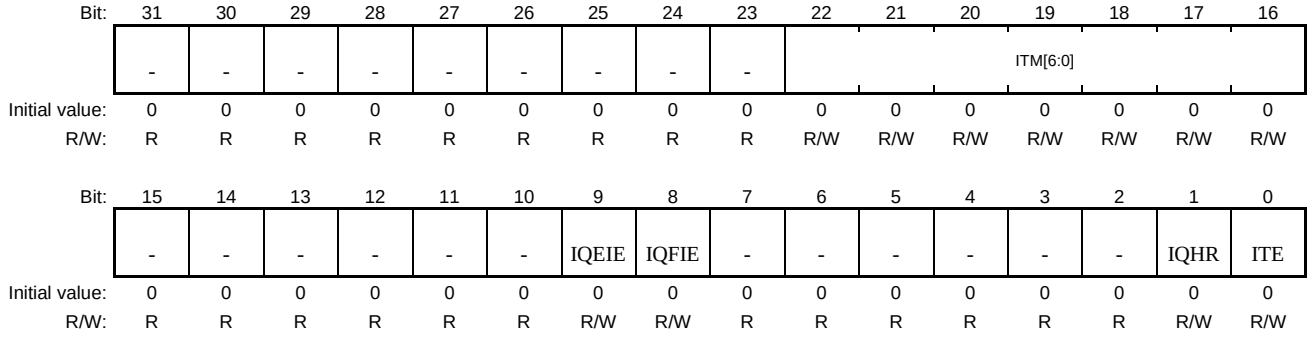
Number of source message buffer to be transferred from the Message RAM to OBF Shadow.

If the number of the first message buffer of the receive FIFO is written to this register the Message Handler transfers the message buffer addressed by the GET Index (GIDX, see section 103.3.11) to OBF Shadow.

### 103.2.12 Data transfer control register

#### 103.2.12.1 FRITC [h'0800]

FlexRay Input Transfer Configuration register




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:23] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [22:16] | ITM | 0 | R/W | Input queue Table Max<br>These bits configure the number of entries in the input pointer table the input buffer handler is capable to maintain in the input queue. |
| [15:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [9] | IQEIE | 0 | R/W | Input Queue Empty Interrupt Enable<br>0: Disabled<br>1: Enabled |
| [8] | IQFIE | 0 | R/W | Input Queue Full Interrupt Enable<br>0: Disabled<br>1: Enabled |
| [7:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1] | IQHR | 0 | R/W | Input Queue Halt Request<br>0: Input queue run request<br>1: Input queue halt request |
| [0] | ITE | 0 | R/W | Input Transfer Enable<br>0: Operation Disable request<br>1: Operation Enable request |

(1) **FRITC.ITM**

Input queue Table Max

The user can only write to this bit when **FRITS.ITS** is '0'.

These bits configure the number of entries in the input pointer table the input buffer handler is capable to maintain in the input queue.

Valid values are h'00 (1 queue entry) to h'7F (128 queue entries).

Note that each entry requires two long words in the input pointer table.

(2) **FRITC.IQEIE**

Input Queue Empty Interrupt Enable

This bit controls the input queue empty interrupt.

0: Disabled
No interrupt will be requested and the input queue empty interrupt line will be released if pending.

1: Enabled
Input queue empty interrupt will be asserted when **FRITS.IQEIS** is '1'.

(3) **FRITC.IQFIE**

Input Queue Full Interrupt Enable

This bit controls the input queue full interrupt.

0: Disabled
No interrupt will be requested and the input queue full interrupt line will be released if pending.

1: Enabled
Input queue full interrupt will be asserted when **FRITS.IQFIS** is '1'.

(4) **FRITC.IQHR**

Input Queue Halt Request

The **IQHR** bit should not be set to '1' when **FRITS.ITS** is '0'.

This bit requests a halt of the input queue.

The status of the halt request is shown in the **FRITS.IQH** register.
Refer to section 103.3.16.1.(5) about usage of this bit.

0: Input queue run request
The input queue resumes their operation.

1: Input queue halt request
The input queue gets halted. An active input transfer will be completed but no further transfer request will start.

(5) **FRITC.ITE**

Input Transfer Enable

The user should only set this bit to '1' when **FRIBCR.IBSYS** is '0'.

The user should only set this bit to '0' when **FRITC.IQHR** '0'. Otherwise committed input transfers get lost.

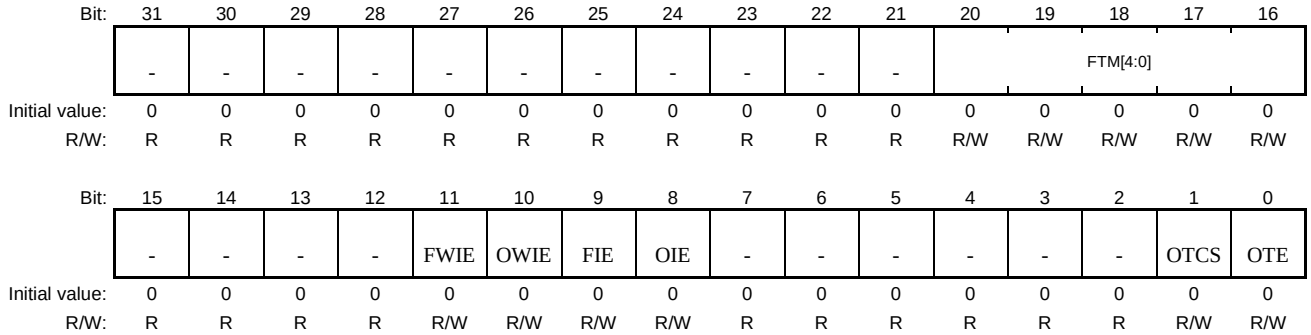
This bit controls the operation mode of the input transfer queue.
The operation status of the input transfer queue function is shown in **FRITS.ITS**.
Refer to section 103.3.16.1.(1) about usage of this bit.

0: Operation Disable request
The input transfer queue gets disabled when it becomes empty.

1: Operation Enable request
The input transfer queue gets enabled. Input data structures are transferred to the FlexRay internal message RAM.

### 103.2.12.2 FROTC [h'0804]

FlexRay Output Transfer Configuration register




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:21] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [20:16] | FTM | 0 | R/W | FIFO Table Max<br>Configures the number of FIFO entries the output transfer handler is capable to maintain in the URAM. |
| [15:12] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [11] | FWIE | 0 | R/W | FIFO transfer Warning Interrupt Enable<br>0: Disabled<br>1: Enabled |
| [10] | OWIE | 0 | R/W | Output transfer Warning Interrupt Enable<br>0: Disabled<br>1: Enabled |
| [9] | FIE | 0 | R/W | FIFO transfer Interrupt Enable<br>0: Disabled<br>1: Enabled |
| [8] | OIE | 0 | R/W | Output transfer Interrupt Enable<br>0: Disabled<br>1: Enabled |
| [7:2] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [1] | OTCS | 0 | R/W | Output Transfer Condition Select<br>0: New data only mode<br>1: New data and status changed mode |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [0] | OTE | 0 | R/W | Output Transfer Enable<br>0: Operation Disable request<br>1: Operation Enable request |

(1) **FROTC.FTM**

FIFO Table Max

The user can only write to these bits when **FROTS.OTS** is '0'.

Configures the number of FIFO entries the output transfer handler is capable to maintain in the URAM.

Valid values are h'00 (1 FIFO entry) to h'1F (32 FIFO entries).

(2) **FROTC.FWIE**

FIFO transfer Warning Interrupt Enable

This bit controls the FIFO transfer warning interrupt.

0: Disabled
No interrupt will be requested and the FIFO transfer warning interrupt line will be released if pending.

1: Enabled
FIFO transfer warning interrupt will be asserted when **FROTS.FWIS** is '1'.

(3) **FROTC.OWIE**

Output transfer Warning Interrupt Enable

This bit controls the output transfer warning interrupt.

0: Disabled
No interrupt will be requested and the output transfer warning interrupt line will be released if pending.

1: Enabled
Output transfer warning interrupt will be asserted when **FROTS.OWIS** is '1'.

(4) **FROTC.FIE**

FIFO transfer Interrupt Enable

This bit controls the FIFO transfer interrupt.

0: Disabled
No interrupt will be requested and the FIFO transfer interrupt line will be released if pending.

1: Enabled
FIFO transfer interrupt will be asserted when **FROTS.FIS** is '1'.

(5) **FROTC.OIE**

Output transfer Interrupt Enable

This bit controls the output transfer interrupt.

0: Disabled
No interrupt will be requested and the output transfer interrupt line will be released if pending.

1: Enabled
Output transfer interrupt will be asserted when **FROTS.OTIS** is '1'.

(6) **FROTC.OTCS**

Output Transfer Condition Select

The user can only write to this bit when **FROTS.OTS** is '0'.

This bit controls the output transfer condition.

0: New data only mode
The **ND** bits in the **FRNDATi** registers are used to detect a transfer condition for dedicated receive buffer

1: New data and status changed mode
The **ND** bits in the **FRNDATi** registers and the **MBC** bits in the **FRMBSC** register are used to detect a transfer condition for dedicated transmit and receive buffer

(7) **FROTC.OTE**

Output Transfer Enable

The user should only set this bit to '1' when **FROBCR.OBSYS** is '0'.

This bit controls the operation mode of the output transfer function.
The operation status of the output buffer transfer function is shown in **FROTS.OTS**.
Refer to section 103.3.16.2.(1) about usage of this bit.

0: Operation Disable request
The output buffer transfer gets disabled.
An active message buffer transfer will be completed but no further transfer will start.

1: Operation Enable request
The output buffer transfer gets enabled. Message buffers are transferred from the FlexRay internal message RAM to output data structures.

The user should not change the ERAY message RAM configuration by writing to the **FRMRC** register.

### 103.2.12.3 FRIBA [h'0808]

FlexRay Input pointer table Base Address register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITA[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ITA[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:2] | ITA | 0 | R/W | Input Table Address<br>These bits configure the base address of the input pointer table. |
| [1:0] | ITA | 0 | R | Input Table Address<br>The address should be 32 bit aligned, thus bits[1:0] are always ‘0’. |

(1) **FRIBA.ITA**

Input Table Address

The user can only write to this bit when **FRITS.ITS** is '0'.

The address should be 32 bit aligned, thus the bits **FRIBA.ITA[1:0]** are always '0'.

These bits configure the base address of the input pointer table.

The table is used for the input transfer queue transferring message buffers from the URAM into the FlexRay internal message RAM.

The size of the input queue is configured in **FRITC.ITM**.

Note that each entry requires two long words in the input pointer table.

### 103.2.12.4 FRFBA [h’080C]

FlexRay FIFO pointer table Base Address register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FTA[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FTA[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:2] | FTA | 0 | R/W | FIFO pointer Table Address<br>These bits configure the base address of the FIFO pointer table. |
| [1:0] | FTA | 0 | R | FIFO pointer Table Address<br>The address should be 32 bit aligned, thus bits[1:0] are always ‘0’. |

(1) **FRFBA.FTA**

FIFO pointer Table Address

The user can only write to this bit when **FROTS.OTS** is '0'.

The address should be 32 bit aligned, thus the bits **FRFBA.FTA[1:0]** are always '0'.

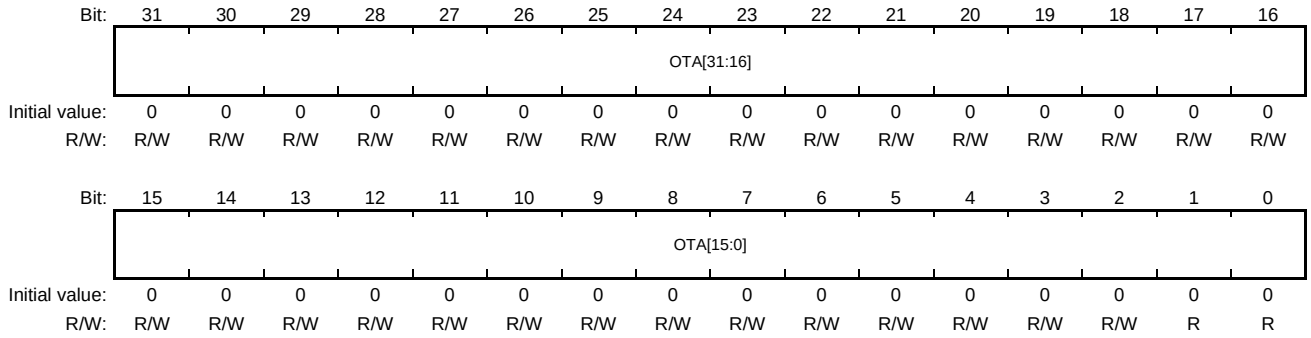
These bits configure the base address of the FIFO pointer table.

The table is used for message buffers transferred from the FlexRay internal FIFO to the URAM.

The size of the FIFO is configured in **FROTC.FTM**.

### 103.2.12.5 FROBA [h'0810]

FlexRay Output pointer table Base Address register




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:2] | OTA | 0 | R/W | Output pointer Table Address<br>These bits configure the base address of the output pointer table |
| [1:0] | OTA | 0 | R | Output pointer Table Address<br>The address should be 32 bit aligned, thus bits[1:0] are always '0' |

(1) **FROBA.OTA**

Output pointer Table Address

The user can only write to this bit when **FROTS.OTS** is '0'.

The address should be 32 bit aligned, thus the bits **FROBA.OTA[1:0]** are always '0'.

These bits configure the base address of the output pointer table.

The table is used for message buffers transferred from the FlexRay internal message RAM to the URAM.

The size of the table depends on the utilisation of the FlexRay internal message RAM and can have up to 128 entries.

### 103.2.12.6 FRIQC [h'0814]

FlexRay Input Queue Control register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | IMBNR[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:7] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [6:0] | IMBNR | 0 | W | Input Message Buffer NumbeR<br>Message buffer number added to the input queue |

(1) **FRIQC.IMBNR**

Input Message Buffer NumbeR

The user can only write to this bit when **FRITS.IQFP** is '0'.

The user should not write to this register when **FRITS.ITS** is '0' or when **FRITC.ITE** is '0'.

These bits are read as "0".

This value specifies the message buffer added to the input queue.
The number has to be identical to **FRWRHS4.IMBNR** (see section 103.3.16.1.(3)) of the input pointer table.
The address to the input data structure has to be provided in the input pointer table at the put index (**FRITS.IPIDX**) before writing to this register.
Writing to this register increases the input put index (**FRITS.IPIDX**).

### 103.2.12.7 FRUIR [h'0818]

FlexRay User Input transfer Request register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | UIDX[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:8] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [7:0] | UIDX | 0 | R/W | User requested Input inDeX<br>Input pointer table index requested for input transfer |

(1) **FRUIR.UIDX**

User requested Input inDeX

The user can only write to this bit when **FRITS.UIRP** is '0'.

The user should not write to this register when **FRITS.ITS** is '0'.
The user should not write to this register when **FRITS.UIRP** is '1'.
The user should not write to this register when **FRITS.IQH** is '1'.
The user should only write **FRITC.ITM+1** to this register.

This value specifies the input pointer table index for the user requested input transfer.
The address to the input data structure has to be provided in the input pointer table at the index **UIDX** before writing to this register.

When writing to this register, the requested input data structure will be transferred from input data structure position to the FlexRay internal message RAM.

In opposite to queued input transfers the related **DA** flag in the **FRDA** register is not influenced by the user input transfer.

### 103.2.12.8 FRUOR [h'081C]

FlexRay User Output transfer Request register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | URDS | - | - | UMBNR[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [9] | URDS | 0 | R/W | User request Read Data Section<br>0: Data section is not transferred<br>1: Data section is transferred |
| [8:7] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [6:0] | UMBNR | 0 | R/W | User requested output Message Buffer NumbeR<br>Message buffer number requested for output transfer |

(1) **FRUOR.URDS**

User request Read Data Section

The user can only write to this bit when **FROTS.UORP** is '0'.

The user should not write to this register when **FROTS.OTS** is '0'.

The user should not write to this register when **FROTS.UORP** is '1'.

0: Data section is not transferred
The data section of the message buffer selected by the bits **UMBNR** is not requested

1: Data section is transferred
The data section of the message buffer selected by the bits **UMBNR** is requested

(2) **FRUOR.UMBNR**

User requested output Message Buffer NumbeR

The user can only write to this bit when **FROTS.UORP** is '0'.

The user should not write to this register when **FROTS.OTS** is '0'.

The user should not write to this register when **FROTS.UORP** is '1'.

The user should restrict **UMBNR** to dedicated receive and transmit buffers when the FlexRay module is not in the CONFIG state.

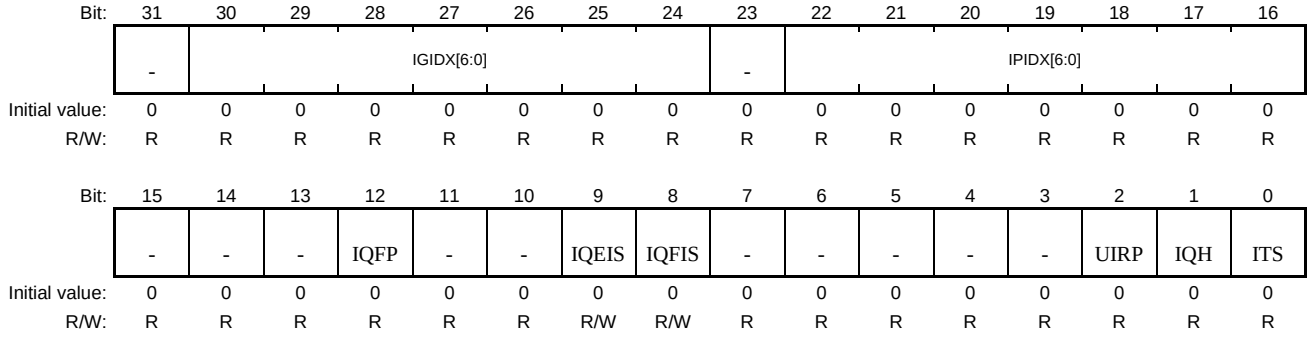
When writing to this register, the header sections and optionally the data section (configurable by **FRUOR.URDS**) of the requested message buffer will be transferred from the FlexRay internal message RAM to the output data structure position defined by the output structure data pointer in the output pointer table.

#### 103.2.12.9 Reserved

### 103.2.13 Data transfer status register

#### 103.2.13.1 FRITS [h'0820]

FlexRay Input Transfer Status register




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [30:24] | IGIDX | 0 | R | Input queue Get InDeX<br>Represents the get index of the input pointer table |
| [23] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [22:16] | IPIDX | 0 | R | Input queue Put InDeX<br>Represents the put index of the input pointer table |
| [15:13] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [12] | IQFP | 0 | R | Input Queue Full condition Pending<br>0: Entries in the input queue are available<br>1: All entries in the input queue are occupied |
| [11:10] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [9] | IQEIS | 0 | R/W | Input Queue Empty Interrupt Status<br>0: No input queue empty condition detected<br>1: Input queue empty condition detected |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [8] | IQFIS | 0 | R/W | Input Queue Full Interrupt Status<br>0: No input queue full condition detected<br>1: Input queue full condition detected |
| [7:3] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [2] | UIRP | 0 | R | User Input transfer Request Pending<br>0: No user input transfer request pending<br>1: User input transfer request pending |
| [1] | IQH | 0 | R | Input Queue Halted<br>0: Input queue not halted<br>1: Input queue halted |
| [0] | ITS | 0 | R | Input Transfer Status<br>0: Disabled<br>1: Enabled |

(1) **FRITS.IGIDX**

Input queue Get InDeX

These bits are only valid when **FRITS.IQH** is '1'

These bits represent the input pointer index the input queue handler will transfer next.
Valid values are h'00 to **FRITC.ITM**.

The get index is incremented when the input data structure has been transferred from the URAM and the related **DA** flag in the **FRDA** register is cleared.

The index is set to h'00 when **FRITS.ITS** changes from '0' to '1'.

(2) **FRITS.IPIDX**

Input queue Put InDeX

These bits represent the index where the next input data structure pointer in the input pointer table should be stored.
Valid values are h'00 to **FRITC.ITM**.

After reaching the maximum value the put index continues from h'00.

The index is incremented when writing to **FRIQC.IMBNR**.
The index is set to h'00 when **FRITS.ITS** changes from '0' to '1'.

(3) **FRITS.IQFP**

Input Queue Full condition Pending

This bit represents that the input queue is full.

There should be no further input transfer requests, by writing to **FRIQC.IMBNR**, as long as **FRITS.IQFP** is '1'.

[Clearing condition]
This bit is cleared when there is one free entry in the input queue.

[Setting condition]
This bit is set when all entries in the input queue are occupied.

(4) **FRITS.IQEIS**

Input Queue Empty Interrupt Status

Writing '0' has no effect on the bit value.

If enabled in **FRITC.IQEIE** the input queue empty interrupt is generated when **FRITS.IQEIS** is '1'.

[Clearing condition]
This bit is cleared when writing a '1' to **FRITS.IQEIS**.
This bit is cleared when **FRITS.ITS** changes from '0' to '1'.

[Setting condition]
This bit is set when all pending input transfers have been processed and consequently the input queue becomes empty.

(5) **FRITS.IQFIS**

Input Queue Full Interrupt Status

Writing '0' has no effect on the bit value.

If enabled in **FRITC.IQFIE** the input queue full interrupt is generated when **FRITS.IQFIS** is '1'.

This flag is intended as interrupt status flag. It does not represent the current input queue status; for this status refer to **FRITS.IQFP.**

[Clearing condition]
This bit is cleared when writing a '1' to **FRITS.IQFIS**.
This bit is cleared when **FRITS.ITS** changes from '0' to '1'.

[Setting condition]
This bit is set when all entries in the input queue are occupied.

(6) **FRITS.UIRP**

User Input transfer Request Pending

This bit represents that a user input transfer is still pending.

There should be no further write access to **FRUIR.UIDX** when this bit is '1'.

[Clearing condition]
This bit is cleared when the user input transfer request is processed by the input transfer handler.

[Setting condition]
This bit is set when writing to **FRUIR.UIDX**.

(7) **FRITS.IQH**

Input Queue Halted

This bit represents the status of the input queue.

There should be no further write access to **FRUIR.UIDX** when this bit is '1'.

[Clearing condition]
This bit is cleared when **FRITC.IQHR** is set to '0'.

[Setting condition]
This bit is set immediately when the **FRITC.IQHR** is set to '1' and there is no ongoing input transfer.

This bit is set only after an ongoing input transfer has been completed and **FRITC.IQHR** is set to '1'.

(8) **FRITS.ITS**

Input Transfer Status

This bit represents the status of the input queue handler.
While this bit is '1', there should be no read or write access to the address area ERAY+h'0400 to ERAY+h'05FF and there should be no CLEAR_RAMS command applied to **FRSUCC1.CMD** register.

The input transfer queue indexes and related status flags are set to '0' when **FRITS.ITS** changes from '0' to '1'.

[Clearing condition]
This bit is cleared immediately when **FRITC.ITE** is set to '0' and there are no pending input transfers
This bit is cleared after all pending requests have been processed and **FRITC.ITE** is '0'.

[Setting condition]
This bit is set when **FRITC.ITE** is set to '1'.

### 103.2.13.2 FROTS [h'0824]

FlexRay Output Transfer Status register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | FFL[5:0] | | | | | | - | - | - | FGIDX[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FWP | OWP | FDA | - | FWIS | OWIS | FIS | OTIS | - | - | - | - | - | UORP | - | OTS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:30] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [29:24] | FFL | 0 | R | FIFO Fill Level<br>Represent the number of unprocessed output FIFO structures |
| [23:21] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [20:16] | FGIDX | 0 | R | FIFO Get InDeX<br>Represent the get index in the FIFO pointer table |
| [15] | FWP | 0 | R | FIFO transfer Warning condition Pending<br>0: No FIFO transfer warning condition pending<br>1: FIFO transfer warning condition pending |
| [14] | OWP | 0 | R | Output transfer Warning condition Pending<br>0: No output transfer warning condition pending<br>1: Output transfer warning condition pending |
| [13] | FDA | 0 | R/W | FIFO Data Available<br>0: No available FIFO structures<br>1: FIFO structures available at current FROTS.FGIDX index |
| [12] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [11] | FWIS | 0 | R/W | FIFO transfer Warning Interrupt Status<br>0: No FIFO transfer warning condition detected<br>1: FIFO transfer warning condition detected |
| [10] | OWIS | 0 | R/W | Output transfer Warning Interrupt Status<br>0: No output transfer warning condition detected<br>1: Output transfer warning condition detected |
| [9] | FIS | 0 | R/W | FIFO transfer Interrupt Status<br>0: No FIFO structure updated in URAM<br>1: FIFO structure updated in URAM |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [8] | OTIS | 0 | R/W | Output transfer Interrupt Status<br>0: No output structure updated in URAM<br>1: Output structure updated in URAM |
| [7:3] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [2] | UORP | 0 | R | User Output transfer Request Pending<br>0: No user output transfer request pending<br>1: User output transfer request pending |
| [1] | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should be always 0 |
| [0] | OTS | 0 | R | Output Transfer Status<br>0: Disabled<br>1: Enabled |

(1) **FROTS.FFL**

FIFO Fill Level

These bits represent the number of available output FIFO structures in the URAM.
Valid values are h'00 to **FROTC.FTM**+1.

The value h'00 represents that the FIFO is empty.
The value **FROTC.FTM**+1 represents that the FIFO is full and no further FIFO transfers will be done.

The FIFO fill level is incremented when a FIFO data structure has been transferred from the FlexRay internal FIFO into the URAM.
The FIFO fill level is decremented when the user releases a FIFO data structure in the URAM by writing '1' to **FROTS.FDA**.
The FIFO fill level is set to h'00 when the bit **FROTS.OTS** changes from '0' to '1'.

(2) **FROTS.FGIDX**

FIFO Get InDeX

These bits represent the index where the current output data structure pointer in the FIFO pointer table is available for reading.
Valid values are h'00 to **FROTC.FTM**.
After reaching the maximum value the get index continues from h'00.

The index is incremented when a FIFO data structure is released by writing '1' to **FROTS.FDA**.
The index is set to h'00 when **FROTS.OTS** changes from '0' to '1'.

(3) **FROTS.FWP**

FIFO transfer Warning condition Pending

This bit represents the FIFO transfer warning condition.

[Clearing condition]
This bit is cleared when there are free output data structures (**FROTS.FFL** <= **FROTC.FTM**).
This bit is cleared when the **FROTS.OTS** changes from '0' to '1'.

[Setting condition]

This bit is set when the output transfer handler detects a transfer condition for FIFO message buffers but there are no free output data structures (**FROTS.FFL** = **FROTC.FTM**+1).

(4) **FROTS.OWP**

Output transfer Warning condition Pending

This bit represents the output transfer warning condition.

[Clearing condition]
This bit is cleared, when all output structure pointers that have a pending output handler transfer condition detected, are released (for dedicated transmit and receive message buffers or a user output transfer request).

[Setting condition]
This bit is set when the output transfer handler detects a transfer condition (for dedicated transmit and receive message buffers or a user output transfer request) but the related output structure pointer was not yet released by the application (data available flag is still set to '1').
This bit is set when the output transfer handler detects a transfer condition for dedicated transmit and receive message buffers but there is a pending input transfer for the same message buffer (data available flag is set to '1' due to the input transfer request).

(5) **FROTS.FDA**

FIFO Data Available

Writing '0' has no effect on the bit value.

When this bit is '1', the next valid output data structure is available.
The related data structure pointer is in the FIFO pointer table at **FROTS.FGIDX**.

Writing '1' to **FROTS.FDA**
increments **FROTS.FGIDX** and
decrements the FIFO fill level (**FROTS.FFL**)
If there are still unprocessed data structures **FROTS.FDA** remains '1'.

[Clearing condition]
This bit is cleared when writing '1' to **FROTS.FDA** and the FIFO fill level becomes h'00.
This bit is cleared when the **FROTS.OTS** changes from '0' to '1'.

[Setting condition]
This bit is set when there is at least one FIFO data structure available in the URAM.

(6) **FROTS.FWIS**

FIFO transfer Warning Interrupt Status

Writing '0' has no effect on the bit value.

If enabled in **FROTC.FWIE** the FIFO transfer warning interrupt is generated when **FROTS.FWIS** is '1'.

[Clearing condition]
This bit is cleared when writing a '1' to **FROTS.FWIS**.
This bit is cleared when the bit **FROTS.OTS** changes from '0' to '1'.

[Setting condition]
This bit is set when the output transfer handler detects a transfer condition for FIFO message buffers but there are no free output data structures (**FROTS.FFL** = **FROTC.FTM**+1).

(7) **FROTS.OWIS**

Output transfer Warning Interrupt Status

Writing '0' has no effect on the bit value.

If enabled in **FROTC.OWIE** the FIFO transfer warning interrupt is generated when **FROTS.OWIS** is '1'.

[Clearing condition]
This bit is cleared when writing a '1' to **FROTS.OWIS**.
This bit is cleared when the bit **FROTS.OTS** changes from '0' to '1'.

[Setting condition]
This bit is set when the output transfer handler detects a transfer condition (for dedicated transmit and receive message buffers or a user output transfer request) but the related output structure pointer was not yet released by the application (data available flag is still set to '1').
This bit is set when the output transfer handler detects a transfer condition for dedicated transmit and receive message buffers but there is a pending input transfer for the same message buffer (data available flag is set to '1' due to the input transfer request).

(8) **FROTS.FIS**

FIFO transfer Interrupt Status

Writing '0' has no effect on the bit value.

If enabled in **FROTC.FIE** the FIFO transfer interrupt is generated when **FROTS.FIS** is '1'.

[Clearing condition]
This bit is cleared when writing a '1' to **FROTS.FIS**.
This bit is cleared when the bit **FROTS.OTS** changes from '0' to '1'.

[Setting condition]
This bit is set when a FIFO data structure is updated by the transfer handler and **FROTS.FFL** changes from h'00 to h'01.

(9) **FROTS.OTIS**

Output transfer Interrupt Status

Writing '0' has no effect on the bit value.

If enabled in **FROTC.OIE** the output transfer interrupt is generated when **FROTS.OTIS** is '1'.

[Clearing condition]
This bit is cleared when writing a '1' to **FROTS.OTIS**.
This bit is cleared when the bit **FROTS.OTS** changes from '0' to '1'.

[Setting condition]
This bit is set when an output data structure is updated by the transfer handler (from a dedicated transmit or receive message buffer or by a user output transfer request).

(10) **FROTS.UORP**

User Output transfer Request Pending

This bit represents that a user output transfer is still pending.

There should be no further write access to **FRUOR.UMBNR** when this bit is '1'.

[Clearing condition]
This bit is cleared when the user output transfer request is processed by the output transfer handler.
This bit is cleared when the bit **FROTS.OTS** changes from '0' to '1'.

[Setting condition]
This bit is set when writing to **FRUOR.UMBNR**.

(11) **FROTS.OTS**

Output Transfer Status

This bit represents the status of the output transfer handler.
While this bit is '1', there should be no read or write access to the address area ERAY+h'0600 to ERAY+h'07FF and there should be no CLEAR_RAMS command applied to **FRSUCC1.CMD** register.

While this bit is '1', the user should not change the ERAY message RAM configuration by writing to the **FRMRC** register.

The output hander transfer indexes and related status flags are set to '0' when **FROTS.OTS** changes from '0' to '1'.

[Clearing condition]
This bit is cleared immediately when **FROTC.OTE** is set to '0' and there are no ongoing output transfers.
This bit is cleared after an ongoing transfer has been completed and **FROTC.OTE** is '0'.

[Setting condition]
This bit is set when **FROTC.OTE** is set to '1'.

### 103.2.13.3 FRAES [h'0828]

FlexRay Access Error Status register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | MAE | FAE | OAE | IAE | EIDX[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:12] | - | 0 | R | Reserved<br>These bits are always read as 0. The write value should be always 0 |
| [11] | MAE | 0 | R/W | Multiple Access Errors<br>0: No multiple access errors occurred<br>1: Multiple access errors occurred |
| [10] | FAE | 0 | R/W | FIFO transfer Access Error<br>0: No access error occurred during FIFO transfer<br>1: Access error occurred during FIFO transfer |
| [9] | OAE | 0 | R/W | Output transfer Access Error<br>0: No access error occurred during output transfer<br>1: Access error occurred during output transfer |
| [8] | IAE | 0 | R/W | Input transfer Access Error<br>0: No access error occurred during input transfer<br>1: Access error occurred during input transfer |
| [7:0] | EIDX | 0 | R | Error InDeX<br>Data structure pointer index number |

(1) **FRAES.MAE**

Multiple Access Errors

Writing '0' has no effect on the bit value.

This bit represents that there were multiple access errors during a data transfer.

[Clearing condition]
This bit is cleared when writing a '1' to **FRAES.MAE**.

[Setting condition]
This bit is set when one of the bits **FRAES.FAE**, **FRAES.OAE** or **FRAES.IAE** are set and
an access to an protected address occurred during a FIFO data transfer or
an access to an protected address occurred during an output data transfer or

an access to an protected address occurred during an input data transfer

(2) **FRAES.FAE**

FIFO transfer Access Error

Writing '0' has no effect on the bit value.

This bit represent that there was an access error during a FIFO data transfer.

[Clearing condition]
This bit is cleared when writing a '1' to **FRAES.FAE**.

[Setting condition]
This bit is set when a URAM access error was detected during a FIFO transfer and the bits **FRAES.OAE**, **FRAES.IAE** and **FRAES.MAE** are '0'.

(3) **FRAES.OAE**

Output transfer Access Error

Writing '0' has no effect on the bit value.

This bit represent that there was an access error during a output data transfer.

[Clearing condition]
This bit is cleared when writing a '1' to **FRAES.OAE**.

[Setting condition]
This bit is set when a URAM access error was detected during an output transfer and the bits **FRAES.FAE**, **FRAES.IAE** and **FRAES.MAE** are '0'.

(4) **FRAES.IAE**

Input transfer Access Error

Writing '0' has no effect on the bit value.

This bit represent that there was an access error during an input data transfer.

[Clearing condition]
This bit is cleared when writing a '1' to **FRAES.IAE**.

[Setting condition]
This bit is set when a URAM access error was detected during an input transfer and the bits **FRAES.OAE**, **FRAES.FAE** and **FRAES.MAE** are '0'.

(5) **FRAES.EIDX**

Error InDeX

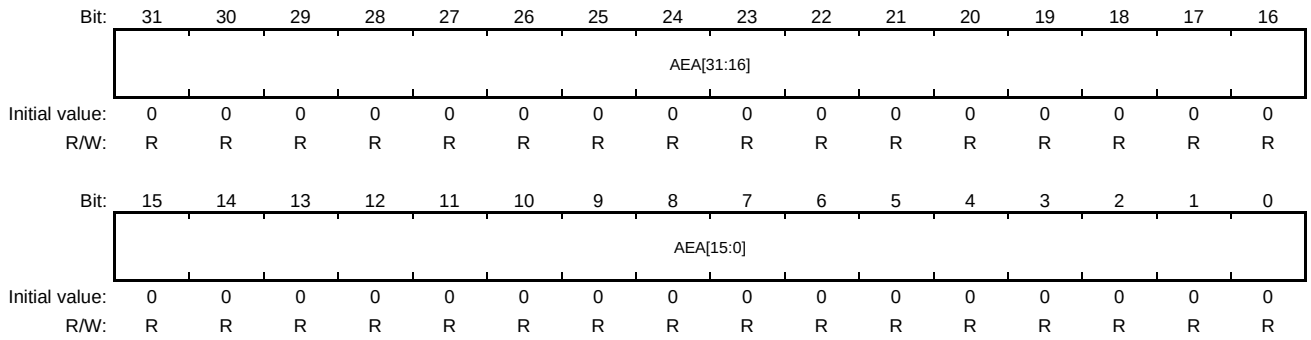
This value is only valid when one of the bits **FRAES.FAE**, **FRAES.OAE** or **FRAES.IAE** is '1'.

When the bit **FRAES.FAE** is '1', **FRAES.EIDX** holds the used FIFO put index when the access error has occurred.
When the bit **FRAES.OAE** is '1', **FRAES.EIDX** holds the used output table entry (which is related to message buffer number) when the access error has occurred.
When the bit **FRAES.IAE** is '1', **FRAES.EIDX** holds the used input pointer table get index when the access error has occurred during the input transfer or user requested input transfer.

These bits are updated when one of the bits **FRAES.FAE**, **FRAES.OAE** or **FRAES.IAE** is changing from '0' to '1'.

### 103.2.13.4 FRAEA [h'082C]

FlexRay Access Error Address register




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:0] | AEA | 0 | R | Access Error Address<br>Address in the URAM when an access error has occurred |

(1) **FRAEA.AEA**

Access Error Address

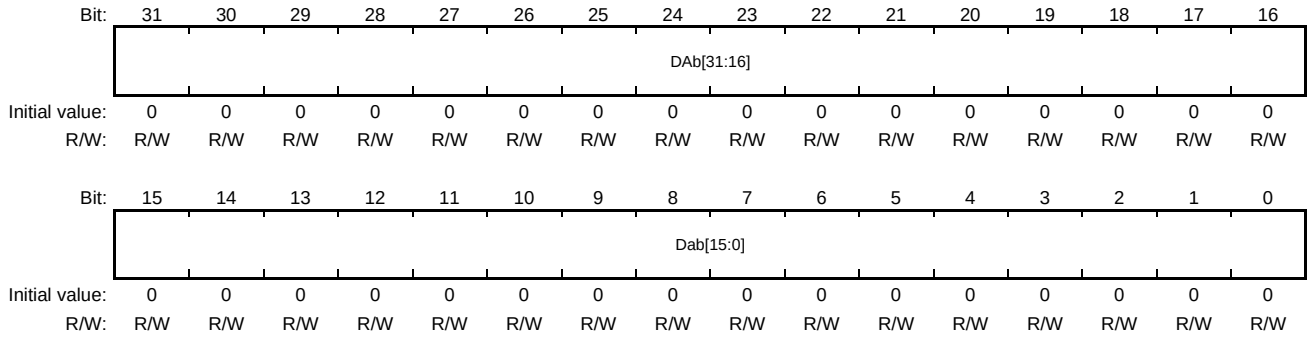
This value is only valid when one of the bits **FRAES.FAE**, **FRAES.OAE** or **FRAES.IAE** is '1'.

These bits represent the address of the access error indicated in the **FRAES** register.

These bits are updated when one of the bits **FRAES.FAE**, **FRAES.OAE** or **FRAES.IAE** is changing from '0' to '1'.

### 103.2.13.5 FRDAi (i=0 to 3) [h'0830 – h'083C]

FlexRay message Data Available register i (i 0 to 3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| [31:0] | DAb | 0 | R/W | Data Available b<br>0 : No data available for destination<br>1 : Data available for destination |

(1) **FRDAi.DAb (b = i*32 to ((i+1)*32)-1)**

Data Available b

The user should not write a '1' to bits that are '0'.
To maintain the status of input transfers, the user should not clear bits related to input transfers.

This register is used for input and output transfers.
Each flag corresponds to a FlexRay message buffer.

[Clearing condition]
Input transfer:
This bit is cleared when the input data structure has been transferred from the URAM. The data structure and the data structure pointer can be changed when the related flag is '0'.

Output transfer:
This bit is cleared when writing a '1' to it.

[Setting condition]
Input transfer:
This bit is set when the corresponding message buffer number has been written to **FRIQC.IMBNR**.
As long as this bit is '1', the input data structure and the data structure pointer corresponding to this input transfer request should not be changed.

Output transfer:
This bit is set when the output data structure corresponding to this message buffer has been updated.
As long as this bit is '1', the data structure is stable; no further update of the data structure will be done by the output handler. While this bit is '1', the application is allowed to change the output data structure pointer in the output pointer table for this message buffer number.

## 103.3 Operation

This chapter describes the FlexRay implementation together with the related FlexRay protocol features. More information about the FlexRay protocol itself can be found in the FlexRay protocol specification [FR_PROT].

### 103.3.1 FlexRay module operation control

#### 103.3.1.1 FlexRay Module enable

After hardware reset or after the FlexRay module has been disabled (following section 103.3.1.2) the FlexRay module is in the reset state (**FROS.OS** is '0') and the clocks of the FlexRay core module are disabled.

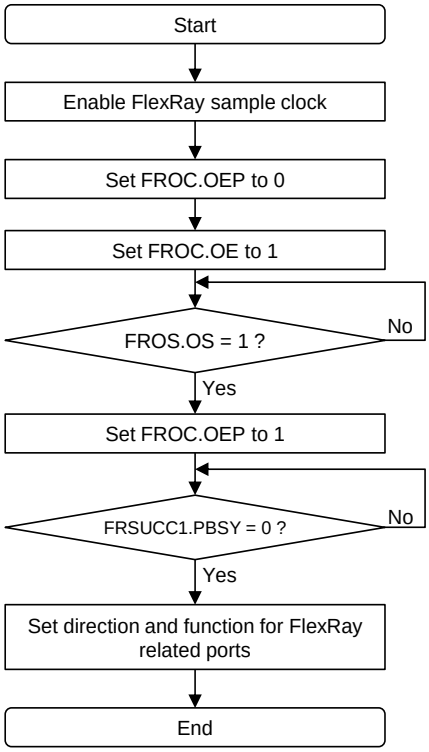



**Figure 103.3 FlexRay enable flow**

#### 103.3.1.2 FlexRay Module disable

The FlexRay module can be disabled at any time. However, it is recommended to disable the FlexRay module using the **FROC.OE** register only when the FlexRay module is in HALT, CONFIG or DEFAULT_CONFIG state. Resetting the FlexRay module in any other state will terminate any ongoing FlexRay communication.
If the data transfer function is used, it is also required to disable this function before disabling the FlexRay module (see section 103.3.16.1.(1) for suspending input transfer function and section 103.3.16.2.(1) for suspending output transfer).
The following flow should be executed to disable the FlexRay module.

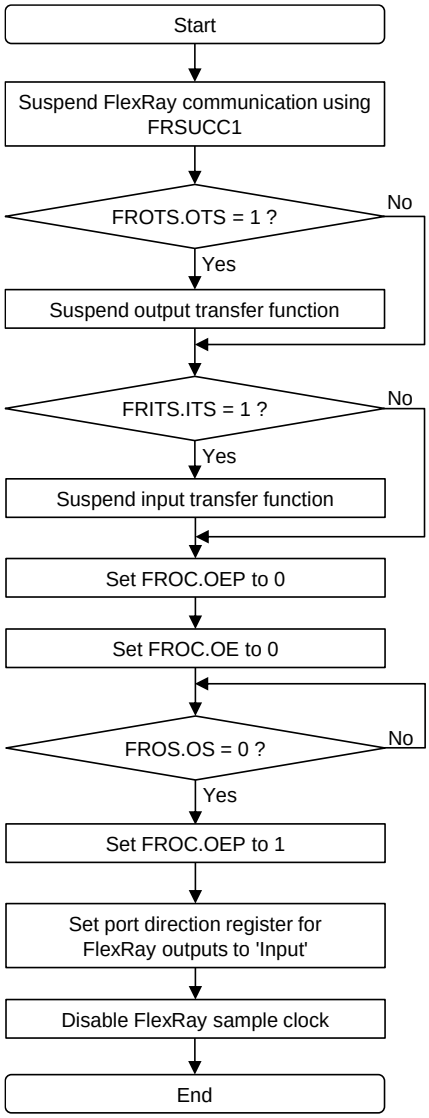



**Figure 103.3.1 FlexRay disable flow**

### 103.3.2 Communication Cycle

Communication on FlexRay networks is based on frames and symbols. The wakeup symbol (WUS) and the collision avoidance symbol (CAS) are transmitted outside the communication cycle to setup the time schedule. Frames and media access test symbols (MTS) are transmitted inside the communication cycle.

A FlexRay communication cycle consists of the following elements:

Static Segment

Dynamic Segment (optional)

Symbol Window (optional)

Network Idle Time (NIT)

Static segment, dynamic segment, and symbol window form the Network Communication Time (NCT). For each communication channel the slot counter starts at 1 and counts up until the end of the dynamic segment is reached. Both channels share the same arbitration grid which means that they use the same synchronized macrotick.

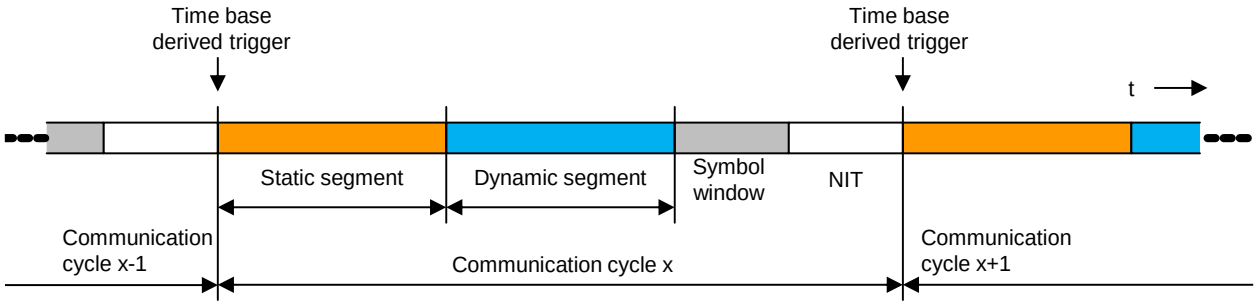



**Figure 103.4 Structure of communication cycle**

#### 103.3.2.1 Static Segment

The Static Segment is characterized by the following features:

Time slots of fixed length (optionally protected by bus guardian)

Start of frame transmission at action point of the respective static slot

Payload length same for all frames on both channels

Parameters:

Number of Static Slots **FRGTUC7.NSS**, Static Slot Length **FRGTUC7.SSL**, Payload Length Static **FRMHDC.SFDL**, Action Point Offset **FRGTUC9.APO**

#### 103.3.2.2 Dynamic Segment

The Dynamic Segment is characterized by the following features:

All controllers have bus access (no bus guardian protection possible)

Variable payload length and duration of slots, different for both channels

Start of transmission at minislot action point

Parameters:

Number of Minislots **FRGTUC8.NMS**, Minislot Length **FRGTUC8.MSL**, Minislot Action Point Offset **FRGTUC9.MAPO**, Start of Latest Transmit (last minislot) **FRMHDC.SLT**

#### 103.3.2.3 Symbol Window

During the symbol window only one media access test symbol (MTS) may be transmitted per channel. MTS symbols are send in NORMAL_ACTIVE state to test the bus guardian.

The symbol window is characterized by the following features:

Send single symbol

Transmission of the MTS symbol starts at the symbol windows action point

Parameters:

Symbol Window Action Point Offset **FRGTUC9.APO** (same as for static slots), Network Idle Time Start **FRGTUC4.NIT**

#### 103.3.2.4 Network Idle Time (NIT)

During network idle time the CC has to perform the following tasks:

Calculate clock correction terms (offset and rate)

Distribute offset correction over multiple macroticks after offset correction start

Perform cluster cycle related tasks
Parameters:

Network Idle Time Start **FRGTUC4.NIT**, Offset Correction Start **FRGTUC4.OCS**

#### 103.3.2.5 Configuration of NIT Start and Offset Correction Start

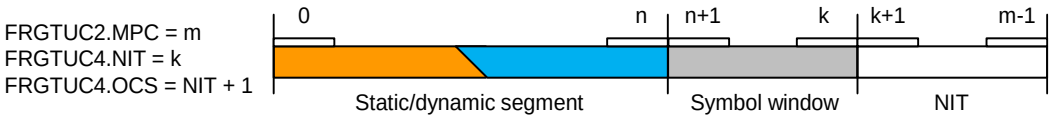



**Figure 103.5 Configuration of NIT start and offset correction start**

The number of macroticks per cycle gMacroPerCycle is assumed to be m. It is configured by programming **FRGTUC2.MPC** = m.

The static / dynamic segment starts with macrotick 0 and ends with macrotick n:
n = static segment length + dynamic segment offset + dynamic segment length - 1MT
n = gNumberOfStaticSlots • gdStaticSlot + dynamic segment offset + gNumberOfMinislots • gdMinislot - 1 MT

The static segment length is configured by **FRGTUC7.SSL** and **FRGTUC7.NSS**.
The dynamic segment length is configured by **FRGTUC8.MSL** and **FRGTUC8.NMS**.

The dynamic segment offset is:
If gdActionPointOffset <= gdMinislotActionPointOffset:

dynamic segment offset = 0 MT

Else if gdActionPointOffset > gdMinislotActionPointOffset:

dynamic segment offset = gdActionPointOffset -gdMinislotActionPointOffset

The NIT starts with macrotick k+1 and ends with the last macrotick of cycle m-1. It has to be configured by setting **FRGTUC4.NIT** = k.

For the FlexRay module the offset correction start is required to be **FRGTUC4.OCS** => **FRGTUC4.NIT** + 1 = k+1.

The length of symbol window results from the number of macroticks between the end of the static / dynamic segment and the beginning of the NIT. It can be calculated by k - n.

### 103.3.3 Communication Modes

The FlexRay Protocol Specification [FR_PROT] defines the Time-Triggered Distributed (TT-D) mode.

#### 103.3.3.1 Time-triggered Distributed (TT-D)

In TT-D mode the following configurations are possible:

Pure static: Minimum 2 static slots + symbol window (optional)

Mixed static/dynamic: Minimum 2 static slots + dynamic segment + symbol window (optional)
A minimum of two coldstart nodes needs to be configured for distributed time-triggered operation. Two fault-free coldstart nodes are necessary for the cluster startup. Each startup frame must be a sync frame, therefore all coldstart nodes are sync nodes.

### 103.3.4 Clock Synchronization

In TT-D mode a distributed clock synchronization is used. Each node individually synchronizes itself to the cluster by observing the timing of received sync frames from other nodes.

#### 103.3.4.1 Global Time

Activities in a FlexRay node, including communication, are based on the concept of a global time, even though each individual node maintains its own view of it. It is the clock synchronization mechanism that differentiates the FlexRay cluster from other node collections with independent clock mechanisms. The global time is a vector of two values; the cycle (cycle counter) and the cycle time (macrotick counter).

**Cluster specific:**

Macrotick (MT) = basic unit of time measurement in a FlexRay network, a macrotick consists of an integer number of microticks (µT)

Cycle length = duration of a communication cycle in units of macroticks (MT)

#### 103.3.4.2 Local Time

Internally, nodes time their behaviour with microtick resolution. Microticks are time units derived from the oscillator clock tick of the specific node. Therefore microticks are controller-specific units. They may have different duration in different controllers. The precision of a node's local time difference measurements is a microtick (µT).

**Node specific:**

Oscillator clock -> prescaler -> microtick (µT)

µT = basic unit of time measurement in a CC, clock correction is done in units of µTs

Cycle counter + macrotick counter = nodes local view of the global time

#### 103.3.4.3 Synchronization Process

Clock synchronization is performed by means of sync frames. Only preconfigured nodes (sync nodes) are allowed to send sync frames. In a two-channel cluster a sync node has to send its sync frame on both channels.

For synchronization in FlexRay the following constraints have to be considered:

Max. one sync frame per node in one communication cycle

Max. 15 sync frames per cluster in one communication cycle

Every node has to use a preconfigured number of sync frames (**FRGTUC2.SNM**) for clock synchronization

Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of sync frames received during the static segment is measured. In a two channel cluster the sync node has to be configured to send sync frames on both channels. The calculation of correction terms is done during NIT (offset: every cycle, rate: every odd cycle) by using an FTM algorithm. For details see FlexRay protocol specification [FR_PROT], chapter 8.

(1) **Offset (phase) Correction**

Only deviation values measured and stored in the current cycle used

For a two channel node the smaller value will be taken

Calculation during NIT of every communication cycle

Offset correction value calculated in even cycles used for error checking only

Checked against limit values

Correction value is a signed integer number of µTs

Correction done in odd numbered cycles, distributed over the macroticks beginning at offset correction start up to cycle end (end of NIT) to shift nodes next start of cycle (MTs lengthened / shortened)

(2) **Rate (frequency) Correction**

Pairs of deviation values measured and stored in even / odd cycle pair used
For a two channel node the average of the differences from the two channels is used
Calculated during NIT of odd numbered cycles
Cluster drift damping is performed using global damping value
Checked against limit values
Correction value is a signed integer number of µTs
Distributed over macroticks comprising the next even / odd cycle pair (MTs lengthened / shortened)

(3) **Sync Frame Transmission**

Sync frame transmission is only possible from Message buffer 0 and 1. Message buffer 1 may be used for sync frame transmission in case that sync frames should have different payloads on the two channels. In this case bit **FRMRC.SPLM** has to be programmed to '1'.

Message buffers used for sync frame transmission have to be configured with the key slot ID and can be (re)configured in DEFAULT_CONFIG or CONFIG state only. For nodes transmitting sync frames **FRSUCC1.TXSY** must be set to '1'.

(4) **External Clock Synchronization**

During normal operation, independent clusters can drift significantly. If synchronous operation across independent clusters is desired, external synchronization is necessary; even though the nodes within each cluster are synchronized. This can be accomplished with synchronous application of host-deduced rate and offset correction terms to the clusters.

External offset / rate correction value is a signed integer
External offset / rate correction value is added to calculated offset / rate correction value
Aggregated offset / rate correction term (external + internal) is not checked against configured limits

### 103.3.5 Error Handling

The implemented error handling concept is intended to ensure that, in case of a lower layer protocol error in one single node, communication between non-affected nodes can be maintained. In some cases, higher layer program activity is required for the CC to resume normal operation. A change of the error handling state will set **FREIR.PEMC** and may trigger an interrupt to the Host if enabled. The actual error mode is signaled by **FRCCEV.ERRM**.

**Table 103.3 Error modes of the POC (degradation model)**

| Error Mode | Activity |
|---|---|
| ACTIVE | Full operation, State: NORMAL_ACTIVE<br>The CC is fully synchronized and supports the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers **FREIR** and **FRSIR**. |
| PASSIVE | Reduced operation, State: NORMAL_PASSIVE, CC self rescue allowed<br>The CC stops transmitting frames and symbols, but received frames are still processed. Clock synchronization mechanisms are continued based on received frames. No active contribution to the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers **FREIR** and **FRSIR**. |
| COMM_HALT | Operation halted, State: HALT, CC self rescue not allowed<br>The CC stops frame and symbol processing, clock synchronization processing, and the macrotick generation. The host has still access to error and status information by reading the error and status interrupt flags from registers **FREIR** and **FRSIR**. The bus drivers are disabled. |

#### 103.3.5.1 Clock Correction Failed Counter

When the Clock Correction Failed Counter reaches the "maximum without clock correction passive" limit defined by **FRSUCC3.WCP**, the POC transits from NORMAL_ACTIVE to NORMAL_PASSIVE state. When it reaches the "maximum without clock correction fatal" limit defined by **FRSUCC3.WCF**, it transits from NORMAL_ACTIVE or NORMAL_PASSIVE to HALT state.

The Clock Correction Failed Counter **FRCCEV.CCFC** allows the Host to monitor the duration of the inability of a node to compute clock correction terms after the CC passed protocol startup phase. It will be incremented by one at the end of any odd communication cycle during which either the missing offset correction **FRSFS.MOCS** or the missing rate correction **FRSFS.MRCS** flag is set.

The Clock Correction Failed Counter is reset to zero at the end of an odd communication cycle if neither the missing offset correction **FRSFS.MOCS** nor the missing rate correction **FRSFS.MRCS** flag is set.

The Clock Correction Failed Counter stops incrementing when the "maximum without clock correction fatal" value **FRSUCC3.WCF** is reached (i.e. incrementing the counter at its maximum value will not cause it to wrap around back to zero). The Clock Correction Failed Counter is initialized to zero when the CC enters READY state or when NORMAL_ACTIVE state is entered.

Note: The transition to HALT state is prevented if **FRSUCC1.HCSE** is not set.

#### 103.3.5.2 Passive to Active Counter

The passive to active counter controls the transition of the POC from NORMAL_PASSIVE to NORMAL_ACTIVE state. **FRSUCC1.PTA** defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the CC is allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state. If **FRSUCC1.PTA** is set to zero the CC is not allowed to transit from NORMAL_PASSIVE to NORMAL_ACTIVE state.

#### 103.3.5.3 HALT Command

In case the Host wants to stop FlexRay communication of the local node it can bring the CC into HALT state by asserting the HALT command. This can be done by writing **FRSUCC1.CMD** = "0110". In order to shut down communication on an entire FlexRay network, a higher layer protocol is required to assure that all nodes apply the HALT command at the same time.

The POC state from which the transition to HALT state took place can be read from **FRCCSV.PSL**.

When called in NORMAL_ACTIVE or NORMAL_PASSIVE state the POC transits to HALT state at the end of the current cycle. When called in any other state **FRSUCC1.CMD** will be reset to "0000" = command_not_accepted and bit **FREIR.CNA** is set to ’1’. If enabled an interrupt to the Host is generated.

#### 103.3.5.4 FREEZE Command

In case the Host detects a severe error condition it can bring the CC into HALT state by asserting the FREEZE command. This can be done by writing **FRSUCC1.CMD** = "0111". The FREEZE command triggers the entry of the HALT state immediately regardless of the actual POC state.

The POC state from which the transition to HALT state took place can be read from **FRCCSV.PSL**.

### 103.3.6 Communication Controller States

#### 103.3.6.1 Communication Controller State Diagram

(1) **Reserved**

(2) **Reserved**

#### 103.3.6.2 DEFAULT_CONFIG State

In DEFAULT_CONFIG state, the CC is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state.

The CC enters this state

When the reset is applied (HW reset or SW reset)

When exiting from HALT state
To leave DEFAULT_CONFIG state the Host has to write **FRSUCC1.CMD** = "0001". The CC then transits to CONFIG state.

#### 103.3.6.3 CONFIG State

In CONFIG state, the CC is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state. This state is used to initialize the CC configuration.

The CC enters this state

When exiting from DEFAULT_CONFIG state

When exiting from READY state
When the state has been entered via HALT and DEFAULT_CONFIG state, the Host can analyse status information and configuration. Before leaving CONFIG state the Host has to assure that the configuration is fault-free.

To leave CONFIG state, the Host has to perform the unlock sequence as described is section 103.2.2.1. Directly after unlocking the CONFIG state the Host has to write **FRSUCC1.CMD** to enter the next state.

**Note:** Status bits **FRMHDS[14:0]**, registers **FRTXRQ1/2/3/4**, and status data stored in the Message RAM are not affected by the transition of the POC from CONFIG to READY state.

When the CC is in CONFIG state it is also possible to bring the CC into a power saving mode by halting the module clocks (bus clock and sample clock). To do this the Host has to assure that all Message RAM transfers have finished before turning off the clocks.

#### 103.3.6.4 READY State

After unlocking CONFIG state and writing **FRSUCC1.CMD** = "0010" the CC enters READY state. From this state the CC can transit to WAKEUP state and perform a cluster wakeup or to STARTUP state to perform a coldstart or to integrate into a running cluster.

The CC enters this state

When exiting from CONFIG, WAKEUP, STARTUP, NORMAL_ACTIVE, or NORMAL_PASSIVE state by writing **FRSUCC1.CMD** = "0010" (READY command).

The CC exits from this state

To CONFIG state by writing **FRSUCC1.CMD** = "0001" (CONFIG command)

To WAKEUP state by writing **FRSUCC1.CMD** = "0011" (WAKEUP command)

To STARTUP state by writing **FRSUCC1.CMD** = "0100" (RUN command)
Internal counters and the CC status flags are reset when the CC enters STARTUP state.

**Note:** Status bits **FRMHDS[14:0]**, registers **FRTXRQ1/2/3/4**, and status data stored in the Message RAM are not affected by the transition of the POC from READY to STARTUP state.

### 103.3.6.5 WAKEUP State

The description below is intended to help configuring wakeup for the FlexRay IP-module. A detailed description of the wakeup procedure together with the respective SDL diagrams can be found in the FlexRay protocol specification [FR_PROT], section 7.1.

The CC enters this state

When exiting from READY state by writing **FRSUCC1.CMD** = "0011" (WAKEUP command).
The CC exits from this state to READY state

After complete non-aborted transmission of wakeup pattern

After WUP reception

After detecting a WUP collision

After reception of a frame header

By writing **FRSUCC1.CMD** = "0010" (READY command)
The cluster wakeup must precede the communication startup in order to ensure that all nodes in a cluster are awake. The minimum requirement for a cluster wakeup is that all bus drivers are supplied with power. A bus driver has the ability to wake up the other components of its node when it receives a wakeup pattern on its channel. At least one node in the cluster needs an external wakeup source.

The Host completely controls the wakeup procedure. It is informed about the state of the cluster by the bus driver and the CC and configures bus guardian (if available) and CC to perform the cluster wakeup. The CC provides to the Host the ability to transmit a special wakeup pattern on each of its available channels separately. The CC needs to recognize the wakeup pattern only during WAKEUP state.

Wakeup may be performed on only one channel at a time. The Host has to configure the wakeup channel while the CC is in CONFIG state by writing **FRSUCC1.WUCS**. The CC ensures that ongoing communication on this channel is not disturbed. The CC cannot guarantee that all nodes connected to the configured channel awake upon the transmission of the wakeup pattern, since these nodes cannot give feedback until the startup phase. The wakeup procedure enables single-channel devices in a two-channel system to trigger the wakeup, by only transmitting the wakeup pattern on the single channel to which they are connected. Any coldstart node that deems a system startup necessary will then wake the remaining channel before initiating communication startup.

The wakeup procedure tolerates any number of nodes simultaneously trying to wakeup a single channel and resolves this situation such that only one node transmits the pattern. Additionally the wakeup pattern is collision resilient, so even in the presence of a fault causing two nodes to simultaneously transmit a wakeup pattern, the resulting collided signal can still wake the other nodes.

After wakeup the CC returns to READY state and signals the change of the wakeup status to the Host by setting flag **FRSIR.WST**. The wakeup status vector can be read from **FRCCSV.WSV**. If a valid wakeup pattern was received also either flag **FRSIR.WUPA** or flag **FRSIR.WUPB** is set.

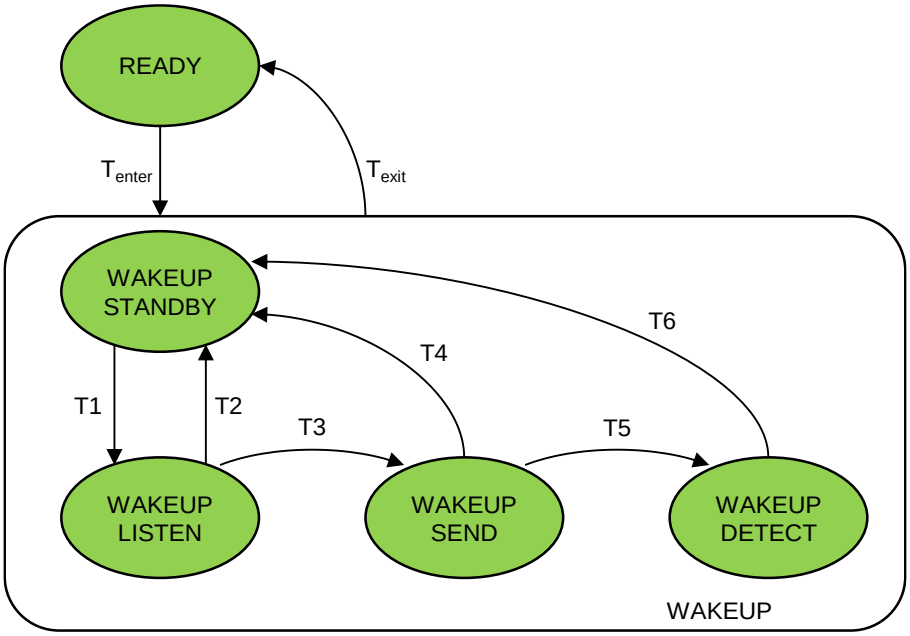



**Figure 103.6 Structure of POC state WAKEUP**

**Table 103.6 State transitions WAKEUP**

| T# | Condition | From | To |
|---|---|---|---|
| enter | Host commands change to WAKEUP state by writing **FRSUCC1.CMD** = "0011" (WAKEUP command) | READY | WAKEUP |
| 1 | CHI command WAKEUP triggers wakeup FSM to transit to WAKEUP_LISTEN state | WAKEUP_STANDBY | WAKEUP_LISTEN |
| 2 | Received WUP on wakeup channel selected by bit **FRSUCC1.WUCS** OR frame header on either available channel | WAKEUP_LISTEN | WAKEUP_STANDBY |
| 3 | Timer event | WAKEUP_LISTEN | WAKEUP_SEND |
| 4 | Complete, non-aborted transmission of wakeup pattern | WAKEUP_SEND | WAKEUP_STANDBY |
| 5 | Collision detected | WAKEUP_SEND | WAKEUP_DETECT |
| 6 | Wakeup timer expired OR WUP detected on wakeup channel selected by bit **FRSUCC1.WUCS** OR frame header received on either available | WAKEUP_DETECT | WAKEUP_STANDBY |
| exit | Wakeup completed (after T2 or T4 or T6) OR Host commands change to READY state by writing **FRSUCC1.CMD** = "0010" (READY command). This command also resets the wakeup FSM to WAKEUP_STANDBY state | WAKEUP | READY |

The WAKEUP_LISTEN state is controlled by the wakeup timer and the wakeup noise timer. The two timers are controlled by the parameters listen timeout **FRSUCC2.LT** and listen timeout noise **FRSUCC2.LTN**. Listen timeout enables a fast cluster wakeup in case of a noise free environment, while listen timeout noise enables wakeup under more difficult conditions regarding noise interference.

In WAKEUP_SEND state the CC transmits the wakeup pattern on the configured channel and checks for collisions. After return from wakeup the Host has to bring the CC into STARTUP state by CHI command RUN.

In WAKEUP_DETECT state the CC attempts to identify the reason for the wakeup collision detected in WAKEUP_SEND state. The monitoring is bounded by the expiration of listen timeout as configured by **FRSUCC2.LT**. Either the detection of a wakeup pattern indicating a wakeup attempt by another node or the reception of a frame header

indicating ongoing communication, causes the direct transition to READY state. Otherwise WAKEUP_DETECT is left after expiration of listen timeout; in this case the reason for wakeup collision is unknown.

The Host has to be aware of possible failures of the wakeup and act accordingly. It is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal time it takes another coldstart node to become awake and to be configured.

The FlexRay Protocol Specification [FR_PROT] recommends that two different CCs shall awake the two channels.

(1) **Host activities**

The host must coordinate the wakeup of the two channels and must decide whether, or not, to wake a specific channel. The sending of the wakeup pattern is initiated by the Host. The wakeup pattern is detected by the remote BDs and signaled to their local Host.

**Wakeup procedure controlled by Host (single-channel wakeup):**

Configure the CC in CONFIG state
- Select wakeup channel by programming bit **FRSUCC1.WUCS**

Check local BDs whether a WUP was received

Activate BD of selected wakeup channel

Command CC to enter READY state

Command CC to start wakeup on the configured channel by writing **FRSUCC1.CMD** = "0011"
- CC enters WAKEUP
- CC returns to READY state and signals status of wakeup attempt to the Host

Wait predefined time to allow the other nodes to wakeup and configure themselves

Coldstart node:
- In a dual channel cluster wait for WUP on the other channel
- Reset coldstart inhibit flag **FRCCSV.CSI** by writing **FRSUCC1.CMD** = "1001" (ALLOW_COLDSTART command)

Command CC to enter startup by writing **FRSUCC1.CMD** = "0100" (RUN command)

**Wakeup procedure triggered by BD:**

Wakeup recognized by BD

BD triggers power-up of Host (if required)

BD signals wakeup event to Host

Host configures its local CC

If necessary, Host commands wakeup of second channel and waits predefined time to allow the other nodes to wakeup and configure themselves

Host commands CC to enter STARTUP state by writing **FRSUCC1.CMD** = "0100" (RUN command)

(2) **Wakeup pattern (WUP)**

The wakeup pattern (WUP) is composed of at least two wakeup symbols (WUS). Wakeup symbol and wakeup pattern are configured by registers **FRPRTC1** and **FRPRTC2**.

Single channel wakeup, wakeup symbol may not be sent on both channels at the same time

Wakeup symbol collision resilient for at least two sending nodes (two overlapping wakeup symbols always recognizable)

Wakeup symbol must be configured identical in all nodes of a cluster

Wakeup symbol transmit low time configured by **FRPRTC2.TXL**

Wakeup symbol idle time used to listen for activity on the bus, configured by **FRPRTC2.TXI**

A wakeup pattern composed of at least two Tx-wakeup symbols needed for wakeup

Number of repetitions configurable by **FRPRTC1.RWP** (2 to 63 repetitions)

Wakeup symbol receive window length configured by **FRPRTC1.RXW**

Wakeup symbol receive low time configured by **FRPRTC2.RXL**

Wakeup symbol receive idle time configured by **FRPRTC2.RXI**

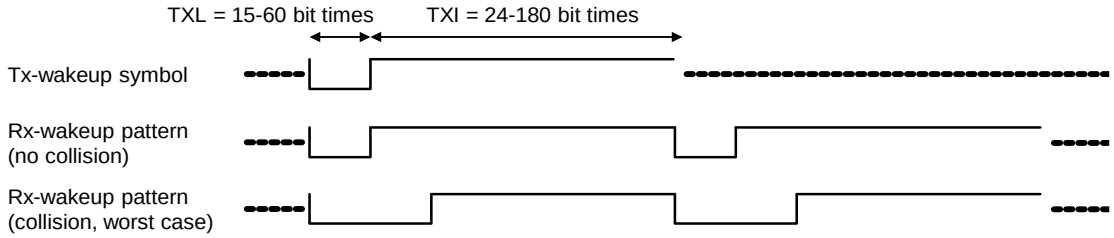



**Figure 103.7 Timing of wakeup pattern**

#### 103.3.6.6 STARTUP State

The description below is intended to help configuring startup for the FlexRay IP-module. A detailed description of the startup procedure together with the respective SDL diagrams can be found in the FlexRay protocol specification [FR_PROT], section 7.2.

Any node entering STARTUP state that has coldstart capability should assure that both channels attached have been awakened before initiating coldstart.

It cannot be assumed that all nodes and stars need the same amount of time to become completely awake and to be configured. Since at least two nodes are necessary to start up the cluster communication, it is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal amount of time it takes another coldstart node to become awake, to be configured and to enter startup. It may require several hundred milliseconds (depending on the hardware used) before all nodes and stars are completely awakened and configured.

Startup is performed on all channels synchronously. During startup, a node only transmits startup frames. Startup frames are both sync frames and null frames during startup.

A fault-tolerant, distributed startup strategy is specified for initial synchronization of all nodes. In general, a node may enter NORMAL_ACTIVE state via (see Figure 103.8):

Coldstart path initiating the schedule synchronization (leading coldstart node)

Coldstart path joining other coldstart nodes (following coldstart node)

Integration path integrating into an existing communication schedule (all other nodes)

A coldstart attempt begins with the transmission of a collision avoidance symbol (CAS). Only a coldstart node that had transmitted the CAS transmits frames in the first four cycles after the CAS, it is then joined firstly by the other coldstart nodes and afterwards by all other nodes.

A coldstart node has bits **FRSUCC1.TXST** and **FRSUCC1.TXSY** set to ’1’. Message buffer 0 holds the key slot ID which defines the slot number where the startup frame is send. In the frame header of the startup frame the startup frame indicator bit is set.

In clusters consisting of three or more nodes, at least three nodes shall be configured to be coldstart nodes. In clusters consisting of two nodes, both nodes must be coldstart nodes. At least two fault-free coldstart nodes are necessary for the cluster to startup.

Each startup frame must also be a sync frame; therefore each coldstart node will also be a sync node. The number of coldstart attempts is configured by **FRSUCC1.CSA**.

A non-coldstart node requires at least two startup frames from distinct nodes for integration. It may start integration before the coldstart nodes have finished their startup. It will not finish its startup until at least two coldstart nodes have finished their startup.

Both non-coldstart nodes and coldstart nodes start passive integration via the integration path as soon as they receive sync frames from which to derive the TDMA schedule information. During integration, the node has to adapt its own clock to the global clock (rate and offset) and has to make its cycle time consistent with the global schedule observable at the network. Afterwards, these settings are checked for consistency with all available network nodes. The node can only leave the integration phase and actively participate in communication when these checks are passed.

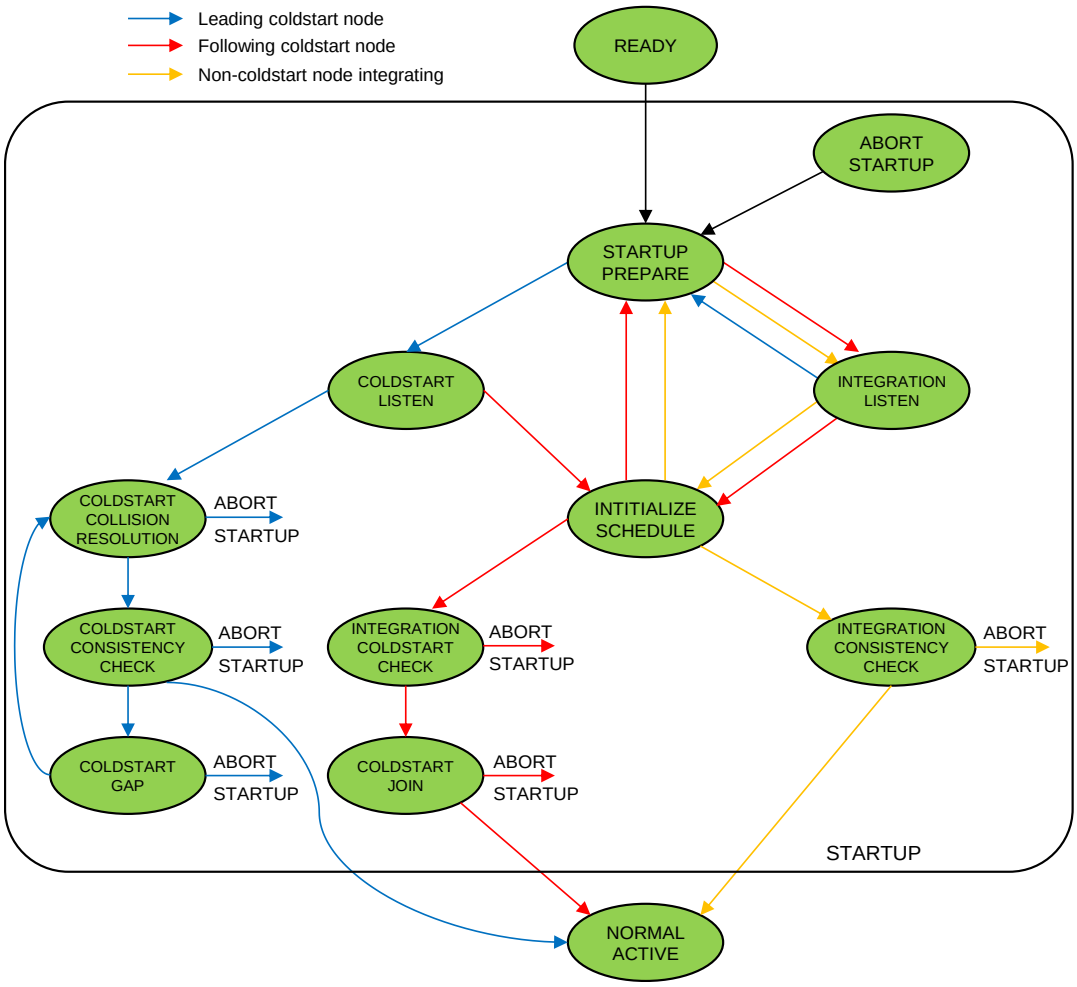



**Figure 103.8 State diagram time-triggered startup**

(1) **Coldstart Inhibit Mode**

In coldstart inhibit mode the node is prevented from initializing the TDMA communication schedule. If bit **FRCCSV.CSI** is set, the node is not allowed to initialize the cluster communication, i.e. entering the coldstart path is prohibited. The node is allowed to integrate to a running cluster or to transmit startup frames after another coldstart node started the initialization of the cluster communication.

The coldstart inhibit bit **FRCCSV.CSI** is set whenever the POC enters READY state. The bit has to be cleared under control of the Host by CHI command ALLOW_COLDSTART (**FRSUCC1.CMD** = "1001")

(2) **Startup Timeouts**

The CC supplies two different µT timers supporting two timeout values, startup timeout and startup noise timeout. The two timers are started when the CC enters the COLDSTART_LISTEN state. The expiration of either of these timers causes the node to leave the initial sensing phase (COLDSTART_LISTEN state) with the intention of starting up communication.

**Note:** The startup and startup noise timers are identical with the wakeup and wakeup noise timers and use the same configuration values **FRSUCC2.LT** and **FRSUCC2.LTN**.

**(a) Startup Timeout**

The startup timeout limits the listen time used by a node to determine if there is already communication between other nodes or at least one coldstart node actively requesting the integration of others. The startup timer is configured by programming **FRSUCC2.LT** (see section 103.2.5.2 ).

The startup timeout is:
pdListenTimeout = **FRSUCC2.LT**

The startup timer is restarted upon:

Entering the COLDSTART_LISTEN state

Both channels reaching idle state while in COLDSTART_LISTEN state
The startup timer is stopped:

If communication channel activity is detected on one of the configured channels while the node is in the COLDSTART_LISTEN state

When the COLDSTART_LISTEN state is left
Once the startup timeout expires, neither an overflow nor a cyclic restart of the timer is performed. The timer status is kept for further processing by the startup state machine.

**(b) Startup Noise Timeout**

At the same time the startup timer is started for the first time (transition from STARTUP_PREPARE state to COLDSTART_LISTEN state), the startup noise timer is started. This additional timeout is used to improve reliability of the startup procedure in the presence of noise. The startup noise timeout is configured by programming **FRSUCC2.LTN** (see section 103.2.5.2).

The startup noise timeout is:
pdListenTimeout * gListenNoise = **FRSUCC2.LT** * (**FRSUCC2.LTN** + 1)

The startup noise timer is restarted upon:

Entering the COLDSTART_LISTEN state

Reception of correctly decoded headers or CAS symbols while the node is in COLDSTART_LISTEN state
The startup noise timer is stopped when the COLDSTART_LISTEN state is left.

Once the startup noise timeout expires, neither an overflow nor a cyclic restart of the timer is performed. The status is kept for further processing by the startup state machine. Since the startup noise timer won't be restarted when random

channel activity is sensed, this timeout defines the fall-back solution that guarantees that a node will try to start up the communication cluster even in the presence of noise.

(3) **Path of leading Coldstart Node (initiating coldstart)**

When a coldstart node enters COLDSTART_LISTEN, it listens to its attached channels.

If no communication is detected, the node enters the COLDSTART_COLLISION_RESOLUTION state and commences a coldstart attempt. The initial transmission of a CAS symbol is succeeded by the first regular cycle. This cycle has the number zero.

From cycle zero on, the node transmits its startup frame. Since each coldstart node may perform a coldstart attempt, it may occur that several nodes simultaneously transmit the CAS symbol and enter the coldstart path. This situation is resolved during the first four cycles after CAS transmission.

As soon as a node that initiates a coldstart attempt receives a CAS symbol or a frame header during these four cycles, it re-enters the COLDSTART_LISTEN state. Thereby, only one node remains in this path. In cycle four, other coldstart nodes begin to transmit their startup frames.

After four cycles in COLDSTART_COLLISION_RESOLUTION state, the node that initiated the coldstart enters the COLDSTART_CONSISTENCY_CHECK state. It collects all startup frames from cycle four and five and performs the clock correction. If the clock correction does not deliver any errors and it has received at least one valid startup frame pair, the node leaves COLDSTART_CONSISTENCY_CHECK and enters NORMAL_ACTIVE state.

The number of coldstart attempts that a node is allowed to perform is configured by **FRSUCC1.CSA**. The number of remaining coldstarts attempts can be read from **FRCCSV.RCA**. The number of remaining coldstart attempts is reduced by one for each attempted coldstart. A node may enter the COLDSTART_LISTEN state only if this value is larger than one and it may enter the COLDSTART_COLLISION_RESOLUTION state only if this value is larger than zero. If the number of coldstart attempts is one, coldstart is inhibited but integration is still possible.

(4) **Path of following Coldstart Node (responding to leading Coldstart Node)**

When a coldstart node enters the COLDSTART_LISTEN state, it tries to receive a valid pair of startup frames to derive its schedule and clock correction from the leading coldstart node.

As soon as a valid startup frame has been received the INITIALIZE_SCHEDULE state is entered. If the clock synchronization can successfully receive a matching second valid startup frame and derive a schedule from this, the INTEGRATION_COLDSTART_CHECK state is entered.

In INTEGRATION_COLDSTART_CHECK state it is assured that the clock correction can be performed correctly and that the coldstart node from which this node has initialized its schedule is still available. The node collects all sync frames and performs clock correction in the following double-cycle. If clock correction does not signal any errors and if the node continues to receive sufficient frames from the same node it has integrated on, the COLDSTART_JOIN state is entered.

In COLDSTART_JOIN state following coldstart nodes begin to transmit their own startup frames and continue to do so in subsequent cycles. Thereby, the leading coldstart node and the nodes joining it can check if their schedules agree with each other. If the clock correction signals any error, the node aborts the integration attempt. If a node in this state sees at least one valid startup frame during all even cycles in this state and at least one valid startup frame pair during all double cycles in this state, the node leaves COLDSTART_JOIN state and enters NORMAL_ACTIVE state. Thereby it leaves STARTUP at least one cycle after the node that initiated the coldstart.

(5) **Path of Non-coldstart Node**

When a non-coldstart node enters the INTEGRATION_LISTEN state, it listens to its attached channels.

As soon as a valid startup frame has been received, the INITIALIZE_SCHEDULE state is entered. If the clock synchronization can successfully receive a matching second valid startup frame and derive a schedule from this, the INTEGRATION_CONSISTENCY_CHECK state is entered.

In INTEGRATION_CONSISTENCY_CHECK state the node verifies that the clock correction can be performed correctly and that enough coldstart nodes (at least 2) are sending startup frames that agree with the node's own schedule. Clock correction is activated, and if any errors are signaled, the integration attempt is aborted.

During the first even cycle in this state, either two valid startup frames or the startup frame of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

During the first double-cycle in this state, either two valid startup frame pairs or the startup frame pair of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

If after the first double-cycle less than two valid startup frames are received within an even cycle, or less than two valid startup frame pairs are received within a double-cycle, the startup attempt is aborted.

Nodes in this state need to see two valid startup frame pairs for two consecutive double-cycles each to be allowed to leave STARTUP and enter NORMAL_OPERATION. Consequently, they leave startup at least one double-cycle after the node that initiated the coldstart and only at the end of a cycle with an odd cycle number.

#### 103.3.6.7 NORMAL_ACTIVE State

As soon as the node that transmitted the first CAS symbol (resolving the potential access conflict and entering STARTUP via coldstart path) and one additional node have entered the NORMAL_ACTIVE state, the startup phase for the cluster has finished. In the NORMAL_ACTIVE state, all configured messages are scheduled for transmission. This includes all data frames as well as the sync frames. Rate and offset measurement is started in all even cycles (even / odd cycle pairs required).

In NORMAL_ACTIVE state the CC supports regular communication functions

The CC performs transmissions and reception on the FlexRay bus as configured

Clock synchronization is running

The Host interface is operational

The CC exits from that state to

HALT state by writing **FRSUCC1.CMD** = "0110" (HALT command, at the end of the current cycle)

HALT state by writing **FRSUCC1.CMD** = "0111" (FREEZE command, immediately)

HALT state due to change of the error state from ACTIVE to COMM_HALT

NORMAL_PASSIVE state due to change of the error state from ACTIVE to PASSIVE

READY state by writing **FRSUCC1.CMD** = "0010" (READY command)

#### 103.3.6.8 NORMAL_PASSIVE State

NORMAL_PASSIVE state is entered from NORMAL_ACTIVE state when the error state changes from ACTIVE to PASSIVE.

In NORMAL_PASSIVE state, the node is able to receive all frames (node is fully synchronized and performs clock synchronization). Contrary to the NORMAL_ACTIVE state, the node does not actively participate in communication, i.e. neither symbols nor frames are transmitted.

In NORMAL_PASSIVE state

The CC performs reception on the FlexRay bus

The CC does not transmit any frames or symbols on the FlexRay bus

Clock synchronization is running

The Host interface is operational

The CC exits from this state to
HALT state by writing **FRSUCC1.CMD** = "0110" (HALT command, at the end of the current cycle)
HALT state by writing **FRSUCC1.CMD** = "0111" (FREEZE command, immediately)
HALT state due to change of the error state from PASSIVE to COMM_HALT
NORMAL_ACTIVE state due to change of the error state from PASSIVE to ACTIVE. The transition takes place when **FRCCEV.PTAC** equals **FRSUCC1.PTA** - 1
To READY state by writing **FRSUCC1.CMD** = "0010" (READY command)

#### 103.3.6.9 HALT State

In this state all communication (reception and transmission) is stopped.

The CC enters this state
By writing **FRSUCC1.CMD** = "0110" (HALT command) while the CC is in NORMAL_ACTIVE or NORMAL_PASSIVE state
By writing **FRSUCC1.CMD** = "0111" (FREEZE command) from all states
When exiting from NORMAL_ACTIVE state because the clock correction failed counter reached the "maximum without clock correction fatal" limit and **FRSUCC1.HCSE** is set
When exiting from NORMAL_PASSIVE state because the clock correction failed counter reached the "maximum without clock correction fatal" limit and **FRSUCC1.HCSE** is set
The CC exits from this state to DEFAULT_CONFIG state
By writing **FRSUCC1.CMD** = "0001" (CONFIG command)
When the CC enters HALT state, all configuration and status data is maintained for analysing purposes.

When the Host writes **FRSUCC1.CMD** = "0110" (HALT command), the CC sets bit **FRCCSV.HRQ** and enters HALT state at the next end of cycle.

When the Host writes **FRSUCC1.CMD** = "0111" (FREEZE command), the CC enters HALT state immediately and sets bit **FRCCSV.FSI**.

The POC state from which the transition to HALT state took place can be read from **FRCCSV.PSL**.

### 103.3.7 Network Management

The accrued Network Management (NM) vector can be read from registers **FRNMV1/2/3**. The CC performs a bit-wise OR operation over all NM vectors out of all received valid NM frames with the Payload Preamble Indicator (PPI) bit set. Only static frames may be configured to hold NM information. The CC updates the NM vector at the end of each cycle.

The length of the NM vector can be configured from 0 to 12 bytes by **FRNEMC.NML**. The NM vector length must be configured identically in all nodes of a cluster.

To configure a transmit buffer to send FlexRay frames with the PPI bit set, bit PPIT in the header section of the respective transmit buffer has to be set via **FRWRHS1.PPIT**. In addition the Host has to write the NM information to the data section of the respective transmit buffer.

The evaluation of the NM vector has to be done by the application running on the Host.

For information about the byte alignment of the received NM vector in the **FRNMV1/2/3** registers see section 103.3.17.

**Note 1:** In case a message buffer is configured for transmission / reception of network management frames, the payload length configured in header 2 of that message buffer should be equal or greater than the length of the NM vector configured by **FRNEMC.NML**.

**Note 2:** When the CC transits to HALT state, the cycle count is not incremented and therefore the NM vector is not updated. In this case **FRNMV1/2/3** holds the value from the cycle before.

### 103.3.8 Filtering and Masking

Filtering is done by comparison of the configuration of assigned message buffers against actual slot and cycle counter values and channel ID (channel A, B). A message buffer is only updated / transmitted if the required matches occur.

Filtering is done on:

Slot Counter

Cycle Counter

Channel ID

The following filter combinations for acceptance / transmit filtering are allowed:

Slot Counter + Channel ID

Slot Counter + Cycle Counter + Channel ID

All configured filters must match in order to store a received message in a message buffer.

**Note:** For the FIFO the acceptance filter is configured by the FIFO Rejection Filter (**FRFRF**) and the FIFO Rejection Filter Mask (**FRFRFM**).

A message will be transmitted in the time slot corresponding to the configured frame ID on the configured channel(s). If cycle counter filtering is enabled the configured cycle filter value must also match.

#### 103.3.8.1 Slot Counter Filtering

Every transmit and receive buffer contains a frame ID stored in the header section. This frame ID is compared against the actual slot counter value in order to assign receive and transmit buffers to the corresponding slot.

If two or more message buffers are configured with the same frame ID and channel ID, and if they have a matching cycle counter filter value for the same slot, then the message buffer with the lowest message buffer number is used.

#### 103.3.8.2 Cycle Counter Filtering

Cycle counter filtering is based on the notion of a cycle set. For filtering purposes, a match is detected if any one of the elements of the cycle set is matched. The cycle set is defined by the cycle code field in header section 1 of each message buffer.

If message buffer 0 resp. 1 is configured to hold the startup / sync frame or the single slot frame by bits **FRSUCC1.TXST**, **FRSUCC1.TXSY**, and **FRSUCC1.TSM**, cycle counter filtering for message buffer 0 resp. 1 shall be disabled.

**Note:** Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay network is not allowed.

The set of cycle numbers belonging to a cycle set is determined as described in Table 103.7.

**Table 103.7 Definition of cycle set**

| Cycle Code | Matching Cycle Counter Values |
|---|---|
| 0b000000x | all Cycles |
| 0b000001c | every second Cycle at (Cycle Count)mod2 = c |

| | | | |
|---|---|---|---|
| 0b00001cc | every fourth Cycle | at (Cycle Count)mod4 | = cc |
| 0b0001ccc | every eighth Cycle | at (Cycle Count)mod8 | = ccc |
| 0b001cccc | every sixteenth Cycle | at (Cycle Count)mod16 | = cccc |
| 0b01ccccc | every thirty-second Cycle | at (Cycle Count)mod32 | = ccccc |
| 0b1cccccc | every sixty-fourth Cycle | at (Cycle Count)mod64 | = cccccc |

Table 103.8 below gives some examples for valid cycle sets to be used for cycle counter filtering:

**Table 103.8 Examples for valid cycle sets**

| Cycle Code | Matching Cycle Counter Values |
|---|---|
| 0b0000011 | 1-3-5-7- …. -63 |
| 0b0000100 | 0-4-8-12- …. -60 |
| 0b0001110 | 6-14-22-30- …. -62 |
| 0b0011000 | 8-24-40-56 |
| 0b0100011 | 3-35 |
| 0b1001001 | 9 |

The received message is stored only if the cycle counter value of the cycle during which the message is received matches an element of the receive buffer's cycle set. Other filter criteria must also be met.

The content of a transmit buffer is transmitted on the configured channel(s) when an element of the cycle set matches the current cycle counter value. Other filter criteria must also be met.

#### 103.3.8.3 Channel ID Filtering

There is a 2-bit channel filtering field (CH) located in the header section of each message buffer in the Message RAM. It serves as a filter for receive buffers, and as a control field for transmit buffers (see Table 103.9).

**Table 103.9 Channel filtering configuration**

| CH[1:0] | Transmit Buffer<br>transmit frame | Receive Buffer<br>store valid receive frame |
|---|---|---|
| 00 | no transmission | ignore frame |
| 01 | on channel A | received on channel A |
| 10 | on channel B | received on channel B |
| 11 | on both channels<br>(static segment only) | received on channel A or B<br>(store first semantically valid frame, static segment only) |

The contents of a transmit buffer is transmitted on the channels specified in the channel filtering field when the slot counter filtering and cycle counter filtering criteria are also met. Only in static segment a transmit buffer may be set up for transmission on both channels (CH = "11").

Valid received frames are stored if they are received on the channels specified in the channel filtering field when the slot counter filtering and cycle counter filtering criteria are also met. Only in static segment a receive buffer may be setup for reception on both channels (CH = "11").

**Note:** If a message buffer is configured for the dynamic segment and both bits of the channel filtering field are set to '1', no frames are transmitted resp. received frames are ignored (same function as CH = "00").

#### 103.3.8.4 FIFO Filtering

For FIFO filtering there is one rejection filter and one rejection filter mask available. The FIFO filter consists of channel filter **FRFRF.CH**, frame ID filter **FRFRF.FID**, and cycle counter filter **FRFRF.CYF**. Registers **FRFRF** and **FRFRFM** can be configured in DEFAULT_CONFIG or CONFIG state only. The filter configuration in the header section of message buffers belonging to the FIFO is ignored.

The 7-bit cycle counter filter determines the cycle set to which frame ID and channel rejection filter are applied. In cycles not belonging to the cycle set specified by **FRFRF.CYF**, all frames are rejected.

A valid received frame is stored in the FIFO if channel ID, frame ID, and cycle counter are not rejected by the configured rejection filter and rejection filter mask, and if there is no matching dedicated receive buffer.

### 103.3.9 Transmit Process

#### 103.3.9.1 Static Segment

For the static segment, if there are several messages pending for transmission, the message with the frame ID corresponding to the next sending slot is selected for transmission.

The data section of transmit buffers assigned to the static segment can be updated until the end of the preceding time slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

#### 103.3.9.2 Dynamic Segment

In the dynamic segment, if several messages are pending, the message with the highest priority (lowest frame ID) is selected next. In the dynamic segment different slot counter sequences on channel A and channel B are possible (concurrent sending of different frame IDs on both channels).

The data section of transmit buffers assigned to the dynamic segment can be updated until the end of the preceding slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

The start of latest transmit configured by **FRMHDC.SLT** defines the maximum minislot value allowed before inhibiting new frame transmission in the dynamic segment of the current cycle.

#### 103.3.9.3 Transmit Buffers

FlexRay message buffers can be configured as transmit buffers by programming bit CFG in the header section of the respective message buffer to ’1’ via **FRWRHS1**.

There exist the following possibilities to assign a transmit buffer to the CC channels:

Static segment: channel A or channel B,
channel A and channel B

Dynamic segment: channel A or channel B

Message buffer 0 resp. 1 is dedicated to hold the startup frame, the sync frame, or the designated single slot frame as configured by **FRSUCC1.TXST**, **FRSUCC1.TXSY**, and **FRSUCC1.TSM**. In this case, it can be reconfigured in DEFAULT_CONFIG or CONFIG state only. This ensures that any node transmits at most one startup / sync frame per communication cycle. Transmission of startup / sync frames from other message buffers is not possible.

All other message buffers configured for transmission in static or dynamic segment are reconfigurable during runtime depending on the configuration of **FRMRC.SEC** (see section 103.3.12.1 ). Due to the organization of the data partition in the Message RAM (reference by data pointer), reconfiguration of the configured payload length and the data pointer in the header section of a message buffer may lead to erroneous configurations.

If a message buffer is reconfigured (header section updated) during runtime, it may happen that this message buffer is not send out in the respective communication cycle.

The CC does not have the capability to calculate the header CRC. The Host is supposed to provide the header CRCs for all transmit buffers. If network management is required, the Host has to set the **FRWRHS1.PPIT** bit in the header

section of the respective message buffer to ’1’ and write the network management information to the data section of the message buffer (see section 103.3.7).

The payload length field configures the payload length in 2-byte words. If the configured payload length of a static transmit buffer is shorter than the payload length configured for the static segment by **FRMHDC.SFDL**, the CC generates padding bytes to ensure that frames have proper physical length. The padding pattern is logical zero.

**Note:** In case of an odd payload length (**FRWRHS2.PLC** = 1,3,5,...) the application has to write zero to the last 16 bit of the message buffers data section to ensure that the padding pattern is all zero.

Each transmit buffer provides a transmission mode flag **FRWRHS1.TXM** that allows the Host to configure the transmission mode for the transmit buffer. If this bit is set, the transmitter operates in the single-shot mode. If this bit is cleared, the transmitter operates in the continuous mode.

In single-shot mode the CC resets the respective **TXR** flag after transmission has completed. Now the Host may update the transmit buffer.

In continuous mode, the CC does not reset the respective transmission request flag **TXR** after successful transmission. In this case a frame is sent out each time the filter criteria match. The **TXR** flag can be reset by the Host by writing the respective message buffer number to the **FRIBCR** register while bit **FRIBCM.STXRH** is set to ’0’.

If two or more transmit buffers meet the filter criteria simultaneously, the transmit buffer with the lowest message buffer number will be transmitted in the respective slot.

#### 103.3.9.4 Frame Transmission

The following steps are required to prepare a message buffer for transmission:

Configure the transmit buffer in the Message RAM via **FRWRHS1**, **FRWRHS2**, and **FRWRHS3**

Write the data section of the transmit buffer via **FRWRDSn**

Transfer the configuration and message data from Input Buffer to the Message RAM by writing the number of the target message buffer to register **FRIBCR**

If configured in register **FRIBCM**, the transmission request flag **TXR** for the respective message buffer will be set as soon as the transfer has completed, and the message buffer is ready for transmission.

Check whether the message buffer has been transmitted by checking the respective **TXR** bit (**FRTXRQn.TXR** = ’0’) in the **FRTXRQ1/2/3/4** registers (single-shot mode only).

After transmission has completed, the respective **TXR** flag in the **FRTXRQ1/2/3/4** register is reset (single-shot mode), and, if bit **MBI** in the header section of the message buffer is set, flag **FRSIR.TXI** is set to ’1’. If enabled, an interrupt is generated.

#### 103.3.9.5 Null Frame Transmission

If in static segment the Host does not set the transmission request flag before transmit time, the CC transmits a null frame with the null frame indication bit set to ’0’ and the payload data set to zero.

In the following cases the CC transmits a null frame:

If the message buffer with the lowest message buffer number matching the filter criteria does not have its transmission request flag set (**TXR** = ’0’).

No transmit buffer configured for the slot has a cycle counter filter that matches the current cycle. In this case, no message buffer status **FRMBS** is updated.

Null frames are not transmitted in the dynamic segment.

### 103.3.10 Receive Process

#### 103.3.10.1 Dedicated Receive Buffers

A portion of the FlexRay message buffers can be configured as dedicated receive buffers by programming bit **CFG** in the header section of the respective message buffer to ’0’ via **FRWRHS1**.

The following possibilities exist to assign a receive buffer to the CC channels:

Static segment: channel A or channel B,
channel A and channel B (the CC stores the first semantically valid frame)

Dynamic segment: channel A or channel B

The CC transfers the payload data of valid received messages from the shift register of the FlexRay channel protocol controller (channel A or B) to the receive buffer with the matching filter configuration. A receive buffer stores all frame elements except the frame CRC.

All message buffers configured for reception in static or dynamic segment are reconfigurable during runtime depending on the configuration of **FRMRC.SEC** (see section 103.3.12.1). If a message buffer is reconfigured (header section updated) during runtime it may happen that in the respective communication cycle a received message is lost.

If two or more receive buffers meet the filter criteria simultaneously, the receive buffer with the lowest message buffer number is updated with the received message.

#### 103.3.10.2 Frame Reception

The following steps are required to prepare a dedicated message buffer for reception:

Configure the receive buffer in the Message RAM via **FRWRHS1**, **FRWRHS2**, and **FRWRHS3**

Transfer the configuration from Input Buffer to the Message RAM by writing the number of the target message buffer to register **FRIBCR**

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal acceptance filtering process which takes place every time the CC receives a message. The first matching receive buffer is updated from the received message.

If a valid payload segment was stored in the data section of a message buffer, the respective **ND** flag in the **FRNDAT1/2/3/4** registers is set, and, if bit **MBI** in the header section of that message buffer is set, flag **FRSIR.RXI** is set to ’1’. If enabled, an interrupt is generated.

In case that bit **ND** was already set when the Message Handler updates the message buffer, bit **FRMBS.MLST** of the respective message buffer is set and the unprocessed message data is lost.

If no frame, a null frame, or a corrupted frame was received in a slot, the data section of the message buffer configured for this slot is not updated. In this case only the respective message buffer status **FRMBS** is updated.

When the Message Handler changed the message buffer status **FRMBS** in the header section of a message buffer, the respective **MBC** flag in the **FRMBSC1/2/3/4** registers is set, and if bit **MBI** in the header section of that message buffer is set, flag **FRSIR.MBSI** is set to ’1’. If enabled an interrupt is generated.

If the payload length of a received frame **PLR** is longer than the value programmed by **PLC** in the header section of the respective message buffer, the data field stored in the message buffer is truncated to that length.

To read a receive buffer from the Message RAM via the Output Buffer, proceed as described in section 103.3.12.2.(2).

**Note:** The **ND** and **MBC** flags are automatically cleared by the Message Handler when the payload data and the header of a received message have been transferred to the Output Buffer, respectively.

#### 103.3.10.3 Null Frame Reception

The payload segment of a received null frame is not copied into the matching dedicated receive buffer. If a null frame has been received, only the message buffer status **FRMBS** of the matching message buffer is updated from the received null frame. All bits in header 2 and 3 of the matching message buffer remain unchanged. They are updated from received data frames only.

When the Message Handler changed the message buffer status **FRMBS** in the header section of a message buffer, the respective **MBC** flag in the **FRMBSC1/2/3/4** register is set, and if bit **MBI** in the header section of that message buffer is set, flag **FRSIR.MBSI** is set to '1'. If enabled, an interrupt is generated.

### 103.3.11 FIFO Function

#### 103.3.11.1 Description

A group of the message buffers can be configured as a cyclic First-In-First-Out (FIFO) buffer. The group of message buffers belonging to the FIFO is contiguous in the register map starting with the message buffer referenced by **FRMRC.FFB** and ending with the message buffer referenced by **FRMRC.LCB**. Up to 127 message buffers can be assigned to the FIFO.

Every valid incoming message not matching with any dedicated receive buffer but passing the programmable FIFO filter is stored into the FIFO. In this case frame ID, payload length, receive cycle count, and the message buffer status **FRMBS** of the addressed FIFO message buffer are overwritten with frame ID, payload length, receive cycle count, and the status from the received frame. Bit **FRSIR.RFNE** shows that the FIFO is not empty, bit **FRSIR.RFCL** is set when the receive FIFO fill level **FRFSR.RFFL** is equal or greater than the critical level as configured by **FRFCL.CL**, bit **FREIR.RFO** shows that a FIFO overrun has been detected. If enabled, interrupts are generated.

If null frames are not rejected by the FIFO rejection filter, the null frames will be treated like data frames when they are stored into the FIFO.

There are two index registers associated with the FIFO. The PUT Index Register (PIDX) is an index to the next available location in the FIFO. When a new message has been received it is written into the message buffer addressed by the PIDX register. The PIDX register is then incremented and addresses the next available message buffer. If the PIDX register is incremented past the highest numbered message buffer of the FIFO, the PIDX register is loaded with the number of the first (lowest numbered) message buffer in the FIFO chain. The GET Index Register (GIDX) is used to address the next message buffer of the FIFO to be read. The GIDX register is incremented after transfer of the contents of a message buffer belonging to the FIFO to the Output Buffer. The PUT Index Register and the GET Index Register are not accessible by the Host.

The FIFO is completely filled when the PUT index (PIDX) reaches the value of the GET index (GIDX). When the next message is written to the FIFO before the oldest message has been read, both PUT index and GET index are incremented and the new message overwrites the oldest message in the FIFO. This will set FIFO overrun flag **FREIR.RFO**.

A FIFO non empty status is detected when the PUT index (PIDX) differs from the GET index (GIDX). In this case flag **FRSIR.RFNE** is set. This indicates that there is at least one received message in the FIFO. The FIFO empty, FIFO not empty, and the FIFO overrun states are explained in Figure 103.9 for a three message buffer FIFO.

The programmable FlexRay FIFO Rejection Filter (**FRFRF**) defines a filter pattern for messages to be rejected. The FIFO filter consists of channel filter, frame ID filter, and cycle counter filter. If bit **FRFRF.RSS** is set to '1', all messages received in the static segment are rejected by the FIFO. If bit **FRFRF.RNF** is set to '1', received null frames are not stored in the FIFO.

The FlexRay FIFO Rejection Filter Mask (**FRFRFM**) specifies which bits of the frame ID filter in the FIFO Rejection Filter register are marked 'don't care' for rejection filtering.

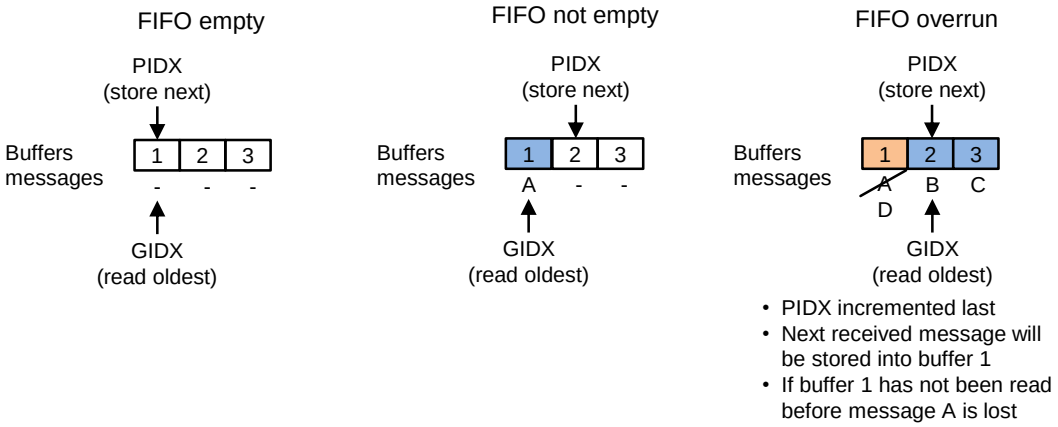



**Figure 103.9 FIFO status: empty, not empty, overrun**

### 103.3.11.2 Configuration of the FIFO

(Re)configuration of message buffers belonging to the FIFO is only possible when the CC is in DEFAULT_CONFIG or CONFIG state. While the CC is in DEFAULT_CONFIG or CONFIG state, the FIFO function is not available.

For all message buffers belonging to the FIFO the payload length configured should be programmed to the same value via **FRWRHS2.PLC**. The data pointer to the first 32-bit word of the data section of the respective message buffer in the Message RAM has to be configured via **FRWRHS3.DP**.

All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask. The values configured in the header sections of the message buffers belonging to the FIFO are, with exception of **DP** and **PLC**, irrelevant.

**Note 1:** It is recommended to program the **MBI** bits of the message buffers belonging to the FIFO to ’0’ via **FRWRHS1.MBI** to avoid generation of RX interrupts.

**Note 2:** If the payload length of a received frame is longer than the value programmed by **FRWRHS2.PLC** in the header section of the respective message buffer, the data field stored in a message buffer of the FIFO is truncated to that length.

### 103.3.11.3 Access to the FIFO

(1) **Using output buffer**

For FIFO access outside DEFAULT_CONFIG and CONFIG state, the Host has to trigger a transfer from the Message RAM to the Output Buffer by writing the number of the first message buffer of the FIFO (referenced by **FRMRC.FFB**) to the register **FROBCR**. The Message Handler then transfers the message buffer addressed by the GET Index Register (GIDX) to the Output Buffer. After this transfer the GET Index Register (GIDX) is incremented.

(2) **Using data transfer function**

Messages received into the FIFO can be also transferred into the URAM using the Output data transfer function as described in section 103.3.16.2.

### 103.3.12 Message Handling

The Message Handler controls data transfers between the Input / Output Buffer and the Message RAM and between the Message RAM and the two Transient Buffers.

Access to the message buffers stored in the Message RAM is done under control of the Message Handler state machine. This avoids conflicts between accesses of the two FlexRay channel protocol controllers and the Host to the Message RAM.

Frame IDs of message buffers assigned to the static segment have to be in the range from 1 to **FRGTUC7.NSS**. Frame IDs of message buffers assigned to the dynamic segment have to be in the range from **FRGTUC7.NSS** + 1 to 2047.

Received messages with no matching dedicated receive buffer (static or dynamic segment) are stored in the receive FIFO (if configured) if they pass the FIFO rejection filter.

This section describes the Host access to the Message buffer content using the input buffer and output buffer function. In section 103.3.16 the access to the Message buffer content using the data transfer function is described.

#### 103.3.12.1 Reconfiguration of Message Buffers

In case that an application needs to operate with more than 128 different messages, static and dynamic message buffers may be reconfigured during FlexRay operation. This is done by updating the header section of the respective message buffer via Input Buffer registers **FRWRHS1/2/3**.

Reconfiguration has to be enabled via control bits **FRMRC.SEC** in the Message RAM Configuration register.

If a message buffer has not been transmitted / updated from a received frame before reconfiguration starts, the respective message is lost.

The point in time when a reconfigured message buffer is ready for transmission / reception according to the reconfigured frame ID depends on the actual state of the slot counter when the update of the header section has completed. Therefore it may happen that a reconfigured message buffer is not transmitted / updated from a received frame in the cycle where it was reconfigured.

The Message RAM is scanned according to Table 103.10 below:

**Table 103.10 Scan of Message RAM**

| Start of Scan in Slot | Scan for Slots |
|---|---|
| 1 | 2…15, 1 (next cycle) |
| 8 | 16…23, 1 (next cycle) |
| 16 | 24…31, 1 (next cycle) |
| 24 | 32…39, 1 (next cycle) |
| … | … |

A Message RAM scan is terminated with the start of NIT regardless whether it has completed or not. The scan of the Message RAM for slots 2 to 15 starts at the beginning of slot 1 of the actual cycle. The scan of the Message RAM for slot 1 is done in the cycle before by checking in parallel to each scan of the Message RAM whether there is a message buffer configured for slot 1 of the next cycle.

The number of the first dynamic message buffer is configured by **FRMRC.FDB**. In case a Message RAM scan starts while the CC is in dynamic segment, the scan starts with the message buffer number configured by **FRMRC.FDB**.

In case a message buffer should be reconfigured to be used in slot 1 of the next cycle, the following has to be considered:
If the message buffer to be reconfigured for slot 1 is part of the "Static Buffers", it will only be found if it is reconfigured

before the last Message RAM scan in the static segment of the actual cycle evaluates this message buffer.
If the message buffer to be reconfigured for slot 1 is part of the "Static + Dynamic Buffers", it will be found if it is reconfigured before the last Message RAM scan in the actual cycle evaluates this message buffer.
The start of NIT terminates the Message RAM scan. In case the Message RAM scan has not evaluated the reconfigured message buffer until this point in time, the message buffer will not be considered for the next cycle.

**Note:** Reconfiguration of message buffers may lead to the loss of messages and therefore has to be used very carefully. In worst case (reconfiguration in consecutive cycles) it may happen that a message buffer is never transmitted / updated from a received frame.

### 103.3.12.2 Host access to Message RAM

The message transfer between Input Buffer and Message RAM as well as between Message RAM and Output Buffer is triggered by the Host by writing the number of the target / source message buffer to be accessed to **FRIBCR** or **FROBCR** register.

The **FRIBCM** and **FROBCM** registers can be used to write / read header and data section of the selected message buffer separately.

If bit **FRIBCM.STXR** is set to = ’1’, the transmission request flag **FRTXRQn.TXR** of the selected message buffer is automatically set after the message buffer has been updated. If bit **FRIBCM.STXR** is reset to ’0’, the transmission request flag **FRTXRQn.TXR** of the selected message buffer is reset. This can be used to stop transmission from message buffers operated in continuous mode.

Input Buffer (IBF) and Output Buffer (OBF) are build up as a double buffer structure. One half of this double buffer structure is accessible by the Host (IBF Host / OBF Host), while the other half (IBF Shadow / OBF Shadow) is accessed by the Message Handler for data transfers between IBF / OBF and Message RAM.

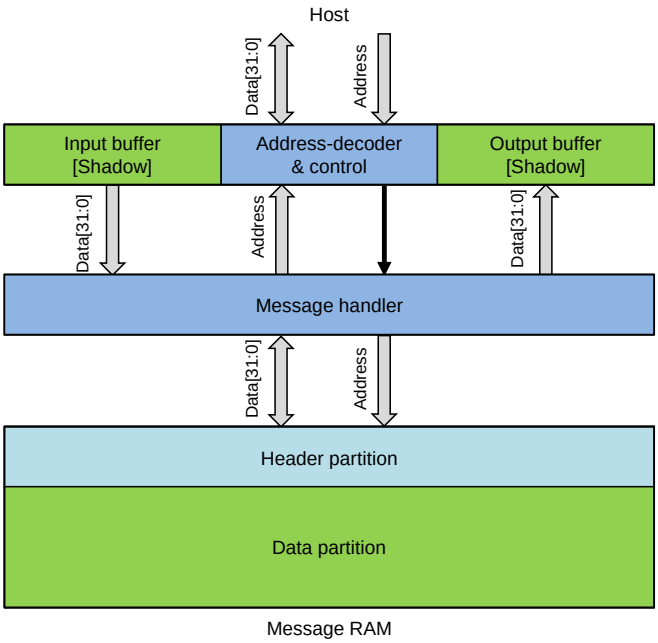



**Figure 103.10 Host access to Message RAM**

(1) **Data Transfer from Input Buffer to Message RAM**

To configure / update a message buffer in the Message RAM, the Host has to write the data to **FRWRDSn** and the header to **FRWRHS1**/**2**/**3**. The specific action is selected by configuring the FlexRay Input Buffer Command Mask **FRIBCM**.

When the Host writes the number of the target message buffer in the Message RAM to **FRIBCR.IBRH**, IBF Host and IBF Shadow are swapped (see Figure 103.11).

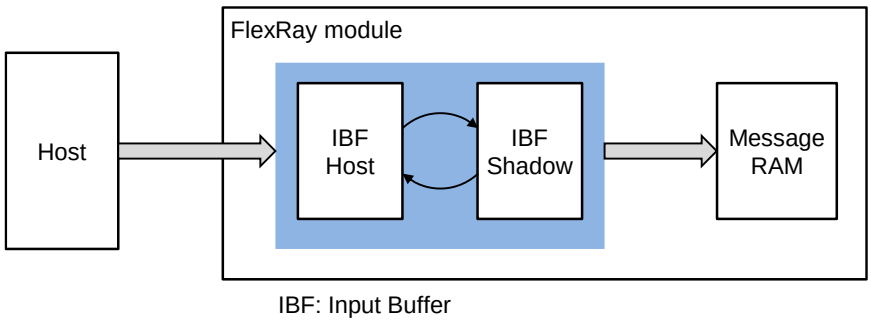



**Figure 103.11 Double buffer structure Input Buffer**

In addition the bits in the **FRIBCM** and **FRIBCR** registers are also swapped to keep them attached to the respective IBF section (see Figure 103.12).

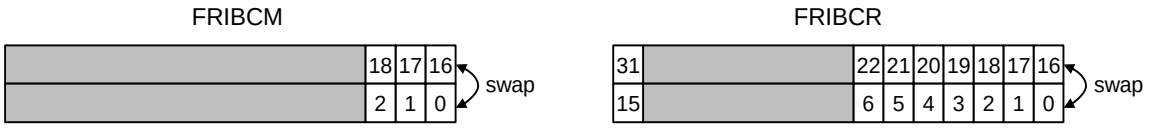



**Figure 103.12 Swapping of FRIBCM and FRIBCR bits**

With this write operation bit **FRIBCR.IBSYS** is set to '1'. The Message Handler then starts to transfer the contents of IBF Shadow to the message buffer in the Message RAM selected by **FRIBCR.IBRS**.

While the Message Handler transfers the data from IBF Shadow to the target message buffer in the Message RAM, the Host may write the next message to IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, bit **FRIBCR.IBSYS** is set back to '0' and the next transfer to the Message RAM may be started by the Host by writing the respective target message buffer number to **FRIBCR.IBRH**.

If a write access to **FRIBCR.IBRH** occurs while **FRIBCR.IBSYS** is '1', **FRIBCR.IBSYH** is set to '1'. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, **FRIBCR.IBSYH** is reset to '0', **FRIBCR.IBSYS** remains set to '1', and the next transfer to the Message RAM is started. In addition the message buffer numbers stored under **FRIBCR.IBRH** and **FRIBCR.IBRS** and the command mask flags are also swapped.

**Example of a 8/16/32-bit Host access sequence:**
Configure / update n-th message buffer via IBF
Wait until **FRIBCR.IBSYH** is reset
Write data section to **FRWRDSn**
Write header section to **FRWRHS1/2/3**
Write Command Mask: write **FRIBCM.STXRH**, **FRIBCM.LDSH**, **FRIBCM.LHSH**
Demand data transfer to target message buffer: write **FRIBCR.IBRH**
Configure / update (n+1)th message buffer via IBF
Wait until **FRIBCR.IBSYH** is reset
Write data section to **FRWRDSn**
Write header section to **FRWRHS1/2/3**

Write Command Mask: write **FRIBCM.STXRH**, **FRIBCM.LDSH**, **FRIBCM.LHSH**

Demand data transfer to target message buffer: write **FRIBCR.IBRH**

…

**Note:** Any write access to IBF while **FRIBCR.IBSYH** is '1' will set error flag **FREIR.IIBA** to '1'. In this case the write access has no effect.

**Table 103.11 Assignment of FRIBCM bits**

| Pos. | Access | Bit | Function |
|---|---|---|---|
| 18 | r | STXRS | Set Transmission Request Shadow ongoing or finished |
| 17 | r | LDSS | Load Data Section Shadow ongoing or finished |
| 16 | r | LHSS | Load Header Section Shadow ongoing or finished |
| 2 | r/w | STXRH | Set Transmission Request Host |
| 1 | r/w | LDSH | Load Data Section Host |
| 0 | r/w | LHSH | Load Header Section Host |

**Table 103.12 Assignment of FRIBCR bits**

| Pos. | Access | Bit | Function |
|---|---|---|---|
| 31 | r | IBSYS | IBF Busy Shadow, signals ongoing transfer from IBF Shadow to Message RAM |
| 22…16 | r | IBRS | IBF Request Shadow, number of message buffer currently / lately updated |
| 15 | r | IBSYH | IBF Busy Host, transfer request pending for message buffer referenced by IBRH |
| 6…0 | r/w | IBRH | IBF Request Host, number of message buffer to be updated next |

(2) **Data Transfer from Message RAM to Output Buffer**

To read a message buffer from the Message RAM, the Host has to write to register **FROBCR** to trigger the data transfer as configured in **FROBCM**. After the transfer has completed, the Host can read the transferred data from **FRRDDSn**, **FRRDHS1**/**2**/**3**, and **FRMBS**.

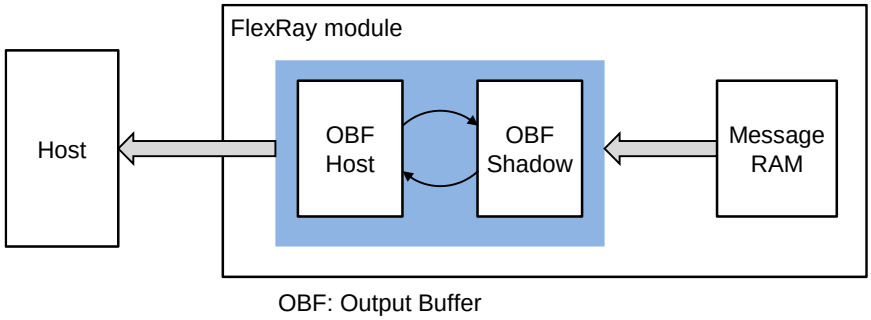



**Figure 103.13 Double buffer structure Output Buffer**

OBF Host and OBF Shadow as well as bits **FROBCM.RHSS**, **FROBCM.RDSS**, **FROBCM.RHSH**, **FROBCM.RDSH** and bits **FROBCR.OBRS**, **FROBCR.OBRH** are swapped under control of bits **FROBCR.VIEW** and **FROBCR.REQ**.

Writing bit **FROBCR.REQ** to ’1’ copies bits **FROBCM.RHSS**, **FROBCM.RDSS** and bits **FROBCR.OBRS** to an internal storage (see Figure 103.14).

After setting **FROBCR.REQ** to ’1’, **FROBCR.OBSYS** is set to ’1’, and the transfer of the message buffer selected by **FROBCR.OBRS** from the Message RAM to OBF Shadow is started. After the transfer between the Message RAM and OBF Shadow has completed, the **FROBCR.OBSYS** bit is set back to ’0’. Bits **FROBCR.REQ** and **FROBCR.VIEW** can only be set to ’1’ while **FROBCR.OBSYS** is ’0’.

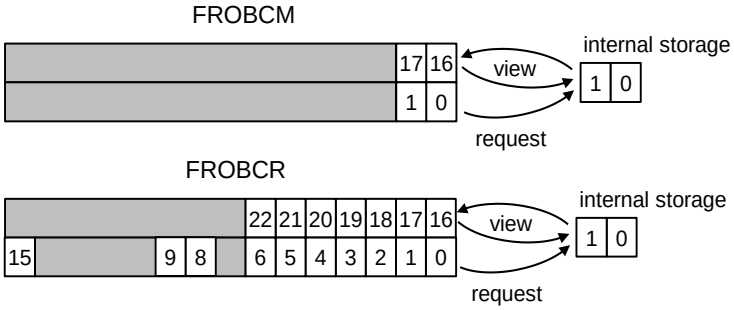



**Figure 103.14 Swapping of FROBCM and FROBCR bits**

OBF Host and OBF Shadow are swapped by setting bit **FROBCR.VIEW** to ’1’ while bit **FROBCR.OBSYS** is ’0’ (see Figure 103.13).

In addition bits **FROBCR.OBRH** and bits **FROBCM.RHSH**, **FROBCM.RDSH** are swapped with the registers internal storage thus assuring that the message buffer number stored in **FROBCR.OBRH** and the mask configuration stored in **FROBCM.RHSH**, **FROBCM.RDSH** matches the transferred data stored in OBF Host (see Figure 103.14).

Now the Host can read the transferred message buffer from OBF Host while the Message Handler may transfer the next message from the Message RAM to OBF Shadow.

If bits **REQ** and **VIEW** are set to ’1’ with the same write access while **FROBCR.OBSYS** is ’0’, **FROBCR.OBSYS** is automatically set to ’1’ and OBF Shadow and OBF Host are swapped. Additionally mask bits **FROBCM.RDSH** and **FROBCM.RHSH** are swapped with the registers internal storage to keep them attached to the respective Output Buffer transfer. Afterwards **FROBCR.OBRS** is copied to the register internal storage, mask bits **FROBCM.RDSS** and **FROBCM.RHSS** are copied to register **FROBCM** internal storage, and the transfer of the selected message buffer from the Message RAM to OBF Shadow is started. While the transfer is ongoing the Host can read the message buffer transferred by the previous transfer from OBF Host. When the current transfer between Message RAM and OBF Shadow has completed, this is signaled by setting **FROBCR.OBSYS** back to ’0’.

**Example of an 8/16/32-bit Host access to a single message buffer:**
If a single message buffer has to be read out, two separate write accesses to **FROBCR.REQ** and **FROBCR.VIEW** are necessary:

Wait until **FROBCR.OBSYS** is reset

Write Output Buffer Command Mask **FROBCM.RHSS**, **FROBCM.RDSS**

Request transfer of message buffer to OBF Shadow by writing **FROBCR.OBRS** and **FROBCR.REQ** (in case of and 8-bit Host interface, **FROBCR.OBRS** has to be written before **FROBCR.REQ**).

Wait until **FROBCR.OBSYS** is reset

Toggle OBF Shadow and OBF Host by writing **FROBCR.VIEW** = ’1’

Read out transferred message buffer by reading **FRRDDSn**, **FRRDHS1**/**2**/**3**, and **FRMBS**

**Example of an 8/16/32-bit Host access sequence:**
Request transfer of 1st message buffer to OBF Shadow

Wait until **FROBCR.OBSYS** is reset

Write Output Buffer Command Mask **FROBCM.RHSS**, **FROBCM.RDSS** for 1st message buffer

Request transfer of 1st message buffer to OBF Shadow by writing **FROBCR.OBRS** and **FROBCR.REQ** (in case of an 8-bit Host interface, **FROBCR.OBRS** has to be written before **FROBCR.REQ**).

Toggle OBF Shadow and OBF Host to read out 1st transferred message buffer and request transfer of 2nd message buffer:

Wait until **FROBCR.OBSYS** is reset

Write Output Buffer Command Mask **FROBCM.RHSS**, **FROBCM.RDSS** for 2nd message buffer

Toggle OBF Shadow and OBF Host and start transfer of 2nd message buffer to OBF Shadow simultaneously by writing **FROBCR.OBRS** of 2nd message buffer, **FROBCR.REQ**, and **FROBCR.VIEW** (in case of and 8-bit Host interface, **FROBCR.OBRS** has to be written before **FROBCR.REQ** and **FROBCR.VIEW**).

Read out 1st transferred message buffer by reading **FRRDDSn**, **FRRDHS1**/**2**/**3**, and **FRMBS**

…

Demand access to last requested message buffer without request of another message buffer:

Wait until **FROBCR.OBSYS** is reset

Demand access to last transferred message buffer by writing **FROBCR.VIEW**

Read out last transferred message buffer by reading **FRRDDSn**, **FRRDHS1/2/3**, and **FRMBS**

**Table 103.13 Assignment of FROBCM bits**

| Pos. | Access | Bit | Function |
|---|---|---|---|
| 17 | r | RDSH | Data Section available for Host access |
| 16 | r | RHSH | Header Section available for Host access |
| 1 | r/w | RDSS | Read Data Section Shadow |
| 0 | r/w | RHSS | Read Header Section Shadow |

**Table 103.14 Assignment of FROBCR bits**

| Pos. | Access | Bit | Function |
|---|---|---|---|
| 22…16 | r | OBRH | OBF Request Host,<br>number of message buffer available for Host access |
| 15 | r | OBSYS | OBF Busy Shadow,<br>signals ongoing transfer from Message RAM to OBF Shadow |
| 9 | r/w | REQ | Request Transfer from Message RAM to OBF Shadow |
| 8 | r/w | VIEW | View OBF Shadow, swap OBF Shadow and OBF Host |
| 6…0 | r/w | OBRS | OBF Request Shadow, number of message buffer for next request |

### 103.3.12.3 FlexRay Protocol Controller access to Message RAM

The two Transient Buffers (TBF A,B) are used to buffer the data for transfer between the two FlexRay Protocol Controllers and the Message RAM.

Each Transient Buffer is build up as a double buffer, able to store two complete FlexRay messages. There is always one buffer assigned to the corresponding Protocol Controller while the other one is accessible by the Message Handler.

If e.g. the Message Handler writes the next message to be send to Transient Buffer Tx, the FlexRay Channel Protocol Controller can access Transient Buffer Rx to store the message it is actually receiving. During transmission of the message stored in Transient Buffer Tx, the Message Handler transfers the last received message stored in Transient Buffer Rx to the Message RAM (if it passes acceptance filtering) and updates the respective message buffer.

Data transfers between the Transient Buffers and the shift registers of the FlexRay Channel Protocol Controllers are done in words of 32 bit. This enables the use of a 32 bit shift register independent of the length of the FlexRay messages.

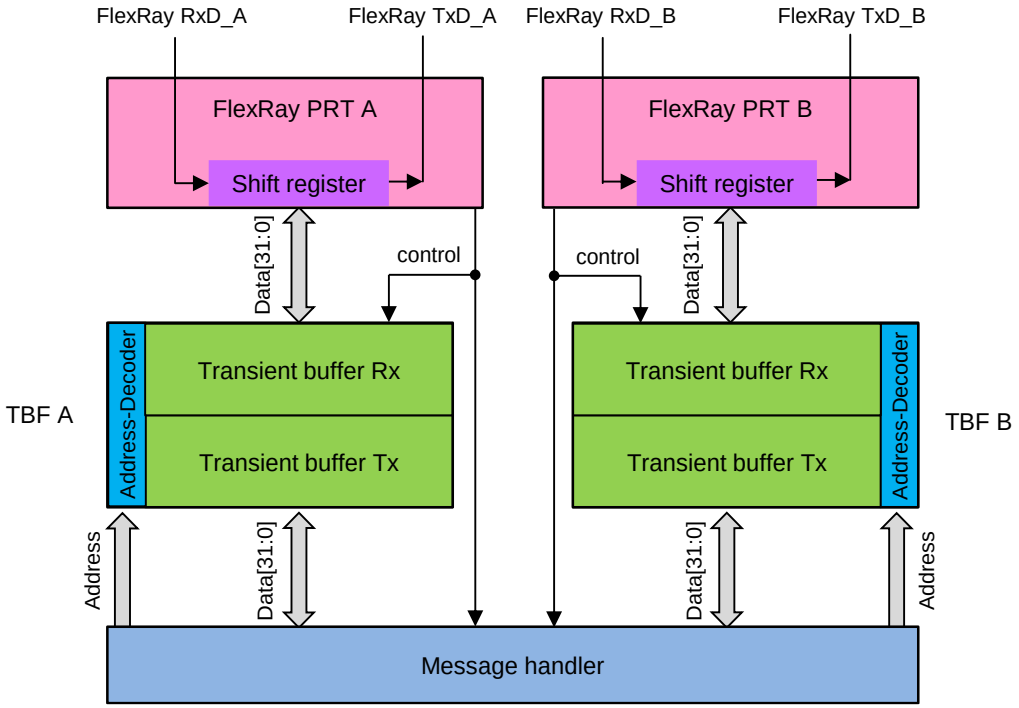



**Figure 103.15 Access to Transient Buffers**

### 103.3.13 Message RAM

To avoid conflicts between Host access to the Message RAM and FlexRay message reception / transmission, the Host cannot directly access the message buffers in the Message RAM. These accesses are handled via the Input and Output Buffers. The Message RAM is able to store up to 128 message buffers depending on the configured payload length.

The Message RAM is able to store up to 2048 32-bit words. To achieve the required flexibility with respect to different numbers of data bytes per FlexRay frame (0…254), the Message RAM has a structure as shown in **Figure 103.16**.

The data partition is allowed to start at Message RAM word number: (**FRMRC.LCB** + 1) * 4

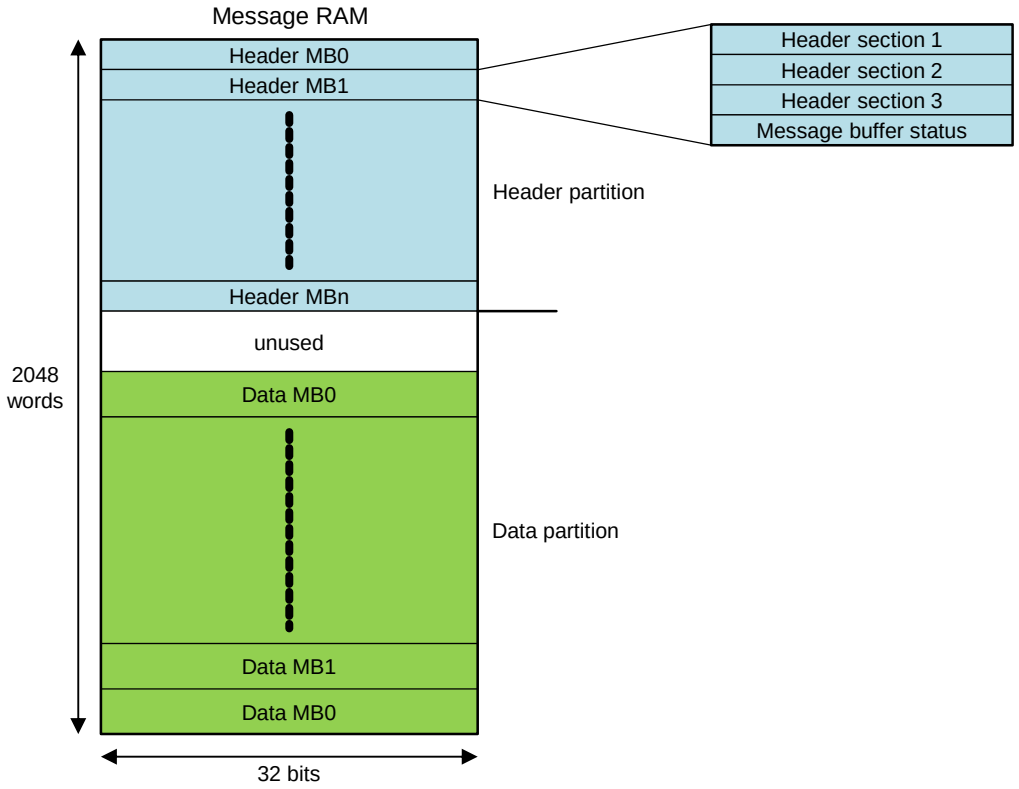



**Figure 103.16 Configuration example of message buffers in the Message RAM**

**Header Partition**
Stores header sections of the configured message buffers:

Supports a maximum of 128 message buffers

Each message buffer has a header section of four 32 bit words

Header section 3 of each message buffer holds the 11-bit data pointer to the respective data section in the data partition

**Data Partition**
Flexible storage of data sections with different length. Some maximum values are:

30 message buffers with 254 byte data section each

Or 56 message buffers with 128 byte data section each

Or 128 message buffers with 48 byte data section each

**Restriction:** header partition + data partition may not occupy more than 2048 32-bit words.

### 103.3.13.1 Header Partition

The elements used for configuration of a message buffer as well as the actual message buffer status are stored in the header partition of the Message RAM as listed in received valid data frames only.

Table 103.3 below. Configuration of the header sections of the message buffers is done via IBF (**FRWRHS1/2/3**). Read access to the header sections are done via OBF (**FRRDHS1/2/3** + **FRMBS**). The data pointer has to be calculated by the programmer to define the starting point of the data section for the respective message buffer in the data partition of the Message RAM. The data pointer should not be modified during runtime. For message buffers belonging to the receive FIFO (re)configuration is possible in DEFAULT_CONFIG or CONFIG state only.

The header section of each message buffer occupies four 32-bit words in the header partition of the Message RAM. The header of message buffer 0 starts with the first word in the Message RAM.

For transmit buffers the Header CRC has to be calculated by the Host.

Payload Length Received **PLR**, Receive Cycle Count **RCC**, Received on Channel Indicator **RCI**, Startup Frame Indicator **SFI**, Sync Frame Indicator **SYN**, Null Frame Indicator **NFI**, Payload Preamble Indicator **PPI**, and Reserved Bit **RES** are updated from received valid data frames only.

**Table 103.15 Header section of a message buffer in the Message RAM**

| Bit Word | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | MBI | TXM | PPIT | CFG | CH | | | Cycle code | | | | | | | | | | | | Frame ID | | | | | | | | | | |
| 1 | | Payload length received | | | | | | | | Payload length configured | | | | | | | | | | | | Tx buffer: Header CRC configured<br>Rx buffer: Header CRC received | | | | | | | | | | |
| 2 | | | RES | PPI | NFI | SYN | SFI | RCI | | | Receive cycle count | | | | | | | | | | | Data pointer | | | | | | | | | | |
| 3 | | | RESS | PPIS | NFIS | SYNS | SFIS | RCIS | | | Cycle count status | | | | | | FTB | FTA | | MLST | ESB | ESA | TCIB | TCIA | SVOB | SVOA | CEOB | CEOA | SEOB | SEOA | VFRB | VFRA |
| … | … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Frame configuration
Filter configuration
Message buffer control
Message RAM configuration
Updated from received data frame
Message buffer status (MBS)
unused

(1) **Header section 1 (word 0)**

Write access via **FRWRHS1**, read access via **FRRDHS1**:

Frame ID - Slot counter filtering configuration
Cycle Code - Cycle counter filtering configuration
CH - Channel filtering configuration
CFG - Message buffer direction configuration: receive / transmit
PPIT - Payload Preamble Indicator Transmit
TXM - Transmit mode configuration: single-shot / continuous
MBI - Message buffer receive / transmit interrupt enable

(2) **Header section 2 (word 1)**

Write access via **FRWRHS2**, read access via **FRRDHS2**:

Header CRC - Transmit Buffer: Configured by the Host (calculated from frame header)
- Receive Buffer: Updated from received frame

Payload Length Configured - Length of data section (2-byte words) as configured by the Host
Payload Length Received - Length of payload segment (2-byte words) stored from received frame

(3) **Header section 3 (word 2)**

Write access via **FRWRHS3**, read access via **FRRDHS3**:

Data Pointer - Pointer to the beginning of the corresponding data section in the data partition
Read access via **FRRDHS3** , valid for receive buffers only, updated from received frames:
Receive Cycle Count - Cycle count from received frame
RCI - Received on Channel Indicator

SFI - Startup Frame Indicator
SYN - Sync Frame Indicator
NFI - Null Frame Indicator
PPI - Payload Preamble Indicator
RES - Reserved bit

(4) **Message Buffer Status FRMBS (word 3)**

Read access via **FRMBS**, updated by the CC at the end of the configured slot.

VFRA - Valid Frame Received on channel A
VFRB - Valid Frame Received on channel B
SEOA - Syntax Error Observed on channel A
SEOB - Syntax Error Observed on channel B
CEOA - Content Error Observed on channel A
CEOB - Content Error Observed on channel B
SVOA - Slot boundary Violation Observed on channel A
SVOB - Slot boundary Violation Observed on channel B
TCIA - Transmission Conflict Indication channel A
TCIB - Transmission Conflict Indication channel B
ESA - Empty Slot Channel A
ESB - Empty Slot Channel B
MLST - Message LoST
FTA - Frame Transmitted on Channel A
FTB - Frame Transmitted on Channel B
Cycle Count Status - Actual cycle count when status was updated
RCIS - Received on Channel Indicator Status
SFIS - Startup Frame Indicator Status
SYNS - Sync Frame Indicator Status
NFIS - Null Frame Indicator Status
PPIS - Payload Preamble Indicator Status
RESS - Reserved bit Status

### 103.3.13.2 Data Partition

The data partition of the Message RAM stores the data sections of the message buffers configured for reception / transmission as defined in the header partition. The number of data bytes for each message buffer can vary from 0 to 254. To optimize the data transfer between the shift registers of the two FlexRay Protocol Controllers and the Message RAM as well as between the Host interface and the Message RAM, the physical width of the Message RAM is set to 4 bytes.

The data partition starts after the last word of the header partition. When configuring the message buffers in the Message RAM the programmer has to assure that the data pointers point to addresses within the data partition. Table 103.16 below shows an example how the data sections of the configured message buffers can be stored in the data partition of the Message RAM.

The beginning and the end of a message buffer’s data section is determined by the data pointer and the payload length configured in the message buffer’s header section, respectively. This enables a flexible usage of the available RAM space for storage of message buffers with different data length.

If the size of the data section is an odd number of 2-byte words, the remaining 16 bits in the last 32bit word are unused (see Table 103.16 below).

**Table 103.16 Example for structure of the data partition in the Message RAM**

| Bit Word | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| … | unused | | | | | | | | unused | | | | | | | | unused | | | | | | | | unused | | | | | | | |
| … | unused | | | | | | | | unused | | | | | | | | unused | | | | | | | | unused | | | | | | | |
| … | MBn data3 | | | | | | | | MBn data2 | | | | | | | | MBn data1 | | | | | | | | MBn data0 | | | | | | | |
| … | … | | | | | | | | … | | | | | | | | … | | | | | | | | … | | | | | | | |
| … | … | | | | | | | | … | | | | | | | | … | | | | | | | | … | | | | | | | |
| … | MBn data(m) | | | | | | | | MBn data(m-1) | | | | | | | | MBn data(m-2) | | | | | | | | MBn data(m-3) | | | | | | | |
| … | … | | | | | | | | … | | | | | | | | … | | | | | | | | … | | | | | | | |
| … | … | | | | | | | | … | | | | | | | | … | | | | | | | | … | | | | | | | |
| … | … | | | | | | | | … | | | | | | | | … | | | | | | | | … | | | | | | | |
| … | MB1 data3 | | | | | | | | MB1 data2 | | | | | | | | MB1 data1 | | | | | | | | MB1 data0 | | | | | | | |
| … | … | | | | | | | | … | | | | | | | | … | | | | | | | | … | | | | | | | |
| 2046 | MB0 data3 | | | | | | | | MB0 data2 | | | | | | | | MB0 data1 | | | | | | | | MB0 data0 | | | | | | | |
| 2047 | unused | | | | | | | | unused | | | | | | | | MB0 data5 | | | | | | | | MB0 data4 | | | | | | | |

### 103.3.13.3 Message Data Integrity Check

There is a data integrity checking mechanism implemented in the FlexRay core to assure the integrity of the data stored in the related RAM. Each RAM has a checksum generator / checker attached as shown in Figure 103.17.

When data is written to a RAM, the local checksum generator generates the checksum. The checksum is stored together with the respective data word. The checksum is checked each time a data word is read from a RAM.

If a checksum error is detected, the respective access error flag is set. The access error flags **FRMHDS.AMR**, **FRMHDS.ATBF1**, **FRMHDS.ATBF2** and the faulty message buffer indicators **FRMHDS.FMBD**, **FRMHDS.MFMB**, **FRMHDS.FMB** are located in the FlexRay Message Handler Status register. These single access error flags control the error interrupt flag **FREIR.AERR**.

Figure 103.17 shows the data paths between the Input Buffer, Transient Buffer and Message RAM.

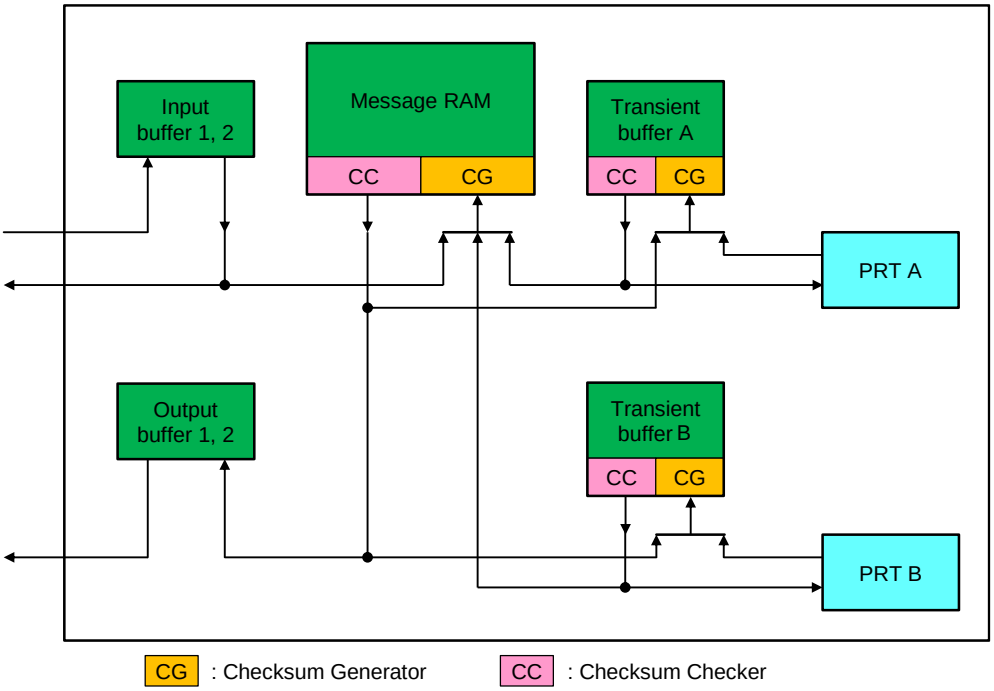



**Figure 103.17 Checksum generation and check**

When an access error has been detected the following actions will be performed:

**In all cases**

The respective access error flag in **FRMHDS** register is set

The access error flag **FREIR.AERR** is set and, if enabled, a module interrupt to the Host will be generated.

**Additionally in specific cases**

1) Access error during data transfer from Input Buffer 1,2 to Message RAM when reading header section of respective message buffer from Message RAM.

**FRMHDS.AMR** is set

**FRMHDS.FMBD** bit is set to indicate that **FRMHDS.FMB** points to a faulty message buffer

**FRMHDS.FMB** indicates the number of the faulty message buffer

The data section of the respective message buffer is not updated

Transmit buffer: Transmission request for the respective message buffer is not set

2) Access error during scan of header sections in Message RAM

**FRMHDS.AMR** is set

**FRMHDS.FMBD** bit is set to indicate that **FRMHDS.FMB** points to a faulty message buffer

**FRMHDS.FMB** indicates the number of the faulty message buffer

Ignore message buffer (message buffer is skipped)

3) Access error during data transfer from Message RAM to Transient Buffer 1, 2

**FRMHDS.AMR** is set

**FRMHDS.FMBD** bit is set to indicate that **FRMHDS.FMB** points to a faulty message buffer

**FRMHDS.FMB** indicates the number of the faulty message buffer

Frame not transmitted, frames already in transmission are invalidated by setting the frame CRC to zero

4) Access error during data transfer from Transient Buffer 1, 2 to Message RAM when reading header section of respective message buffer from Message RAM:

**FRMHDS.AMR** is set

**FRMHDS.FMBD** bit is set to indicate that **FRMHDS.FMB** points to a faulty message buffer

**FRMHDS.FMB** indicates the number of the faulty message buffer

The data section of the respective message buffer is not updated

5) Access error during data transfer from Message RAM to Output Buffer

The access error flag **FRMHDS.AMR** is set

**FRMHDS.FMBD** bit is set to indicate that **FRMHDS.FMB** points to a faulty message buffer

**FRMHDS.FMB** indicates the number of the faulty message buffer

6) Access error during a data transfer from Transient Buffer 1, 2 to Protocol Controller 1, 2

**FRMHDS.ATBF1, 2** bit is set

Frames already in transmission are invalidated by setting the frame CRC to zero

7) Access error during data transfer from Transient Buffer 1, 2 to Message RAM when reading Transient Buffer RAM 1, 2:

**FRMHDS.ATBF1, 2** bit is set

**FRMHDS.FMBD** bit is set to indicate that **FRMHDS.FMB** points to a faulty message buffer

**FRMHDS.FMB** indicates the number of the faulty message buffer

8) Access error during data read of Transient Buffer RAM 1, 2:

When an access error occurs when the Message Handler read a frame with network management information (PPI = '1') from the Transient Buffer RAM 1, 2 the corresponding network management vector register **FRNMV1/2/3** is not updated from this frame.

#### 103.3.13.4 Host Handling of Access Errors

Access error caused by transient bit flips can be fixed by:

(1) **Self-healing**

Access errors located in the Data Section of Message RAM, Transient Buffer RAM A or Transient Buffer RAM B are overwritten with the next write access to the disturbed bit(s) caused by Host access or by FlexRay communication.

(2) **CLEAR_RAMS Command**

When called in DEFAULT_CONFIG or CONFIG state POC command CLEAR_RAMS initializes the message RAM to zero.

(3) **Temporary Unlocking of Header Section**

An access error in the header section of a locked message buffer can be fixed by a transfer from the Input Buffer to the locked buffer Header Section. For this transfer, the write access to the **FRIBCR** (specifying the message buffer number) must be immediately preceded by the unlock sequence normally used to leave CONFIG state (see section 103.2.2.1).

For that single transfer the respective message buffer header is unlocked, regardless whether it belongs to the FIFO or whether its locking is controlled by **FRMRC.SEC**, and will be updated with new data.

### 103.3.14 Module Interrupt

In general, interrupts provide a close link to the protocol timing as they are triggered almost immediately when an error or status change is detected by the CC, a frame is received or transmitted, a configured timer interrupt is activated, or a stop watch event occurred. This enables the Host to react very quickly on specific error conditions, status changes, or

timer events. On the other hand too many interrupts can cause the Host to miss deadlines required for the application. Therefore the CC supports enable / disable controls for each individual interrupt source separately.

An interrupt may be triggered when

An error was detected

A status flag is set

A timer reaches a preconfigured value

A message transfer from Input Buffer to Message RAM or from Message RAM to Output Buffer has completed

A message transfer from the URAM to Message RAM or from Message RAM to URAM has completed

A stop watch event occurred

Tracking status and generating interrupts when a status change or an error occurs are two independent tasks. Regardless of whether an interrupt is enabled or not, the corresponding status is tracked and indicated by the CC. The Host has access to the actual status and error information by reading registers **FREIR**, **FRSIR**, **FROS**, **FROTS** and **FRITS**.

The general purpose interrupt lines to the Host, FlexRay Interrupt 0, FlexRay Interrupt 1, are controlled by the enabled interrupts in **FREIES** and **FRSIES**. In addition each of the two interrupt lines can be enabled / disabled separately by programming bit **FRILE.EINT0** and **FRILE.EINT1**.

The input data transfer interrupt lines to the Host, FlexRay input queue empty interrupt, FlexRay input queue full interrupt, are controlled by the enabled interrupts in **FRITS**. In addition each of the input data transfer interrupts can be enabled / disabled separately by programming the related bits in **FRITC.**

The output data transfer interrupt lines to the Host, FlexRay FIFO transfer warning interrupt, FlexRay output transfer warning interrupt, FlexRay FIFO transfer interrupt, FlexRay output transfer interrupt, are controlled by the enabled interrupts in **FROTS**. In addition each of the output data transfer interrupts can be enabled / disabled separately by programming the related bits in **FROTC.**

The three timer interrupts lines to the Host are controlled by the enabled interrupts in **FROS**. In addition each of the interrupt lines can be enabled / disabled separately by programming bit **FROC.T0IE**, **FROC.T1IE** and **FROC.T2IE**.

### 103.3.15 Assignment of FlexRay Configuration Parameters

**Table 103.17 FlexRay configuration parameters**

| Parameter | Bit(field) |
|---|---|
| pKeySlotusedForStartup | **FRSUCC1.TXST** |
| pKeySlotUsedForSync | **FRSUCC1.TXSY** |
| gColdStartAttempts | **FRSUCC1.CSA** |
| pAllowPassiveToActive | **FRSUCC1.PTA** |
| pWakeupChannel | **FRSUCC1.WUCS** |
| pSingleSlotEnabled | **FRSUCC1.TSM** |
| pAllowHaltDueToClock | **FRSUCC1.HCSE** |
| pChannels | **FRSUCC1.CCHA/B** |
| pdListenTimeOut | **FRSUCC2.LT** |
| gListenNoise | **FRSUCC2.LTN** |
| gMaxWithoutClockCorrectionPassive | **FRSUCC3.WCP** |
| gMaxWithoutClockCorrectionFatal | **FRSUCC3.WCF** |
| gNetworkManagementVectorLength | **FRNEMC.NML** |
| gdTSSTransmitter | **FRPRTC1.TSST** |
| gdCASRxLowMax | **FRPRTC1.CASM** |
| gdSampleClockPeriod | **FRPRTC1.BRP** |

| | |
|---|---|
| pSamplesPerMicrotick | **FRPRTC1.BRP** |
| gdWakeupSymbolRxWindow | **FRPRTC1.RXW** |
| pWakeupPattern | **FRPRTC1.RWP** |
| gdWakeupSymbolRxIdle | **FRPRTC2.RXI** |
| gdWakeupSymbolRxLow | **FRPRTC2.RXL** |
| gdWakeupSymbolTxIdle | **FRPRTC2.TXI** |
| gdWakeupSymbolTxLow | **FRPRTC2.TXL** |
| gPayloadLengthStatic | **FRMHDC.SFDL** |
| pLatestTx | **FRMHDC.SLT** |
| pMicroPerCycle | **FRGTUC1.UT** |
| gMacroPerCycle | **FRGTUC2.MPC** |
| gSyncNodeMax | **FRGTUC2.SNM** |
| pMicroInitialOffset[A] | **FRGTUC3.UIOA** |
| pMicroInitialOffset[B] | **FRGTUC3.UIOB** |
| pMacroInitialOffset[A] | **FRGTUC3.MIOA** |
| pMacroInitialOffset[B] | **FRGTUC3.MIOB** |
| gdNIT | **FRGTUC4.NIT** |
| gOffsetCorrectionStart | **FRGTUC4.OCS** |
| pDelayCompensation[A] | **FRGTUC5.DCA** |
| pDelayCompensation[B] | **FRGTUC5.DCB** |
| pClusterDriftDamping | **FRGTUC5.CDD** |
| pDecodingCorrection | **FRGTUC5.DEC** |
| pdAcceptedStartupRange | **FRGTUC6.ASR** |
| pdMaxDrift | **FRGTUC6.MOD** |
| gdStaticSlot | **FRGTUC7.SSL** |
| gNumberOfStaticSlots | **FRGTUC7.NSS** |
| gdMinislot | **FRGTUC8.MSL** |
| gNumberOfMinislots | **FRGTUC8.NMS** |
| gdActionPointOffset | **FRGTUC9.APO** |
| gdMinislotActionPointOffset | **FRGTUC9.MAPO** |
| gdDynamicSlotIdlePhase | **FRGTUC9.DSI** |
| pOffsetCorrectionOut | **FRGTUC10.MOC** |
| pRateCorrectionOut | **FRGTUC10.MRC** |
| pExternOffsetCorrection | **FRGTUC11.EOC** |
| pExternRateCorrection | **FRGTUC11.ERC** |

### 103.3.16 Usage of data transfer

A mechanism is implemented to allow storage of FlexRay messages directly into the URAM (user RAM) and have transfers between the FlexRay internal message RAM and the URAM and vice versa with minimum CPU support. The data in the URAM should be indexed by data structure pointers located in data pointer tables stored in the URAM.
Data transfer from the URAM to the FlexRay internal message RAM (input transfer) needs to be initiated by the application. These transfers can be used to configure message buffers or to update transmit data.
A data transfer from the FlexRay internal message RAM to the URAM (output transfer) is initiated automatically by a reception into a receive message buffer or FlexRay internal FIFO or by a change in the slot status. It can be initiated also by a specific user transfer request.
The input and output data transfer can be activated independently. When the input data transfer is activated the application should not directly access message buffers using the FlexRay input buffer. When the output data transfer is activated the application should not directly access message buffers using the FlexRay output buffer.

### 103.3.16.1 Input data transfer

When the automatic input data transfer function is enabled, committed input data structures are transferred from the URAM to the FlexRay internal message RAM with minimum CPU support.

(1) **Activation and deactivation**

The input data transfer function should be activated before usage. The activation of the input transfer handler initializes the input queue put index (**FRITS.IPIDX**) and get index (**FRITS.IGIDX**) to zero. Also the interrupt status flags in the **FRITS** register (**IQEIS** and **IQFIS**) are set to '0'.

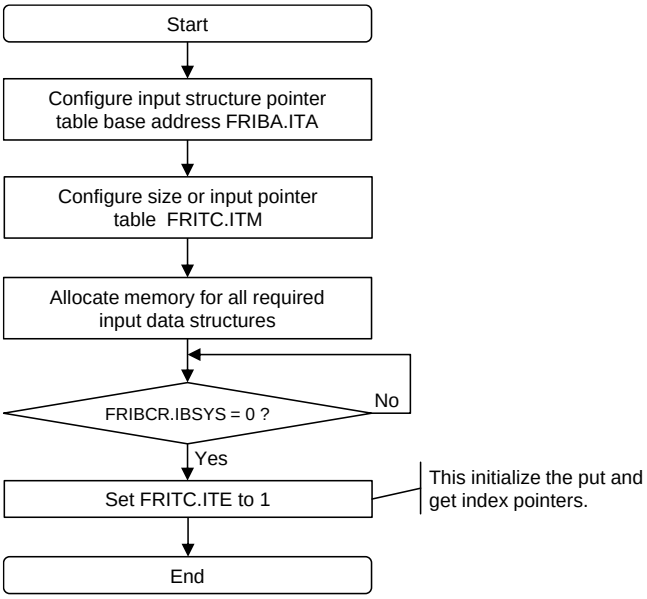



**Figure 103.18 Input transfer enable flow**

A deactivation request of the input transfer function can be made at any time. The input queue put index and the input queue status are maintained independently from the input transfer function state.

Before the transfer function gets disabled (status indicated by **FRITS.ITS**), user requested input transfers and all committed input transfers will be completed.

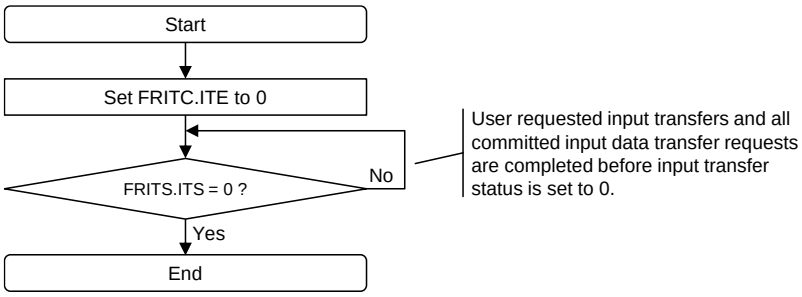



**Figure 103.19 Input transfer disable flow**

(2) **Input data structure**

The application has to reserve a location in the URAM to provide the content for message buffer configuration (input data structure).
The location of this input data structure needs to be defined by an input data structure pointer also located in the URAM.

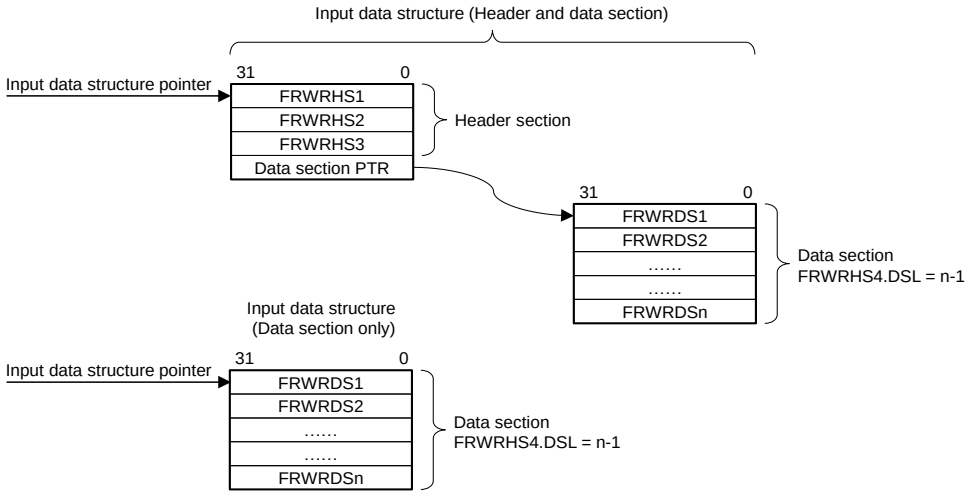



**Figure 103.20 Input data structure**

In general the input data structure consists of two sections, the header and the data section.
The header section consists of **FRWRHS1**, **FRWRHS2**, **FRWRHS3** and the data section pointer.

For information about the bit alignment and bit function within the header section refer to section 103.3.13.1.

Depending on the settings in the control field (**FRWRHS4**) located in the input pointer table, the data structure pointer is a reference to the address of **FRWRHS1** or **FRWRDS1**. The data structure pointer has to be aligned to a 32 bit address.

If the bit **LHS** in the address related to **FRWRHS4** is set to '1' it is required to provide a valid header section. In this case **FRWRHS1** is the first element of the data structure.

If the bit **LHS** in the address related to **FRWRHS4** is set to '0' a header section is not required. In this case **FRWRDS1** is the first element of the data structure.

If the bit **LDS** in the address related to **FRWRHS4** is set to '1' it is required to provide a valid data section. The pointer to the data section is a reference to the address of the first payload long word (**FRWRDS1**) and has to be aligned to a 32 bit address.

If the bit **LDS** in the address related to **FRWRHS4** is set to '0' a data section is not required. The data section pointer is not evaluated by the input handler.

The byte order for the FlexRay payload data in the input data structure is determined by **FROC.BEC**. For information about the payload data alignment within the data section refer to section 103.3.13.2 and section 103.3.17.

The length of the data section and the size to be allocated in the URAM depends on the configuration of the bits **DSL** in the address related to **FRWRHS4**.

For the transfer into the FlexRay core internal message RAM the number of 16 bit words configured by **FRWRHS2.PLC** is used. The application has to ensure, that a proper number of data words is provided in the URAM. In case the buffer is configured by **FRWRHS2.PLC** to hold an odd payload length, the application has to write zero to the last 16 bit of the payload section to ensure that the padding data is all zero.

(3) **Input pointer table**

To transfer data from the input data structures located in the URAM to the FlexRay internal message RAM the related input data structure pointer and control field needs to be added to the input pointer table which is located in the URAM. The location of the first element of this table is identified by the input pointer table base address (**FRIBA.ITA**). This base address has to be aligned to a 32 bit address.

The maximum number of input requests that can be queued is defined by the Input queue Table Max register (**FRITC.ITM**).

Each Input pointer table entry requires two long words. The required address range of the input pointer table for the queued transfer requests can be calculated by

$$input_pointer_table_size[bytes] = (((\mathrm{FRITC.ITM} + 1) * 2) * 4)$$

**Equation 4-1**

The input pointer entry for the user requested input transfer should be added after the end of the input pointer table. The pointer table index related to this entry and hence the number to be written to **FRUIR.UIDX**, is **FRITC.ITM**+1. The address in the input pointer table related to the user requested input transfer (user input address) can be calculated by

$$user_input_address = \mathrm{FRIBA.ITA} + input_pointer_table_size$$

**Equation 4-2**

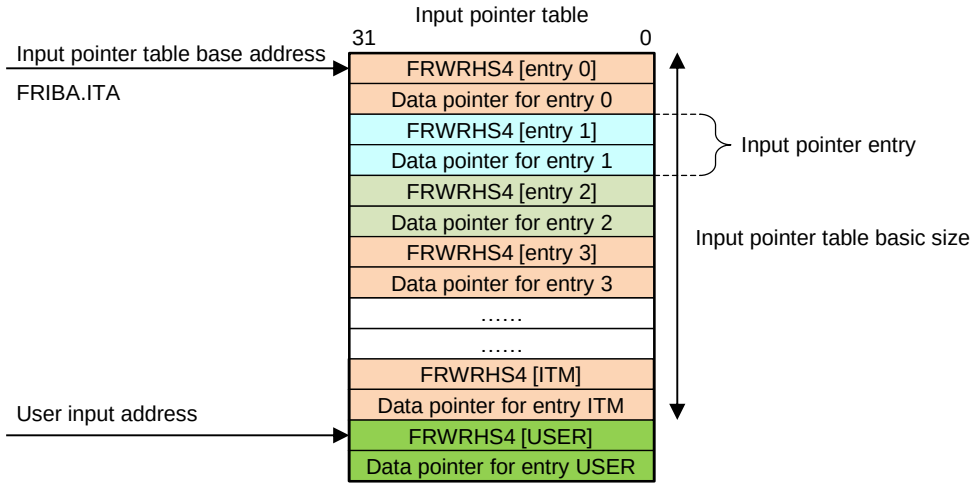



**Figure 103.21 Input pointer table**

The input pointer table holds the control field **FRWRHS4** and the pointers to the URAM location where the message buffer content (header section and/or data section) is stored.
The application has to write **FRWRHS4** and the input data structure pointer at the addresses in the input pointer table related to the put index position before a transfer request is initiated.

**FRWRHS4:**

| b31 | b30 | b29 | b28 | b27 | b26 | b25 | b24 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| b23 | b22 | b21 | b20 | b19 | b18 | b17 | b16 |
| — | — | DSL[5:0] | | | | | |
| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 |
| — | — | — | — | INV | STR | LDS | LHS |
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| — | IMBNR[6:0] | | | | | | |

**Bit Functions**

| Bit | Symbol | Bit Name | Function |
|---|---|---|---|
| b[31:22] | - | Reserved | The read value is undefined. The write value should be always '0'. |
| b[21:16] | DSL | Data Section Length | Specifies the length of the data section in terms of 32 bit values. |
| b[15:12] | - | Reserved | The read value is undefined. The write value should be always '0'. |
| b11 | INV | Invalidate entry | 0: The data structure is valid and will be transferred to the FlexRay internal message RAM.<br>1: The data structure is invalid. FlexRay internal message RAM is not updated using this input pointer entry. |
| b10 | STR | Set transmission request | 0: The bit **FRTXRQn.TXR** for the message buffer selected by the bits **IMBNR** is set to '0'. No data from this message buffer is transmitted. |

| Bit | Symbol | Bit Name | Function |
|---|---|---|---|
| | | | 1: The bit **FRTXRQn.TXR** for the message buffer selected by the bits **IMBNR** is set to '1' to release the message buffer for transmission The application should not set the bit **STR** to '1' for receive buffers. |
| b9 | LDS | Load data section | 0: No update of data section.<br>1: Data section for the message buffer selected by the bits **IMBNR** is updated. |
| b8 | LHS | Load header section | 0: no update of header section.<br>1: Header section for the message buffer selected by the bits **IMBNR** is updated. |
| b7 | - | Reserved | The read value is undefined. The write value should be always '0'. |
| b[6:0] | IMBNR | Message buffer number to be updated | Selects the target message buffer number in the FlexRay internal message RAM for transfer |

Note that the **LHS** bit should not be set for protected message buffers.

The bit **LDS** defines if the data section of the message buffer selected by the bits **IMBNR** should be updated.
If **LDS** is set to '1' (**DSL** + 1) 32 bit words of payload data are transferred from the URAM to the message buffer selected by the bits **IMBNR**.
If **LDS** is set to '0' no payload data is transferred from the URAM.
Note that the payload transferred is independent from the configured payload length (bits **PLC** in the address related to **FRWRHS2**).

The bit **INV** can be used to invalidate a committed data structure. This bit should be only used to cancel the transfer of committed data structures when the input queue is halted (see section 103.3.16.1.(5)).

When this bit is set to '1' the message buffer number **IMBNR** is not updated. When the bit is set to '0' the message buffer number **IMBNR** is updated.

(4) **Transfer function of input data structure**

To use the input data structure transfer function the input transfer has to be activated (see section 103.3.16.1.(1)). The activation process requires the setup of the input pointer table (see section 103.3.16.1.(3)) in order to specify the source location (input data structures) for the data structures to be transferred. When the input transfer gets enabled the get index pointer is initialized to zero.

All FlexRay internal message buffers can be updated using the input transfer queue which is built in the input pointer table. The application has to write the pointer and control field (table entry) to the data structure to be transferred into the input pointer table. For that purpose the application has to maintain a put index for the input pointer table that indicates where the pointer has to be written to.
To commit this table entry to the input handler, the application has to write the target message buffer number to the input queue control register (**FRIQC.IMBNR**). Afterwards the application has to increment the application internal put index. By writing to the input queue control register the data available flag (**FRDAi.DA[IMBNR]**) is automatically set to '1'. The input transfer handler also maintains the put index pointer in the status register (**FRITS.IPIDX**).

In case the input queue gets full (number of queued input transfer requests is equal to the input queue table size) **FRITS.IQFP** and **FRITS.IQFIS** are set to '1'. The input queue full condition pending flag (**FRITS.IQFP**) changes from '1' to '0' when there are entries in the input queue available, whereby the input queue full interrupt status flag (**FRITS.IQFIS**) needs to be cleared by the application.
The application should not make any further write access to the bit **IMBNR** in the **FRIQC** register as long as the bit **IQFP** in the **FRITS** register is '1'.

In case the input queue gets empty (number of queued input transfer requests changes to zero) **FRITS.IQEIS** is set to '1'. The input queue empty interrupt status flag (**FRITS.IQEIS**) needs to be cleared by the application.

The transfer of the input data structures to the FlexRay message RAM is controlled by a get index pointer which is handled inside the FlexRay module and flagged in **FRITS.IGIDX**. Note that the index is referring to the input entry and not the address offset in the input pointer table.
If the input queue is not empty, the transfer handler reads out the input pointer table entry of the transfer queue and starts the transfer of the input data structure from the address the input pointer is referring to. When all required data words are transferred to the FlexRay module, the data available flag for the transferred message buffer number is set to '0' and the get index in the transfer handler is incremented by one.

In case of an invalidated data structure (see section 103.3.16.1.(5)) no FlexRay internal message buffer is updated and the related data available flag is automatically set to '0'. The change of the data available flag can be used to confirm the cancellation a transmit request

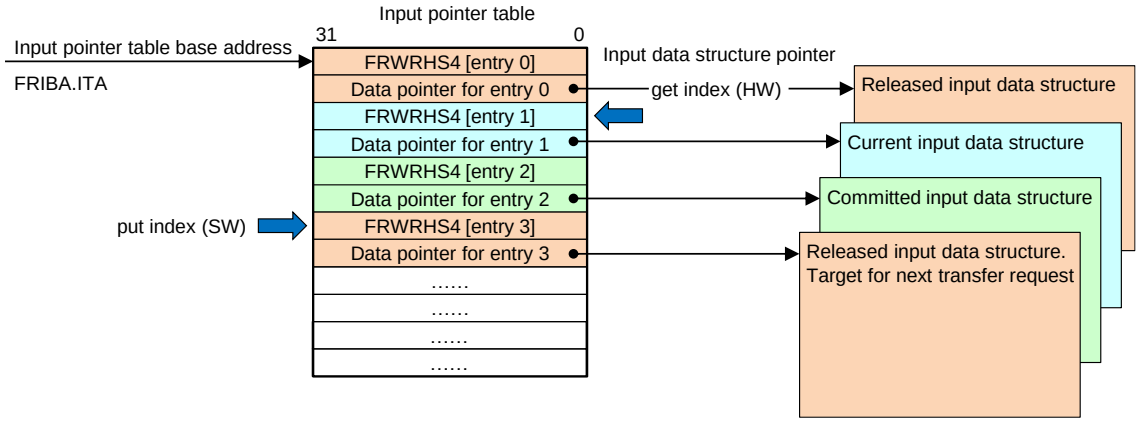



**Figure 103.22 Input pointer table**

Receive message buffers can be also configured using the input data transfer by setting up the required header sections and mark only the header section (**FRWRHS4.LDS** = '0', **FRWRHS4.LHS** = '1') to be updated in the FlexRay module.

(5) **Halting the input queue**

Committed data structures cannot be removed, but can be invalidated or updated when the input queue is halted.
To cancel data structures already committed to the input queue, the queue can be halted by writing a '1' to **FRITC.IQHR**.

After the ongoing input transfer has been completed the queue is halted and **FRITS.IQH** changes from '0' to '1'.

To invalidate an entry of the input queue **FRWRHS4.INV** has to be set to '1'. All other bits in **FWRHS4** should be unchanged.

Following flow shall be used to analyse whether a committed message has been already transferred to the FlexRay internal message RAM or not.

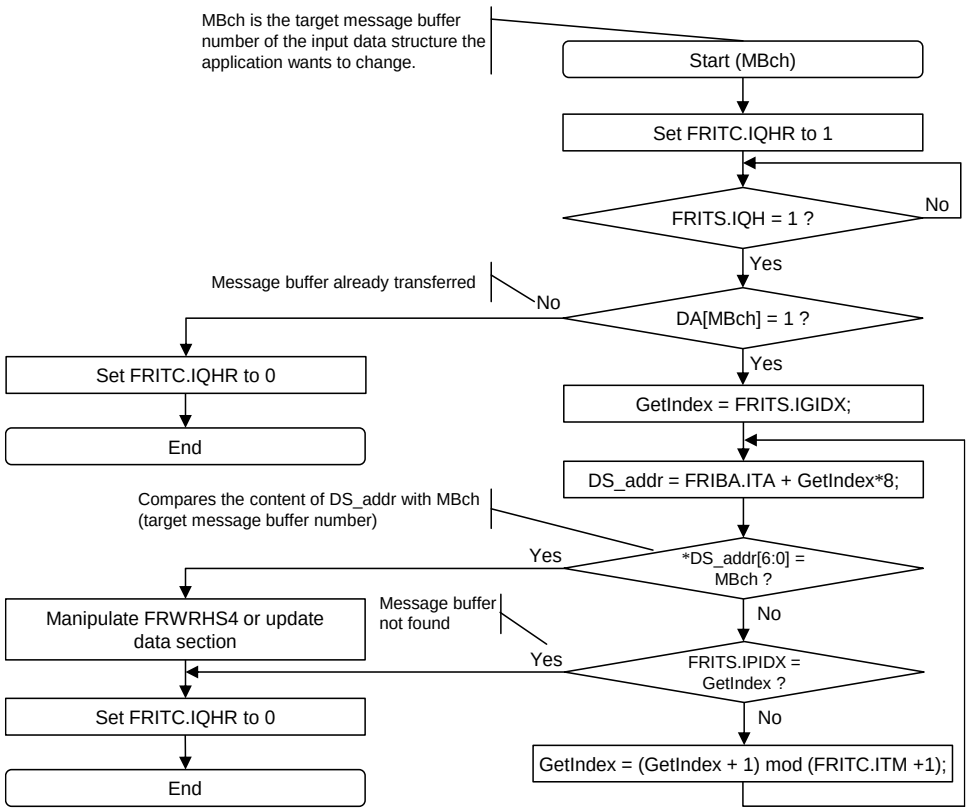



**Figure 103.23** Input table analysis

In case the message buffer was already transferred to the FlexRay internal message RAM, the user input transfer request can be used to bypass the actual queue and update the required message buffer (see section 103.3.16.2.(6)).

(6) **Transfer function of user requested input transfers**

To use this function the input transfer has to be activated (see section 103.3.16.1.(1)).

The application is capable, by using **FRUIR.UIDX,** to request a transfer of an input data structure. The user input transfer request is serviced first.

The application has to write the pointer and control field (table entry) to the data structure to be transferred into the input pointer table. The table entry for the user input transfer request should be added after the end of the input pointer table (see section 103.3.16.1.(3)).
To commit this table entry to the input handler, the application has to write the index (**FRITC.ITM**+1) to the user input transfer request register (**FRUIR.UIDX**).

By writing to the user input transfer request register, the user input transfer request pending flag (**FRITS.UIRP**) is automatically set to '1'.
As long this flag is '1' the application should not make any further user input transfer requests.

The user input transfer request pending flag (**FRITS.UIRP**) changes from '1' to '0' when the requested input transfer is completed. As next the pending transfers are processed.

### 103.3.16.2 Output data transfer

When the output data transfer function is enabled, received messages (either in dedicated message buffers or in the FlexRay receive FIFO) are transferred to the URAM by the output data handler. The output data handler can also transfer the message buffer content to the URAM on application request. When enabled the output handler is also capable to initiate a transfer when the message buffer status has changed.

(1) **Activation and deactivation**

The output data transfer function should be activated before usage. The activation of the output transfer hander will initialize the FIFO put and get index pointer and FIFO fill level (**FROTS.FGIDX** and **FROTS.FFL**) to zero, set the bits **FROTS.FDA**, **FROTS.OWP**, **FROTS.FWP** and **FROTS.UORP** to '0'. Also the interrupt status flags **FROTS.OTIS**, **FROTS.FIS**, **FROTS.OWIS** and **FROTS.FWIS**) are set to '0'.
The activation has no influence to the data available flags (**FRDAi.DA**) which are related to the dedicated buffers; these flags have to be cleared by the application.

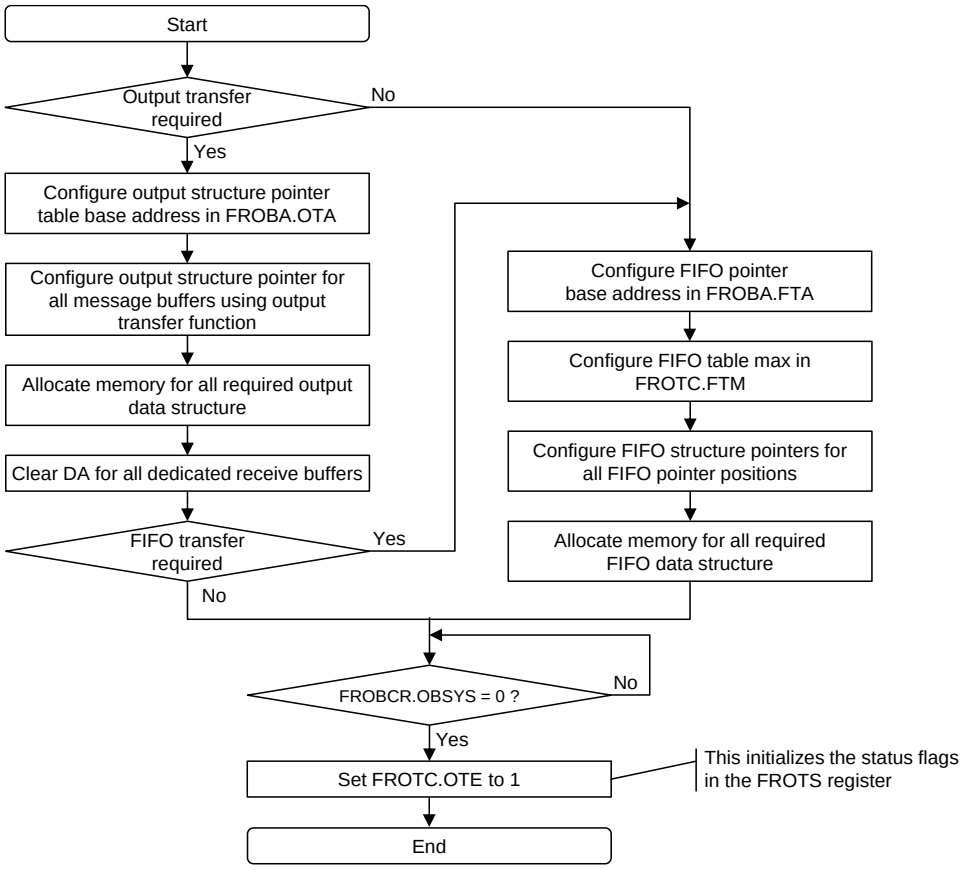



**Figure 103.24 Output transfer enable flow**

A deactivation request of the output data transfer function can be made at any time. An ongoing transfer will be completed and the completion of this transfer will be flagged. During this time **FROTS.OTS** remains '1'.
When **FROTS.OTS** changes from '1' to '0', the output transfer function is deactivated. The data available status flags and the FIFO get index are still maintained when the output transfer function is disabled.

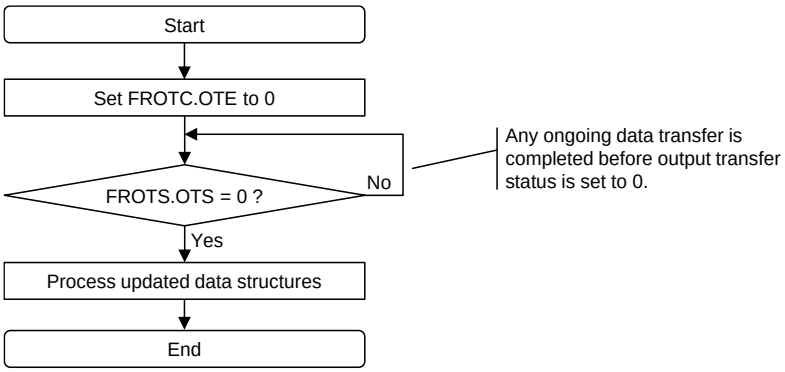



**Figure 103.25 Output transfer disable flow**

(2) **Output transfer data structure**

The data in the URAM is stored in an output data structure. The location of the output data structures are determined by output data structure pointers also located in the URAM. The output data structure and indexing is visualized in Figure 103.26.

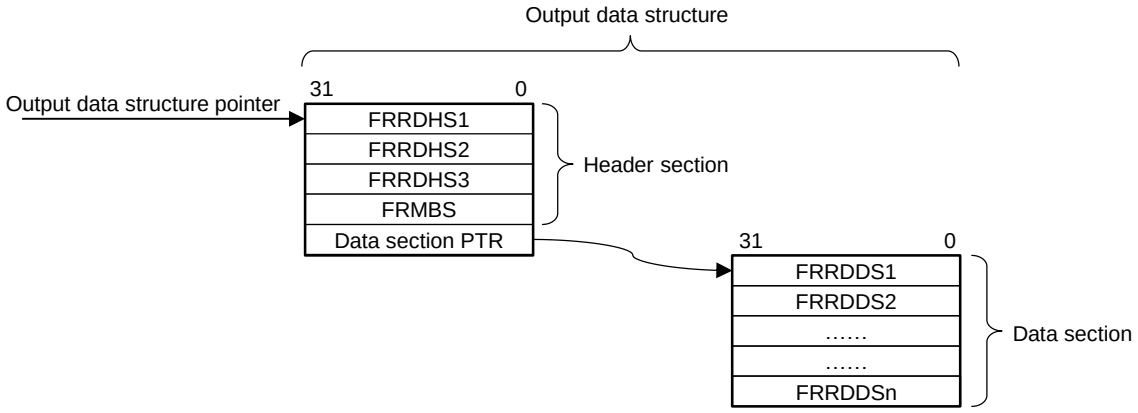



**Figure 103.26 Output data structure**

The output data structure consists of two sections, the header and data section. The header section consists of **FRRDHS1**, **FRRDHS2**, **FRRDHS3**, **FRMBS** and the data section pointer. **FRRDHS1** is the first element of the structure and has to be aligned to a 32 bit address. The data structure pointer is a reference to the address of **FRRDHS1**. For information about the bit alignment and bit function within the header section refer to section 103.3.13.1.

**FRRDDS1** is the first element of the data section. The data section pointer is a reference to the address of **FRRDDS1** and has to be aligned to a 32 bit address. The byte order for the FlexRay payload data in the output data structure is determined by the bit **BEC** in the **FROC** register. For information about the payload data alignment within the data section refer to section 103.3.13.2.

The length of the data section as well as the total structure size to be allocated in the URAM depends on the configured payload length (**FRRDHS2.PLC**) of the related message buffer. In case the configured payload length is an odd number

of words or the received payload length (**FRRDHS2.PLR**) is smaller than the configured payload length, the remaining data words in the URAM are unused and should not be used by the application.

The output data structure is identical for all three kinds of output transfers. In case only the header section is transferred the data section pointer is not evaluated by the output handler and the data section remains unchanged.

(3) **Output pointer table**

For the output data transfer function the application needs to setup an output pointer table in the URAM.
The location of the first element of this table should be programmed into the output pointer table base address (**FROBA.OTA**). This base address has to be aligned to a 32 bit address.
The size of the output pointer table is defined by the maximum of: the last configured dedicated message buffer and the highest message buffer number which will be used for the user output transfer request.

The output pointer table holds pointers (output data structure pointers) to the URAM location where a memory space is reserved for the target message buffer content (header section and data section).
There is a fixed linear relationship between the address of the entries in the output pointer table and the number of the related message buffers (see Figure 103.27): the output pointer table starts with the entry for message buffer number 0 at the address configured in **FROBA.OTA** and continues in ascending order for each following message buffer number, by 32 bit aligned address (e.g. message buffer 1 at address **OTA** + 4, message buffer 2 at address **OTA** + 8, etc.) for all possible message buffers.

When a set **ND** bit is the only transfer condition (**FROTC.OTCS** is set to 0) only message buffers configured as a dedicated receive buffer or that will be used for user output transfer requests need have valid pointer entries.

When a set **ND** bit or a set **MBC** bit is the transfer condition (**FROTC.OTCS** is set to 1) all dedicated receive buffer and dedicated transmit buffers need to have valid pointer entries.

(4) **FIFO output pointer table**

The FlexRay module internal FIFO can be extended by a queued buffer structure in the URAM.
If the FlexRay module internal FIFO is used the application needs to setup the FIFO output pointer table. The location of the first element of this table is identified by the FIFO pointer table base address (**FRFBA.FTA**). This base address has to be aligned to a 32 bit address.
The size of the FIFO pointer table and hence the maximum number of messages that can be added to the queue, is defined by FIFO Table Max (**FROTC.FTM**).

The FIFO pointer table holds pointers (output data structure pointers) to the URAM location where a memory space is reserved the target message buffer content (header section and data section). For each table entry a data pointer shall be configured in this table.

(5) **Transfer function of dedicated message buffers**

To use this transfer function the output transfer has to be activated (see section 103.3.16.2.(1)). The activation process requires to setup the output pointer table (see section 103.3.16.2.(3)) in order to specify the destination location (output data structures) for the data to transfer. Figure 103.27 shows how the output pointer table references the output data structures.

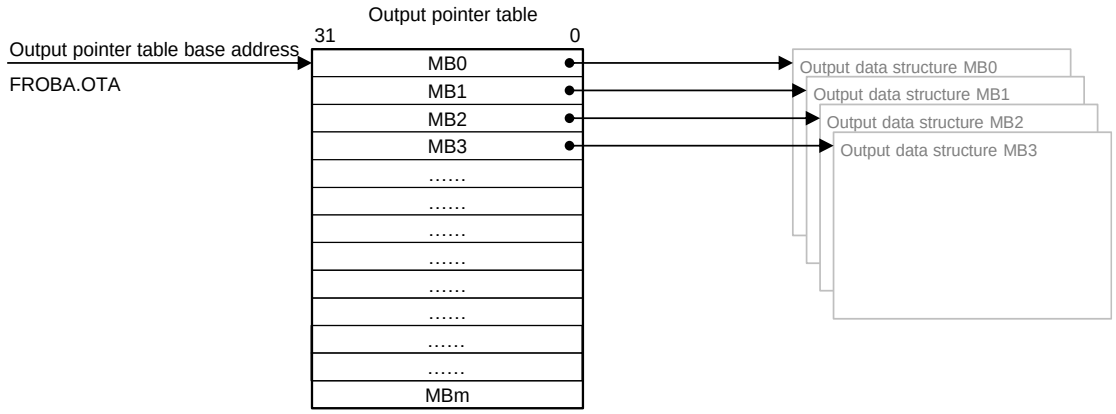



**Figure 103.27 Output data structure and indexing**

With **FROTC.OTCS** the output transfer condition can be selected between the 'New data only mode' and the 'New data and status changed mode'.

In the 'New data only mode' an output data transfer is initiated when a valid FlexRay data frame has been stored into a dedicated receive buffer which causes the related **ND** flag in the **FRNDATn** register to set. The **ND** flag in the **FRNDATn** register is automatically set to '0' during the transfer procedure. The header section is also transferred and hence the **MBC** flag in the **FRMBSCn** register is set to '0'.

In the 'New data and status changed mode' an output data transfer is initiated as described in the 'New data only mode'. In addition an output data transfer is initiated when only the message buffer status has been changed which causes the related **MBC** flag in the **FRMBSCn** register to be set. In this case only the header section is transferred. The **MBC** flag in the **FRMBSCn** register is automatically set to '0' during the transfer procedure.

After transferring the message buffer data from the FlexRay internal message RAM to the output data structure the corresponding data available flag in the **FRDAi** (i from 0 to 3) registers is set to '1'. The update of the output data structure is also flagged by the setting of the output transfer interrupt status flag (**FROTS.OTIS**).

As long as the data available flag remains '1' the corresponding output data structure will not be updated.

In the case
the data available flag is '1' and a valid received message was stored or
when **FROTC.OTCS** is '1' and the message buffer status was updated,
the output transfer warning interrupt flag (**FROTS.OWIS**) is set to '1' notifying the application that new data is available but the output data structure transfer cannot be processed. In addition **FROTS.OWP** is set to '1' that continuously flags that status of the output transfer warning condition.

If a valid receive message in the FlexRay internal message RAM is overwritten by an additional receive message, the message lost flag (**FRMBS.MLST**) is set to '1'. This flag can be evaluated after the message buffer has been transferred into an output data structure.

Following sections are giving a guidance how output data structures can be handled.

**(a) Data section copy method**

One option is to copy the information from the output data structure to a different location of the URAM and then release the output data structure by clearing the related data available flag. The application should use the copied information for further processing.

**(b) Data structure pointer method**

A different option is to modify the output data structure pointer in the output pointer table and to release the output data structure by clearing the related data available flag. The changed output data pointer should refer to a free data structure. The application should use the old data structure for further processing.

**(c) Data section pointer method**

A third option is to modify the data section pointer in the output data structure and to release the output data structure by clearing the related data available flag. The changed data section pointer should refer to a free memory area. The application should use the old data section for further processing by forwarding the data section pointer.

(6) **Transfer function of FIFO message buffers**

To use this buffer transfer function the output transfer has to be activated (see section 103.3.16.2.(1)). The activation process requires the setup of the FIFO pointer table (see section 103.3.16.2.(4)) in order to specify a location in the URAM reserved for the storage of the required output data structures.

A FIFO data transfer is initiated when a valid FlexRay data frame has been stored in the FlexRay internal FIFO.
After transfers from the internal FIFO to the output data structure, the FIFO interrupt status flag (**FROTS.FIS**) and **FROTS.FDA** are set to '1'. The bit **FROTS.FIS** can be used as an interrupt source. The bit **FROTS.FDA** indicates that the FIFO is not empty.
Up to **FROTC.FTM** output structures configured in the **FROTC** register can be queued.

The transfer to the extended FIFO buffer structure is controlled by index pointers. This put index is controlled by the FIFO transfer handler and is incremented after transferring a message to the output data structure.
The FIFO reception handler also maintains a get index which is flagged in **FROTS.FGIDX**. The value of this get index is known by the application by either reading the status or maintaining a software variable. The get index (initially set to the zero) is incremented by one when the application releases the oldest entry of the FIFO queue by writing '1' to **FROTS.FDA**. By comparing the put index and the get index the FIFO handler knows about the current fill level of the queued buffer structure.
The current FIFO fill level is flagged in **FROTS.FFL**. When **FROTS.FDA** is '1', there is at least one entry in the FIFO queue.

In case the queued buffer structure in the URAM is full (**FROTS.FFL** = **FROTC.FTM**+1), no further transfers are initialized, new messages remain in the FlexRay internal FIFO and the FIFO transfer warning interrupt status flag (**FROTS.FWIS**) is set to '1'.
In case the FlexRay internal FIFO structure becomes full, messages in the FlexRay internal FIFO structure may get overwritten. The related status flags and configuration registers of the FlexRay core module can be used to generate desired warning notifications.

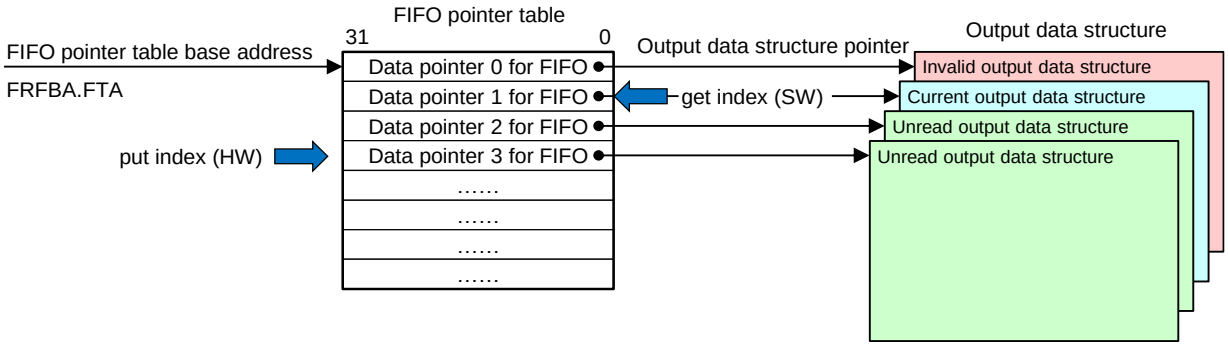



**Figure 103.28 FIFO pointer table**

(7) **Transfer function of user output transfer requests**

To use this transfer function the output transfer has to be activated (see section 103.3.16.2.(1)). The activation process requires to setup the output pointer table (see section 103.3.16.2.(3)) in order to specify the location in the URAM reserved for the transfer of the data (output data structures).

The application is capable, by using **FRUOR.UMBNR**, to request a transfer of dedicated message buffer to an output data structure. Except in CONFIG state, message buffers which are part of the FlexRay internal FIFO should not be requested.
The header section is always transferred to the output data structure. The transfer of the data section can be enabled by setting **FRUOR.URDS** to '1'. The selected message buffer content is stored in the output data structure location determined by the pointers in the output pointer table.
The data available status and transfer blocking by the **DA** bits in the **FRDAi** register is also used for the user requested transfers. Therefore the **DA** bit in the **FRDAi** register related to the requested buffer number (**FRUOR.UMBNR**) should be released before making the transfer request.
After writing to **FRUOR.UMBNR**, the bit **FROTS.UORP** is set to '1' to indicate that there is a pending user transfer request. When the transfer has been processed the bit **FROTS.UORP** is set to '0', the bit **FROTS.OTIS** is set to '1' and the **DA** bit in the **FRDAi** register related to the requested buffer number (**FRUOR.UMBNR**) is set to '1'.
User output transfer requests cannot be queued. The application should check the bit **FROTS.UORP** before writing to **FRUOR.UMBNR**.

User output transfer requests should not be made for message buffers which are pending in the input transfer queue.

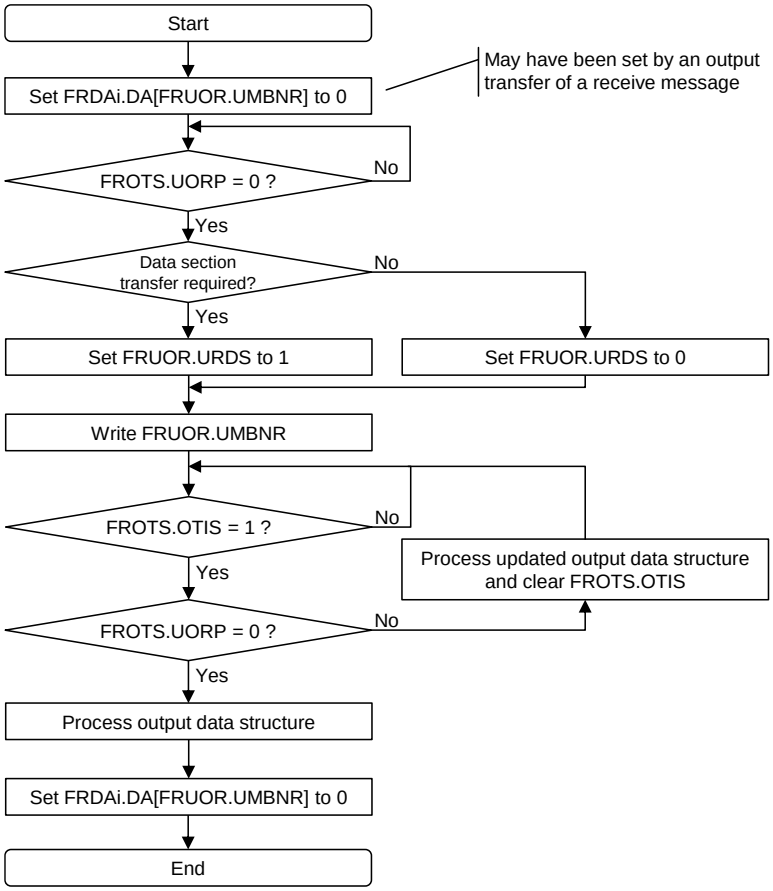



**Figure 103.29 User output transfer request flow**

Note that it may be possible that the data structure addressed by a user request is being updated due to a receive message buffer update (which causes the bit **DA** in the **FRDAi** register being set). This set **DA** flags inhibits the user output transfer request. Therefore polling **FROTS.UORP** is not a secure method to identify when the transfer of a requested message buffer has been completed. The bits **FROTS.OTIS** or the **DA** bit in the **FRDAi** register can be used instead. The exact flow depends on the software architecture.

#### 103.3.16.3 Data structure transfer scheduling

Cyclically the different types of transfer requests are checked. In order to guarantee a certain transfer time the different types of transfers have different priorities.

Use requested input transfers have highest priority followed by the transfer of data structures committed into the active input transfer queue. No new output transfer will be started as long as there is a pending input transfer request.

The three output transfer request types are checked in a specific order:

1) All dedicated message buffers in ascending order
   When **FROTC.OTCS** is set to '0', set flags in the **FRNDATn** register are causing a transfer of the message buffer to the output data structure if the destination area is free (**DA** bit in the **FRDAi** register is ‘0’).
   When **FROTC.OTCS** is set to '1', set flags in the **FRNDATn** register or set flags in the **FRMBSC1/2/3/4** register are causing a transfer of the message buffer to the output data structure if the destination area is free (**DA** bit in the **FRDAi** register is ‘0’).

2) FlexRay internal FIFO
   When the FlexRay internal FIFO is not empty and there is a free destination area, one FIFO message is transferred into the output data structure addressed by the FIFO pointer table.
3) User output request
   If there is a pending user output transfer request, one message buffer is transferred into the corresponding output data structure.

The check sequence is suspended when an input transfer occurs.

#### 103.3.16.4 Behaviour in case of data transfer access error

The memory areas accessed by the data transfer function may be protected by a memory protection unit (MPU). When the MPU flags an access to a protected address caused by an input or output transfer, an access error event is generated and the related bit in the **FRAES** register is set.

The ongoing transfer is immediately terminated but succeeding transfers are processed and may generate further access errors. Any following access errors are only flagged in **FRAES.MAE**. The other status flags are not updated.

(1) **Access error during input transfer**

When an access error occurs during an input transfer:

The ongoing transfer is immediately terminated. The FlexRay internal message RAM will not be updated

The address, the FlexRay module wanted to access to, is captured in the **FRAEA** register

**FRAES.IAE** is set to '1'

The input pointer table index is flagged in **FRAES.EIDX**

In case of an normal input transfer the to the transfer related **DA** flag in the **FRDAi** register is set to '0'

In case of an user input transfer request **FRITS.UIRP** is set to '0'

With the given status information the application is able to identify and correct the faulty data structure.
In addition the application needs to clear the input access error flag (**FRAES.IAE**).

(2) **Access error during output transfer**

When an access error occurs during an output transfer:

The ongoing transfer is immediately terminated but the update of the data structure may have started.
The address, the FlexRay module wanted to access to, is captured in the **FRAEA** register
**FRAES.OAE** is set to '1'
The output pointer table index is flagged in **FRAES.EIDX**
In case of an normal output transfer the to the transfer related **DA** flag in the **FRDAi** register remains '0' and no outputt transfer interrupt is generated
In case of an user output transfer request **FROTS.UORP** is set to '0'
With the given status information the application is able to identify and correct the faulty data structure. The data structure in the URAM cannot be treated as valid.
In addition the application needs to clear the output access error flag (**FRAES.OAE**).

The FlexRay module internal transfer of the message buffer is completed before the URAM access error is detected. The output transfer will not be re-initiated. To avoid loss of data, the application can perform a user output transfer request of this message buffer to a correct URAM location.

(3) **Access error during FIFO transfer**

When an access error occurs during an FIFO transfer:

The ongoing transfer is immediately terminated
The address, the FlexRay module wanted to access to, is captured in the **FRAEA** register
**FRAES.FAE** is set to '1'
The FIFO pointer table index is flagged in **FRAES.EIDX**
The FIFO index pointer are not changed and hence the FIFO status flags are unchanged
With the given status information the application is able to identify and correct the faulty data structure.
In addition the application needs to clear the FIFO access error flag (**FRAES.FAE**).

The data in the URAM cannot be treated as valid and is not released to the application. The message cannot be recovered.

### 103.3.16.5 Behaviour in case of RAM read errors

The FlexRay internal message RAM has an ECC checking mechanism. In case a not correctable RAM read error occurs the application has to analyse the status in the **FRMHDS** register and react as described in section 103.3.16.3. The input and output transfer handler reacts also on these errors detected in the message RAM when the error is related to an active transfer.

In addition the Transient Buffer RAM A and Transient Buffer RAM B have an ECC checking mechanism as well. Not correctable RAM read errors does not impact the data transfer functionality but have to be handled as described in section 103.3.13.1.(4).

In all cases data causing a read error is never transferred to the URAM. If there is no recovery available in the application, the message is lost.

(1) **Read error during transfer from TBF to MBF**

This internal transfer is done for each valid received FlexRay message.

A read error can only occur when reading the header section in the FlexRay Message RAM (see read error flags in **FRMHDS**). In this case the message buffer needs to be re-configured.

For dedicated receive message buffers the related **ND** flag in the **FRNDATn** register will not get set. Consequently the affected message buffer will not be transferred to the output data structure.

For the FlexRay internal FIFO buffers the **ND** flag in the **FRNDATn** register is not set but the FlexRay internal FIFO put index is incremented. Due to this, a transfer procedure from the FlexRay internal FIFO buffer to the output buffer is started. However if the read error is still present in the header section the update of the output data structure will not start (see 6.16.5.2); thus the data in the URAM remains correct.

Note that the correction or any other reconfiguration of FIFO related message buffers while there are pending FIFO transfers may result in incorrect data in the URAM. It is strongly recommended to deactivate the output data transfer before starting the reconfiguration and flush the FlexRay internal FIFO before to reactivate the output data transfer.

(2) **Read error during transfer from MBF to OBF**

This internal transfer is done for every output data transfer (dedicated reception, FIFO, user requested).

A read error can occur in the header and data section (see read error flags in **FRMHDS**). In both cases the message gets lost. If the error is located in the header section, the message buffer needs to be re-configured. If the error is located in the data section, the error is corrected with the next data section update.

When a read error occurs during the transfer from the message RAM to the output buffer the output data structure will not be updated and the data available will not be set to '1', the FIFO put index and the FIFO fill level is not changed.

In case of an user output transfer request **FROTS.UORP** is set to '0' even if there was no update of the output data structure.

(3) **Read error during transfer from IBF to MBF**

This internal transfer is done for every input data transfer.

A read error can occur only when there is no update of the header section requested (the bit **LHS** in **FRWRHS4** is set to "0") due to the reading of the header section from the message RAM (see read error flags in **FRMHDS**). In this case the message buffer needs to be re-configured.

When a read error occurs during the input data transfer the actually transferred message in the input queue gets lost.

(4) **Message RAM read errors**

Read errors when reading the header section are flagged in the **FRMHDS** register.

Depending on the buffer type and set buffer protection a reconfiguration of the message buffer may not be possible.

The input transfer function cannot be used to reconfigure a locked message buffer using the method described in section 103.3.13.4.(3)

Before reconfiguring a locked buffers the user should disable the input transfer function and the output transfer function.

#### 103.3.16.6 Reserved

### 103.3.17 Byte alignment

The alignment of the bytes received by the FlexRay protocol and the alignment of the bytes required by the application may be different. The FlexRay module provides with **FROC.BEC** a byte alignment function to support different byte ordering styles.

Figure 103.30 shows the payload byte alignment within a FlexRay frame.

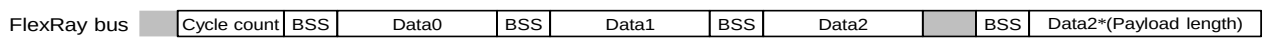


**Figure 103.30 Byte alignment on the FlexRay bus**

### 103.3.17.1 Little endian alignment

When **FROC.BEC** is '0' the byte alignment is set to Little endian.

(1) **FRNMVn (n=1 to 3)**

The byte alignment of the NMV bytes is as below.

(2) **FRWRDSn (n=1 to 64)**

The byte alignment for the message payload in the FlexRay input buffer and in the input data structure is as below.

FRWRDSn.MD[7:0]= $Data_{4n-4}$
FRWRDSn.MD[15:8]= $Data_{4n-3}$
FRWRDSn.MD[23:16]= $Data_{4n-2}$
FRWRDSn.MD[31:24]= $Data_{4n-1}$

Transmission order on the FlexRay bus is **FRWRDSn.MD**[7:0], **FRWRDSn.MD**[15:8], **FRWRDSn.MD**[23:16], **FRWRDSn.MD**[31:24] with the most significant bit transmitted first.

(3) **FRRDDSn (n=1 to 64)**

The byte alignment for the message payload in the FlexRay output buffer and in the output data structure is as below.

FRRDDSn.MD[7:0]= $Data_{4n-4}$
FRRDDSn.MD[15:8]= $Data_{4n-3}$
FRRDDSn.MD[23:16]= $Data_{4n-2}$
FRRDDSn.MD[31:24]= $Data_{4n-1}$

Reception order on the FlexRay bus is **FRRDDSn.MD**[7:0], **FRRDDSn.MD**[15:8], **FRRDDSn.MD**[23:16], **FRRDDSn.MD**[31:24] with the most significant bit received first.

### 103.3.17.2 Big endian alignment

When **FROC.BEC** is '1' the byte alignment is set to Big endian.

(1) **FRNMVn (n=1 to 3)**

The byte alignment of the NMV bytes is as below.

(2) **FRWRDSn (n=1 to 64)**

The byte alignment for the message payload in the FlexRay input buffer and in the input data structure is as below.

FRWRDSn.MD[7:0]= $Data_{4n-1}$
FRWRDSn.MD[15:8]= $Data_{4n-2}$
FRWRDSn.MD[23:16]= $Data_{4n-3}$
FRWRDSn.MD[31:24]= $Data_{4n-4}$

Transmission order on the FlexRay bus is **FRWRDSn.MD**[31:24], **FRWRDSn.MD**[23:16], **FRWRDSn.MD**[15:8], **FRWRDSn.MD**[7:0] with the most significant bit transmitted first.

(3) **FRRDDSn (n=1 to 64)**

The byte alignment for the message payload in the FlexRay output buffer and in the output data structure is as below.

FRRDDSn.MD[7:0]= $Data_{4n-1}$
FRRDDSn.MD[15:8]= $Data_{4n-2}$
FRRDDSn.MD[23:16]= $Data_{4n-3}$
FRRDDSn.MD[31:24]= $Data_{4n-4}$

Reception order on the FlexRay bus is **FRRDDSn.MD**[31:24], **FRRDDSn.MD**[23:16], **FRRDDSn.MD**[15:8], **FRRDDSn.MD**[7:0] with the most significant bit received first.

## 103.4 Usage Notes

No description on this section

## 103.5 Safety Mechanism

### 103.5.1 EDC/ECC for FlexRay

#### 103.5.1.1 Overview

EDC/ECC protection is applied to internal buffers (MBF, TBF A, TBF B) of FlexRay.

#### 103.5.1.2 Hardware description

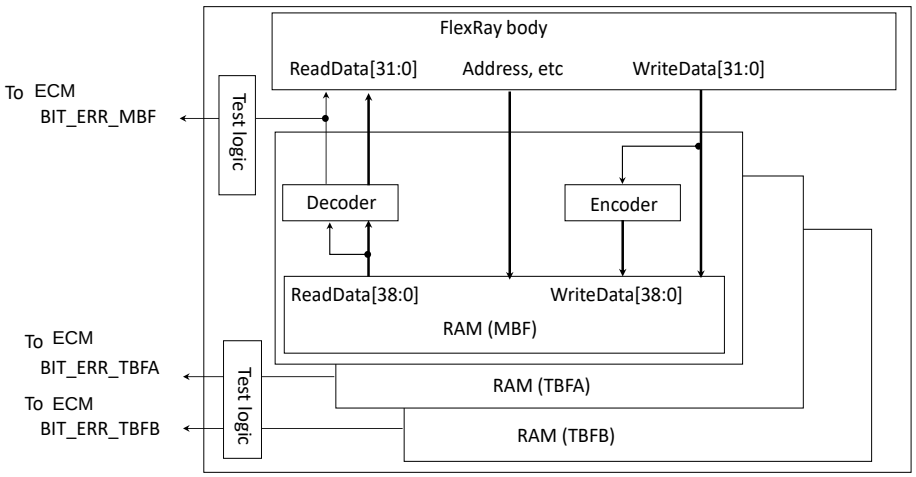



**Figure 103.31 EDC protection for FlexRay module internal buffers.**

**Table 103.18 Register configuration**

| Register name | Abbreviation | R/W | Address | Initial Value | Access size | |
|---|---|---|---|---|---|---|
| FlexRay Memory (MBF/TBFA/TBFB) Bit Error | FRMBE | R/W | H' E775 700C | H' 0000 0000 | 32 | |

(1) **FlexRay memory (MBF/TBFA/TBFB) Bit Error (FRMBE)**

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | mbfecc | mbfedc | tbfaecc | tbfaedc | tbfbecc | tbfbedc | frayecc |
| initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 7 | - | All 0 | R | Reserved<br>The write value should be always 0. |
| 6 | mbfecc | 0 | R/W | ECC_ERR_MBF signal test<br>0: User Signal<br>1: Test Signal (set to 1) |
| 5 | mbfedc | 0 | R/W | EDC_ERR_MBF signal test<br>0: User Signal<br>1: Test Signal (set to 1) |
| 4 | tbfaecc | 0 | R/W | ECC_ERR_TBFA signal test<br>0: User Signal<br>1: Test Signal (set to 1) |
| 3 | tbfaedc | 0 | R/W | EDC_ERR_TBFA signal test<br>0: User Signal<br>1: Test Signal (set to 1) |
| 2 | tbfbecc | 0 | R/W | ECC_ERR_TBFB signal test<br>0: User Signal<br>1: Test Signal (set to 1) |
| 1 | tbfbedc | 0 | R/W | EDC_ERR_TBFB signal test<br>0: User Signal<br>1: Test Signal (set to 1) |
| 0 | frayecc | 0 | R/W | FlexRay Buffer RAM (MBF/TBFA/TBFB) Error Correction mode select<br>0: EDC<br>1: ECC |

# 104. PCIe Controller

## 104.1 Overview

This module performs PCIe controls, and transfers data between this LSI internal bus (AXI bus) and the PCI devices connected to the PCIe Controller. This module facilitates the design of systems using the PCI Express while at the same time permitting high-speed data transfers in a compact system.

This module functions as a bus bridge connecting the PCI Express to this LSI internal bus (AXI bus), and provides transfer channels between PCI Express devices and devices connected to the LSI internal bus (AXI bus).

This LSI incorporates a PCI Express physical module that supports dual lanes and one channel.

This module has two modes: a Root Port and an Endpoint and can operate as a Root Port or Endpoint that is defined in the PCI Express specification.

This module supports AVS.

Note:

### 104.1.1 Features

This module has the following features.

- The AXI bus clocks are S0D2_HSCφ.
- The AXI bus support 32bit address.
- The APB bus clocks are S0D4_HSCφ.
- Functions as a bridge that connects the PCI Express to the LSI internal bus (AXI bus)
  - Generates a PCIe packet from the AXI bus transaction to the PCI area
  - Generates a AXI bus transaction from a PCI Express packet
- Supports high-speed data transfer between PCI Express devices and internal bus (AXI bus) via the internal DMAC
- Supports the subset of PCI Express Base Specification Revision 4.0 (Sep. 27, 2017)
- Operates as a PCI Express Endpoint or a PCI Express Root Port
- Supports the requester and completer functions for the PCIe transaction
- Incorporates configuration registers and the following capability structures
  - Power management capability structure
  - MSI capability structure (supports masking function and pending bits.)
  - PCI Express capability structure
  - Advanced Error Reporting (AER) Capability structure
  - Device Serial Number Capability structure
  - Vendor-Specific Extended Capability ( for RAS DP, RAS D.E.S., and PTM. )
  - Latency Tolerance Reporting (LTR) Capability structure
  - Secondary PCI Express Extended Capability structure
  - L1 PM Substates Extended Capability structure
  - Precision Time Management (PTM) Capability structure
  - Data Link Feature Extended Capability structure
  - Physical Layer 16.0 GT/s Extended Capability structure
  - Lane Margining at the Receiver Extended Capability structure
- Supports interrupts by INTx and MSI

- Supports one lane (x1), two lanes (x2) or four lanes (x4).
- Supports multiple speeds (2.5 GT/s, 5.0 GT/s, 8.0 GT/s and 16.0 GT/s).
- Automatically performs link training and link configurations
- Supports the D+/D-line automatic swap function (lane polarity inversion)
- Achieves QoS through transaction ordering.
- Supports only VC0.
- Automatically performs flow control by managing six types of credits
- Automatic handling function by hardware
- Buffer for holding error packet headers and error indication status provided
- Supports the link power control function (L0, L0s, L1 states and L1 PM Substates)
- Supports device power control function (D0, $D3_{hot}$, and $D3_{cold}$ states).
- Supports 256-byte max payload size.
- Supports completion timeout disable.
- Supports Multi-Function Device (Up to two functions)
- Supports FLR(Function Level Reset)
- Resizable BAR

Note: This module does not support the following PCI Express functions.

- Expansion ROM for system boot
- Card bus
- Phantom function
- Power-management event (PME) handling by hardware
- Slot function (including Hot Plug)
- SR-IOV
  - However supports for external SR-IOV capable EndPoint devices is provided in RootComplex mode
- BIST function
- ARI forwarding function
- Port arbitration function
- MSI-X interrupt
  - However supports MSI-X by INTC-AP
- Surprise Down error function
- TLP Prefix
- Peer-to-peer transfer
- ECRC Generation and Checking
- M-PCIe

Note: The configuration parameter of the PCIe controller in this document is shown as below.

- CC_MAX_MSTR_TAGS_AXI : 64
- CC_NUM_DMA_RD_CHAN : 8
- CC_NUM_DMA_WR_CHAN : 8
- CX_ACS_ENABLE : false (0)
- CX_ARI_ENABLE : false (0)
- CX_ATOMIC_ROUTING_EN : false (0)
- CX_ATU_MAX_REGION_SIZE : 4G (0)
- CX_ATU_MIN_REGION_SIZE : 4k (4096)
- CX_ATU_NUM_INBOUND_REGIONS : 16
- CX_ATU_NUM_OUTBOUND_REGIONS : 16
- CX_AUTO_LANE_FLIP_CTRL_EN : true (1)
- CX_COMM_NFTS : 200
- CX_GEN3_EQ_COEFQ_DEPTH : 8
- CX_IDO_ENABLE : false (0)
- CX_L1_SUBSTATES_ENABLE : true (1)

- CX_LBC_EXT_AW : 32
- CX_NPRFX : 0
- CX_PIPE_VER : PIPE4.4.1 (2)
- CX_PIPE43_P0_P1CPM : 1: P0 -> P1.CPM
- CX_PIPE43_P1CPM_ENCODING : 0x4
- CX_PTM_ENABLE : true (1)
- CX_PTM_EXTERNAL_MASTER_TIME : true (1)
- CX_SRIOV_ENABLE : false (0)
- DEFAULT_CLK_PM_CAP : 0x0
- DEFAULT_L0S_EXIT_LATENCY : 2us to 4us (0x6)
- DEFAULT_L1_EXIT_LATENCY : 32us to 64us (0x6)
- DEFAULT_TARGET : Drop (0x0)
- FLT_Q_ADDR_WIDTH : 64
- MULTI_DEVICE_AND_BUS_PER_FUNC_EN : false (0)
- SLAVE_BUS_ADDR_WIDTH : 64
- CX_NL : x4
- DEFAULT_FAST_LINK_SCALING_FACTOR : false (0)
- CC_DMA_EFETCH_EN : true (1)

### 104.1.2 Block Diagram

Figure 104.1 shows a block diagram of the PCIe Controller.

The PCIe Controller comprises a bridge, PCIe Controller, and PCIe Controller physical unit.

The bridge provides the bridge function of connecting the internal bus (AXI bus and APB bus) to the PCI Express controller.

The PCIe Controller is a block that controls the PCI Express packet transmission and reception, and it implements the functions of transaction and data link layers that are defined by the PCI Express standards. The configuration registers defined by the PCI Express standards are installed in this block.

The PCIe Controller physical unit is a block that controls PCI Express transmission paths and implements the physical layer function that is defined by the PCI Express standards.

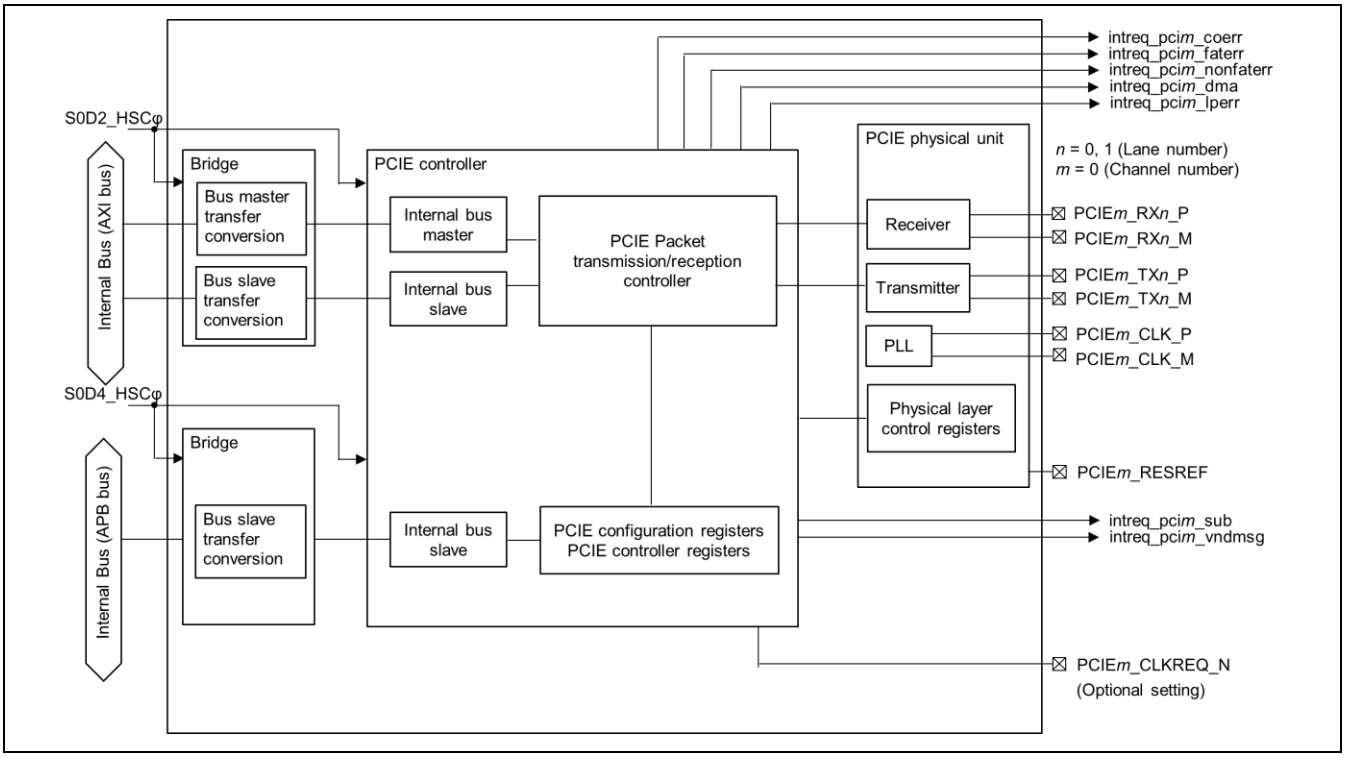



**Figure 104.1 Block Diagram**

### 104.1.3 External Pins

Table 104.1 shows the pin configuration.

**Table 104.1 Pin Configuration**

| Signal Name | Signal Conforming to PCI Standard | I/O | Description |
|---|---|---|---|
| PCIE*m*_CLK_P<br>PCIE*m*_CLK_M | REFCLK+<br>REFCLK- | Input | Reference clock inputs to the PLL incorporated in the PCIe Controller module (differential inputs). 100-MHz clock should be applied. |
| PCIE*m*_TX*n*_P<br>PCIE*m*_TX*n*_M<br>(*n*=0,1) | PETp*N*<br>PETn*N*<br>(*N*=0,1) | Output | Transmit data pins used by a physical module, where 2.5, 5.0, 8.0 or 16.0 GHz signals are propagated. (Differential output) |
| PCIE*m*_RX*n*_P<br>PCIE*m*_RX*n*_M<br>(*n*=0,1) | PERp*N*<br>PERn*N*<br>(*N*=0,1) | Input | Receive data pins used by a physical module, where 2.5, 5.0, 8.0 or 16.0 GHz signals are propagated. (Differential input) |
| PCIE*m*_RESREF | - | InOut | Reference Resistor Connection<br>Attach a 200-Ω ±1% ±100 ppm/°C precision resistor-to-ground on the board. |
| PCIE*m*_CLKREQ_N | CLKREQ# | I/O | Open drain pins to enable or disable the platform reference clocks.<br>These pins appear by pin multiplexing setting. |

*m=0*

### 104.1.4 Register Configuration

Table 104.3 shows the PCIe Controller registers. In Table 104.3, the Address Offset columns show an offset for register from the PCIe Controller base address.The actual access address is obtained by adding each offset to the PCIe Controller base address. Table 104.2 shows the PCIe Controller base address. These registers should be accessed through the APB bus. Only the little endian mode can be used for this module.

**Table 104.2 PCIe Controller Base Address**

| | PCI Express |
|---|---|
| PCIE0 Base Address | H'E65D 0000 |

**Table 104.3 PCIe Controller Register Configuration**

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| PCIE*m* PCI Configuration for Function0 (Common address in Root Complex mode and Endpoint mode) | | | | | | | √ |
| PCI configuration register 0 function 0 | PCICONF0F0 | R | R | H'0000 | H'xxxx xxxx | 32 | √ |
| PCI configuration register 1 function 0 | PCICONF1F0 | R/W | R/W | H'0004 | H'xxxx xxxx | 32 | √ |
| PCI configuration register 2 function 0 | PCICONF2F0 | R | R | H'0008 | H'xxxx xxxx | 32 | √ |
| PCI configuration register 3 function 0 | PCICONF3F0 | R/W | R/W | H'000C | H'xxxx xxxx | 32 | √ |
| PCI configuration register 4 function 0 | PCICONF4F0 | *2 | *3 | H'0010 | *2 | 32 | √ |
| PCI configuration register 5 function 0 | PCICONF5F0 | *4 | *3 | H'0014 | *4 | 32 | √ |
| PCI configuration register 6 function 0 | PCICONF6F0 | Header TYPE00:*5<br>Header TYPE01: R/W | Header TYPE00: *3<br>Header TYPE01: — | H'0018 | Header TYPE00:*5<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 7 function 0 | PCICONF7F0 | Header TYPE00:*6<br>Header TYPE01: R/W | Header TYPE00: *3<br>Header TYPE01: — | H'001C | Header TYPE00:*6<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 8 function 0 | PCICONF8F0 | Header TYPE00:*7<br>Header TYPE01: R/W | Header TYPE00: *3<br>Header TYPE01: — | H'0020 | Header TYPE00:*7<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 9 function 0 | PCICONF9F0 | Header TYPE00:*8<br>Header TYPE01: R/W | Header TYPE00: *3<br>Header TYPE01: — | H'0024 | Header TYPE00:*8<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 10 function 0 | PCICONF10F0 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'0028 | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 11 function 0 | PCICONF11F0 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'002C | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 12 function 0 | PCICONF12F0 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'0030 | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| PCI configuration register 13 function 0 | PCICONF13F0 | R/W | R | H'0034 | H'xxxx xxxx | 32 | √ |
| PCI configuration register 14 function 0 | PCICONF14F0 | Header TYPE00: R<br>Header TYPE01: R | Header TYPE00: R<br>Header TYPE01: — | H'0038 | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 15 function 0 | PCICONF15F0 | Header TYPE00: R/W<br>Header TYPE01: R/W | Header TYPE00: R/W<br>Header TYPE01: — | H'003C | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI power management capability register 0 function 0 | PMCAP0F0 | R | R | H'0040 | H'xxxx xxxx | 32 | √ |
| PCI power management capability register 1 function 0 | PMCAP1F0 | R/W | R/W | H'0044 | H'xxxx xxxx | 32 | √ |
| MSI capability register 0 function 0 | MSICAP0F0 | R/W | R/W | H'0050 | H'xxxx xxxx | 32 | √ |
| MSI capability register 1 function 0 | MSICAP1F0 | R/W | R/W | H'0054 | H'xxxx xxxx | 32 | √ |
| MSI capability register 2 function 0 | MSICAP2F0 | R/W | R/W | H'0058 | H'xxxx xxxx | 32 | √ |
| MSI capability register 3 function 0 | MSICAP3F0 | R/W | R/W | H'005C | H'xxxx xxxx | 32 | √ |
| MSI capability register 4 function 0 | MSICAP4F0 | R/W | R/W | H'0060 | H'xxxx xxxx | 32 | √ |
| MSI capability register 5 function 0 | MSICAP5F0 | R | R | H'0064 | H'xxxx xxxx | 32 | √ |
| PCI Express capability register 0 function0<br>PCI Express capability list register<br>PCI Express capabilities register | EXPCAP0F0 | R/W | R | H'0070 | | 32 | √ |
| PCI Express capability register 1 function0<br>Device capabilities register | EXPCAP1F0 | R/W | R | H'0074 | | 32 | √ |
| PCI Express capability register 2 function0<br>Device control register<br>Device status register | EXPCAP2F0 | R/W | R/W | H'0078 | | 32 | √ |
| PCI Express capability register 3 function0<br>Link capabilities register | EXPCAP3F0 | R/W | R | H'007C | | 32 | √ |
| PCI Express capability register 4 function0<br>Link control register | EXPCAP4F0 | R/W | R/W | H'0080 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Link status register | | | | | | | |
| PCI Express capability register 5 function0<br>Slot capabilities register | EXPCAP5F0 | Header TYPE00: R<br>Header TYPE01: R | Header TYPE00: R<br>Header TYPE01: — | H'0084 | Header TYPE00: H'0000 0000<br>Header TYPE01: H'0000 0000 | 32 | √ |
| PCI Express capability register 6 function0<br>Slot control register<br>Slot status register | EXPCAP6F0 | Header TYPE00: R<br>Header TYPE01: R | Header TYPE00: R<br>Header TYPE01: — | H'0088 | Header TYPE00: H'0000 0000<br>Header TYPE01: H'0040 0000 | 32 | √ |
| PCI Express capability register 7 function0<br>Root control register<br>Root capabilities register | EXPCAP7F0 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'008C | Header TYPE00: H'0000 0000<br>Header TYPE01: H'0000 0000 | 32 | √ |
| PCI Express capability register 8 function0<br>Root status register | EXPCAP8F0 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'0090 | Header TYPE00: H'0000 0000<br>Header TYPE01: H'0000 0000 | 32 | √ |
| PCI Express capability register 9 function0<br>Device capabilities 2 register | EXPCAP9F0 | R/W | R | H'0094 | | 32 | √ |
| PCI Express capability register 10 function0<br>Device control 2 register<br>Device status 2 register | EXPCAP10F0 | R/W | R/W | H'0098 | | 32 | √ |
| PCI Express capability register 11 function0<br>Link capabilities 2 register | EXPCAP11F0 | R | R | H'009C | | 32 | √ |
| PCI Express capability register 12 function0<br>Link control 2 register<br>Link status 2 register | EXPCAP12F0 | R/W | R/W | H'00A0 | | 32 | √ |
| Advanced Error Reporting Capability Register 0 function 0 | AERCAP0F0 | R/W | R | H'0100 | | 32 | √ |
| Advanced Error Reporting Capability Register 1 function 0 | AERCAP1F0 | R/W | R/W | H'0104 | | 32 | √ |
| Advanced Error Reporting Capability Register 2 function 0 | AERCAP2F0 | R/W | R/W | H'0108 | | 32 | √ |
| Advanced Error Reporting Capability Register 3 function 0 | AERCAP3F0 | R/W | R/W | H'010C | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Advanced Error Reporting Capability Register 4 function 0 | AERCAP4F0 | R/W | R/W | H'0110 | | 32 | √ |
| Advanced Error Reporting Capability Register 5 function 0 | AERCAP5F0 | R/W | R/W | H'0114 | | 32 | √ |
| Advanced Error Reporting Capability Register 6 function 0 | AERCAP6F0 | R/W | R/W | H'0118 | | 32 | √ |
| Advanced Error Reporting Capability Register 7 function 0 | AERCAP7F0 | R | R | H'011C | | 32 | √ |
| Advanced Error Reporting Capability Register 8 function 0 | AERCAP8F0 | R | R | H'0120 | | 32 | √ |
| Advanced Error Reporting Capability Register 9 function 0 | AERCAP9F0 | R | R | H'0124 | | 32 | √ |
| Advanced Error Reporting Capability Register 10 function 0 | AERCAP10F0 | R | R | H'0128 | | 32 | √ |
| Advanced Error Reporting Capability Register 11 function 0 | AERCAP11F0 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: R/W | H'012C | | 32 | √ |
| Advanced Error Reporting Capability Register 12 function 0 | AERCAP12F0 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: R/W | H'0130 | | 32 | √ |
| Advanced Error Reporting Capability Register 13 function 0 | AERCAP13F0 | Header TYPE00: R<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'0134 | | 32 | √ |
| Device Serial Number Capability Register 0 function 0 | SERNUMCAP0F0 | R/W | R | H'0148 | | 32 | √ |
| Device Serial Number Capability Register 1 function 0 | SERNUMCAP1F0 | R | R | H'014C | | 32 | √ |
| Device Serial Number Capability Register 2 function 0 | SERNUMCAP2F0 | R | R | H'0150 | | 32 | √ |
| Secondary PCI Express Extended Capability Register 0 function 0 | SEXPECAP0F0 | R/W | R | H'0158 | | 32 | √ |
| Secondary PCI Express Extended Capability Register 1 function 0 | SEXPECAP1F0 | R | R | H'015C | | 32 | √ |
| Secondary PCI Express Extended Capability Register 2 function 0 | SEXPECAP2F0 | R/W | R/W | H'0160 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Secondary PCI Express Extended Capability Register 3 function 0 | SEXPECAP3F0 | R/W | R | H'0164 | | 32 | √ |
| Secondary PCI Express Extended Capability Register 4 function 0 | SEXPECAP4F0 | R/W | R | H'0168 | | 32 | √ |
| Physical Layer 16.0 GT/s Extended Capability Register 0 function 0 | PHY16ECAP0F0 | R/W | R | H'0178 | | 32 | √ |
| Physical Layer 16.0 GT/s Extended Capability Register 1 function 0 | PHY16ECAP1F0 | R | R | H'017C | | 32 | √ |
| Physical Layer 16.0 GT/s Extended Capability Register 2 function 0 | PHY16ECAP2F0 | R | R | H'0180 | | 32 | √ |
| Physical Layer 16.0 GT/s Extended Capability Register 3 function 0 | PHY16ECAP3F0 | R/W | R/W | H'0184 | | 32 | √ |
| Physical Layer 16.0 GT/s Extended Capability Register 4 function 0 | PHY16ECAP4F0 | R/W | R/W | H'0188 | | 32 | √ |
| Physical Layer 16.0 GT/s Extended Capability Register 5 function 0 | PHY16ECAP5F0 | R/W | R/W | H'018C | | 32 | √ |
| Physical Layer 16.0 GT/s Extended Capability Register 6 function 0 | PHY16ECAP6F0 | R/W | R/W | H'0190 | | 32 | √ |
| Physical Layer 16.0 GT/s Extended Capability Register 7 function 0 | PHY16ECAP7F0 | R/W | R | H'0198 | | 32 | √ |
| Lane Margining at the Receiver Extended Capability Register 0 function 0 | LNMGNCAP0F0 | R/W | R | H'019C | | 32 | √ |
| Lane Margining at the Receiver Extended Capability Register 1 function 0 | LNMGNCAP1F0 | R | R | H'01A0 | | 32 | √ |
| Lane Margining at the Receiver Extended Capability Register 2 function 0 | LNMGNCAP2F0 | R/W | R/W | H'01A4 | | 32 | √ |
| Lane Margining at the Receiver Extended Capability Register 3 function 0 | LNMGNCAP3F0 | R/W | R/W | H'01A8 | | 32 | √ |
| Lane Margining at the Receiver Extended Capability Register 4 function 0 | LNMGNCAP4F0 | R/W | R/W | H'01AC | | 32 | √ |
| Lane Margining at the Receiver Extended Capability Register 5 function 0 | LNMGNCAP5F0 | R/W | R/W | H'01B0 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Latency Tolerance Reporting (LTR) Capability Register 0 function 0 | LTRCAP0F0 | R/W | R | H'01B4 | | 32 | √ |
| Latency Tolerance Reporting (LTR) Capability Register 1 function 0 | LTRCAP1F0 | R/W | R/W | H'01B8 | | 32 | √ |
| L1 PM Substates Extended Capability Register 0 function 0 | L1PSCAP0F0 | R/W | R | H'01BC | | 32 | √ |
| L1 PM Substates Extended Capability Register 1 function 0 | L1PSCAP1F0 | R/W | R | H'01C0 | | 32 | √ |
| L1 PM Substates Extended Capability Register 2 function 0 | L1PSCAP2F0 | R/W | R/W | H'01C4 | | 32 | √ |
| L1 PM Substates Extended Capability Register 3 function 0 | L1PSCAP3F0 | R/W | R/W | H'01C8 | | 32 | √ |
| Vendor-Specific Extended Capability Register 0 function 0 | VSECAP0F0 | R/W | R | H'01CC | | 32 | √ |
| Vendor-Specific Extended Capability Register 1 function 0 | VSECAP1F0 | R | R | H'01D0 | | 32 | √ |
| Vendor-Specific Extended Capability Register 2 function 0 | VSECAP2F0 | R/W | R/W | H'01D4 | | 32 | √ |
| Vendor-Specific Extended Capability Register 3 function 0 | VSECAP3F0 | R | R | H'01D8 | | 32 | √ |
| Vendor-Specific Extended Capability Register 4 function 0 | VSECAP4F0 | R/W | R/W | H'01DC | | 32 | √ |
| Vendor-Specific Extended Capability Register 5 function 0 | VSECAP5F0 | R | R | H'01E0 | | 32 | √ |
| Vendor-Specific Extended Capability Register 6 function 0 | VSECAP6F0 | R/W | R/W | H'01FC | | 32 | √ |
| Vendor-Specific Extended Capability Register 7 function 0 | VSECAP7F0 | R/W | R/W | H'0200 | | 32 | √ |
| Vendor-Specific Extended Capability Register 8 function 0 | VSECAP8F0 | R/W | R/W | H'0204 | | 32 | √ |
| Vendor-Specific Extended Capability Register 9 function 0 | VSECAP9F0 | R/W | R/W | H'0208 | | 32 | √ |
| Vendor-Specific Extended Capability Register 10 function 0 | VSECAP10F0 | R/W | R/W | H'020C | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Vendor-Specific Extended Capability Register 11 function 0 | VSECAP11F0 | R/W | R/W | H'0210 | | 32 | √ |
| Vendor-Specific Extended Capability Register 12 function 0 | VSECAP12F0 | R/W | R/W | H'0214 | | 32 | √ |
| Vendor-Specific Extended Capability Register 13 function 0 | VSECAP13F0 | R/W | R/W | H'0218 | | 32 | √ |
| Vendor-Specific Extended Capability Register 14 function 0 | VSECAP14F0 | R/W | R/W | H'021C | | 32 | √ |
| Vendor-Specific Extended Capability Register 15 function 0 | VSECAP15F0 | R/W | R/W | H'0220 | | 32 | √ |
| Vendor-Specific Extended Capability Register 16 function 0 | VSECAP16F0 | R/W | R/W | H'0224 | | 32 | √ |
| Vendor-Specific Extended Capability Register 17 function 0 | VSECAP17F0 | R/W | R/W | H'0228 | | 32 | √ |
| Vendor-Specific Extended Capability Register 18 function 0 | VSECAP18F0 | R/W | R/W | H'022C | | 32 | √ |
| Vendor-Specific Extended Capability Register 19 function 0 | VSECAP19F0 | R/W | R/W | H'0230 | | 32 | √ |
| Vendor-Specific Extended Capability Register 20 function 0 | VSECAP20F0 | R/W | R/W | H'0234 | | 32 | √ |
| Vendor-Specific Extended Capability Register 21 function 0 | VSECAP21F0 | R/W | R/W | H'0238 | | 32 | √ |
| Vendor-Specific Extended Capability Register 22 function 0 | VSECAP22F0 | R/W | R/W | H'023C | | 32 | √ |
| Vendor-Specific Extended Capability Register 23 function 0 | VSECAP23F0 | R/W | R/W | H'0240 | | 32 | √ |
| Vendor-Specific Extended Capability Register 24 function 0 | VSECAP24F0 | R/W | R/W | H'0244 | | 32 | √ |
| Vendor-Specific Extended Capability Register 25 function 0 | VSECAP25F0 | R/W | R/W | H'0248 | | 32 | √ |
| Vendor-Specific Extended Capability Register 26 function 0 | VSECAP26F0 | R/W | R/W | H'024C | | 32 | √ |
| Vendor-Specific Extended Capability Register 27 function 0 | VSECAP27F0 | R/W | R/W | H'0250 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Vendor-Specific Extended Capability Register 28 function 0 | VSECAP28F0 | R/W | R/W | H'0254 | | 32 | √ |
| Vendor-Specific Extended Capability Register 29 function 0 | VSECAP29F0 | R/W | R/W | H'0258 | | 32 | √ |
| Vendor-Specific Extended Capability Register 30 function 0 | VSECAP30F0 | R/W | R/W | H'026C | | 32 | √ |
| Vendor-Specific Extended Capability Register 31 function 0 | VSECAP31F0 | R/W | R/W | H'0270 | | 32 | √ |
| Vendor-Specific Extended Capability Register 32 function 0 | VSECAP32F0 | R/W | R/W | H'027C | | 32 | √ |
| Vendor-Specific Extended Capability Register 33 function 0 | VSECAP33F0 | R/W | R/W | H'0280 | | 32 | √ |
| Vendor-Specific Extended Capability Register 34 function 0 | VSECAP34F0 | R/W | R/W | H'0284 | | 32 | √ |
| Vendor-Specific Extended Capability Register 35 function 0 | VSECAP35F0 | R/W | R/W | H'0288 | | 32 | √ |
| Vendor-Specific Extended Capability Register 36 function 0 | VSECAP36F0 | R/W | R/W | H'028C | | 32 | √ |
| Vendor-Specific Extended Capability Register 37 function 0 | VSECAP37F0 | R/W | R/W | H'0290 | | 32 | √ |
| Vendor-Specific Extended Capability Register 38 function 0 | VSECAP38F0 | R/W | R/W | H'029C | | 32 | √ |
| Vendor-Specific Extended Capability Register 39 function 0 | VSECAP39F0 | R/W | R/W | H'02A0 | | 32 | √ |
| Vendor-Specific Extended Capability Register 40 function 0 | VSECAP40F0 | R/W | R/W | H'02A4 | | 32 | √ |
| Vendor-Specific Extended Capability Register 41 function 0 | VSECAP41F0 | R | R | H'02AC | | 32 | √ |
| Vendor-Specific Extended Capability Register 42 function 0 | VSECAP42F0 | R | R | H'02B0 | | 32 | √ |
| Vendor-Specific Extended Capability Register 43 function 0 | VSECAP43F0 | R | R | H'02B4 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 0 function 0 | VSE2CAP0F0 | R | R | H'02CC | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Vendor-Specific Extended 2 Capability Register 1 function 0 | VSE2CAP1F0 | R | R | H'02D0 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 2 function 0 | VSE2CAP2F0 | R/W | R/W | H'02D4 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 3 function 0 | VSE2CAP3F0 | R/W | R/W | H'02D8 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 4 function 0 | VSE2CAP4F0 | R | R | H'02DC | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 5 function 0 | VSE2CAP5F0 | R/W | R/W | H'02E0 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 6 function 0 | VSE2CAP6F0 | R | R | H'02E4 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 7 function 0 | VSE2CAP7F0 | R/W | R/W | H'02E8 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 8 function 0 | VSE2CAP8F0 | R | R | H'02EC | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 9 function 0 | VSE2CAP9F0 | R | R | H'02F0 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 10 function 0 | VSE2CAP10F0 | R/W | R/W | H'02F4 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 11 function 0 | VSE2CAP11F0 | R/W | R/W | H'02F8 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 12 function 0 | VSE2CAP12F0 | R | R | H'02FC | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 13 function 0 | VSE2CAP13F0 | R | R | H'0300 | | 32 | √ |
| Data Link Feature Extended Capability Register 0 function 0 | DLFECAP0F0 | R/W | R | H'0304 | | 32 | √ |
| Data Link Feature Extended Capability Register 1 function 0 | DLFECAP1F0 | R | R | H'0308 | | 32 | √ |
| Data Link Feature Extended Capability Register 2 function 0 | DLFECAP2F0 | R/W | R | H'030C | | 32 | √ |
| Precision Time Management (PTM) Capability Register 0 function 0 | PTMCAP0F0 | R/W | R | H'0310 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Precision Time Management (PTM) Capability Register 1 function 0 | PTMCAP1F0 | R/W | R | H'0314 | | 32 | √ |
| Precision Time Management (PTM) Capability Register 2 function 0 | PTMCAP2F0 | R/W | R/W | H'0318 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 0 function 0 | PTMRSCAP0F0 | Header TYPE00: —<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: — | H'031C | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 1 function 0 | PTMRSCAP1F0 | Header TYPE00: —<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: — | H'0320 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 2 function 0 | PTMRSCAP2F0 | Header TYPE00: —<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: R | H'0324 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 3 function 0 | PTMRSCAP3F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'0328 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 4 function 0 | PTMRSCAP4F0 | Header TYPE00: —<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: — | H'032C | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 5 function 0 | PTMRSCAP5F0 | Header TYPE00: —<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: — | H'0330 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 6 function 0 | PTMRSCAP6F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'0334 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 7 function 0 | PTMRSCAP7F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'0338 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 8 function 0 | PTMRSCAP8F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'033C | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 9 function 0 | PTMRSCAP9F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'0340 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| PTM Responder Capability Structure (VSEC) Register 10 function 0 | PTMRSCAP10F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'0344 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 11 function 0 | PTMRSCAP11F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'0348 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 12 function 0 | PTMRSCAP12F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'034C | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 13 function 0 | PTMRSCAP13F0 | Header TYPE00: —<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'0350 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 14 function 0 | PTMRSCAP14F0 | Header TYPE00: —<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: R/W | H'0354 | | 32 | √ |
| PTM Responder Capability Structure (VSEC) Register 15 function 0 | PTMRSCAP15F0 | Header TYPE00: —<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: R/W | H'0358 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 0 function 0 | PTMRQCAP0F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'031C | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 1 function 0 | PTMRQCAP1F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'0320 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 2 function 0 | PTMRQCAP2F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'0324 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 3 function 0 | PTMRQCAP3F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0328 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 4 function 0 | PTMRQCAP4F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'032C | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| PTM Requestor Capability Structure (VSEC) Register 5 function 0 | PTMRQCAP5F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0330 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 6 function 0 | PTMRQCAP6F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0334 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 7 function 0 | PTMRQCAP7F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0338 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 8 function 0 | PTMRQCAP8F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'033C | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 9 function 0 | PTMRQCAP9F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0340 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 10 function 0 | PTMRQCAP10F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0344 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 11 function 0 | PTMRQCAP11F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0348 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 12 function 0 | PTMRQCAP12F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'034C | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 13 function 0 | PTMRQCAP13F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0350 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 14 function 0 | PTMRQCAP14F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0354 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 15 function 0 | PTMRQCAP15F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0358 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 16 function 0 | PTMRQCAP16F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'035C | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 17 function 0 | PTMRQCAP17F0 | Header TYPE00: R | Header TYPE00: R | H'0360 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| | | Header TYPE01: — | Header TYPE01: — | | | | |
| PTM Requestor Capability Structure (VSEC) Register 18 function 0 | PTMRQCAP18 F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'0364 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 19 function 0 | PTMRQCAP19 F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'0368 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 20 function 0 | PTMRQCAP20 F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'036C | | 32 | √ |
| Resizable BAR Capability Header function 0 | RESBAR_CAP _HDR _F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'039C | | 32 | √ |
| Resizable BAR0 Capability Register function 0 | RESBAR_CAP _REG_0_REG _F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'03A0 | | 32 | √ |
| Resizable BAR0 Control Register function 0 | RESBAR_CTR L_REG_0_REG _F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'03A4 | | 32 | √ |
| Resizable BAR2 Capability Register function 0 | RESBAR_CAP _REG_2_REG _F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'03A8 | | 32 | √ |
| Resizable BAR2 Control Register function 0 | RESBAR_CTR L_REG_2_REG _F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'03AC | | 32 | √ |
| Resizable BAR5 Capability Register function 0 | RESBAR_CAP _REG_5_REG _F0 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'03B0 | | 32 | √ |
| Resizable BAR5 Control Register function 0 | RESBAR_CTR L_REG_5_REG _F0 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'03B4 | | 32 | √ |
| Port Logic Registers 0 | PRTLGC0 | R/W | R/W | H'0700 | H'308D 102F | 32 | √ |
| Port Logic Registers 1 | PRTLGC1 | R/W | R/W | H'0704 | H'FFFF FFFF | 32 | √ |
| Port Logic Registers 2 | PRTLGC2 | R/W | R/W | H'0708 | H'0000 0000 | 32 | √ |
| Port Logic Registers 3 | PRTLGC3 | R/W | R/W | H'070C | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Port Logic Registers 4 | PRTLGC4 | R/W | R/W | H'0710 | | 32 | √ |
| Port Logic Registers 5 | PRTLGC5 | R/W | R/W | H'0714 | | 32 | √ |
| Port Logic Registers 6 | PRTLGC6 | R/W | R/W | H'0718 | | 32 | √ |
| Port Logic Registers 7 | PRTLGC7 | R/W | R/W | H'071C | | 32 | √ |
| Port Logic Registers 8 | PRTLGC8 | R/W | R/W | H'0720 | | 32 | √ |
| Port Logic Registers 9 | PRTLGC9 | R/W | R/W | H'0724 | | 32 | √ |
| Port Logic Registers 10 | PRTLGC10 | R | R | H'0728 | | 32 | √ |
| Port Logic Registers 11 | PRTLGC11 | R | R | H'072C | | 32 | √ |
| Port Logic Registers 12 | PRTLGC12 | R | R | H'0730 | | 32 | √ |
| Port Logic Registers 13 | PRTLGC13 | R | R | H'0734 | | 32 | √ |
| Port Logic Registers 14 | PRTLGC14 | R | R | H'0738 | | 32 | √ |
| Port Logic Registers 15 | PRTLGC15 | R/W | R/W | H'073C | | 32 | √ |
| Port Logic Registers 16 | PRTLGC16 | R | R | H'0740 | | 32 | √ |
| Port Logic Registers 17 | PRTLGC17 | R | R | H'0744 | | 32 | √ |
| Port Logic Registers 18 | PRTLGC18 | R/W | R/W | H'0748 | | 32 | √ |
| Port Logic Registers 19 | PRTLGC19 | R/W | R/W | H'074C | | 32 | √ |
| Port Logic Registers 20 | PRTLGC20 | R/W | R/W | H'0750 | | 32 | √ |
| Port Logic Registers 21 | PRTLGC21 | R/W | R/W | H'080C | | 32 | √ |
| Port Logic Registers 22 | PRTLGC22 | R | R | H'0810 | | 32 | √ |
| Port Logic Registers 23 | PRTLGC23 | R/W | R/W | H'0814 | | 32 | √ |
| Port Logic Registers 24 | PRTLGC24 | R/W | R/W | H'081C | | 32 | √ |
| Port Logic Registers 25 | PRTLGC25 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'0820 | | 32 | √ |
| Port Logic Registers 26 | PRTLGC26 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'0824 | | 32 | √ |
| Port Logic Registers 27+3*i (*i* = 0 to 7) | PRTLGC27+3*i (*i* = 0 to 7) | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'0828+H'C*i (*i* = 0 to 7) | | 32 | √ |
| Port Logic Registers 28+3*i (*i* = 0 to 7) | PRTLGC28+3*i (*i* = 0 to 7) | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'082C+H'C*i (*i* = 0 to 7) | | 32 | √ |
| Port Logic Registers 29+3*i (*i* = 0 to 7) | PRTLGC29+3*i (*i* = 0 to 7) | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'0830+H'C*i (*i* = 0 to 7) | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Port Logic Registers 51 | PRTLGC51 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'0888 | | 32 | √ |
| Port Logic Registers 52 | PRTLGC52 | R/W | R/W | H'088C | | 32 | √ |
| Port Logic Registers 53 | PRTLGC53 | R/W | R/W | H'0890 | | 32 | √ |
| Port Logic Registers 54 | PRTLGC54 | R/W | R/W | H'08A8 | | 32 | √ |
| Port Logic Registers 55 | PRTLGC55 | R/W | R/W | H'08AC | | 32 | √ |
| Port Logic Registers 56 | PRTLGC56 | R/W | R/W | H'08B4 | | 32 | √ |
| Port Logic Registers 57 | PRTLGC57 | R/W | R/W | H'08B8 | | 32 | √ |
| Port Logic Registers 58 | PRTLGC58 | R/W | R/W | H'08BC | | 32 | √ |
| Port Logic Registers 59 | PRTLGC59 | R/W | R/W | H'08C0 | | 32 | √ |
| Port Logic Registers 60 | PRTLGC60 | R/W | R/W | H'08C4 | | 32 | √ |
| Port Logic Registers 61 | PRTLGC61 | R/W | R/W | H'08C8 | | 32 | √ |
| Port Logic Registers 62 | PRTLGC62 | R/W | R/W | H'08CC | | 32 | √ |
| Port Logic Registers 63 | PRTLGC63 | R/W | R/W | H'08D0 | | 32 | √ |
| Port Logic Registers 64 | PRTLGC64 | R/W | R/W | H'08D4 | | 32 | √ |
| Port Logic Registers 65 | PRTLGC65 | R/W | R/W | H'08D8 | | 32 | √ |
| Port Logic Registers 66 | PRTLGC66 | R/W | R/W | H'08E0 | | 32 | √ |
| Port Logic Registers 67 | PRTLGC67 | R/W | R/W | H'08E4 | | 32 | √ |
| Port Logic Registers 68 | PRTLGC68 | R/W | R/W | H'08E8 | | 32 | √ |
| Port Logic Registers 69 | PRTLGC69 | R/W | R/W | H'08F0 | | 32 | √ |
| Port Logic Registers 70 | PRTLGC70 | R/W | R/W | H'08F4 | | 32 | √ |
| Port Logic Registers 73 | PRTLGC73 | R/W | R/W | H'0930 | | 32 | √ |
| Port Logic Registers 74 | PRTLGC74 | R/W | R/W | H'0934 | | 32 | √ |
| Port Logic Registers 75 | PRTLGC75 | R/W | R/W | H'0938 | | 32 | √ |
| Port Logic Registers 80 | PRTLGC80 | R/W | R/W | H'0960 | | 32 | √ |
| Port Logic Registers 81 | PRTLGC81 | R/W | R/W | H'0964 | | 32 | √ |
| Port Logic Registers 82 | PRTLGC82 | R/W | R/W | H'0B20 | | 32 | √ |
| Port Logic Registers 83 | PRTLGC83 | R/W | R/W | H'0B24 | | 32 | √ |
| Port Logic Registers 84 | PRTLGC84 | R | R | H'0B28 | | 32 | √ |
| Port Logic Registers 85 | PRTLGC85 | R | R | H'0B2C | | 32 | √ |
| Port Logic Registers 86 | PRTLGC86 | R/W | R | H'0B30 | | 32 | √ |
| Port Logic Registers 87 | PRTLGC87 | R/W | R/W | H'0B40 | | 32 | √ |
| Port Logic Registers 88 | PRTLGC88 | R/W | R/W | H'0B44 | | 32 | √ |
| Port Logic Registers 89 | PRTLGC89 | R/W | R/W | H'0B70 | | 32 | √ |
| Port Logic Registers 90 | PRTLGC90 | R/W | R/W | H'0B74 | | 32 | √ |
| Port Logic Registers 91 | PRTLGC91 | R/W | R/W | H'0B80 | | 32 | √ |
| Port Logic Registers 92 | PRTLGC92 | R/W | R/W | H'0B84 | | 32 | √ |
| Port Logic Registers 93 | PRTLGC93 | R/W | R/W | H'0B90 | | 32 | √ |
| | PCIE*m* PCI Configuration for Function1 (Endpoint mode only) | | | | | | √ |
| PCI configuration register 0 function 1 | PCICONF0F1 | R | R | H'1000 | H'xxxx xxxx | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| PCI configuration register 1 function 1 | PCICONF1F1 | R/W | R/W | H'1004 | H'xxxx xxxx | 32 | √ |
| PCI configuration register 2 function 1 | PCICONF2F1 | R | R | H'1008 | H'xxxx xxxx | 32 | √ |
| PCI configuration register 3 function 1 | PCICONF3F1 | R/W | R/W | H'100C | H'xxxx xxxx | 32 | √ |
| PCI configuration register 4 function 1 | PCICONF4F1 | *2 | *3 | H'1010 | *2 | 32 | √ |
| PCI configuration register 5 function 1 | PCICONF5F1 | *4 | *3 | H'1014 | *4 | 32 | √ |
| PCI configuration register 6 function 1 | PCICONF6F1 | Header TYPE00:*5<br>Header TYPE01: R/W | Header TYPE00: *3<br>Header TYPE01: — | H'1018 | Header TYPE00:*5<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 7 function 1 | PCICONF7F1 | Header TYPE00:*6<br>Header TYPE01: R/W | Header TYPE00: *3<br>Header TYPE01: — | H'101C | Header TYPE00:*6<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 8 function 1 | PCICONF8F1 | Header TYPE00:*7<br>Header TYPE01: R/W | Header TYPE00: *3<br>Header TYPE01: — | H'1020 | Header TYPE00:*7<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 9 function 1 | PCICONF9F1 | Header TYPE00:*8<br>Header TYPE01: R/W | Header TYPE00: *3<br>Header TYPE01: — | H'1024 | Header TYPE00:*8<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 10 function 1 | PCICONF10F1 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'1028 | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 11 function 1 | PCICONF11F1 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'102C | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 12 function 1 | PCICONF12F1 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'1030 | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI configuration register 13 function 1 | PCICONF13F1 | R/W | R | H'1034 | H'xxxx xxxx | 32 | √ |
| PCI configuration register 14 function 1 | PCICONF14F1 | Header TYPE00: R | Header TYPE00: R | H'1038 | Header TYPE00: H'xxxx xxxx | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| | | Header TYPE01: R | Header TYPE01: — | | Header TYPE01: H'xxxx xxxx | | |
| PCI configuration register 15 function 1 | PCICONF15F1 | Header TYPE00: R/W<br>Header TYPE01: R/W | Header TYPE00: R/W<br>Header TYPE01: — | H'103C | Header TYPE00: H'xxxx xxxx<br>Header TYPE01: H'xxxx xxxx | 32 | √ |
| PCI power management capability register 0 function 1 | PMCAP0F1 | R | R | H'1040 | H'xxxx xxxx | 32 | √ |
| PCI power management capability register 1 function 1 | PMCAP1F1 | R/W | R/W | H'1044 | H'xxxx xxxx | 32 | √ |
| MSI capability register 0 function 1 | MSICAP0F1 | R/W | R/W | H'1050 | H'xxxx xxxx | 32 | √ |
| MSI capability register 1 function 1 | MSICAP1F1 | R/W | R/W | H'1054 | H'xxxx xxxx | 32 | √ |
| MSI capability register 2 function 1 | MSICAP2F1 | R/W | R/W | H'1058 | H'xxxx xxxx | 32 | √ |
| MSI capability register 3 function 1 | MSICAP3F1 | R/W | R/W | H'105C | H'xxxx xxxx | 32 | √ |
| MSI capability register 4 function 1 | MSICAP4F1 | R/W | R/W | H'1060 | H'xxxx xxxx | 32 | √ |
| MSI capability register 5 function 1 | MSICAP5F1 | R | R | H'1064 | H'xxxx xxxx | 32 | √ |
| PCI Express capability register 0 function1<br>PCI Express capability list register<br>PCI Express capabilities register | EXPCAP0F1 | R/W | R | H'1070 | | 32 | √ |
| PCI Express capability register 1 function1<br>Device capabilities register | EXPCAP1F1 | R/W | R | H'1074 | | 32 | √ |
| PCI Express capability register 2 function1<br>Device control register<br>Device status register | EXPCAP2F1 | R/W | R/W | H'1078 | | 32 | √ |
| PCI Express capability register 3 function1<br>Link capabilities register | EXPCAP3F1 | R/W | R | H'107C | | 32 | √ |
| PCI Express capability register 4 function1<br>Link control register<br>Link status register | EXPCAP4F1 | R/W | R/W | H'1080 | | 32 | √ |
| PCI Express capability register 5 function1<br>Slot capabilities register | EXPCAP5F1 | Header TYPE00: R<br>Header TYPE01: R | Header TYPE00: R<br>Header TYPE01: — | H'1084 | Header TYPE00: H'0000 0000 | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| | | | | | Header TYPE01: H'0000 0000 | | |
| PCI Express capability register 6 function1<br>Slot control register<br>Slot status register | EXPCAP6F1 | Header TYPE00: R<br>Header TYPE01: R | Header TYPE00: R<br>Header TYPE01: — | H'1088 | Header TYPE00: H'0000 0000<br>Header TYPE01: H'0040 0000 | 32 | √ |
| PCI Express capability register 7 function1<br>Root control register<br>Root capabilities register | EXPCAP7F1 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'108C | Header TYPE00: H'0000 0000<br>Header TYPE01: H'0000 0000 | 32 | √ |
| PCI Express capability register 8 function1<br>Root status register | EXPCAP8F1 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: R<br>Header TYPE01: — | H'1090 | Header TYPE00: H'0000 0000<br>Header TYPE01: H'0000 0000 | 32 | √ |
| PCI Express capability register 9 function1<br>Device capabilities 2 register | EXPCAP9F1 | R/W | R | H'1094 | | 32 | √ |
| PCI Express capability register 10 function1<br>Device control 2 register<br>Device status 2 register | EXPCAP10F1 | R/W | R/W | H'1098 | | 32 | √ |
| PCI Express capability register 11 function1<br>Link capabilities 2 register | EXPCAP11F1 | R | R | H'109C | | 32 | √ |
| PCI Express capability register 12 function1<br>Link control 2 register<br>Link status 2 register | EXPCAP12F1 | R/W | R/W | H'10A0 | | 32 | √ |
| Advanced Error Reporting Capability Register 0 function 1 | AERCAP0F1 | R/W | R | H'1100 | | 32 | √ |
| Advanced Error Reporting Capability Register 1 function 1 | AERCAP1F1 | R/W | R/W | H'1104 | | 32 | √ |
| Advanced Error Reporting Capability Register 2 function 1 | AERCAP2F1 | R/W | R/W | H'1108 | | 32 | √ |
| Advanced Error Reporting Capability Register 3 function 1 | AERCAP3F1 | R/W | R/W | H'110C | | 32 | √ |
| Advanced Error Reporting Capability Register 4 function 1 | AERCAP4F1 | R/W | R/W | H'1110 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Advanced Error Reporting Capability Register 5 function 1 | AERCAP5F1 | R/W | R/W | H'1114 | | 32 | √ |
| Advanced Error Reporting Capability Register 6 function 1 | AERCAP6F1 | R/W | R/W | H'1118 | | 32 | √ |
| Advanced Error Reporting Capability Register 7 function 1 | AERCAP7F1 | R | R | H'111C | | 32 | √ |
| Advanced Error Reporting Capability Register 8 function 1 | AERCAP8F1 | R | R | H'1120 | | 32 | √ |
| Advanced Error Reporting Capability Register 9 function 1 | AERCAP9F1 | R | R | H'1124 | | 32 | √ |
| Advanced Error Reporting Capability Register 10 function 1 | AERCAP10F1 | R | R | H'1128 | | 32 | √ |
| Advanced Error Reporting Capability Register 11 function 1 | AERCAP11F1 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: R/W | H'112C | | 32 | √ |
| Advanced Error Reporting Capability Register 12 function 1 | AERCAP12F1 | Header TYPE00: R<br>Header TYPE01: R/W | Header TYPE00: —<br>Header TYPE01: R/W | H'1130 | | 32 | √ |
| Advanced Error Reporting Capability Register 13 function 1 | AERCAP13F1 | Header TYPE00: R<br>Header TYPE01: R | Header TYPE00: —<br>Header TYPE01: R | H'1134 | | 32 | √ |
| Device Serial Number Capability Register 0 function 1 | SERNUMCAP0F1 | R/W | R | H'1148 | | 32 | √ |
| Device Serial Number Capability Register 1 function 1 | SERNUMCAP1F1 | R | R | H'114C | | 32 | √ |
| Device Serial Number Capability Register 2 function 1 | SERNUMCAP2F1 | R | R | H'1150 | | 32 | √ |
| Vendor-Specific Extended Capability Register 0 function 1 | VSECAP0F1 | R/W | R | H'11CC | | 32 | √ |
| Vendor-Specific Extended Capability Register 1 function 1 | VSECAP1F1 | R | R | H'11D0 | | 32 | √ |
| Vendor-Specific Extended Capability Register 2 function 1 | VSECAP2F1 | R/W | R/W | H'11D4 | | 32 | √ |
| Vendor-Specific Extended Capability Register 3 function 1 | VSECAP3F1 | R | R | H'11D8 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Vendor-Specific Extended Capability Register 4 function 1 | VSECAP4F1 | R/W | R/W | H'11DC | | 32 | √ |
| Vendor-Specific Extended Capability Register 5 function 1 | VSECAP5F1 | R | R | H'11E0 | | 32 | √ |
| Vendor-Specific Extended Capability Register 6 function 1 | VSECAP6F1 | R/W | R/W | H'11FC | | 32 | √ |
| Vendor-Specific Extended Capability Register 7 function 1 | VSECAP7F1 | R/W | R/W | H'1200 | | 32 | √ |
| Vendor-Specific Extended Capability Register 8 function 1 | VSECAP8F1 | R/W | R/W | H'1204 | | 32 | √ |
| Vendor-Specific Extended Capability Register 9 function 1 | VSECAP9F1 | R/W | R/W | H'1208 | | 32 | √ |
| Vendor-Specific Extended Capability Register 10 function 1 | VSECAP10F1 | R/W | R/W | H'120C | | 32 | √ |
| Vendor-Specific Extended Capability Register 11 function 1 | VSECAP11F1 | R/W | R/W | H'1210 | | 32 | √ |
| Vendor-Specific Extended Capability Register 12 function 1 | VSECAP12F1 | R/W | R/W | H'1214 | | 32 | √ |
| Vendor-Specific Extended Capability Register 13 function 1 | VSECAP13F1 | R/W | R/W | H'1218 | | 32 | √ |
| Vendor-Specific Extended Capability Register 14 function 1 | VSECAP14F1 | R/W | R/W | H'121C | | 32 | √ |
| Vendor-Specific Extended Capability Register 15 function 1 | VSECAP15F1 | R/W | R/W | H'1220 | | 32 | √ |
| Vendor-Specific Extended Capability Register 16 function 1 | VSECAP16F1 | R/W | R/W | H'1224 | | 32 | √ |
| Vendor-Specific Extended Capability Register 17 function 1 | VSECAP17F1 | R/W | R/W | H'1228 | | 32 | √ |
| Vendor-Specific Extended Capability Register 18 function 1 | VSECAP18F1 | R/W | R/W | H'122C | | 32 | √ |
| Vendor-Specific Extended Capability Register 19 function 1 | VSECAP19F1 | R/W | R/W | H'1230 | | 32 | √ |
| Vendor-Specific Extended Capability Register 20 function 1 | VSECAP20F1 | R/W | R/W | H'1234 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Vendor-Specific Extended Capability Register 21 function 1 | VSECAP21F1 | R/W | R/W | H'1238 | | 32 | √ |
| Vendor-Specific Extended Capability Register 22 function 1 | VSECAP22F1 | R/W | R/W | H'123C | | 32 | √ |
| Vendor-Specific Extended Capability Register 23 function 1 | VSECAP23F1 | R/W | R/W | H'1240 | | 32 | √ |
| Vendor-Specific Extended Capability Register 24 function 1 | VSECAP24F1 | R/W | R/W | H'1244 | | 32 | √ |
| Vendor-Specific Extended Capability Register 25 function 1 | VSECAP25F1 | R/W | R/W | H'1248 | | 32 | √ |
| Vendor-Specific Extended Capability Register 26 function 1 | VSECAP26F1 | R/W | R/W | H'124C | | 32 | √ |
| Vendor-Specific Extended Capability Register 27 function 1 | VSECAP27F1 | R/W | R/W | H'1250 | | 32 | √ |
| Vendor-Specific Extended Capability Register 28 function 1 | VSECAP28F1 | R/W | R/W | H'1254 | | 32 | √ |
| Vendor-Specific Extended Capability Register 29 function 1 | VSECAP29F1 | R/W | R/W | H'1258 | | 32 | √ |
| Vendor-Specific Extended Capability Register 30 function 1 | VSECAP30F1 | R/W | R/W | H'126C | | 32 | √ |
| Vendor-Specific Extended Capability Register 31 function 1 | VSECAP31F1 | R/W | R/W | H'1270 | | 32 | √ |
| Vendor-Specific Extended Capability Register 32 function 1 | VSECAP32F1 | R/W | R/W | H'127C | | 32 | √ |
| Vendor-Specific Extended Capability Register 33 function 1 | VSECAP33F1 | R/W | R/W | H'1280 | | 32 | √ |
| Vendor-Specific Extended Capability Register 34 function 1 | VSECAP34F1 | R/W | R/W | H'1284 | | 32 | √ |
| Vendor-Specific Extended Capability Register 35 function 1 | VSECAP35F1 | R/W | R/W | H'1288 | | 32 | √ |
| Vendor-Specific Extended Capability Register 36 function 1 | VSECAP36F1 | R/W | R/W | H'128C | | 32 | √ |
| Vendor-Specific Extended Capability Register 37 function 1 | VSECAP37F1 | R/W | R/W | H'1290 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Vendor-Specific Extended Capability Register 38 function 1 | VSECAP38F1 | R/W | R/W | H'129C | | 32 | √ |
| Vendor-Specific Extended Capability Register 39 function 1 | VSECAP39F1 | R/W | R/W | H'12A0 | | 32 | √ |
| Vendor-Specific Extended Capability Register 40 function 1 | VSECAP40F1 | R/W | R/W | H'12A4 | | 32 | √ |
| Vendor-Specific Extended Capability Register 41 function 1 | VSECAP41F1 | R | R | H'12AC | | 32 | √ |
| Vendor-Specific Extended Capability Register 42 function 1 | VSECAP42F1 | R | R | H'12B0 | | 32 | √ |
| Vendor-Specific Extended Capability Register 43 function 1 | VSECAP43F1 | R | R | H'12B4 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 0 function 1 | VSE2CAP0F1 | R | R | H'12CC | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 1 function 1 | VSE2CAP1F1 | R | R | H'12D0 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 2 function 1 | VSE2CAP2F1 | R/W | R/W | H'12D4 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 3 function 1 | VSE2CAP3F1 | R/W | R/W | H'12D8 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 4 function 1 | VSE2CAP4F1 | R | R | H'12DC | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 5 function 1 | VSE2CAP5F1 | R/W | R/W | H'12E0 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 6 function 1 | VSE2CAP6F1 | R | R | H'12E4 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 7 function 1 | VSE2CAP7F1 | R/W | R/W | H'12E8 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 8 function 1 | VSE2CAP8F1 | R | R | H'12EC | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 9 function 1 | VSE2CAP9F1 | R | R | H'12F0 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 10 function 1 | VSE2CAP10F1 | R/W | R/W | H'12F4 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Vendor-Specific Extended 2 Capability Register 11 function 1 | VSE2CAP11F1 | R/W | R/W | H'12F8 | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 12 function 1 | VSE2CAP12F1 | R | R | H'12FC | | 32 | √ |
| Vendor-Specific Extended 2 Capability Register 13 function 1 | VSE2CAP13F1 | R | R | H'1300 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 0 function 1 | PTMRQCAP0F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'131C | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 1 function 1 | PTMRQCAP1F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'1320 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 2 function 1 | PTMRQCAP2F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'1324 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 3 function 1 | PTMRQCAP3F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1328 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 4 function 1 | PTMRQCAP4F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'132C | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 5 function 1 | PTMRQCAP5F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1330 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 6 function 1 | PTMRQCAP6F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1334 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 7 function 1 | PTMRQCAP7F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1338 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 8 function 1 | PTMRQCAP8F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'133C | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| PTM Requestor Capability Structure (VSEC) Register 9 function 1 | PTMRQCAP9F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1340 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 10 function 1 | PTMRQCAP10F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1344 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 11 function 1 | PTMRQCAP11F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1348 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 12 function 1 | PTMRQCAP12F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'134C | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 13 function 1 | PTMRQCAP13F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1350 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 14 function 1 | PTMRQCAP14F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1354 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 15 function 1 | PTMRQCAP15F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1358 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 16 function 1 | PTMRQCAP16F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'135C | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 17 function 1 | PTMRQCAP17F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1360 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 18 function 1 | PTMRQCAP18F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'1364 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 19 function 1 | PTMRQCAP19F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'1368 | | 32 | √ |
| PTM Requestor Capability Structure (VSEC) Register 20 function 1 | PTMRQCAP20F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'136C | | 32 | √ |
| Resizable BAR Capability Header function 1 | RESBAR_CAP_HDR _F1 | Header TYPE00: R | Header TYPE00: R | H'139C | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| | | Header TYPE01: — | Header TYPE01: — | | | | |
| Resizable BAR0 Capability Register function 1 | RESBAR_CAP_REG_0_REG_F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'13A0 | | 32 | √ |
| Resizable BAR0 Control Register function 1 | RESBAR_CTRL_REG_0_REG_F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'13A4 | | 32 | √ |
| Resizable BAR2 Capability Register function 1 | RESBAR_CAP_REG_2_REG_F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'13A8 | | 32 | √ |
| Resizable BAR2 Control Register function 1 | RESBAR_CTRL_REG_2_REG_F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'13AC | | 32 | √ |
| Resizable BAR5 Capability Register function 1 | RESBAR_CAP_REG_5_REG_F1 | Header TYPE00: R<br>Header TYPE01: — | Header TYPE00: R<br>Header TYPE01: — | H'13B0 | | 32 | √ |
| Resizable BAR5 Control Register function 1 | RESBAR_CTRL_REG_5_REG_F1 | Header TYPE00: R/W<br>Header TYPE01: — | Header TYPE00: R/W<br>Header TYPE01: — | H'13B4 | | 32 | √ |
| PCIE*m* PCIE Shadow Register | | | | | | | √ |
| BAR0 MASK Register function 0 | BAR0MASKF0 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2010 | | 32 | √ |
| BAR1 MASK Register function 0 | BAR1MASKF0 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2014 | | 32 | √ |
| BAR2 MASK Register function 0 | BAR2MASKF0 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2018 | | 32 | √ |
| BAR3 MASK Register function 0 | BAR3MASKF0 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'201C | | 32 | √ |
| BAR4 MASK Register function 0 | BAR4MASKF0 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2020 | | 32 | √ |
| BAR5 MASK Register function 0 | BAR5MASKF0 | Header TYPE00: W | Header TYPE00: — | H'2024 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| | | Header TYPE01: — | Header TYPE01: — | | | | |
| EXP ROM MASK Register function 0 | EROMMASKF0 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2030 | | 32 | √ |
| BAR0 MASK Register function 1 | BAR0MASKF1 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2810 | | 32 | √ |
| BAR1 MASK Register function 1 | BAR1MASKF1 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2814 | | 32 | √ |
| BAR2 MASK Register function 1 | BAR2MASKF1 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2818 | | 32 | √ |
| BAR3 MASK Register function 1 | BAR3MASKF1 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'281C | | 32 | √ |
| BAR4 MASK Register function 1 | BAR4MASKF1 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2820 | | 32 | √ |
| BAR5 MASK Register function 1 | BAR5MASKF1 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2824 | | 32 | √ |
| EXP ROM MASK Register function 1 | EROMMASKF1 | Header TYPE00: W<br>Header TYPE01: — | Header TYPE00: —<br>Header TYPE01: — | H'2830 | | 32 | √ |
| PCIE*m* PCIE Control Register (Common for All functions) | | | | | | | √ |
| ATU Capabilities Registers [0+i*18] (*i* = 0 to 15) | ATUCAP[0+*i**18] | R/W | — | H'3000 + *i**H'200 + H'0 | | 32 | √ |
| ATU Capabilities Registers [1+i*18] (*i* = 0 to 15) | ATUCAP[1+*i**18] | R/W | — | H'3000 + *i**H'200 + H'4 | | 32 | √ |
| ATU Capabilities Registers [2+i*18] (*i* = 0 to 15) | ATUCAP[2+*i**18] | R/W | — | H'3000 + *i**H'200 + H'8 | | 32 | √ |
| ATU Capabilities Registers [3+i*18] (*i* = 0 to 15) | ATUCAP[3+*i**18] | R/W | — | H'3000 + *i**H'200 + H'C | | 32 | √ |
| ATU Capabilities Registers [4+i*18] (*i* = 0 to 15) | ATUCAP[4+*i**18] | R/W | — | H'3000 + *i**H'200 + H'10 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| ATU Capabilities Registers [5+i*18] (*i* = 0 to 15) | ATUCAP[5+*i**18] | R/W | — | H'3000 + *i**H'200 + H'14 | | 32 | √ |
| ATU Capabilities Registers [6+i*18] (*i* = 0 to 15) | ATUCAP[6+*i**18] | R/W | — | H'3000 + *i**H'200 + H'18 | | 32 | √ |
| ATU Capabilities Registers [7+i*18] (*i* = 0 to 15) | ATUCAP[7+*i**18] | R | — | H'3000 + *i**H'200 + H'1C | | 32 | √ |
| ATU Capabilities Registers [8+i*18] (*i* = 0 to 15) | ATUCAP[8+*i**18] | R | — | H'3000 + *i**H'200 + H'20 | | 32 | √ |
| ATU Capabilities Registers [9+i*18] (*i* = 0 to 15) | ATUCAP[9+*i**18] | R/W | — | H'3000 + *i**H'200 + H'100 | | 32 | √ |
| ATU Capabilities Registers [10+i*18] (*i* = 0 to 15) | ATUCAP[10+*i**18] | R/W | — | H'3000 + *i**H'200 + H'104 | | 32 | √ |
| ATU Capabilities Registers [11+i*18] (*i* = 0 to 15) | ATUCAP[11+*i**18] | R/W | — | H'3000 + *i**H'200 + H'108 | | 32 | √ |
| ATU Capabilities Registers [12+i*18] (*i* = 0 to 15) | ATUCAP[12+*i**18] | R/W | — | H'3000 + *i**H'200 + H'10C | | 32 | √ |
| ATU Capabilities Registers [13+i*18] (*i* = 0 to 15) | ATUCAP[13+*i**18] | R/W | — | H'3000 + *i**H'200 + H'110 | | 32 | √ |
| ATU Capabilities Registers [14+i*18] (*i* = 0 to 15) | ATUCAP[14+*i**18] | R/W | — | H'3000 + *i**H'200 + H'114 | | 32 | √ |
| ATU Capabilities Registers [15+i*18] (*i* = 0 to 15) | ATUCAP[15+*i**18] | R/W | — | H'3000 + *i**H'200 + H'118 | | 32 | √ |
| ATU Capabilities Registers [16+i*18] (*i* = 0 to 15) | ATUCAP[16+*i**18] | R | — | H'3000 + *i**H'200 + H'11C | | 32 | √ |
| ATU Capabilities Registers [17+i*18] (*i* = 0 to 15) | ATUCAP[17+*i**18] | R | — | H'3000 + *i**H'200 + H'120 | | 32 | √ |
| DMA Capabilities Registers 0 | DMACAP0 | R/W | — | H'5000 | | 32 | √ |
| DMA Capabilities Registers 1 | DMACAP1 | R/W | — | H'5008 | | 32 | √ |
| DMA Capabilities Registers 2 | DMACAP2 | R/W | — | H'500C | | 32 | √ |
| DMA Capabilities Registers 3 | DMACAP3 | R/W | — | H'5010 | | 32 | √ |
| DMA Capabilities Registers 4 | DMACAP4 | R/W | — | H'5018 | | 32 | √ |
| DMA Capabilities Registers 5 | DMACAP5 | R/W | — | H'501C | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| DMA Capabilities Registers 6 | DMACAP6 | R/W | — | H'502C | | 32 | √ |
| DMA Capabilities Registers 7 | DMACAP7 | R/W | — | H'5030 | | 32 | √ |
| DMA Capabilities Registers 8 | DMACAP8 | R/W | — | H'5038 | | 32 | √ |
| DMA Capabilities Registers 9 | DMACAP9 | R/W | — | H'503C | | 32 | √ |
| DMA Capabilities Registers 10 | DMACAP10 | R/W | — | H'504C | | 32 | √ |
| DMA Capabilities Registers 11 | DMACAP11 | R/W | — | H'5054 | | 32 | √ |
| DMA Capabilities Registers 12 | DMACAP12 | R | — | H'5058 | | 32 | √ |
| DMA Capabilities Registers 13 | DMACAP13 | R | — | H'505C | | 32 | √ |
| DMA Capabilities Registers 14 | DMACAP14 | R/W | — | H'5060 | | 32 | √ |
| DMA Capabilities Registers 15 | DMACAP15 | R/W | — | H'5064 | | 32 | √ |
| DMA Capabilities Registers 16 | DMACAP16 | R/W | — | H'5068 | | 32 | √ |
| DMA Capabilities Registers 17 | DMACAP17 | R/W | — | H'506C | | 32 | √ |
| DMA Capabilities Registers 18 | DMACAP18 | R/W | — | H'5070 | | 32 | √ |
| DMA Capabilities Registers 19 | DMACAP19 | R/W | — | H'5074 | | 32 | √ |
| DMA Capabilities Registers 20 | DMACAP20 | R/W | — | H'5078 | | 32 | √ |
| DMA Capabilities Registers 21 | DMACAP21 | R/W | — | H'507C | | 32 | √ |
| DMA Capabilities Registers 22 | DMACAP22 | R/W | — | H'5090 | | 32 | √ |
| DMA Capabilities Registers 23 | DMACAP23 | R/W | — | H'50A0 | | 32 | √ |
| DMA Capabilities Registers 24 | DMACAP24 | R/W | — | H'50A8 | | 32 | √ |
| DMA Capabilities Registers 25 | DMACAP25 | R/W | — | H'50AC | | 32 | √ |
| DMA Capabilities Registers 26 | DMACAP26 | R | — | H'50B4 | | 32 | √ |
| DMA Capabilities Registers 27 | DMACAP27 | R | — | H'50B8 | | 32 | √ |
| DMA Capabilities Registers 28 | DMACAP28 | R/W | — | H'50C4 | | 32 | √ |
| DMA Capabilities Registers 29 | DMACAP29 | R/W | — | H'50CC | | 32 | √ |
| DMA Capabilities Registers 30 | DMACAP30 | R/W | — | H'50D0 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| DMA Capabilities Registers 31 | DMACAP31 | R/W | — | H'50D4 | | 32 | √ |
| DMA Capabilities Registers 32 | DMACAP32 | R/W | — | H'50D8 | | 32 | √ |
| DMA Capabilities Registers 33 | DMACAP33 | R/W | — | H'50DC | | 32 | √ |
| DMA Capabilities Registers 34 | DMACAP34 | R/W | — | H'50E0 | | 32 | √ |
| DMA Capabilities Registers 35 | DMACAP35 | R/W | — | H'50E4 | | 32 | √ |
| DMA Capabilities Registers 36 | DMACAP36 | R/W | — | H'50E8 | | 32 | √ |
| DMA Capabilities Registers [37+*i**16] (*i* = 0 to 7) | DMACAP[37+*i**16] | R/W | — | H'5200 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [38+*i**16] (*i* = 0 to 7) | DMACAP[38+*i**16] | R/W | — | H'5208 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [39+*i**16] (*i* = 0 to 7) | DMACAP[39+*i**16] | R/W | — | H'520C + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [40+*i**16] (*i* = 0 to 7) | DMACAP[40+*i**16] | R/W | — | H'5210 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [41+*i**16] (*i* = 0 to 7) | DMACAP[41+*i**16] | R/W | — | H'5214 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [42+*i**16] (*i* = 0 to 7) | DMACAP[42+*i**16] | R/W | — | H'5218 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [43+*i**16] (*i* = 0 to 7) | DMACAP[43+*i**16] | R/W | — | H'521C + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [44+*i**16] (*i* = 0 to 7) | DMACAP[44+*i**16] | R/W | — | H'5220 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [45+*i**16] (*i* = 0 to 7) | DMACAP[45+*i**16] | R/W | — | H'5300 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [46+*i**16] (*i* = 0 to 7) | DMACAP[46+*i**16] | R/W | — | H'5308 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [47+*i**16] (*i* = 0 to 7) | DMACAP[47+*i**16] | R/W | — | H'530C + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [48+*i**16] (*i* = 0 to 7) | DMACAP[48+*i**16] | R/W | — | H'5310 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [49+*i**16] (*i* = 0 to 7) | DMACAP[49+*i**16] | R/W | — | H'5314 + *i**H'200 | | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| DMA Capabilities Registers [50+*i**16] (*i* = 0 to 7) | DMACAP[50+*i**16] | R/W | — | H'5318 + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [51+*i**16] (*i* = 0 to 7) | DMACAP[51+*i**166] | R/W | — | H'531C + *i**H'200 | | 32 | √ |
| DMA Capabilities Registers [52+*i**16] (*i* = 0 to 7) | DMACAP[52+*i**16] | R/W | — | H'5320 + *i**H'200 | | 32 | √ |
| Application registers PCIEC | | | | | | | √ |
| PCIe Mode Setting Register 0 | PCIEMSR0 | RW | — | H'6200 | 0x00000000 | 32 | √ |
| PCIe Mode Setting Register 1 | PCIEMSR1 | RW | — | H'6204 | 0x00000000 | 32 | √ |
| PCIe Mode Setting Register 2 | PCIEMSR2 | RW | — | H'6208 | 0x00000000 | 32 | √ |
| PCIe Mode Setting Register 3 | PCIEMSR3 | RW | — | H'620C | 0x00000000 | 32 | √ |
| PCIe Reset Control Register 0 | PCIERSTCTRL0 | RW | — | H'6210 | 0x00010000 | 32 | √ |
| PCIe Reset Control Register 1 | PCIERSTCTRL1 | RW | — | H'6214 | 0x00010000 | 32 | √ |
| PCIe Reset Control Register 2 | PCIERSTCTRL2 | RW | — | H'6218 | 0x00000000 | 32 | √ |
| PCIe Reset Control Register 3 | PCIERSTCTRL3 | RW | — | H'621C | 0x00000000 | 32 | √ |
| PCIe Reset Status Register | PCIERSTSTS | R | — | H'6220 | 0x00000000 | 32 | √ |
| MSI Control Register 0 | MSICTRL0 | RW | — | H'6224 | 0x00000000 | 32 | √ |
| MSI Control Register 1 | MSICTRL1 | RW | — | H'6228 | 0x00000000 | 32 | √ |
| MSI Control Register 2 | MSICTRL2 | RW | — | H'622C | 0x00000000 | 32 | √ |
| MSI Control Register 3 | MSICTRL3 | RW | — | H'6230 | 0x00000000 | 32 | √ |
| MSI Control Register 4 | MSICTRL4 | R | — | H'6234 | 0x00000000 | 32 | √ |
| MSI Control Register 5 | MSICTRL5 | R | — | H'6238 | 0x00000000 | 32 | √ |
| PCIe Message Transmission | PCIEMSGTX | RW | — | H'623C | 0x00000000 | 32 | √ |
| PCIe Message Reception 0 | PCIEMSGRX0 | R | — | H'6240 | 0x00000000 | 32 | √ |
| PCIe Message Reception 1 | PCIEMSGRX1 | R | — | H'6244 | 0x00000000 | 32 | √ |
| PCIe Message Reception | PCIEMSGRX2 | R | — | H'6248 | 0x00000000 | 32 | √ |
| PCIe Message Reception Status | PCIEMSGRCPSTS | R | — | H'624C | 0x00000000 | 32 | √ |
| PCIe LTR Message Control 0 | PCIELTRMSGCTRL0 | RW | — | H'6250 | 0x00000000 | 32 | √ |
| PCIe LTR Message Control 1 | PCIELTRMSGCTRL1 | RW | — | H'6254 | 0x00000000 | 32 | √ |
| PCIe PTM Control | PCIEPTMCTRL | RW | — | H'6258 | 0x00000000 | 32 | √ |
| PCIe PTM Status 0 | PCIEPTMSTS0 | R | — | H'625C | 0x00000000 | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| PCIe PTM Status 1 | PCIEPTMSTS1 | R | — | H'6260 | 0x00000000 | 32 | √ |
| PCIe PTM Status 2 | PCIEPTMSTS2 | R | — | H'6264 | 0x00000000 | 32 | √ |
| PCIe PTM Status 3 | PCIEPTMSTS3 | R | — | H'6268 | 0x00000000 | 32 | √ |
| PCIe PTM Status 4 | PCIEPTMSTS4 | R | — | H'626C | 0x00000000 | 32 | √ |
| PCIe Power Management Control | PCIEPWRMNGCTRL | RW | — | H'6270 | 0x00000010 | 32 | √ |
| PCIe Power Management Status 0 | PCIEPWRMNGSTS0 | R | — | H'6274 | 0x00000000 | 32 | √ |
| PCIe Power Management Status 1 | PCIEPWRMNGSTS1 | R | — | H'6278 | 0x00000000 | 32 | √ |
| PCIe Error Status 0 | PCIEERRSTS0 | R | — | H'627C | 0x00000000 | 32 | √ |
| PCIe Interrupt Status 0 | PCIEINTSTS0 | R | — | H'6284 | 0x00000000 | 32 | √ |
| PCIe DMA Interrupt Status | PCIEDMAINTSTS | R | — | H'628C | 0x00000000 | 32 | √ |
| PCIe INTx Interrupt Status | PCIEINTXSTS | R | — | H'6290 | 0x00000000 | 32 | √ |
| PCIe Interrupt Status 1 | PCIEINTSTS1 | R | — | H'6294 | 0x00000000 | 32 | √ |
| PCIe RAS D.E.S Control | PCIERASDESCTRL | RW | — | H'6298 | 0x00000000 | 32 | √ |
| PCIe Receive CPL Timeout Status | PCIERCVCPLTOSTS | R | — | H'629C | 0x00000000 | 32 | √ |
| PCIe Transmit Control 0 | PCIETXCNTRL0 | RW | — | H'62A0 | 0x00000000 | 32 | √ |
| PCIe Transmit Control 1 | PCIETXCNTRL1 | RW | — | H'62A4 | 0x00000000 | 32 | √ |
| PCIe Target Completion LUT 0 | PCIETRGTCMPLUT0 | R | — | H'62A8 | 0x00000000 | 32 | √ |
| PCIe Target Completion LUT 1 | PCIETRGTCMPLUT1 | R | — | H'62AC | 0x00000000 | 32 | √ |
| PCIe Mode Information | PCIEMODEINFO | R | — | H'62B4 | 0x00000000 | 32 | √ |
| PCIe Configuration Information 0 | PCIECNFGINFO0 | R | — | H'62B8 | 0x00000000 | 32 | √ |
| PCIe Configuration Information 1 | PCIECNFGINFO1 | R | — | H'62BC | 0x00000000 | 32 | √ |
| PCIe Margining Control | PCIEMARGINGCTRL | RW | — | H'62C0 | 0x00000000 | 32 | √ |
| PCIe Reset Status Enable Register | PCIERSTSTSEN | RW | — | H'6500 | 0x00000000 | 32 | √ |
| PCIe Message Reception Status Enable | PCIEMSGRCPSTSEN | RW | — | H'6504 | 0x00000000 | 32 | √ |
| PCIe Power Management Status 0 Enable | PCIEPWRMNGSTS0EN | RW | — | H'6508 | 0x00000000 | 32 | √ |
| PCIe Error Status 0 Enable | PCIEERRSTS0EN | RW | — | H'650C | 0x00000000 | 32 | √ |
| PCIe Interrupt Status 0 Enable | PCIEINTSTS0EN | RW | — | H'6510 | 0x00000000 | 32 | √ |
| PCIe DMA Interrupt Status Enable | PCIEDMAINTSTSEN | RW | — | H'6514 | 0x00000000 | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| PCIe INTx Interrupt Status Enable | PCIEINTXSTSEN | RW | — | H'6518 | 0x00000000 | 32 | √ |
| PCIe Interrupt Status 1 Enable | PCIEINTSTS1EN | RW | — | H'651C | 0x00000000 | 32 | √ |
| PCIe Target Completion LUT 1 Enable | PCIETRGTCMPLUT1EN | RW | — | H'6520 | 0x00000000 | 32 | √ |
| PCIe Reset Status Clear Register | PCIERSTSTSCLR | RW | — | H'6530 | 0x00000000 | 32 | √ |
| PCIe Message Reception Status Clear | PCIEMSGRCPSTSCLR | RW | — | H'6534 | 0x00000000 | 32 | √ |
| PCIe Power Management Status 0 Clear | PCIEPWRMNGSTS0CLR | RW | — | H'6538 | 0x00000000 | 32 | √ |
| PCIe Error Status 0 Clear | PCIEERRSTS0CLR | RW | — | H'653C | 0x00000000 | 32 | √ |
| PCIe Interrupt Status 0 Clear | PCIEINTSTS0CLR | RW | — | H'6540 | 0x00000000 | 32 | √ |
| PCIe INTx Interrupt Status Clear | PCIEINTXSTSCLR | RW | — | H'6548 | 0x00000000 | 32 | √ |
| PCIe Interrupt Status 1 Clear | PCIEINTSTS1CLR | RW | — | H'654C | 0x00000000 | 32 | √ |
| PCIe Target Completion LUT 1 Clear | PCIETRGTCMPLUT1CLR | RW | — | H'6550 | 0x00000000 | 32 | √ |
| PCIe PTM Status 0 Clear | PCIEPTMSTS0CLR | RW | — | H'6554 | 0x00000000 | 32 | √ |
| PCIe Message Transmission Clear | PCIEMSGTXCLR | RW | — | H'6564 | 0x00000000 | 32 | √ |
| PIPE current link operating rate | PMRATE | R | — | H'6600 | 0x00000000 | 32 | √ |
| SII: BDF Signals | SIIBDF | RW | — | H'6700 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 0 | CNVID0 | RW | — | H'6900 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 1 | CNVID1 | RW | — | H'6904 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 2 | CNVID2 | RW | — | H'6908 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 3 | CNVID3 | RW | — | H'690C | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 4 | CNVID4 | RW | — | H'6910 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 5 | CNVID5 | RW | — | H'6914 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 6 | CNVID6 | RW | — | H'6918 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 7 | CNVID7 | RW | — | H'691C | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 8 | CNVID8 | RW | — | H'6920 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 9 | CNVID9 | RW | — | H'6924 | 0x00000000 | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Convert Requester ID to OSID Register 10 | CNVID10 | RW | — | H'6928 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 11 | CNVID11 | RW | — | H'692C | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 12 | CNVID12 | RW | — | H'6930 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 13 | CNVID13 | RW | — | H'6934 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 14 | CNVID14 | RW | — | H'6938 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Register 15 | CNVID15 | RW | — | H'693C | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 0 | CNVIDMSK0 | RW | — | H'6980 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 1 | CNVIDMSK1 | RW | — | H'6984 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 2 | CNVIDMSK2 | RW | — | H'6988 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 3 | CNVIDMSK3 | RW | — | H'698C | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 4 | CNVIDMSK4 | RW | — | H'6990 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 5 | CNVIDMSK5 | RW | — | H'6994 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 6 | CNVIDMSK6 | RW | — | H'6998 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 7 | CNVIDMSK7 | RW | — | H'699C | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 8 | CNVIDMSK8 | RW | — | H'69A0 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 9 | CNVIDMSK9 | RW | — | H'69A4 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 10 | CNVIDMSK10 | RW | — | H'69A8 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 11 | CNVIDMSK11 | RW | — | H'69AC | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 12 | CNVIDMSK12 | RW | — | H'69B0 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 13 | CNVIDMSK13 | RW | — | H'69B4 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 14 | CNVIDMSK14 | RW | — | H'69B8 | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Mask Register 15 | CNVIDMSK15 | RW | — | H'69BC | 0x00000000 | 32 | √ |
| Convert Requester ID to OSID Control Register | CNVOSIDCTRL | RW | — | H'6A00 | 0x00000000 | 32 | √ |
| INTC address | AXIINTCADDR | RW | — | H'6C00 | 0x00000000 | 32 | √ |
| INTC control & mask | AXIINTCCONT | RW | — | H'6C04 | 0x00000000 | 32 | √ |
| Reference Clock Control Signals PHY0 Register | REFCLKCTRLP0 | RW | — | H'70B8 | 0x00000004 | 32 | √ |

| Register Name | Abbreviation | APB*1 R/W | PCI*1 R/W | Address Offset | Initial Value | Access Size | R-Car V4M |
|---|---|---|---|---|---|---|---|
| Reference Clock Control Signals PHY1 Register | REFCLKCTRLP1 | RW | — | H'72B8 | 0x00000000 | 32 | √ |

Notes: 1. APB: APB bus (internal bus)
PCI: PCI local bus
'—' in R/W column: Access prohibited.
For details of R/W attribute for each bit, see section 104.2, Register Description.
2. Depends on the setting of the local address mask register 0.
3. R/W changes by the state. RootPort:R/W.EndPoint:R.
4. Depends on the setting of the local address mask register 1.
5. Depends on the setting of the local address mask register 2.
6. Depends on the setting of the local address mask register 3.
7. Depends on the setting of the local address mask register 4.
8. Depends on the setting of the local address mask register 5.
9. *m* is channel number. (*m* = 0)

Do not write to any addresses other than listed above. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed above are undefined.

### 104.1.5 Address Map

Table 104.5 shows the PCIe Controller Address Map.

**Table 104.5 PCIe Controller Address Map**

| Address | Space name | Note |
|---|---|---|
| H'3000_0000 – H'3FFF_FFFF | PCIE0 Memory | |
| H'E65D_0000 – H' E65D_0FFF | PCIE0 PCI Configuration for Function 0 | Common address in Root Complex mode and Endpoint mode |
| H'E65D_1000 – H' E65D_1FFF | PCIE0 PCI Configuration for Function 1 | Endpoint mode only |
| H'E65D_2000 – H' E65D_27FF | PCIE0 PCI shadow register for Function 0 | Common address in Root Complex mode and Endpoint mode |
| H'E65D_2800 – H' E65D_2FFF | PCIE0 PCI shadow register for Function 1 | Endpoint mode only |
| H'E65D_3000 – H' E65D_7FFF | PCIE0 PCIe Control Register | Common for All functions |
| H'FE00_0000 – H' FE3F_FFFF | PCIE0 Configuration for Remote | This area can be used for Memory |

### 104.1.6 Connected Module

**Table 104.6** Connected modules

| Module name | Connected module name | Function of connected module |
|---|---|---|
| PCIEC | AP-System Core | Access the Registers |
| | CPG | Output clocks |
| | Module Standby | Control to stop clocks |
| | Software Reset | Execute software reset |
| | INTC-AP, REALTIME CORE | Control to interrupt |
| | IMP-X6 | Image Processing |
| | RT-DMAC | DMA controller |

## 104.2 Register Description

[Legend for the Register Description]

A bit assignment figure is shown for each register. The initial value and R/W attribute are indicated for each bit.

| | |
|---|---|
| Bit: | Bit number or bit range |
| Bit Name: | Bit name or field name |
| Initial value: | Register value after a reset |
| —: | Undefined value |
| R/W: | Readable/writable. The written value can be read. |
| RW1C: | Readable/writable. The bit is initialized when 1 is written. Writing 0 is ignored. |
| R: | Read-only. The write value should always be 0. (If there is a direction on a read or write value in the Description column, set the bit as directed.) |
| RWS*: | Sticky-read write bit. Readable/writable. The bit is not initialized by a hot reset. |
| RW1CS*: | Sticky-read-only status bit. The bit is initialized when 1 is written. Writing 0 is ignored. The bit is not initialized by a hot reset. |
| RSVDP: | Reserved and Preserved – Reserved for future RW implementations. Register bits are read-only and must return zero when read. Software must preserve the value read for writes to bits. |

Note: * RWS and RW1CS are only used for the configuration registers. For details, see section 104.4.1, Write/Read Attributes of Configuration Registers.

### 104.2.1 PCI Configuration Register 0 function *i* (PCICONF0F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEVICE_ID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| PCI R/W | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VENDOR_ID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| PCI R/W | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note: *1: R/W changes by the TYPE. TYPE00: R, TYPE01: -

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31to 16 | DEVICE_ID | H'0032 | *1 | R | Device ID<br>Indicates the device ID.<br>Reflects a value set in the IDSETR0.Device ID set.<br>Indicates the device ID assigned to the PCI device vendor. |
| 15 to 0 | VENDOR_ID | H'1912 | *1 | R | Vendor ID<br>Indicates the vendor ID.<br>Reflects a value set in the IDSETR0.Vendor ID set.<br>Indicates the PCI device vendor ID. |

### 104.2.2 PCI Configuration Register 1 function *i* (PCICONF1F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPE | SSE | RMA | RTA | STA | DEVSEL TIMING | | MDPE | FBBTCAP | — | 66MCAP | CAPL | INTST | — | — | IMMR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | RW1C | RW1C | RW1C | RW1C | RW1C | R | R | RW1C | R | R | R | R | R | R | R | R |
| Internal bus R/W | RW1C | RW1C | RW1C | RW1C | RW1C | R | R | RW1C | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | INTDIS | — | SERRE | IDSS | PERS | VGAPS | MWIE | SC | BME | MSE | IOSE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R/W | R | R/W | R | R/W | R | R | R | R/W | *1 | *1 |
| Internal bus R/W | R | R | R | R | R | R/W | R | R/W | R | R/W | R | R | R | R/W | R/W | R/W |

Note: *1: R/W changes by the TYPE. TYPE00: R, TYPE01: -

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | DPE | 0 | RW1C | RW1C | Detected Parity Error<br>Indicates that a parity error has been detected on the primary bus side.<br>• Root Port:<br>This bit is set by a Function whenever it receives a Poisoned TLP, regardless of the state the Parity Error Response bit in the Command register.<br>The bit is set when the Poisoned TLP is received by a Function's primary side.<br>• Endpoint:<br>This bit is set by a Function whenever it receives a Poisoned TLP, regardless of the state the Parity Error Response bit in the Command register. |
| 30 | SSE | 0 | RW1C | RW1C | Signaled System Error<br>Indicates that a system error has been detected.<br>• Root Port:<br>This bit is set when a Function sends an ERR_FATAL or ERR_NONFATAL Message, and the SERR# Enable bit in the Command register is 1b..<br>• Endpoint:<br>This bit is set when a Function sends an ERR_FATAL or ERR_NONFATAL message, and the SERR# Enable bit in the Command register is 1b.<br>For Functions that do not send ERR_FATAL or ERR_NONFATAL messages, the controller hardwires this bit to 0b. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 29 | RMA | 0 | RW1C | RW1C | Received Master Abort<br>This bit is set when a Requester receives a Completion with Unsupported Request Completion status.<br>• Root Port:<br>The bit is set when the Unsupported Request is received by a Function's primary side.<br>• Endpoint:<br>For Functions that do not make Non-Posted Requests on their own behalf, the controller hardwires this bit to 0b. |
| 28 | RTA | 0 | RW1C | RW1C | Received Target Abort<br>This bit is set when a Requester receives a Completion with Completer Abort Completion status.<br>• Root Port:<br>The bit is set when the Completer Abort is received by a Function's primary side.<br>• Endpoint:<br>For Functions that do not make Non-Posted Requests on their own behalf, the controller hardwires this bit to 0b. |
| 27 | STA | 0 | RW1C | RW1C | Signaled Target Abort<br>This bit is set when a Function completes a Posted or Non-Posted Request as a Completer Abort error.<br>• Root Port:<br>This applies to a Function when the Completer Abort was generated by its primary side.<br>• Endpoint:<br>The controller hardwires this bit to 0b for Functions that do not signal Completer Abort. |
| 26, 25 | DEVSEL TIMING | All 0 | R | R | DEVSEL Timing<br>Fixed to 0. These bits are not used for PCI Express. |
| 24 | MDPE | 0 | RW1C | RW1C | Master Data Parity Error<br>This bit is set by a Function if the Parity Error Response bit in the Command register is 1b and either of the following two conditions occurs. If the Parity Error Response bit is 0b, this bit is never set.<br>• Root Port:<br>1. Port receives a Poisoned Completion going downstream<br>2. Port transmits a Poisoned Request upstream<br>• Endpoint:<br>1. Function receives a Poisoned Completion<br>2. Function transmits a Poisoned Request |
| 23 | FBBTCAP | 0 | R | R | Fast Back to Back Transaction Capable<br>Fixed to 0. This bit is not used for PCI Express. |
| 22 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 21 | 66MCAP | 0 | R | R | 66 MHz Capable<br>This bit is always read as 0. The write value should always be 0.<br>This bit is not used for PCI Express. |
| 20 | CAPL | 1 | R | R | Capabilities List<br>Indicates that the extended capabilities list exists in the PCI compatible configuration space. This IP supports the extended capabilities list.<br>This bit must not be set to a value other than 1. |
| 19 | INTST | 0 | R | R | Interrupt Status<br>When set, indicates that an INTx emulation interrupt is pending internally in the Function.<br>Setting the Interrupt Disable bit has no effect on the state of this bit.<br>For Functions that do not generate INTx interrupts, the controller hardwires this bit to 0b. |
| 18 to 17 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | IMMR | 0 | R | R | Immediate Readiness.<br>This optional bit, when set, indicates the Function is guaranteed to be ready to successfully complete valid configuration accesses at any time following any reset that the host is capable of issuing Configuration Requests to this Function. |
| 15 to 11 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 10 | INTDIS | 0 | R/W | R/W | Interrupt Disable<br>Controls the ability of a Function to generate INTx emulation interrupts. When set, Functions are prevented from asserting INTx interrupts. |
| 9 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 8 | SERRE | 0 | R/W | R/W | SERR# Enable.<br>Enables an error message transmission when a non-fatal/fatal error is detected.<br>Setting this bit to 1 enables an error message transmission. |
| 7 | IDSS | 0 | R | R | IDSEL Stepping/Wait Cycle Control<br>This bit is always read as 0. The write value should always be 0. This bit is not used for PCI Express. |
| 6 | PERS | 0 | R/W | R/W | Parity Error Response<br>This bit controls the logging of poisoned TLPs in the Master Data Parity Error bit in the Status register. |
| 5 | VGAPS | 0 | R | R | VGA Plate Snoop<br>This bit is always read as 0. The write value should always be 0. This bit is not used for PCI Express. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 4 | MWIE | 0 | R | R | Memory Write and Invalidate Enable<br>This bit is always read as 0. The write value should always be 0.<br>This bit is not used for PCI Express. |
| 3 | SC | 0 | R | R | Special Cycle Enable<br>This bit is always read as 0. The write value should always be 0.<br>This bit is not used for PCI Express. |
| 2 | BME | 0 | R/W | R/W | Bus Master Enable<br>• Root Port:<br>This bit controls forwarding of Memory or I/O requests by a port in the Upstream direction.<br>When this bit is 0b, Memory and I/O Requests received at a Root Port must be handled as Unsupported Requests (UR) For Non-Posted Requests a Completion with UR completion status must be returned.<br>This bit does not affect forwarding of Completions in either the Upstream or Downstream direction.<br>The forwarding of Requests other than Memory or I/O Requests is not controlled by this bit.<br>• Endpoint:<br>Controls the ability of a Function to issue Memory and I/O Read/Write requests.<br>・When this bit is set, the Function is allowed to issue Memory or I/O Requests.<br>・When this bit is clear, the Function is not allowed to issue any Memory or I/O Requests.<br>Requests other than Memory or I/O Requests are not controlled by this bit. |
| 1 | MSE | 0 | *1 | R/W | Memory Space Enable<br>Enables an access to memory space.<br>When this bit is 0, the memory request to this device is handled as an unsupported request. |
| 0 | IOSE | 0 | *1 | R/W | IO Space Enable<br>Enables an access to I/O space.<br>When this bit is 0, the IO request to this device is handled as an unsupported request. |

### 104.2.3 PCI Configuration Register 2 function *i* (PCICONF2F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLASS CODE | | | | | | | | SUB CLASS CODE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG IF | | | | | | | | REVISION ID | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note: *1: R/W changes by the TYPE. TYPE00: R, TYPE01: -

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | CLASS CODE | H'00 | *1 | R | Class Code<br>Indicates the class code. |
| 23 to 16 | SUB CLASS CODE | All 0 | *1 | R | Sub Class Code<br>Indicates the sub-class code. |
| 15 to 8 | PROG IF | All 0 | *1 | R | Prog IF<br>Indicates the prog IF. |
| 7 to 0 | REVISION ID | All 0 | *1 | R | Revision ID<br>Indicates the revision ID. |

### 104.2.4 PCI Configuration Register 3 function *i* (PCICONF3F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BSTCAP | STBST | — | — | BSTCODE | | | | SFMF | Header Type | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Master Latency Timer | | | | | | | | Cache Line Size | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | BSTCAP | 0 | R | R | BIST Capable<br>Indicates the BIST function support status.<br>0: BIST function is not supported.<br>1: BIST function is supported.<br>Fixed to 0. These bits are not used. |
| 30 | STBST | 0 | R | R | Start BIST<br>Indicates the BIST function execution status.<br>0: BIST function has ended.<br>1: BIST function execution is in progress.<br>Fixed to 0. These bits are not used. |
| 29, 28 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 24 | BSTCODE | H'0 | R | R | BIST Completion Code<br>Indicates the BIST completion status code.<br>H'0: Passed BIST.<br>Non-zero values mean the Function failed.<br>Fixed to 0. These bits are not used. |
| 23 | SFMF | * | R | R | Single Function/ Multi-function<br>Indicates whether single function device or multi-function device is used.<br>・ When set, indicates that the Device may contain multiple Functions, but not necessarily. Software is permitted to probe for Functions other than Function 0.<br>・ When clear, software must not probe for Functions other than Function 0 unless explicitly indicated by another mechanism, such as an ARI or SR-IOV Capability structure.<br>・ * When in function 0 init value is 0, when in function 1 initial value is 1. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 22 to 16 | Header Type | H'00 | R | R | Header Type<br>Specifies a layout of the configuration registers.<br>Initial device type is Endpoint. When in RC mode initial value is H’01, when in EP mode initial value is H’00<br>H'00: Type00 layout<br>H'01: Type01 layout<br>• Root Port:<br>Set Type01.<br>• Endpoint:<br>Set Type00. |
| 15 to 8 | Master Latency Timer | All 0 | R | R | Master Latency Timer<br>Fixed to 0. These bits are not used. |
| 7 to 0 | Cache Line Size | All 0 | R/W | R/W | Cache Line Size<br>The Cache Line Size register is programmed by the system firmware or the operating system to system cache line size. |

### 104.2.5 PCI Configuration Register 4 function *i* (PCICONF4F*i*) *i* = 0, 1

**Header TYPE00 Only**

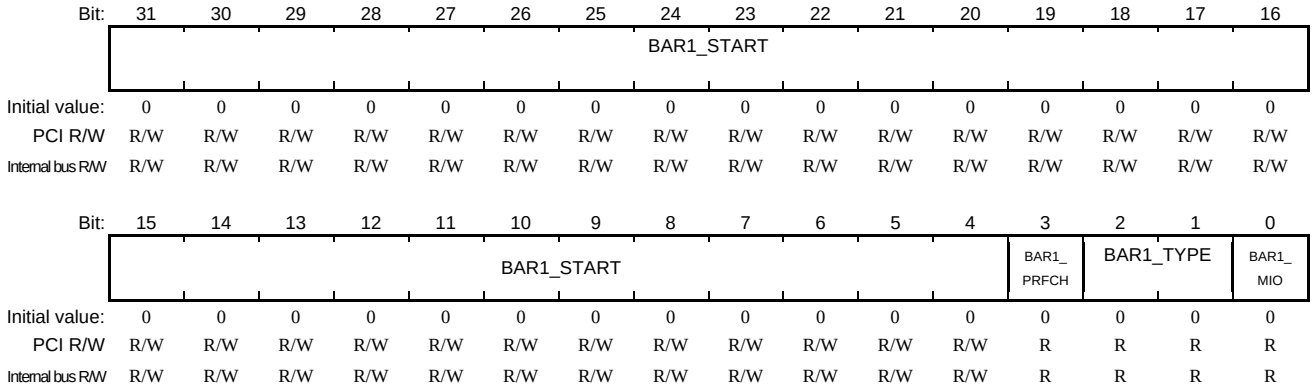
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAR0_START | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAR0_START | | | | | | | | | | | | BAR0_PRFCH | BAR0_TYPE | | BAR0_MIO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | BAR0_START | 0 | R/W | R/W | BAR0 Base Address.<br>▪ Memory Space: Base Address.<br>▪ IO Space : Base Address. |
| 3 | BAR0_PRFCH | 0 | R | R | BAR0 Prefetchable.<br>▪ Memory Space: Set to one if data is pre-fetchable.<br>BAR0 Base Address.<br>▪ IO Space : Base Address. |
| 2 to 1 | BAR0_TYPE | H'2 | R | R | BAR0 Type.<br>■ Memory Space:<br>Base Address registers that map into Memory Space can be 32 bits or 64 bits wide (to support mapping into a 64-bit address space). The bit encoding is as follows:<br>00 : Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space.<br>10 : Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.<br>01/11 : Reserved.<br>BAR0 Base Address.<br>■ IO Space:<br>Bit 2 is Base Address.<br>Bit 1 is reserved and must return 0. |
| 0 | BAR0_MIO | 0 | R | R | BAR0 Memory Space Indicator.<br>This bit is used to determine whether the register maps into Memory or I/O Space. Base Address registers that map to Memory Space must return a 0b. Base Address registers that map to I/O Space must return a 1b. |

### 104.2.6 PCI Configuration Register 5 function *i* (PCICONF5F*i*) *i* = 0, 1

**Header TYPE00 Only**




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | BAR1_START | 0 | R/W | R/W | BAR1 Base Address.<br>▪ Memory Space: Base Address.<br>▪ IO Space : Base Address. |
| 3 | BAR1_PRFCH | 0 | R | R | BAR1 Prefetchable.<br>▪ Memory Space: Set to one if data is pre-fetchable.<br>BAR1 Base Address.<br>▪ IO Space : Base Address. |
| 2 to 1 | BAR1_TYPE | H'0 | R | R | BAR1 Type.<br>■ Memory Space:<br>Base Address registers that map into Memory Space can be 32 bits or 64 bits wide (to support mapping into a 64-bit address space). The bit encoding is as follows:<br>00 : Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space.<br>10 : Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.<br>01/11 : Reserved.<br>BAR1 Base Address.<br>■ IO Space:<br>Bit 2 is Base Address.<br>Bit 1 is reserved and must return 0. |
| 0 | BAR1_MIO | 0 | R | R | BAR1 Memory Space Indicator.<br>This bit is used to determine whether the register maps into Memory or I/O Space. Base Address registers that map to Memory Space must return a 0b. Base Address registers that map to I/O Space must return a 1b. |

### 104.2.7 PCI Configuration Register 6 function *i* (PCICONF6F*i*) *i* = 0, 1

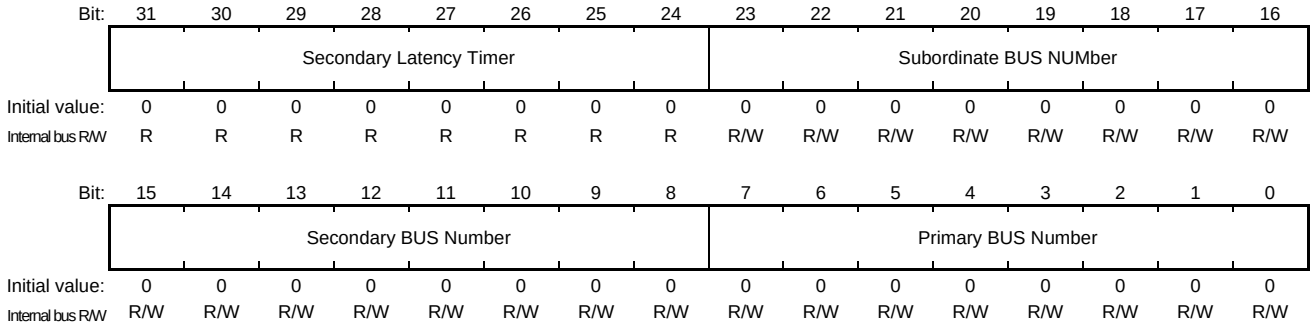
**Header TYPE00**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAR2_START | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAR2_START | | | | | | | | | | | | BAR2_PRFCH | BAR2_TYPE | | BAR2_MIO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | BAR2_START | 0 | R/W | R/W | BAR2 Base Address.<br>▪ Memory Space: Base Address.<br>▪ IO Space : Base Address. |
| 3 | BAR2_PRFCH | 0 | R | R | BAR2 Prefetchable.<br>▪ Memory Space: Set to one if data is pre-fetchable.<br>BAR2 Base Address.<br>▪ IO Space : Base Address. |
| 2 to 1 | BAR2_TYPE | H'2 | R | R | BAR2 Type.<br>■ Memory Space:<br>Base Address registers that map into Memory Space can be 32 bits or 64 bits wide (to support mapping into a 64-bit address space). The bit encoding is as follows:<br>00 : Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space.<br>10 : Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.<br>01/11 : Reserved.<br>BAR2 Base Address.<br>■ IO Space:<br>Bit 2 is Base Address.<br>Bit 1 is reserved and must return 0. |
| 0 | BAR2_MIO | 0 | R | R | BAR2 Memory Space Indicator.<br>This bit is used to determine whether the register maps into Memory or I/O Space. Base Address registers that map to Memory Space must return a 0b. Base Address registers that map to I/O Space must return a 1b. |

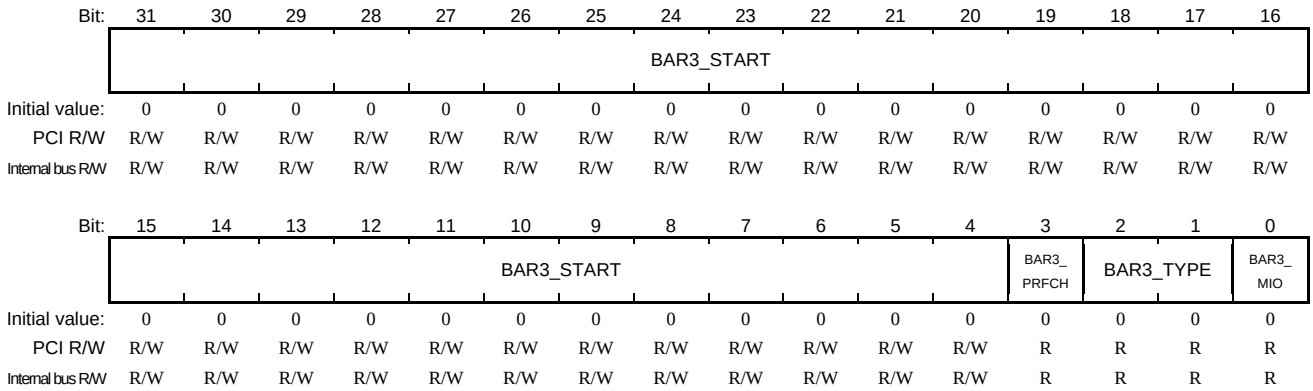
**Header TYPE01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | Secondary Latency Timer | All 0 | R | Secondary Latency Timer<br>Fixed to 0. These bits are not used for PCI Express. |
| 23 to 16 | Subordinate BUS NUMber | All 0 | R/W | Subordinate Bus Number<br>The Subordinate Bus Number register is used to record the bus number of the highest numbered PCI bus segment which is behind (or subordinate to) the bridge. |
| 15 to 8 | Secondary BUS Number | All 0 | R/W | Secondary Bus Number<br>The Secondary Bus Number register is used to record the bus number of the PCI bus segment to which the secondary interface of the bridge is connected. |
| 7 to 0 | Primary BUS Number | All 0 | R/W | Primary Bus Number<br>This register is not used by PCIExpress Functions. It is implemented for compatibility with legacy software. |

### 104.2.8 PCI Configuration Register 7 function *i* (PCICONF7F*i*) *i* = 0, 1

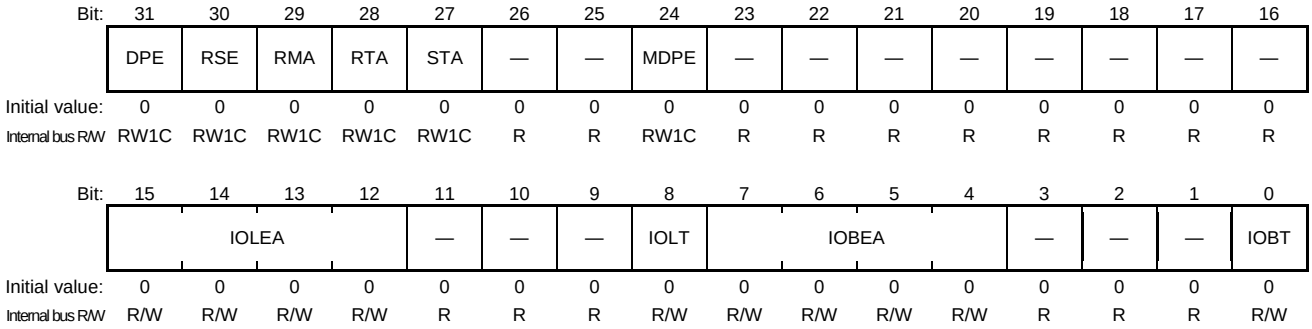
**Header TYPE00**




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | BAR3_START | 0 | R/W | R/W | BAR3 Base Address.<br>▪ Memory Space: Base Address.<br>▪ IO Space : Base Address. |
| 3 | BAR3_PRFCH | 0 | R | R | BAR3 Prefetchable.<br>▪ Memory Space: Set to one if data is pre-fetchable.<br>BAR3 Base Address.<br>▪ IO Space : Base Address. |
| 2 to 1 | BAR3_TYPE | H'0 | R | R | BAR3 Type.<br>■ Memory Space:<br>Base Address registers that map into Memory Space can be 32 bits or 64 bits wide (to support mapping into a 64-bit address space). The bit encoding is as follows:<br>00 : Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space.<br>10 : Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.<br>01/11 : Reserved.<br>BAR3 Base Address.<br>■ IO Space:<br>Bit 2 is Base Address.<br>Bit 1 is reserved and must return 0. |
| 0 | BAR3_MIO | 0 | R | R | BAR3 Memory Space Indicator.<br>This bit is used to determine whether the register maps into Memory or I/O Space. Base Address registers that map to Memory Space must return a 0b. Base Address registers that map to I/O Space must return a 1b. |

### Header TYPE01

- Secondary status register




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | DPE | 0 | RW1C | Detected Parity Error<br>Indicates that a parity error has been detected on the secondary bus side.<br>This bit is set to 1 upon reception of a poisoned TLP. |
| 30 | RSE | 0 | RW1C | Received System Error<br>Indicates that a system error has been detected.<br>This bit is set to 1 upon reception of an ERR_FATAL/NON_FATAL message. |
| 29 | RMA | 0 | RW1C | Received Master Abort<br>Indicates that a master abort has been detected on the secondary bus side.<br>This bit is set to 1 upon reception of a completion with the unsupported request completion status. |
| 28 | RTA | 0 | RW1C | Received Target Abort<br>Indicates that a target abort has been detected on the secondary bus side.<br>This bit is set to 1 upon reception of a completion with the completer abort completion status. |
| 27 | STA | 0 | RW1C | Signaled Target Abort<br>Indicates that a target abort has been transmitted to the secondary bus side.<br>This bit is set to 1 upon transmission of a completion with the completer abort completion status. |
| 26, 25 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 24 | MDPE | 0 | RW1C | Master Data Parity Error<br>Indicates that a master data parity error has occurred on the secondary bus side. This bit is set when either of the following conditions is satisfied.<br>1. A poisoned completion has been received.<br>2. A poisoned request has been transmitted. |
| 23 to 16 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 15 to 12 | IOLEA | All 0 | R/W | IO Limit End Address<br>These bits correspond to the address[15:12] of IO address range. |
| 11 to 9 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | IOLT | 0 | R/W | IO Limit Type<br>This bit encodes the IO addressing capability of the bridge.<br>0: 16-bit I/O address<br>1: 32-bit I/O address |
| 7 to 4 | IOBEA | All 0 | R/W | IO Base End Address<br>According to the PCI standard, sets the address[15:12] of the base address of the IO transaction to be transferred to the primary bus.<br>In this IP, this field has no effect on the hardware. |
| 3 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | IOBT | 0 | R/W | IO Base Type<br>According to the PCI standard, indicates the base address decoding format of the IO transactions to be transferred to the primary bus.<br>0: 16-bit I/O address<br>1: 32-bit I/O address |

### 104.2.9 PCI Configuration Register 8 function *i* (PCICONF8F*i*) *i* = 0, 1

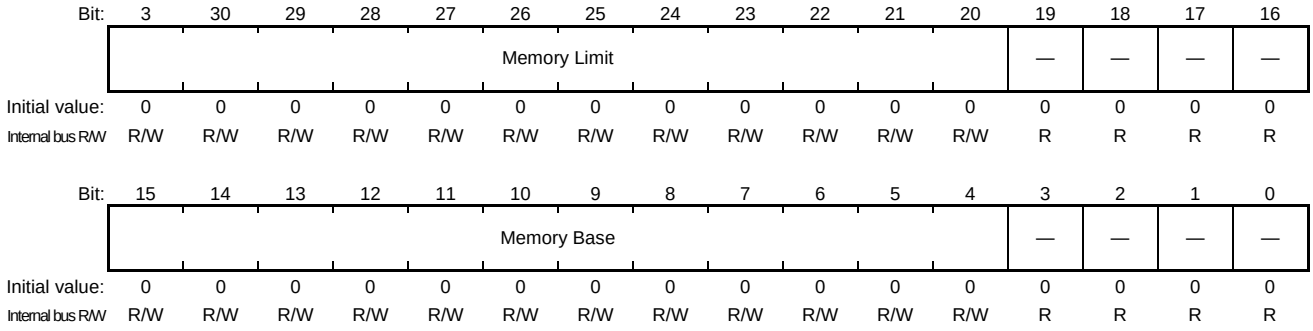
**Header TYPE00**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAR4_START | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAR4_START | | | | | | | | | | | | BAR4_PRFCH | BAR4_TYPE | | BAR4_MIO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | BAR4_START | 0 | R/W | R/W | BAR4 Base Address.<br>▪ Memory Space: Base Address.<br>▪ IO Space : Base Address. |
| 3 | BAR4_PRFCH | 0 | R | R | BAR4 Prefetchable.<br>▪ Memory Space: Set to one if data is pre-fetchable.<br>BAR4 Base Address.<br>▪ IO Space : Base Address. |
| 2 to 1 | BAR4_TYPE | H'0 | R | R | BAR4 Type.<br>■ Memory Space:<br>Base Address registers that map into Memory Space can be 32 bits or 64 bits wide (to support mapping into a 64-bit address space). The bit encoding is as follows:<br>00 : Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space.<br>10 : Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.<br>01/11 : Reserved.<br>BAR4 Base Address.<br>■ IO Space:<br>Bit 2 is Base Address.<br>Bit 1 is reserved and must return 0. |
| 0 | BAR4_MIO | 1 | R | R | BAR4 Memory Space Indicator.<br>This bit is used to determine whether the register maps into Memory or I/O Space. Base Address registers that map to Memory Space must return a 0b. Base Address registers that map to I/O Space must return a 1b. |

### Header TYPE01




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 20 | Memory Limit | All 0 | R/W | Memory Limit<br>These bits correspond to the upper 12 address bits, Address[31:20], of 32-bit addresses. |
| 19 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 4 | Memory Base | All 0 | R/W | Memory Base<br>These bits correspond to the upper 12 address bits, Address[31:20], of 32-bit addresses. |
| 3 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.10 PCI Configuration Register 9 function *i* (PCICONF9F*i*) *i* = 0, 1

**Header TYPE00**

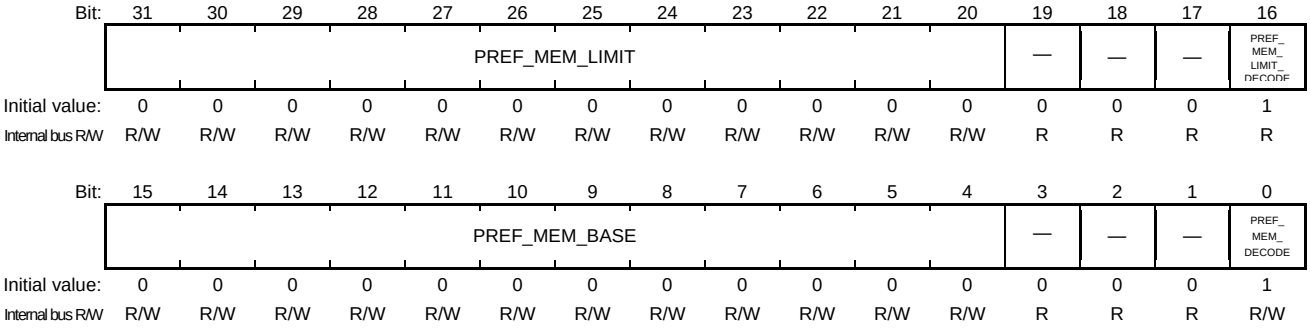
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAR5_START | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BAR5_START | | | | | | | | | | | | BAR5_ PRFCH | BAR5_TYPE | | BAR5_ MIO |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | BAR5_START | 0 | R/W | R/W | BAR5 Base Address.<br>▪ Memory Space: Base Address.<br>▪ IO Space : Base Address. |
| 3 | BAR5_PRFCH | 0 | R | R | BAR5Prefetchable.<br>▪ Memory Space: Set to one if data is pre-fetchable.<br>BAR5 Base Address.<br>▪ IO Space : Base Address. |
| 2 to 1 | BAR5_TYPE | H'0 | R | R | BAR5 Type.<br>■ Memory Space:<br>Base Address registers that map into Memory Space can be 32 bits or 64 bits wide (to support mapping into a 64-bit address space). The bit encoding is as follows:<br>00 : Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space.<br>10 : Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.<br>01/11 : Reserved.<br>BAR5 Base Address.<br>■ IO Space:<br>Bit 2 is Base Address.<br>Bit 1 is reserved and must return 0. |
| 0 | BAR5_MIO | 0 | R | R | BAR5 Memory Space Indicator.<br>This bit is used to determine whether the register maps into Memory or I/O Space. Base Address registers that map to Memory Space must return a 0b. Base Address registers that map to I/O Space must return a 1b. |

**Header TYPE01**

Prefetchable Memory Limit and Base Register




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 20 | PREF_MEM_LIMIT | H'000 | R/W | Prefetchable Memory Limit Address<br>If the Prefetchable Memory Limit register indicates support for 32-bit addressing, then the Prefetchable Limit Upper 32 Bits register is implemented as a read-only register that returns zero when read.<br>If the Prefetchable Memory Limit registers indicate support for 64-bit addressing, then the Prefetchable Limit Upper 32 Bits register is implemented as a read/write register which must be initialized by configuration software.<br>If a 64-bit prefetchable memory address range is supported, the Prefetchable Limit Upper 32 Bits register specifies the upper 32 bits, corresponding to Address[63:32], of the 64-bit limit addresses which specify the prefetchable memory address range. |
| 19 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | PREF_MEM_LIMIT_DECODE | 1 | R | Prefetchable Memory Limit DecodeThis bit encodes whether or not the bridge supports 64-bit addresses.<br>0: 32-bit address<br>1: 64-bit address. |
| 15 to 4 | PREF_MEM_BASE | H'000 | R/W | Prefetchable Memory Base<br>If the Prefetchable Memory Base register indicates support for 32-bit addressing, then the Prefetchable Base Upper 32 Bits register is implemented as a read-only register that returns zero when read.<br>If the PrefetchableMemory Base register indicates support for 64-bit addressing, then the Prefetchable Limit Upper 32 Bits register is implemented as a read/write register which must be initialized by configuration software.<br>If a 64-bit prefetchable memory address range is supported, the Prefetchable Base Upper 32 Bits register spec ifies the upper 32 bits, corresponding to Address[63:32], of the 64-bit base addresses which specify the prefetchable memory address range. |
| 3 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 0 | PREF_MEM_DECODE | 1 | R | Prefetchable Memory Base DecodeThis bit encodes whether or not the bridge supports 64-bit addresses. The value indicates the following:<br>0: 32-bit address<br>1: 64-bit address |

### 104.2.11 PCI Configuration Register 10 function *i* (PCICONF10F*i*) *i* = 0, 1

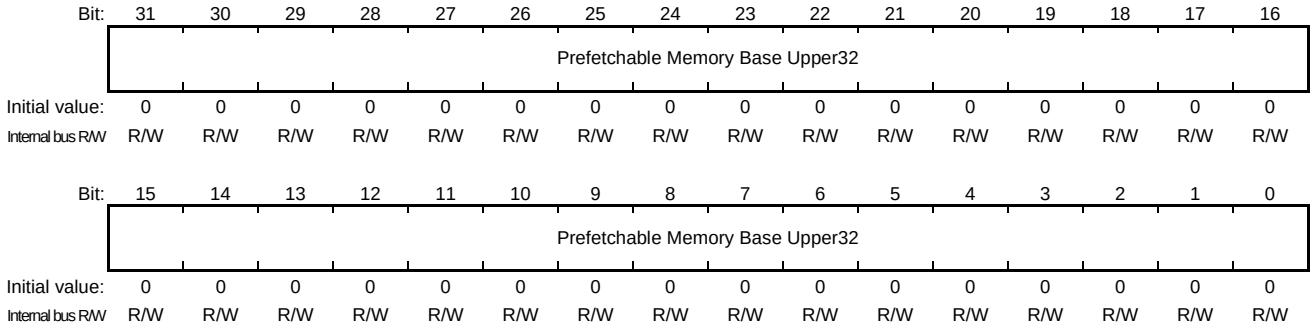
**Header TYPE00**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Card Bus CIS Pointer | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Card Bus CIS Pointer | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | Card Bus CIS Pointer | All 0 | R | R | Card Bus CIS Pointer<br>Its functionality does not apply to PCI Express. It must be hardwired to All 0. |

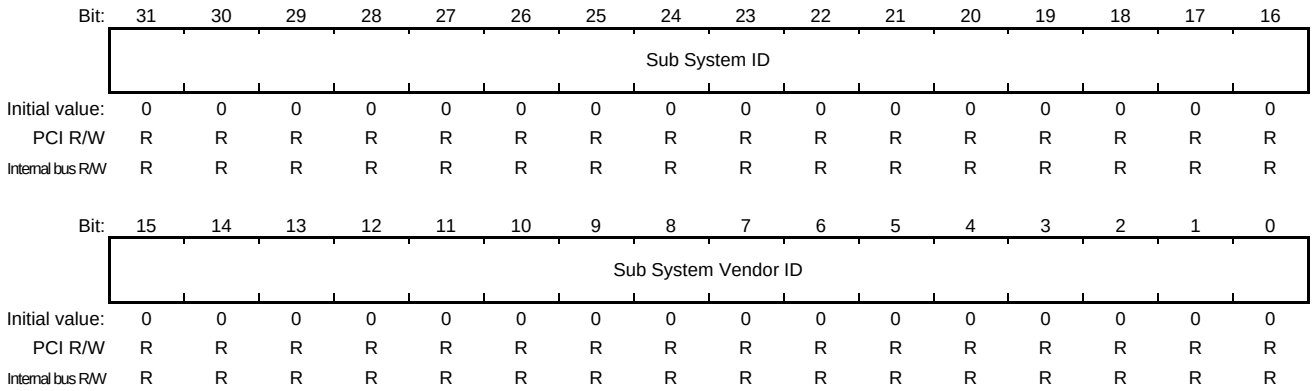
### Header TYPE01




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Prefetchable Memory Base Upper32 | All 0 | R/W | Prefetchable Memory Base Upper 32<br>If the Prefetchable Memory Base register indicates support for 32-bit addressing, then this register is implemented as read-only register that returns zero when read.<br>If the Prefetchable Memory Base register indicate support for 64-bit addressing, then this register is implemented as read/write register which must be initialized by configuration software.<br>This register specifies the upper 32 bits, corresponding to Address[63:32], of the 64-bit base addresses which specify the prefetchable memory address range. |

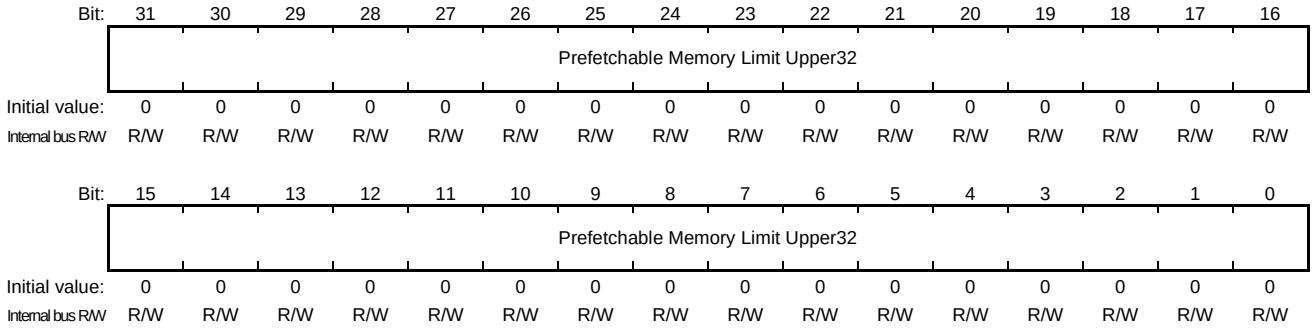
### 104.2.12 PCI Configuration Register 11 function *i* (PCICONF11F*i*) *i* = 0, 1

**Header TYPE00**




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | Sub System ID | All 0 | R | R | Sub System ID<br>Indicates the sub-system ID. |
| 15 to 0 | Sub System Vendor ID | All 0 | R | R | Sub System Vendor ID<br>Indicates the sub-system vendor ID. |

### Header TYPE01




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Prefetchable Memory Limit Upper32 | All 0 | R/W | Prefetchable Memory Limit Upper 32 Bit<br>If the Prefetchable Memory Limit register indicate support for 64-bit addressing, then this register is implemented as read/write register which must be initialized by configuration software.<br>This register specifies the upper 32 bits, corresponding to Address[63:32], of the<br>64-bit base addresses which specify the prefetchable memory address range. |

### 104.2.13 PCI Configuration Register 12 function *i* (PCICONF12F*i*) *i* = 0, 1

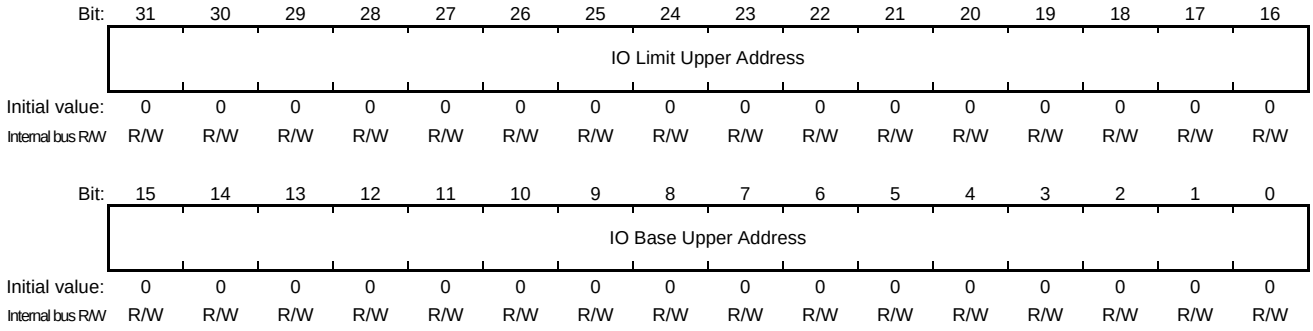
**Header TYPE00**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Expansion ROM BAR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Expansion ROM BAR | | | | | — | — | — | — | — | — | — | — | — | — | EROME |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 11 | Expansion ROM BAR | All 0 | R | R | Expansion ROM BAR<br>Upper 21 bits of the Expansion ROM base address.<br>The number of bits (out of these 21) that a Function actually implements depends on how much address space the Function requires. |
| 10 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | EROME | 0 | R | R | Expansion ROM Enable<br>Expansion ROM Enable. This bit controls whether or not the Function accepts accesses to its expansion ROM.<br>■ When this bit is 0b, the Function's expansion ROM address space is disabled.<br>■ When the bit is 1b, address decoding is enabled using the parameters in the other part of the Expansion ROM Base<br>According to the PCI standard, indicates that access to expansion ROM is enabled. |

### Header TYPE01




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | IO Limit Upper Address | All 0 | R/W | IO Limit Upper 16Bits<br>The I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are optional extensions to the I/O Base and I/O Limit registers.<br>If the I/O Base and I/O Limit registers indicate support for 16-bit I/O address decoding, then the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are implemented as read-only registers which return zero when read.<br>If the I/O Base and I/O Limit registers indicate support for 32-bit I/O addressing, then the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers must be initialized by configuration software so default states are not specified. |
| 15 to 0 | IO Base Upper Address | All 0 | R/W | IO Limit Upper 16Bits<br>The I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are optional extensions to the I/O Base and I/O Limit registers.<br>If the I/O Base and I/O Limit registers indicate support for 16-bit I/O address decoding, then the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are implemented as read-only registers which return zero when read.<br>If the I/O Base and I/O Limit registers indicate support for 32-bit I/O addressing, then the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers must be initialized by configuration software so default states are not specified. |

### 104.2.14 PCI Configuration Register 13 function *i* (PCICONF13F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

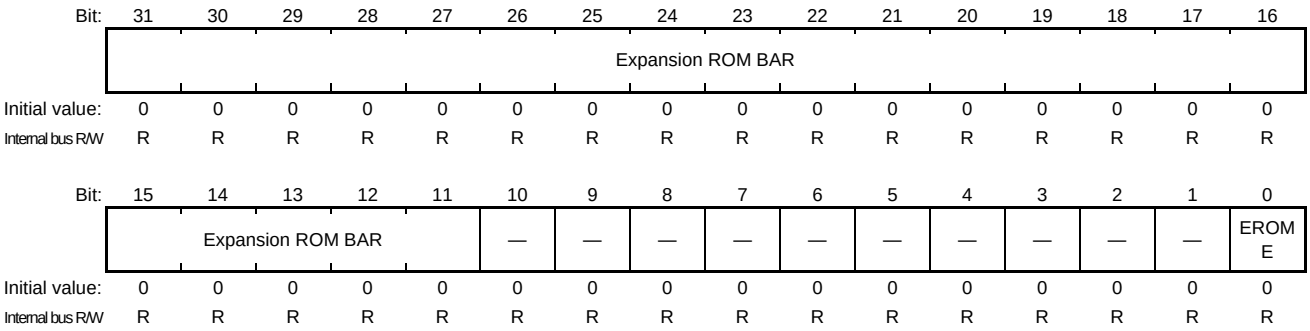
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Capabilities Pointer | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | Capabilities Pointer | H'40 | R | R | Capabilities Pointer<br>This register is used to point to a linked list of capabilities implemented by this Function.<br>Since all PCI Express Functions are required to implement both the Power Management Capability and the PCI Express Capability structure, these structures must be included somewhere in the linked list; this register may point to either of these Capability Structures or to an optional Capability Structure implemented by the Function. The bottom two bits are Reserved and must be set to 00b.<br>Software must mask these bits off before using this register as a pointer in Configuration Space to the first entry of a linked list of new capabilities. |

### 104.2.15 PCI Configuration Register 14 function *i* (PCICONF14F*i*) *i* = 0, 1

**Header TYPE01 only**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 11 | Expansion ROM BAR | All 0 | R | Expansion ROM BAR<br>According to the PCI standard, indicates the value of expansion ROM base address register. |
| 10 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | EROME | 0 | R | Expansion ROM Enable<br>According to the PCI standard, indicates that access to expansion ROM is enabled. |

### 104.2.16 PCI Configuration Register 15 function *i* (PCICONF15F*i*) *i* = 0, 1

**Header TYPE00**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Interrupt Pin | | | | | | | | Interrupt Line | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | Interrupt Pin | All 0 | R | R/W | Interrupt Pin<br>Specifies the legacy interrupt message used by the device at initialization. When no legacy interrupts are used, set 0.<br>H'0: Legacy interrupt is not used.<br>H'1: INTA is used.<br>H'2: INTB is used.<br>H'3: INTC is used.<br>H'4: INTD is used. |
| 7 to 0 | Interrupt Line | H'FF | R/W | R/W | Interrupt Line<br>Indicates information on legacy interrupt routing. |

### Header TYPE01

- Bridge control register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | SECBRST | MABMD | VGA16DEC | VGAE | ISAE | SERRE | PERS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Interrupt Pin | | | | | | | | Interrupt Line | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 22 | SECBRST | 0 | R/W | Secondary Bus Reset<br>Setting this bit triggers a hot reset on the corresponding PCI Express Port. |
| 21 | MABMD | 0 | R | Master Abort Mode<br>This bit is always read as 0. The write value should always be 0.<br>This bit is not used for PCI Express. |
| 20 | VGA16DEC | 0 | R/W | VGA 16-bit Decode<br>According to the PCI standard, enables VGA IO decoding.<br>The following actions are taken based on the value of the VGA_16B_DEC bit:<br>・0b: Execute 10-bit address decodes on VGA I/O accesses<br>・1b: Execute 16-bit address decodes on VGA I/O accesses<br>For Functions that do not support VGA, the controller hardwires this bit to 0b. |
| 19 | VGAE | 0 | R/W | VGA Enable<br>The following actions are taken based on the value of the VGA_EN bit:<br>・0b: Do not forward VGA compatible memory and I/O addresses from the primary to the secondary interface (addresses defined above) unless they are enabled for forwardingby the defined I/O and memory address ranges<br>・1b: Forward VGA compatible memory and I/O addresses (addresses defined above) from the primary interface to the secondary interface (if the I/O Space Enable and Memory Space Enable bits are set) independent of the I/O and memory address ranges and independent of the ISA Enable bit For Functions that do not support VGA, the controller hardwires this bit to 0b. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 18 | ISAE | 0 | R/W | ISA Enable<br>Modifies the response by the bridge to ISA I/O addresses.<br>The following actions are taken based on the value of the ISA_EN bit:<br>■ 0b: Forward downstream all I/O addresses in the address range defined by the I/O Base and I/O Limit registers<br>■ 1b: Forward upstream ISA I/O addresses in the address range defined by the I/O Base and I/O Limit registers that are in the first 64 KB of PCI I/O address space (top 768 bytes of each 1-KB block.) |
| 17 | SERRE | 0 | R/W | SERR Enable<br>Enables ERR_FATAL/ERR_NONFATAL/ERR_COR message transfer (notification) from the secondary IF to the primary IF.<br>1: Enables error message transfer (notification).<br>0: Disables error message transfer (notification). |
| 16 | PERS | 0 | R/W | Parity Error Response<br>This bit controls the logging of poisoned TLPs in the Master Data Parity Error bit in the Secondary Status register. |
| 15 to 8 | Interrupt Pin | All 0 | R/W | Interrupt Pin<br>The Interrupt Pin Registers that identifies the legacy interrupt Message(s) the Function uses.<br>Valid valuesare:<br>・01h, 02h, 03h, and 04h: map to legacy interrupt Messages for INTA, INTB, INTC, and INTD respectively.<br>・00h: indicates that the Function uses no legacy interrupt Message(s).<br>・05h through FFh: Reserved. |
| 7 to 0 | Interrupt Line | H'FF | R/W | Interrupt Line<br>The Interrupt Line register communicates interrupt line routing information. |

### 104.2.17 PCI Power Management Capability Register 0 function *i* (PMCAP0F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMESP | | | | | D2SP | D1SP | AUX Current | | | DSI | IRO RD0 | PMECL K | PCI PM Version | | |
| Initial value: | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Next Capability Pointer | | | | | | | | Capability ID | | | | | | | |
| Initial value: | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 27 | PMESP | H'1B | R | R | Power Management Event Support<br>This 5-bit field indicates the power states in which the Function may generate a PME and/or forward PME Messages.<br>A value of 0b for any bit indicates that the Function is not capable of asserting PME while in that power state.<br>bit(27) X XXX1b - PME can be generated from D0<br>bit(28) X XX1Xb - PME can be generated from D1<br>bit(29) X X1XXb - PME can be generated from D2 (not support D2 Power Management State)<br>bit(30) X 1XXXb - PME can be generated from D3hot<br>bit(31) 1 XXXXb - PME can be generated from D3cold<br>Bit 31 (PME can be asserted from D3cold) represents a special case. |
| 26 | D2SP | 0 | R | R | D2 Support<br>If this bit is Set, this Function supports the D2 Power Management State.<br>Functions that do not support D2 must always return a value of 0b for this bit. (not support D2 Power Management State) |
| 25 | D1SP | 1 | R | R | D1 Support<br>If this bit is Set, this Function supports the D1 Power Management State.<br>Functions that do not support D1 must always return a value of 0b for this bit. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 24 to 22 | AUX Current | H'0 | R | R | AUX Current<br>Indicates the current of 3.3V auxiliary power supply.<br>Set this field at initialization.<br>H'0: 0 mA (without auxiliary power supply)<br>H'1: 55 mA<br>H'2: 100 mA<br>H'3: 160 mA<br>H'4: 220 mA<br>H'5: 270 mA<br>H'6: 320 mA<br>H'7: 375 mA |
| 21 | DSI | 0 | R | R | Device Specific Initialization<br>The DSI bit indicates whether special initialization of this Function is required.<br>When Set indicates that the Function requires a device specific initialization sequence following a transition to the D0 uninitialized state. |
| 20 | IRORD0 | 0 | R | R | Immediate_Readiness_on_Return_to_D0<br>If this bit is a “1”, this Function is guaranteed to be ready to successfully complete valid accesses immediately after being set to D0. When this bit is “1”, for accesses to this Function, software is exempt from all requirements to delay accesses following a transition to D0, including but not limited to the 10 ms delay. |
| 19 | PMECLK | 0 | R | R | PME Clock<br>This bit is always read as 0. The write value should always be 0.<br>This bit is not used for PCI Express. |
| 18 to 16 | PCI PM Version | H'3 | R | R | PCI PM Version<br>Indicate the corresponding PCI PM Interface Specification version. This device corresponds to the PCI PM Interface Specification version 1.2. |
| 15 to 8 | Next Capability Pointer | H'50 | R | R | PM Next Capability Pointer<br>This field provides an offset into the Function’s Configuration Space pointing to the location of next item in the capabilities list. If there are no additional items in the capabilities list, this field is set to 00h. |
| 7 to 0 | Capability ID | H'01 | R | R | Capability ID<br>Capability List ID<br>Indicates the power management capability (H'01). |

### 104.2.18 PCI Power Management Capability Register 1 function *i* (PMCAP1F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DATA | | | | | | | | BPCCE | B2B3SP | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMEST | Data Scale | | Data Select | | | | PMEE | — | — | — | — | NOSRST | — | Power State | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| PCI R/W | RW1C | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | RW1C | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | DATA | All 0 | R | R | Power Data Information Register.<br>Fixed to 0. This function is not supported. |
| 23 | BPCCE | 0 | R | R | Bus Power Clock Control Enable<br>Enables the PCI bus power/clock control function.<br>0: Disables the PCI bus power/clock control function.<br>1: Enables the PCI bus power/clock control function.<br>Set this field to 0 at initialization.<br>Fixed to 0. This function is not supported. |
| 22 | B2B3SP | 0 | R | R | B2 B3 Support<br>Indicates the bus state in the D3Hot state.<br>0: Clock supply to the secondary IF does not stop in the D3Hot state. (B3)<br>1: Clock supply to the secondary IF stops in the D3Hot state. (B2)<br>Set this field to 0 at initialization.<br>Fixed to 0. This function is not supported. |
| 21 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | PMEST | 0 | RW1C | RW1C | PME Status<br>This field is valid when the PME is supported.<br>Indicates the PME generation status.<br>0: A PME message is not being transmitted.<br>1: A PME message is being transmitted. |
| 14, 13 | Data Scale | All 0 | R | R | Data Scale<br>This field indicates the scaling factor to be used when interpreting the value of the Data register.<br>Fixed to 0. This function is not supported. |
| 12 to 9 | Data Select | All 0 | R | R | Data Select<br>This 4-bit field is used to select which data is to be reported through the Data register and Data_Scale field.<br>Fixed to 0. This function is not supported. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 8 | PMEE | 0 | R/W | R/W | PME Enable<br>Enables PME message transmission.<br>0: Disables PME message transmission.<br>1: Enables PME message transmission. |
| 7 to 4 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 3 | NOSRST | 1 | R | R | No Soft Reset<br>Indicates whether internal reset is performed or not when the device makes a transition from the D3Hot state to the D0 state.<br>0: Internal reset is performed when the device makes a transition from D3Hot state to D0 state.<br>1: Internal reset is not performed when the device makes a transition from D3Hot state to D0 state.<br>Set the device function at initialization. |
| 2 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 1, 0 | Power State | B’00 | R/W | R/W | Power State<br>Specifies the power state.<br>B’00: D0<br>B’01: D1<br>B’10: D2 (not support D2 Power Management State, Read/Write value B'10 always read as 0. The write value should always be 0.)<br>B’11: D3Hot |

### 104.2.19 MSI Capability Register 0 function *i* (MSICAP0F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | PMEDE | PMEDC | PVMSK | 64ADCAP | MMESE | | | MMESCAP | | | MSIE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R/W |
| Internal bus R/W | R | R | R | R | R | R/W | R | R | R | R/W | R/W | R/W | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Next Capability Pointer | | | | | | | | Capability ID | | | | | | | |
| Initial value: | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 26 | PMEDE | 0 | R | R/W | Extended Message Data Enable.<br>If Set, the Function is enabled to provide Extended Message Data.<br>If Clear, the Function is not enabled to provide Extended Message Data.<br>This bit must be read-write if the Extended Message Data Capable bit is 1b; otherwise it must be hardwired to 0b. |
| 25 | PMEDC | 0 | R | R | Extended Message Data Capable.<br>If Set, the Function is capable of providing Extended Message Data.<br>If Clear, the Function does not support providing Extended Message Data. |
| 24 | PVMSK | 1 | R | R | MSI Per Vector Masking Capable<br>Indicates whether the per vector masking function is supported or not.<br>0: The per vector masking function is not supported.<br>1: The per vector masking function is supported. |
| 23 | 64ADCAP | 1 | R | R | MSI 64-bit Address Capable<br>Indicates whether transmission of a 64-bit address message is supported or not.<br>0: A 64-bit address message cannot be transmitted.<br>1: A 64-bit address message can be transmitted.<br>This device supports transmission of a 64-bit address message. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 22 to 20 | MMESE | B’000 | R/W | R/W | MSI Multiple Message Enable<br>Sets the number of interrupt vectors that can be transmitted.<br>B’000: 1 vector<br>B’001: 2 vectors<br>B’010: 4 vectors<br>B’011: 8 vectors<br>B’100: 16 vectors<br>B’101: 32 vectors<br>B’110: Reserved<br>B’111: Reserved |
| 19 to 17 | MMESCAP | B’000 | R | R | MSI Multiple Message Capable<br>Sets the number of interrupt vectors that a device can transmit.<br>B’000: 1 vector<br>B’001: 2 vectors<br>B’010: 4 vectors<br>B’011: 8 vectors<br>B’100: 16 vectors<br>B’101: 32 vectors<br>B’110: Reserved<br>B’111: Reserved<br>Set the device function at initialization. |
| 16 | MSIE | 0 | R/W | R/W | MSI Enable<br>Enables the MSI function.<br>0: Disables the MSI function.<br>1: Enables the MSI function. |
| 15 to 8 | Next Capability Pointer | H'70 | R | R | MSI Next Capability Pointer<br>Pointer to the extended capability list.<br>Initial Value: Pointer value calculated at configuration time. |
| 7 to 0 | Capability ID | H'05 | R | R | MSI Capability ID<br>Capability list ID<br>Indicates the MSI capability ID (H'05). |

### 104.2.20 MSI Capability Register 1 function *i* (MSICAP1F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

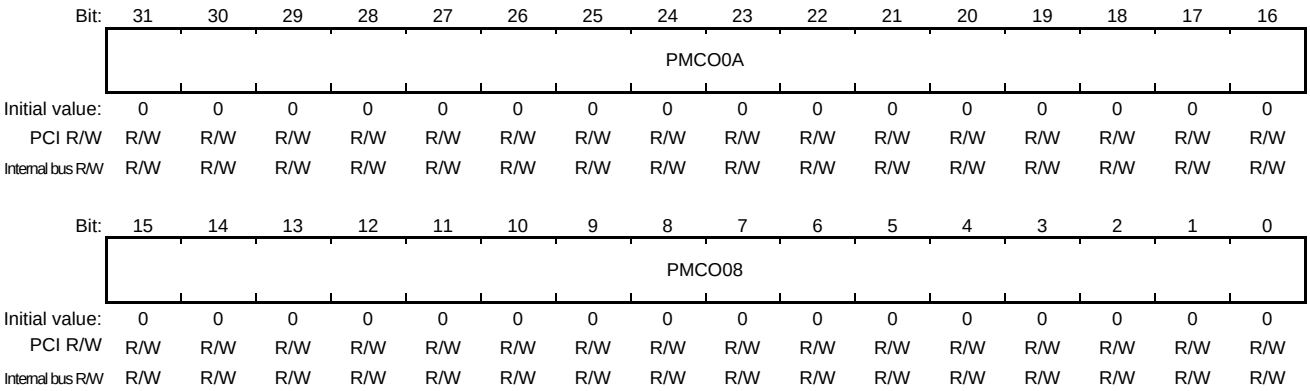
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Lower Message Address | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Lower Message Address | | | | | | | | | | | | | | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 2 | Lower Message Address | All 0 | R/W | R/W | MSI Message Lower Address Field<br>Specifies the address[31:2] of an MSI message during an MSI transmission.<br>The value of this field is used for transmission of an MSI message. |
| 1, 0 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.21 MSI Capability Register 2 function *i* (MSICAP2F*i*) *i* = 0, 1

**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | PMCO0A | All 0 | R/W | R/W | For a 32 bit MSI Message, this field contains Ext MSI Data.<br>For 64-bit it contains upper 16 bits of the Upper Address. |
| 15 to 0 | PMCO08 | All 0 | R/W | R/W | For a 32-bit MSI Message, this field contains Data.<br>For 64-bit it contains lower 16 bits of the Upper Address. |

### 104.2.22 MSI Capability Register 3 function *i* (MSICAP3F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCO0E | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCO0C | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | PMCO0E | All 0 | R/W | R/W | For a 64-bit MSI Message, this field contains Data.<br>For 32-bit, it contains the upper Mask Bits if PVM is enabled. |
| 15 to 0 | PMCO0C | All 0 | R/W | R/W | For a 64-bit MSI Message, this field contains Data.<br>For 32-bit, it contains the lower Mask Bits if PVM is enabled. |

### 104.2.23 MSI Capability Register 4 function *i* (MSICAP4F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCO10 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCO10 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PMCO10 | All 0 | R/W | R/W | Used for MSI when the Per Vector Masking Capable bit (MSICAP0Fi.PVMSK) is set.<br>For 32-bit contains Pending Bits.<br>For 64-bit, contains Mask Bits. |

### 104.2.24 MSI Capability Register 5 function *i* (MSICAP5F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCO14 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMCO14 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PMCO14 | All 0 | R/W | R/W | Used for MSI 64-bit messaging when the Per Vector Masking Capable bit (MSICAP0Fi.PVMSK) is set. Contains Pending Bits. |

### 104.2.25 PCI Express Capability Register 0 function *i* (EXPCAP0F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | Interrupt Message Number | | | | | SLTIMP | Device Port Type | | | | Capability Version | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Next Capability Pointer | | | | | | | | Capability ID | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31, 30 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29 to 25 | Interrupt Message Number | All 0 | R | R | PCIe Interrupt Message Number<br>This field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with any of the status bits of this Capability structure. |
| 24 | SLTIMP | 0 | R | R | PCIe Slot Implemented<br>When Set, this bit indicates that the Link associated with this Port is connected to a slot (as compared to being connected to a system-integrated device or being disabled).<br>This bit is valid for Downstream Ports. This bit is undefined for Upstream Ports. |
| 23 to 20 | Device Port Type | H'0 | R | R | PCIe Device Port Type<br>Indicates the device type.<br>Defined encodings for Functions that implement a Type 00h PCI Configuration Space header are:<br>H'0: PCI Express Endpoint<br>H'1: Legacy PCI Express Endpoint<br>H'9: Root Complex Integrated Endpoint Device<br>H'A: Root Complex Event Collector<br>Defined encodings for Functions that implement a Type 01h PCI Configuration Space header are:<br>H'4: Root Port of PCI Express Root Complex<br>H'5: Upstream Port of PCI Express Switch<br>H'6: Downstream Port of PCI Express Switch<br>H'7: PCI Express to PCI/PCI-X Bridge<br>H'8: PCI/PCI-X to PCI Express Bridge |
| 19 to 16 | Capability Version | H'2 | R | R | PCIe Capability Version Number<br>Indicates the version of this capability. |
| 15 to 8 | NEXT Capability Pointer | H'00 | R | R | PCIe Next Capability Pointer<br>Pointer to the extension capability list<br>Indicates H'00, which is End Of List. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 7 to 0 | Capability ID | H'10 | R | R | Capability ID<br>Capability List ID<br>Indicates the PCI Express capability ID (H'10). |

### 104.2.26 PCI Express Capability Register 1 function *i* (EXPCAP1F*i*) *i* = 0, 1

**Header TYPE00**

- Device capability register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | FLRCAP | CAPSLPLSC | | Captured Slot Power Limit Value | | | | | | | | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBER | — | — | — | EL1ACLAT | | | EL0ACLAT | | | ETAGS | PFS | | MPSS | | |
| Initial value: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 | FLRCAP | 0 | R | R/W | Function Level Reset Capability<br>Indicates that the function level reset function is supported.<br>• Endpoints only.<br>For all other Function types this bit doesn't exist. |
| 27, 26 | CAPSLPLSC | B'00 | R | R | Captured Slot Power Limit Scale<br>Indicates the slot power limit value scale.<br>B'00: 1.0x<br>B'01: 0.1x<br>B'10: 0.01x<br>B'11: 0.001x |
| 25 to 18 | Captured Slot Power Limit Value | H'00 | R | R | Captured Slot Power Limit Value<br>In combination with the Captured Slot Power Limit Scale value, specifies the upper limit on power available to the adapter. |
| 17, 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | RBER | 1 | R | R | Role-Based Error Reporting<br>When Set, this bit indicates that the Function implements the functionality originally defined in the Error Reporting ECN for PCI Express Base Specification, Revision 1.0a, and later incorporated into PCI Express Base Specification, Revision 1.1. |
| 14 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 11 to 9 | EL1ACLAT | B’000 | R | R/W | Endpoint L1 Acceptable Latency<br>Indicates the maximum L1 to L0 transition latency that the device can accept.<br>B’000: L0 recovery must be completed within 1 µs.<br>B’001: L0 recovery must be completed within 2 µs.<br>B’010: L0 recovery must be completed within 4 µs.<br>B’011: L0 recovery must be completed within 8 µs.<br>B’100: L0 recovery must be completed within 16 µs.<br>B’101: L0 recovery must be completed within 32 µs.<br>B’110: L0 recovery must be completed within 64 µs.<br>B’111: No time limit<br>Set the device support status at initialization. |
| 8 to 6 | EL0ACLAT | B’000 | R | R/W | Endpoint L0s Acceptable Latency<br>Indicates the maximum L0s to L0 transition latency that the device can accept.<br>B’000: L0 recovery must be completed within 64 ns.<br>B’001: L0 recovery must be completed within 128 ns.<br>B’010: L0 recovery must be completed within 256 ns.<br>B’011: L0 recovery must be completed within 512 ns.<br>B’100: L0 recovery must be completed within 1 µs.<br>B’101: L0 recovery must be completed within 2 µs.<br>B’110: L0 recovery must be completed within 4 µs.<br>B’111: No time limit |
| 5 | ETAGS | 1 | R | R | Extended Tag Field Supported<br>Indicates the tag ID size supported as the requester ID.<br>0: 5-bit tag<br>1: 8-bit extension tag |
| 4, 3 | PFS | B’00 | R | R | Phantom Function Supported<br>Indicates the phantom function support status for transaction ID extension.<br>B’00: The phantom function is not supported.<br>B’01: The MSB of requester ID function number can be used for the phantom function.<br>B’10: The upper two bits of requester ID function number can be used for the phantom function.<br>B’11: All the three bits of requester ID function number can be used for the phantom function. |
| 2 to 0 | MPSS | B’001 | R | R | Max Payload Size Supported<br>Indicates the maximum payload size supported by the device.<br>B’000: 128 bytes<br>B’001: 256 bytes<br>B’010: 512 bytes<br>B’011: 1024 bytes<br>B’100: 2048 bytes<br>B’101: 4096 bytes<br>B’110: Reserved<br>B’111: Reserved |

### Header TYPE01

- Device capability register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RBER | — | — | — | — | — | — | — | — | — | ETAGS | PFS | | MPSS | | |
| Initial value: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | RBER | 1 | R | R | Role-Based Error Reporting<br>When Set, this bit indicates that the Function implements the functionality originally defined in the Error Reporting ECN for PCI Express Base Specification, Revision 1.0a, and later incorporated into PCI Express Base Specification, Revision 1.1. |
| 14 to 6 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 | ETAGS | 1 | R | R | Extended Tag Field Supported<br>Indicates the tag ID size supported as the requester ID.<br>0: 5-bit tag<br>1: 8-bit extension tag |
| 4, 3 | PFS | B’00 | R | R | Phantom Function Supported<br>Indicates the phantom function support status for transaction ID extension.<br>B’00: The phantom function is not supported.<br>B’01: The MSB of requester ID function number can be used for the phantom function.<br>B’10: The upper two bits of requester ID function number can be used for the phantom function.<br>B’11: All the three bits of requester ID function number can be used for the phantom function. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 2 to 0 | MPSS | B’000 | R | R/W | Max Payload Size Supported<br>Indicates the maximum payload size supported by the device.<br>B’000: 128 bytes<br>B’001: 256 bytes<br>B’010: 512 bytes<br>B’011: 1024 bytes<br>B’100: 2048 bytes<br>B’101: 4096 bytes<br>B’110: Reserved<br>B’111: Reserved |

### 104.2.27 PCI Express Capability Register 2 function *i* (EXPCAP2F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Device status register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | TRPD | AUXPDTCD | URDTCD | FEDTCD | NFEDTCD | CEDTCD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C |

- Device control register

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PCIF | Max Read Request Size | | | ENSNP | AUXPPME | PFE | ETAGE | Max Payload Size | | | ERLOD | URRPE | FERPE | NFERPE | CERPE |
| Initial value: | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | *1 | RWS | *2 | *1 | R/W | R/W | R/W | *1 | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Note: *1: R/W changes by the TYPE. TYPE00:R/W, TYPE01:-

*2: R/W changes by the TYPE. TYPE00:R, TYPE01:-

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 22 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 21 | TRPD | 0 | R | R | Transaction Pending<br>Indicates that there is a non-posted request not yet completed.<br>・ Endpoints:<br>When Set, this bit indicates that the Function has issued Non-Posted Requests that have not been completed.<br>・ Root and Switch Ports:<br>When Set, this bit indicates that a Port has issued Non-Posted Requests on its own behalf (using the Port's own Requester ID) which have not been completed. |
| 20 | AUXPDTCD | 0 | R | R | AUX Power Detected<br>Functions that require Aux power report this bit as Set if Aux power is detected by the Function. |
| 19 | URDTCD | 0 | RW1C | RW1C | Unsupported Request Detected<br>Indicates the unsupported request detection status.<br>0: An unsupported request has not been detected.<br>1: An unsupported request has been detected.<br>This field is set to 1 when an unsupported request is detected. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 18 | FEDTCD | 0 | RW1C | RW1C | Fatal Error Detected Status<br>Indicates the fatal error detection status.<br>0: A fatal error has not been detected.<br>1: A fatal error has been detected.<br>This field is set to 1 when a fatal error is detected. |
| 17 | NFEDTCD | 0 | RW1C | RW1C | Non-fatal Error Detected<br>Indicates the non-fatal error detection status.<br>0: A non-fatal error has not been detected.<br>1: A non-fatal error has been detected.<br>This field is set to 1 when a non-fatal error is detected. |
| 16 | CEDTCD | 0 | RW1C | RW1C | Correctable Error Detected<br>Indicates the correctable error detection status.<br>0: A correctable error has not been detected.<br>1: A correctable error has been detected.<br>This field is set to 1 when a correctable error is detected. |
| 15 | PCIF | 0 | R/W | R/W | Initiate Function Level Reset (applicable to Endpoints with Function Level Reset Capability set to 1b).<br>A write of 1b initiates Function Level Reset to the Function. The value read by software from this bit is always 0b. |
| 14 to 12 | Max Read Request Size | B’010 | R/W | R/W | Max Read Request Size<br>Indicates the maximum size of the read request that can be issued as the requester.<br>B’000: 128 bytes max.<br>B’001: 256 bytes max.<br>B’010: Reserved<br>B’011: Reserved<br>B’100: Reserved<br>B’101: Reserved<br>B’110: Reserved<br>B’111: Reserved |
| 11 | ENSNP | 1 | *1 | R/W | Enable No Snoop<br>Enables the device to issue the no-snoop transaction.<br>0: The no-snoop transaction cannot be issued.<br>1: The no-snoop transaction can be issued. |
| 10 | AUXPPME | 0 | RWS | R/W | AUX Power PM Enable<br>Enables AUX power power management.<br>0: AUX power power management is disabled.<br>1: AUX power power management is enabled. |
| 9 | PFE | 0 | *2 | R/W | Phantom Function Enable<br>Enables the phantom function.<br>0: The phantom function is disabled.<br>1: The phantom function is enabled. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 8 | ETAGE | 0 | *1 | R/W | Extended Tag Field Enable<br>This bit, in combination with the 10-Bit Tag Requester Enable bit in the Device Control 2 register, determines how many Tag field bits a Requester is permitted to use.<br>The following applies when the 10-Bit Tag Requester Enable bit is Clear.<br>If the Extended Tag Field Enable bit is Set, the Function is permitted to use an 8-bit Tag field as a Requester. |
| 7 to 5 | Max Payload Size | B’000 | R/W | R/W | Max Payload Size<br>Indicates the maximum payload size for the device.<br>B’000: 128 bytes max.<br>B’001: 256 bytes max.<br>B’010: 512 bytes max.<br>B’011: 1024 bytes max.<br>B’100: 2048 bytes max.<br>B’101: 4096 bytes max.<br>B’110: Reserved<br>B’111: Reserved |
| 4 | ERLOD | 1 | *1 | R/W | Enabled Relax Ordering<br>Enables the device to issue the relaxed ordering transaction.<br>0: The relaxed ordering transaction cannot be issued.<br>1: The relaxed ordering transaction can be issued. |
| 3 | URRPE | 0 | R/W | R/W | Unsupported Request Reporting Enable<br>Enables the error message report on detection of an unsupported request.<br>0: The FATAL_ERR or NONFATAL_ERR message is not transmitted on detection of an unsupported request.<br>1: The FATAL_ERR or NONFATAL_ERR message is transmitted on detection of an unsupported request. |
| 2 | FERPE | 0 | R/W | R/W | Fatal Error Reporting Enable<br>Enables the error message report on detection of a FATAL_ERR.<br>0: The ERR_FATAL message is not transmitted on detection of a FATAL_ERR.<br>1: The ERR_FATAL message is transmitted on detection of a FATAL_ERR. |
| 1 | NFERPE | 0 | R/W | R/W | Non-fatal Error Reporting Enable<br>Enables the error message report on detection of a NON_FATAL_ERR.<br>0: The ERR_NONFATAL message is not transmitted on detection of a NON FATAL_ERR.<br>1: The ERR_NONFATAL message is transmitted on detection of a NON FATAL_ERR. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | CERPE | 0 | R/W | R/W | Correctable Error Reporting Enable<br>Enables the error message report on detection of a correctable error.<br>0: The ERR_COR message is not transmitted on detection of a correctable error.<br>1: The ERR_COR message is transmitted on detection of a correctable error. |

### 104.2.28 PCI Express Capability Register 3 function *i* (EXPCAP3F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Link capability register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Port Number | | | | | | | | — | PCAOC | LKBWNOTFCAP | DLLACTRPCAP | SDNERPCAP | CLKPM | L1ELAT | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | L1ELAT | L0s Exit Latency | | | ASPM Supported | | Maximum Link Width | | | | | | Supported Link Speeds | | | |
| Initial value: | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | Port Number | All 0 | R | R | Port Number<br>Indicates the port number of this PCI Express port.<br>Specify the port number at initialization. |
| 23 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | PCAOC | 0 | R | R | ASPM Optionality Compliance<br>This bit must be set to 1b in all Functions.<br>Components implemented against certain earlier versions of this specification will have this bit set to 0b. |
| 21 | LKBWNOTFCAP | 1 | R | R | Link Bandwidth Notification Capability<br>A value of 1b indicates support for the Link Bandwidth Notification status and interrupt mechanisms. |
| 20 | DLLACTRPCAP | 1 | R | R | Data Link Layer Active Reporting Capable<br>For a Downstream Port, this bit must be hardwired to 1b if the component supports the optional capability of reporting the DL_Active state of the Data Link Control and Management State Machine.<br>For a hot-plug capable Downstream Port (as indicated by the Hot-Plug Capable bit of the Slot Capabilities register) or a Downstream Port that supports Link speeds greater than 5.0 GT/s, this bit must be hardwired to 1b.<br>For Upstream Ports and components that do not support this optional capability, this bit must be hardwired to 0b. |
| 19 | SDNERPCAP | 1 | R | R | Surprise Down Error Reporting Capable<br>For a Downstream Port, this bit must be Set if the component supports the optional capability of detecting and reporting a Surprise Down error condition.<br>For Upstream Ports and components that do not support this optional capability, this bit must be hardwired to 0b. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 18 | CLKPM | 0 | R | R | Clock Power Management<br>Indicates that the clock can be stopped when the link is in the L1, L2Ready, or L3Ready state.<br>0: Clock cannot be stopped.<br>1: Clock can be stopped.<br>For Downstream Ports, this bit must be hardwired to 0b. |
| 17 to 15 | L1ELAT | B’110 | R | R | L1 Exit Latency<br>Indicates the L1 to L0 transition latency of the device.<br>B’000: less than 1 µs.<br>B’001: 1 µs to less than 2 µs.<br>B’010: 2 µs to less than 4 µs.<br>B’011: 4 µs to less than 8 µs.<br>B’100: 8 µs to less than 16 µs.<br>B’101: 16 µs to less than 32 µs.<br>B’110: 32 µs - 64 µs.<br>B’111: More than 64 µs. |
| 14 to 12 | L0s Exit Latency | B’111 | R | R | L0s Exit Latency<br>Indicates the L0s to L0 transition latency of the device.<br>B’000: Less than 64 ns.<br>B’001: 64 ns to less than 128 ns.<br>B’010: 128 ns to less than 256 ns.<br>B’011: 256 ns to less than 512 ns.<br>B’100: 512 ns to less than 1 µs.<br>B’101: 1 µs to less than 2 µs.<br>B’110: 2 µs - 4 µs<br>B’111: More than 4 µs. |
| 11, 10 | ASPM Supported | B’01 | R | R | ASPM Supported<br>Indicates the ASPM support status.<br>B’00: No ASPM Support<br>B’01: L0s Supported<br>B’10: L1 Supported<br>B’11: L0s and L1 Supported |
| 9 to 4 | Maximum Link Width | H'01 | R | R | Maximum Link Width<br>Indicates the maximum link width.<br>H'00: Reserved<br>H'01: x1<br>H'02: x2<br>H'04: x4<br>H'08: x8<br>H'0C: x12<br>H'10: x16<br>H'20: x32 |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 3 to 0 | Supported Link Speeds | H'4 | R | R | Supported Link Speeds<br>Indicates the maximum link speed.<br>H'1 Supported Link Speeds Vector field bit 0<br>H'2 Supported Link Speeds Vector field bit 1<br>H'3 Supported Link Speeds Vector field bit 2<br>H'4 Supported Link Speeds Vector field bit 3<br>H'5 Supported Link Speeds Vector field bit 4<br>H'6 Supported Link Speeds Vector field bit 5<br>H'7 Supported Link Speeds Vector field bit 6 |

### 104.2.29 PCI Express Capability Register 4 function *i* (EXPCAP4F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Link status register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LKAUTOBWSTS | LKBWMNGSTS | DLLACT | SLCLKCFG | LKTR | — | Negotiated Link Width | | | | | | Current Link Speed | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | *1 | *1 | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | *2 | *2 | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

- Link control register

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PCDSC | | — | — | LKAUTOBWINTE | LKBWMNGINTE | HWAUTOWDIS | ECLKPM | EXTSYNC | CCLKCFG | RTRNLK | LKDIS | RCB | — | ASPM Control | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | *3 | *3 | R | R/W | R/W | R/W | R | R | R/W | R | R/W | R/W |
| Internal bus R/W | R | R | R | R | *4 | *4 | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W |

Note: *1: R/W changes by the state. Root Port:R, Endpoint :-

*2: R/W changes by the state. Root Port:RW1C, Endpoint :R

*3: R/W changes by the state. Root Port:-, Endpoint :R

*4: R/W changes by the state. Root Port:R/W, Endpoint :R

[Root Port mode]

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | LKAUTOBWSTS | 0 | R | RW1C | Link Autonomous Bandwidth Status<br>• Root Port:<br>When a change of link bandwidth due to intentional factor is detected, 1 is set. This bit is cleared to 0 when 1 is written. |
| 30 | LKBWMNGSTS | 0 | R | RW1C | Link Bandwidth Management Status<br>• Root Port:<br>When a change of link bandwidth due to reliability-related factor is detected, 1 is set. This bit is cleared to 0 when 1 is written. |
| 29 | DLLACT | 0 | R | R | Data Link Layer Active<br>Indicates that DLCMSM is in the DL_ACTIVE state.<br>0: Not in the DL_ACTIVE state<br>1: In the DL_ACTIVE state |
| 28 | SLCLKCFG | 0 | R | R/W | Slot Clock Configuration<br>Indicates that the same reference clock as that supplied to the connector by the platform is used.<br>0: The same reference clock is not used.<br>1: The same reference clock is used. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 27 | LKTR | 0 | R | R | Link Training<br>Indicates that link training by MAC LTSSM is in progress.<br>• Root Port:<br>Set to 1 when link training is in progress. |
| 26 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 25 to 20 | Negotiated Link Width | H'04 | R | R | Negotiated Link Width<br>Indicates the link width determined by negotiation between links.<br>H'01: x1<br>H'02: x2<br>H'04: x4<br>H'08: x8<br>H'0C: x12<br>H'10: x16<br>H'20: x32<br>This field value is undefined while a link is not established. |
| 19 to 16 | Current Link Speed | H'1 | R | R | Current Link Speed<br>Indicates the link speed determined by negotiation between links.<br>H’1 Supported Link Speeds Vector field bit 0<br>H’2 Supported Link Speeds Vector field bit 1<br>H’3 Supported Link Speeds Vector field bit 2<br>H’4 Supported Link Speeds Vector field bit 3<br>H’5 Supported Link Speeds Vector field bit 4<br>H’6 Supported Link Speeds Vector field bit 5<br>H’7 Supported Link Speeds Vector field bit 6 |
| 15, 14 | PCDSC | All 0 | R | R | DRS Signaling Control<br>H’0 DRS not Reported<br>H’1 DRS Interrupt Enabled<br>H’2 DRS to FRS Signaling Enabled<br>Fixed to 0. This function is not supported. |
| 13, 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | LKAUTOBWINTE | 0 | — | R/W | Link Autonomous Bandwidth Interrupt Enable<br>• Root Port:<br>When Set, this bit enables the generation of an interrupt to indicate that the Link Autonomous Bandwidth Status bit has been Set. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 10 | LKBWMNGINTE | 0 | — | R/W | Link Bandwidth Management Interrupt Enable<br>• Root Port:<br>When Set, this bit enables the generation of an interrupt to indicate that the Link Bandwidth Management Status bit has been Set. |
| 9 | HWAUTOWDIS | 0 | R | R | Hardware Autonomous Width Disable<br>Disables autonomous change of link width by hardware except for the cases of poor link reliability.<br>0: Autonomous change of link width by hardware is enabled.<br>1: Autonomous change of link width by hardware is disabled.<br>According to the standard, fixing this bit to 0 is allowed if hardware does not provide the autonomous link width changing mechanism.<br>Since this IP does not provide the autonomous link. |
| 8 | ECLKPM | 0 | R/W | R/W | Enable Clock Power Management<br>Enables the clock power management function. |
| 7 | EXTSYNC | 0 | R/W | R/W | Extended SYNC<br>Transmits the ordered set when the MAC LTSSM makes a transition from L0s or in the recovery state. |
| 6 | CCLKCFG | 0 | R/W | R/W | Common Clock Configuration<br>Indicates that the same clock is used by both ends of the link. |
| 5 | RTRNLK | 0 | R | R/W | Retrain Link<br>Directs link retraining.<br>Note: The read value is always 0.<br>• Root Port:<br>Write 1 to allow link retraining. |
| 4 | LKDIS | 0 | R | R/W | Link Disable<br>Disables the link.<br>• Root Port:<br>Write 1 to disable the link. |
| 3 | RCB | 0 | R/W | R/W | Read Completion Boundary<br>Indicates the read completion boundary.<br>0: 64-byte boundary<br>1: 128-byte boundary |
| 2 | Reserved | 0 | R | R | This bit is always read as 0. The write value should always be 0. |
| 1, 0 | ASPM Control | B’00 | R/W | R/W | ASPM Control<br>Sets the ASPM support level.<br>B’00: Disabled<br>B’01: L0s transition enabled<br>B’10: L1 transition enabled<br>B’11: L0s/L1 transition enabled |

[Endpoint mode]

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | LKAUTOBWSTS | 0 | R | R | Link Autonomous Bandwidth Status<br>• Endpoint:<br>Use the initial value. Not need to be changed. |
| 30 | LKBWMNGSTS | 0 | R | R | Link Bandwidth Management Status<br>• Endpoint:<br>Use the initial value. Not need to be changed. |
| 29 | DLLACT | 0 | R | R | Data Link Layer Active<br>Indicates that DLCMSM is in the DL_ACTIVE state.<br>0: Not in the DL_ACTIVE state<br>1: In the DL_ACTIVE state |
| 28 | SLCLKCFG | 0 | R | R/W | Slot Clock Configuration<br>Indicates that the same reference clock as that supplied to the connector by the platform is used.<br>0: The same reference clock is not used.<br>1: The same reference clock is used. |
| 27 | LKTR | 0 | R | R | Link Training<br>Indicates that link training by MAC LTSSM is in progress.<br>• Endpoint:<br>Fixed to 0. |
| 26 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 25 to 20 | Negotiated Link Width | H'04 | R | R | Negotiated Link Width<br>Indicates the link width determined by negotiation between links.<br>H'01: x1<br>H'02: x2<br>H'04: x4<br>H'08: x8<br>H'0C: x12<br>H'10: x16<br>H'20: x32<br>This field value is undefined while a link is not established. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 19 to 16 | Current Link Speed | H'1 | R | R | Current Link Speed<br>Indicates the link speed determined by negotiation between links.<br>H'1 Supported Link Speeds Vector field bit 0<br>H'2 Supported Link Speeds Vector field bit 1<br>H'3 Supported Link Speeds Vector field bit 2<br>H'4 Supported Link Speeds Vector field bit 3<br>H'5 Supported Link Speeds Vector field bit 4<br>H'6 Supported Link Speeds Vector field bit 5<br>H'7 Supported Link Speeds Vector field bit 6 |
| 15, 14 | PCDSC | All 0 | R | R | DRS Signaling Control<br>H'0 DRS not Reported<br>H'1 DRS Interrupt Enabled<br>H'2 DRS to FRS Signaling Enabled<br>Fixed to 0. This function is not supported. |
| 13, 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | LKAUTOBWINTE | 0 | R | R | Link Autonomous Bandwidth Interrupt Enable<br>• Endpoint:<br>Use the initial value. Not need to be changed. |
| 10 | LKBWMNGINTE | 0 | R | R | Link Bandwidth Management Interrupt Enable<br>• Endpoint:<br>Use the initial value. Not need to be changed. |
| 9 | HWAUTOWDIS | 0 | R | R | Hardware Autonomous Width Disable<br>Disables autonomous change of link width by hardware except for the cases of poor link reliability.<br>0: Autonomous change of link width by hardware is enabled.<br>1: Autonomous change of link width by hardware is disabled.<br>According to the standard, fixing this bit to 0 is allowed if hardware does not provide the autonomous link width changing mechanism.<br>Since this IP does not provide the autonomous link. |
| 8 | ECLKPM | 0 | R/W | R/W | Enable Clock Power Management<br>Enables the clock power management function. |
| 7 | EXTSYNC | 0 | R/W | R/W | Extended SYNC<br>Transmits the ordered set when the MAC LTSSM makes a transition from L0s or in the recovery state. |
| 6 | CCLKCFG | 0 | R/W | R/W | Common Clock Configuration<br>Indicates that the same clock is used by both ends of the link. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 5 | RTRNLK | 0 | R | R/W | Retrain Link<br>Directs link retraining.<br>Note: The read value is always 0.<br>• Endpoint:<br>This bit is reserved. The write value should always be 0. |
| 4 | LKDIS | 0 | R | R/W | Link Disable<br>Disables the link.<br>• Endpoint:<br>This bit is reserved. The write value should always be 0. |
| 3 | RCB | 0 | R/W | R/W | Read Completion Boundary<br>Indicates the read completion boundary.<br>0: 64-byte boundary<br>1: 128-byte boundary |
| 2 | Reserved | 0 | R | R | This bit is always read as 0. The write value should always be 0. |
| 1, 0 | ASPM Control | B’00 | R/W | R/W | ASPM Control<br>Sets the ASPM support level.<br>B’00: Disabled<br>B’01: L0s transition enabled<br>B’10: L1 transition enabled<br>B’11: L0s/L1 transition enabled |

### 104.2.30 PCI Express Capability Register 5 function *i* (EXPCAP5F*i*) *i* = 0

**Header TYPE01 Only**

- Slot capabilities register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Physical Slot Number | | | | | | | | | | | | | NOCCSP | EMLKPRS | SLPLSC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SLPLSC | Slot Power Limit Value | | | | | | | | HOTPLGCAP | HOTPLGSPR | PWINDPRS | ATINDPRS | MRLSENPRS | PWCONPRS | ATBTNPRS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 19 | Physical Slot Number | All 0 | R | Physical Slot Number<br>This field indicates the physical slot number attached to this Port. |
| 18 | NOCCSP | 0 | R | No Command Complete Support<br>Indicates that completion of a Hot Plug command is not to be reported.<br>0: Upon completion of a Hot Plug command, it is reported.<br>1: Upon completion of a Hot Plug command, it is not reported. |
| 17 | EMLKPRS | 0 | R | Electro Mechanical Interlock Present<br>Indicates that Electromechanical Interlock is provided for the slot.<br>This field is invalid with this IP. |
| 16, 15 | SLPLSC | B’00 | R | Slot Power Limit Scale<br>Indicates the scale for slot power limit value.<br>B’00: 1.0x<br>B’01: 0.1x<br>B’10: 0.01x<br>B’11: 0.001x |
| 14 to 7 | Slot Power Limit Value | All 0 | R | Slot Power Limit Value<br>Indicates the upper limit of slot power supplied by the slot power supply in Watts.<br>H’F0 = 250 W Slot Power Limit<br>H’F1 = 275 W Slot Power Limit<br>H’F2 = 300 W Slot Power Limit<br>H’F3 to FFh = Reserved for Slot Power Limit values above 300 W |
| 6 | HOTPLGCAP | 0 | R | Hot Plug Capable<br>Indicates the Hot Plug support status in this slot. |
| 5 | HOTPLGSPR | 0 | R | Hot Plug Surprise possible<br>Indicates that the adapter inserted to this slot can be unplugged without advance notice. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 4 | PWINDPRS | 0 | R | Power Indicator Present<br>Indicates that the power indicator is provided. |
| 3 | ATINDPRS | 0 | R | Attention Indicator Present<br>Indicates that the attention indicator is provided. |
| 2 | MRLSENPRS | 0 | R | MRL Sensor Present<br>Indicates that the MRL sensor is provided. |
| 1 | PWCONPRS | 0 | R | Power Controller Present<br>Indicates that the power controller is provided. |
| 0 | ATBTNPRS | 0 | R | Attention Button Present<br>Indicates that the attention button is provided. |

### 104.2.31 PCI Express Capability Register 6 function *i* (EXPCAP6F*i*) *i* = 0

**Header TYPE01 only**

- Slot status register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | DLLSTCHG | EMLKSTS | PRDETST | MRLSENST | COMCPL | PRDETCHG | MRLSENCHG | PWFALDET | ATBTNPRES |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | RW1C | R | R | R | RW1C | RW1C | RW1C | RW1C | RW1C |

- Slot control register

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DLLSTCHGE | EMLKCON | PWCONCON | PWINDCON | | ATINDCON | | HOTPLGINTE | COMCPLINTE | PRDETCHGE | MRLSENCHGE | PWFALDETE | ATBTNPRESE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

/W

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 24 | DLLSTCHG | 0 | RW1C | Data Link Layer State Changed<br>This bit is Set when the value reported in the Data Link Layer Link Active bit of the Link Status register is changed |
| 23 | EMLKSTS | 0 | R | Electromechanical Interlock Status<br>Indicates the Electromechanical Interlock state.<br>0: Not locked by Electromechanical Interlock.<br>1: Locked by Electromechanical Interlock |
| 22 | PRDETST | 0 | R | Presence Detect State<br>Indicates that the adapter is inserted in the slot.<br>0: The slot is empty.<br>1: The adapter is inserted in the slot. |
| 21 | MRLSENST | 0 | R | MRL Sensor State<br>Indicates the MRL sensor state.<br>0: MRL Closed.<br>1: MRL Open |
| 20 | COMCPL | 0 | RW1C | Command Completed<br>Indicates that the Hot Plug command has been completed.<br>0: The Hot Plug command is being executed.<br>1: The Hot Plug command execution has been completed. |
| 19 | PRDETCHG | 0 | RW1C | Presence Detect Changed<br>This bit is Set when the value reported in the Presence Detect State bit is changed. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 18 | MRLSENCHG | 0 | RW1C | MRL Sensor Changed<br>If an MRL sensor is implemented, this bit is Set when a MRL Sensor state change is detected. If an MRL sensor is not implemented, this bit must not be Set. |
| 17 | PWFALDET | 0 | RW1C | Power Fault Detect<br>Indicates that the power fault has been detected. |
| 16 | ATBTNPRES | 0 | RW1C | Attention Button Pressed<br>Indicates that the attention button has been pressed. |
| 15 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 | DLLSTCHGE | 0 | RW | Data Link Layer State Change Enable<br>If the Data Link Layer Link Active Reporting capability is 1b, this bit enables software notification when Data Link Layer Link Active bit is changed. |
| 11 | EMLKCON | 0 | R | Electro Mechanical Interlock Control<br>If an Electromechanical Interlock is implemented, a write of 1b to this bit causes the state of the interlock to toggle.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 10 | PWCONCON | 0 | RW | Power Controller Control<br>If a Power Controller is implemented, this bit when written sets the power state of the slot per the defined encodings.<br>0: Power On<br>1: Power Off |
| 9, 8 | PWINDCON | B’11 | RW | Power Indicator Control<br>If a Power Indicator is implemented, writes to this field set the Power Indicator to the written state.<br>B’00: Reserved<br>B’01: On<br>B’10: Blink<br>B’11: Off |
| 7, 6 | ATINDCON | B’11 | RW | Attention Indicator Control<br>If an Attention Indicator is implemented, writes to this field set the Attention Indicator to the written state.<br>B’00: Reserved<br>B’01: On<br>B’10: Blink<br>B’11: Off |
| 5 | HOTPLGINTE | 0 | RW | Hot Plug Interrupt Enable<br>When Set, this bit enables generation of an interrupt on enabled hot-plug events. |
| 4 | COMCPLINTE | 0 | RW | Command Completed Interrupt Enable<br>If Command Completed notification is supported (if the No Command Completed Support bit in the Slot Capabilities register is 0b), when Set, this bit enables software notification when a hot-plug command is completed by the Hot-Plug Controller. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 3 | PRDETCHGE | 0 | RW | Presence Detect Changed Enable<br>When Set, this bit enables software notification on a presence detect changed event. |
| 2 | MRLSENCHGE | 0 | RW | MRL Sensor Changed Enable<br>When Set, this bit enables software notification on a MRL sensor changed event |
| 1 | PWFALDETE | 0 | RW | Power Fault Detect Enable<br>When Set, this bit enables software notification on a power fault event. |
| 0 | ATBTNPRESE | 0 | RW | Attention Button Pressed Enable<br>When Set to 1b, this bit enables software notification on an attention button pressed event. |

### 104.2.32 PCI Express Capability Register 7 function *i* (EXPCAP7F*i*) *i* = 0

**Header TYPE01 only**

- Root capabilities register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CRSVIS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

- Root control register

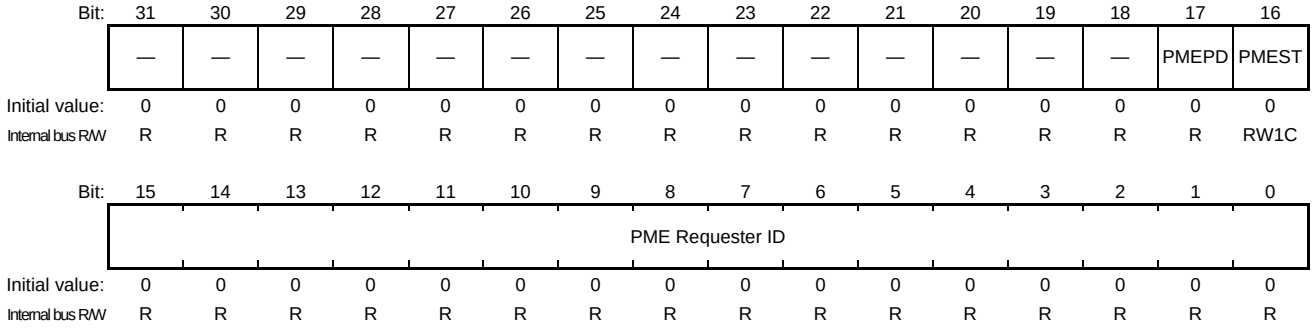
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | CRSVISE | PMEINTE | SERRFEE | SERRNFEE | SERRCEE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | CRSVIS | 0 | R/W | CRS Software Visibility<br>Indicates whether Root Port has the function of returning Configuration Retry Status to the upper software. |
| 15 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | CRSVISE | 0 | R/W | CRS Software Visibility Enable<br>When Set, this bit enables the Root Port to return Configuration Request Retry Status (CRS) Completion Status to software. |
| 3 | PMEINTE | 0 | R/W | PME Interrupt Enable<br>When Set, this bit enables PME interrupt generation upon receipt of a PME Message as reflected in the PME Status bit. |
| 2 | SERRFEE | 0 | R/W | System Error on Fatal Error Enable<br>Enables Root Port to generate the interrupt to the upper software upon reception of an ERR_FATAL message or occurrence of an error reported by Root Port itself. |
| 1 | SERRNFEE | 0 | R/W | System Error on Non-fatal Error Enable<br>Enables Root Port to generate the interrupt to the upper software upon reception of an ERR_NONFATAL message or occurrence of an error reported by Root Port itself. |
| 0 | SERRCEE | 0 | R/W | System Error on Correctable Error Enable<br>Enables Root Port to generate the interrupt to the upper software upon reception of an ERR_COR message or occurrence of an error reported by Root Port itself. |

### 104.2.33 PCI Express Capability Register 8 function *i* (EXPCAP8F*i*) *i* = 0

**Header TYPE01 only**

- Root status register




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17 | PMEPD | 0 | R | PME Pending<br>This bit indicates that another PME is pending when the PME Status bit is Set. |
| 16 | PMEST | 0 | RW1C | PME Status<br>This bit indicates that PME was asserted by the PME Requester indicated in the PME Requester ID field. |
| 15 to 0 | PME Requester ID | All 0 | R | PME Requester ID<br>Indicates the requester ID having issued the PME.<br>This field sets the requester ID on reception of PME. |

### 104.2.34 PCI Express Capability Register 9 function *i* (EXPCAP9F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Device capabilities 2 register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PCFS | — | — | — | — | — | — | — | PCCMETP | | PCCE TPS | PCC EFS | PCOS | | PC10 BTRS | PC10 BTCS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PCLSC | | PCTCS | | PCLS | PCNR EPP | PC12 8CCS | PC64A CS | PC32A CS | PCARS | ARIFWD SUP | CPLTOD ISSUP | Completion Timeout Range Supported | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | PCFS | 0 | R | R | FRS Supported<br>When Set, indicates support for the optional Function Readiness Status (FRS) capability. |
| 30 to 24 | — | All 0 | R | R | Reserved<br>These bits are always read as 0.The write value should always be 0. |
| 23, 22 | PCCMETP | H'0 | R | R | Max Number of End-End TLP Prefixes Supported.<br>Indicates the maximum number of End-End TLP Prefixes supported by this Function.<br>H'1 1 End-End TLP Prefix<br>H'2 2 End-End TLP Prefixes<br>H'3 3 End-End TLP Prefixes<br>H'0 4 End-End TLP Prefixes |
| 21 | PCCETPS | 0 | R | R | End-End TLP Prefix Supported<br>Indicates whether End-End TLP Prefix support is offered by a Function.<br>0 : No Support<br>1 : Support is provided to receive TLPs containing End-End TLP Prefixes. |
| 20 | PCCEFS | 0 | R | R | Extended Fmt Field Supported<br>If Set, the Function supports the 3-bit definition of the Fmt field. If Clear, the Function supports a 2-bit definition of the Fmt field.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 19, 18 | PCOS | H'0 | R | R | OBFF Supported<br>This field indicates if OBFF is supported and, if so, what signaling mechanism is used.<br>H'0 OBFF Not Supported<br>H'1 OBFF supported using Message signaling only<br>H'2 OBFF supported using WAKE# signaling only<br>H'3 OBFF supported using WAKE# and Message signaling<br>Root Ports, Switch Ports, and Endpoints are permitted to implement this capability. |
| 17 | PC10BTRS | 0 | R | R | 10-Bit Tag Requester Supported<br>If this bit is Set, the Function supports 10-Bit Tag Requester capability. |
| 16 | PC10BTCS | 0 | R | R | 10-Bit Tag Completer Supported<br>If this bit is Set, the Function supports 10-Bit Tag Completer capability. |
| 15, 14 | PCLSC | H'0 | R | R | LN System CLS<br>Applicable only to Root Ports and RCRBs<br>H'0 LN Completer either not supported or not in effect<br>H'1 LN Completer with 64-byte cachelines in effect<br>H'2 LN Completer with 128-byte cachelines in effect<br>H'3 Reserved |
| 13, 12 | PCTCS | H'0 | R | R | TPH Completer Supported<br>Value indicates Completer support for TPH or Extended TPH. Applicable only to Root Ports and Endpoints.<br>H'0 : TPH and Extended TPH Completer not supported.<br>H'1 : TPH Completer supported; Extended TPH Completer not supported.<br>H'2 : Reserved.<br>H'3 : Both TPH and Extended TPH Completer supported. |
| 11 | PCLS | 0 | R | R | LTR Mechanism Supported<br>A value of 1b indicates support for the optional Latency Tolerance Reporting (LTR) mechanism.<br>Root Ports, Switches and Endpoints are permitted to implement this capability. |
| 10 | PCNREPP | 0 | R | R | No RO-enabled PR-PR Passing<br>This bit applies only for Switches and RCs that support peer-to-peer traffic between Root Ports. |
| 9 | PC128CCS | 0 | R | R | 128-bit CAS Completer Supported<br>Applicable to Functions with Memory Space BARs as well as all Root Ports. |
| 8 | PC64ACS | 0 | R | R | 64-bit AtomicOp Completer Supported<br>Applicable to Functions with Memory Space BARs as well as all Root Ports. |
| 7 | PC32ACS | 0 | R | R | 32-bit AtomicOp Completer Supported<br>Applicable to Functions with Memory Space BARs as well as all Root Ports. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 6 | PCARS | 0 | R | R | AtomicOp Routing Supported<br>Applicable only to Switch Upstream Ports, Switch Downstream Ports, and Root Ports; must be 0b for other Function types. |
| 5 | ARIFWDSUP | 0 | R | R | ARI Forwarding Support<br>Indicates the ARI Forwarding support status.<br>• Root Port:<br>This IP does not support the ARI Forwarding.<br>Fixed to 0.<br>• Endpoint:<br>Fixed to 0. |
| 4 | CPLTODISSUP | 1 | R | R | Completion Timeout Disable Supported<br>Indicates the Completion Timeout Disable function support status.<br>0: The Completion Timeout Disable function is not supported.<br>1: The Completion Timeout Disable function is supported. |
| 3 to 0 | Completion Timeout Range Supported | H'0 | R | R | Completion Timeout Range Supported<br>Indicates the Completion Timeout Range support status.<br>Timeout range specified by the standard:<br>Range A: 50 µs to 10 ms<br>Range B: 10 ms to 250 ms<br>Range C: 250 ms to 4 s<br>Range D: 4 s to 64 s<br>Supported ranges:<br>H'0: Setting the timeout value is prohibited (fixed).<br>H'1: Range A<br>H'2: Range B<br>H'3: Range A and B<br>H'6: Range B and C<br>H'7: Range A, B and C<br>H'E: Range B, C and D<br>H'F: Range A, B, C and D<br>Other than above: Reserved<br>This IP does not support the timeout value setting. |

### 104.2.35 PCI Express Capability Register 10 function *i* (EXPCAP10F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Device status 2 register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

- Device control 2 register

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PCCETPB | PCOE | | PC10BTRE | — | PCLE | PCICE | PCIRE | PCAEB | PCARE | ARIFWDE | CPLTODIS | Completion Timeout Value | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R/W* | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R/W* | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0.The write value should always be 0. |
| 15 | PCCETPB | 0 | R | R | End-End TLP Prefix Blocking<br>Controls whether the routing function is permitted to forward TLPs containing an End-End TLP Prefix.<br>0 : Forwarding Enabled – Function is permitted to send TLPs with End-End TLP Prefixes.<br>1 : Forwarding Blocked – Function is not permitted to send TLPs with End-End TLP Prefixes. |
| 14, 13 | PCOE | H'0 | R | R | OBFF Enable<br>This field enables the OBFF mechanism and selects the signaling method.<br>H'0 Disabled<br>H'1 Enabled using Message signaling [Variation A]<br>H'2 Enabled using Message signaling [Variation B]<br>H'3 Enabled using WAKE# signaling<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 12 | PC10BTRE | 0 | R | R | 10-Bit Tag Requester Enable<br>This bit, in combination with the Extended Tag Field Enable bit in the Device Control register, determines how many Tag field bits a Requester is permitted to use.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 11 | — | 0 | R | R | Reserved<br>This bit si always read as 0.The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 10 | PCLE | 0 | R/W* | R/W* | LTR Mechanism Enable<br>When Set to 1b, this bit enables Upstream Ports to send LTR messages and Downstream Ports to process LTR Messages.<br>*Note*: RW for function #0 and RsdvP for all other functions. |
| 9 | PCICE | 0 | R | R | IDO Completion Enable<br>If this bit is Set, the Function is permitted to set the ID-Based Ordering (IDO) bit (Attr[2]) of Completions it returns<br>Endpoints, including RC Integrated Endpoints, and Root Ports are permitted to implement this capability.<br>Note: IDO is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 8 | PCIRE | 0 | R | R | IDO Request Enable<br>If this bit is Set, the Function is permitted to set the ID-Based Ordering (IDO) bit (Attr[2]) of Requests it initiates.<br>Note: IDO is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 7 | PCAEB | 0 | R | R | AtomicOp Egress Blocking<br>Applicable and mandatory for Switch Upstream Ports, Switch Downstream Ports, and Root Ports that implement AtomicOp routing capability. |
| 6 | PCARE | 0 | R | R | AtomicOp Requester Enable<br>Applicable only to Endpoints and Root Ports<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 5 | ARIFWDE | 0 | R | R | ARI Forwarding Enable<br>When set, the Downstream Port disables its traditional Device Number field being 0 enforcement when turning a Type 1 Configuration Request into a Type 0 Configuration Request, permitting access to Extended Functions in an ARI Device immediately below the Port. |
| 4 | CPLTODIS | 0 | R/W | R/W | Completion Timeout Disable<br>Sets the Completion Timeout Disable function.<br>0: The Completion Timeout Disable function is disabled.<br>1: The Completion Timeout Disable function is enabled. |
| 3 to 0 | Completion Timeout Value | H'0 | RW | RW | Completion Timeout Value<br>Sets the Completion Timeout value.<br>H'0: Default Range 50 µs to 50 ms<br>H'1: 50 µs to 100 µs<br>H'2: 1 ms to 10 ms<br>H'5: 16 ms to 55 ms<br>H'6: 65 ms to 210 ms<br>H'9: 260 ms to 900 ms<br>H'A: 1 s to 3.5 s<br>H'C: 4 s to 13 s<br>H'D: 17 s to 64 s<br>Other than above: Reserved. |

### 104.2.36 PCI Express Capability Register 11 function *i* (EXPCAP11F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Link capabilities 2 register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | PCTRPDS | PCRPDS | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | PCCLS | PCSLSV | | | | | | | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 24 | PCTRPDS | 1 | R | R | Two Retimers Presence Detect Supported<br>When set to 1b, this bit indicates that the associated Port supports detection and reporting of two Retimers presence. |
| 23 | PCRPDS | 1 | R | R | Retimer Presence Detect Supported<br>When set to 1b, this bit indicates that the associated Port supports detection and reporting of Retimer presence. |
| 22 to 9 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | PCCLS | 0 | R | R | Cross Link Supported<br>When set to 1b, this bit indicates that the associated Port supports crosslinks. |
| 7 to 1 | PCSLSV | H'0 | R | R | Supported Link Speeds Vector<br>This field indicates the supported Link speed(s) of the associated Port.<br>H'0 : 2.5 GT/s<br>H'1 : 5.0 GT/s<br>H'2 : 8.0 GT/s<br>H'3 : 16.0 GT/s |
| 0 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

### 104.2.37 PCI Express Capability Register 12 function *i* (EXPCAP12F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Link status 2 register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | — | — | — | — | — | PCCR | | PCTRPD | PCRPD | PCLER | PCECP3 | PCECP2 | PCECP1 | PCEC | CURDEEMLEV |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | RW1C | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | RW1C | R | R | R | R | R |

- Link control 2 register

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPDEEM | | | | CMPSOS | ETMDFCMP | Transmit Margin | | | SELDEEM | HWATSPDIS | ENTCMP | Target Link Speed | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | *1 | *1 | RWS | RWS | RWS | R | R | RWS | RWS | RWS | RWS | RWS |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W |

Note: *1: R/W changes by the TYPE. TYPE00:RWS, TYPE01:-

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | — | 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 30 to 28 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27, 26 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25, 24 | PCCR | H'3 | R | R | Crosslink Resolution<br>This field indicates the state of the Crosslink negotiation.<br>H'0 Crosslink Resolution is not supported.<br>H'1 Crosslink negotiation resolved as an Upstream Port.<br>H'2 Crosslink negotiation resolved as a Downstream Port.<br>H'3 Crosslink negotiation is not completed. |
| 23 | PCTRPD | 0 | R | R | Two Retimers Presence Detected<br>When set to 1b, this bit indicates that two Retimers were present during the most recent Link negotiation. |
| 22 | PCRPD | 0 | R | R | Retimer Presence Detected<br>When set to 1b, this bit indicates that a Retimer was present during the most recent Link negotiation. |
| 21 | PCLER | 0 | RW1C | RW1C | Link Equalization Request 8.0 GT/s<br>This bit is Set by hardware to request the 8.0 GT/s Link equalization process to be performed on the Link. |
| 20 | PCECP3 | 0 | R | R | Equalization 8.0 GT/s Phase 3 Successful<br>When set to 1b, this bit indicates that Phase 3 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 19 | PCECP2 | 0 | R | R | Equalization 8.0 GT/s Phase 2 Successful<br>When set to 1b, this bit indicates that Phase 2 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. |
| 18 | PCECP1 | 0 | R | R | Equalization 8.0 GT/s Phase 1 Successful<br>When set to 1b, this bit indicates that Phase 1 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. |
| 17 | PCEC | 0 | R | R | cfg_ls2_eq_cmpl |
| 16 | CURDEEMLEV | 1 | R | R | Current De-emphasis Level<br>Indicates the current de-emphasis level during 5-GT/s link operation.<br>0: -6 dB<br>1: -3.5 dB<br>The value is undefined during 2.5-GT/s operation. |
| 15 to 12 | CMPDEEM | H'0 | R/W* | R/W* | Compliance Preset/De-emphasis<br>For 8.0 GT/s and higher Data Rate: This field sets the Transmitter Preset in Polling.<br>H'0: -6 dB<br>H'1: -3.5 dB<br>The value is undefined during 2.5-GT/s operation.<br>*Note: Only function 0 (i = 0) control the component of LINK behavior, In all other functions of that device, this bit is of type RsvdP. |
| 11 | CMPSOS | 0 | *1*2 | R/W*2 | Compliance SOS(Skip Ordered Sets transmission)<br>When this bit is 1, LTSSM periodically inserts the SKP ordered set between compliance patterns.<br>0: The SKP ordered set is not periodically inserted.<br>1: The SKP ordered set is periodically inserted.<br>This bit is applicable when the Link is operating at 2.5 GT/s or 5.0 GT/s data rates only.<br>*2Note: Only function 0 (i = 0) control the component of LINK behavior, In all other functions of that device, this bit is of type RsvdP. |
| 10 | ETMDFCMP | 0 | *1*2 | R/W*2 | Enter Modified Compliance<br>When this bit is 1, LTSSM transmits modified compliance patterns on transition to the Polling.Compliance state.<br>0: Normal compliance patterns are used.<br>1: Modified compliance patterns are used.<br>*2Note: Only function 0 (i = 0) control the component of LINK behavior, In all other functions of that device, this bit is of type RsvdP. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 9 to 7 | Transmit Margin | B'000 | R/W* | R/W* | Transmit Margin<br>Sets the signal level without de-emphasis.<br>B'000: Normal signal level<br>B'001: 800 to 1200 mV for full swing, 400 to 700 mV for reduced swing<br>B'010-(n-1): n is specified by PHY (3 ≤ n ≤ 7).<br>n-111: Reserved<br>Determine n according to the PHY specification.<br>Do not set the value exceeding the PHY specification.<br>*Note: Only function 0 (i = 0) control the component of LINK behavior, In all other functions of that device, this bit is of type RsvdP. |
| 6 | SELDEEM | 0 | R | R | Selectable De-emphasis<br>Indicates the de-emphasis level during 5-GT/s upstream component operation.<br>• Root Port:<br>Set a value appropriate for the PHY at initialization.<br>0: -6 dB<br>1: -3.5 dB<br>• Endpoint:<br>Fixed to 0. |
| 5 | HWATSPDIS | 0 | R | R | Hardware Autonomous Speed Disable<br>Disables autonomous change of link width by hardware except for the cases of poor reliability.<br>0: Autonomous change of link width by hardware is enabled.<br>1: Autonomous change of link width by hardware is disabled.<br>According to the standard, fixing this bit to 0 is allowed if hardware does not provide the autonomous link width changing mechanism. |
| 4 | ENTCMP | 0 | RWS* | R/W | Enter Compliance<br>Setting this bit of both ends of the link forcibly places a link into compliance mode at the specified speed.<br>Used to set compliance mode.<br>*Note: Only function 0 (i = 0) control the component of LINK behavior, In all other functions of that device, this bit is of type RsvdP. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 3 to 0 | Target Link Speed | B'0010 * | RWS* | R/W | Target Link Speed<br>The encoded value specifies a bit location in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the desired target Link speed.<br>B'0001 Supported Link Speeds Vector field bit 0<br>B'0010 Supported Link Speeds Vector field bit 1<br>B'0011 Supported Link Speeds Vector field bit 2<br>B'0100 Supported Link Speeds Vector field bit 3<br>B'0101 Supported Link Speeds Vector field bit 4<br>B'0110 Supported Link Speeds Vector field bit 5<br>B'0111 Supported Link Speeds Vector field bit 6<br>*Note: For a Multi-Function Device associated with an Upstream Port, the field in Function 0 (i = 0) is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of this PCIe, this field is of type RsvdP (intial value = b'0000, attribute RO). |

### 104.2.38 Advanced Error Reporting Capability Register 0 function *i* (AERCAP0F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Advanced Error Reporting Extended Capability Header

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NEXTOFFSET | | | | | | | | | | | | CAPVERSION | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CAPID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | NEXTOFFSET | H'148 | R | R | Next Capability Offset<br>This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. |
| 19 to 16 | CAPVERSION | H'2 | R | R | Capability Version<br>Indicates the version of the Capability structure present. |
| 15 to 0 | CAPID | H'1 | R | R | AER Extended Capability ID.<br>Indicates the nature and format of the Extended Capability. |

### 104.2.39 Advanced Error Reporting Capability Register 1 function *i* (AERCAP1F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Uncorrectable Error Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TPBES | AEBES | — | INTNALES | ACSVS | URES | ECRCES | MTLPES | RECOES | UNEXPCES |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | RW1C | RW1C | R | RW1C | RW1C | RW1C | RW1C | RW1C |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | RW1C | R | RW1C | RW1C | RW1C | RW1C | RW1C |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPLTAES | CMPLTTES | FCPRTCES | POISTES | — | — | — | — | — | — | SUPDWNES | DLPROTES | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | RW1C | RW1C | RW1C | RW1C | R | R | R | R | R | R | RW1C | RW1C | R | R | R | R |
| Internal bus R/W | RW1C | RW1C | RW1C | RW1C | R | R | R | R | R | R | RW1C | RW1C | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 | TPBES | 0 | R | R | TLP Prefix Blocked Error Status.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 24 | AEBES | 0 | R | R | AtomicOp Egress Blocked Status.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 23 | — | 0 | RW1C | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | INTNALES | 0 | RW1C | RW1C | Uncorrectable Internal Error Status. |
| 21 | ACSVS | 0 | R | R | ACS Violation Status.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 20 | URES | 0 | RW1C | RW1C | Unsupported Request Error Status |
| 19 | ECRCES | 0 | RW1C | RW1C | ECRC Error Status |
| 18 | MTLPES | 0 | RW1C | RW1C | Malformed TLP Status |
| 17 | RECOES | 0 | RW1C | RW1C | Receiver Overflow Status |
| 16 | UNEXPCES | 0 | RW1C | RW1C | Unexpected Completion Status |
| 15 | CMPLTAES | 0 | RW1C | RW1C | Completer Abort Status |
| 14 | CMPLTTES | 0 | RW1C | RW1C | Completion Timeout Status |
| 13 | FCPRTCES | 0 | RW1C | RW1C | Flow Control Protocol Error Status |
| 12 | POISTES | 0 | RW1C | RW1C | Poisoned TLP Status |
| 11 to 6 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 5 | SUPDWNES | 0 | RW1C | RW1C | Surprise Down Error Status |
| 4 | DLPROTES | 0 | RW1C | RW1C | Data Link Protocol Error Status |
| 3 to 0 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

### 104.2.40 Advanced Error Reporting Capability Register 2 function *i* (AERCAP2F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Uncorrectable Error Mask Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TPBEM | AEBEM | — | INTNALE | ACSVM | UREM | ECRCEM | MTLPEM | RECOEM | UNEXPCEM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPLTAEM | CMPLTTEM | FCPRTCEM | POISTEM | — | — | — | — | — | — | SUPDWNEM | DLPROTEM | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 | TPBEM | 0 | R | R | TLP Prefix Blocked Error Mask.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 24 | AEBEM | 0 | R | R | AtomicOp Egress Blocked Mask. This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 23 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | INTNALEM | 1 | R/W | R/W | Uncorrectable Internal Error Mask |
| 21 | ACSVM | 0 | R | R | ACS Violation Mask. This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 20 | UREM | 0 | R/W | R/W | Unsupported Request Error Mask |
| 19 | ECRCEM | 0 | R/W | R/W | ECRC Error Mask |
| 18 | MTLPEM | 0 | R/W | R/W | Malformed TLP Mask |
| 17 | RECOEM | 0 | R/W | R/W | Receiver Overflow Mask |
| 16 | UNEXPCEM | 0 | R/W | R/W | Unexpected Completion Mask |
| 15 | CMPLTAEM | 0 | R/W | R/W | Completer Abort Error Mask |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 14 | CMPLTTEM | 0 | R/W | R/W | Completion Timeout Error Mask |
| 13 | FCPRTCEM | 0 | R/W | R/W | Flow Control Protocol Error Mask |
| 12 | POISTEM | 0 | R/W | R/W | Poisoned TLP Mask |
| 11 to 6 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5 | SUPDWNEM | 0 | R | R | Surprise Down Error Mask.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 4 | DLPROTEM | 0 | R/W | R/W | Data Link Protocol Error Mask |
| 3 to 0 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

### 104.2.41 Advanced Error Reporting Capability Register 3 function *i* (AERCAP3F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Uncorrectable Error Severity Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | TPB ESV | AEB ESV | — | INTN ALE | ACS VSV | URE SV | ECRC ESV | MTLP ESV | REC OESV | UNEXP CESV |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMPLT AESV | CMPLT TESV | FCPRT CESV | POIS TESV | — | — | — | — | — | — | SUPDW NESV | DLPRO TESV | — | — | — | — |
| Initial value: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 | TPBESV | 0 | R | R | TLP Prefix Blocked Error Severity. This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 24 | AEBESV | 0 | R | R | AtomicOp Egress Blocked Severity. This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 23 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 22 | INTNALESV | 1 | R/W | R/W | Uncorrectable Internal Error Severity |
| 21 | ACSVSV | 0 | R | R | ACS Violation Severity.<br>This function is not supported. This bit is Reserved, always read as 0, the write value should always be 0. |
| 20 | URESV | 0 | R/W | R/W | Unsupported Request Error Severity |
| 19 | ECRCESV | 0 | R/W | R/W | ECRC Error Severity |
| 18 | MTLPESV | 1 | R/W | R/W | Malformed TLP Severity |
| 17 | RECOESV | 1 | R/W | R/W | Receiver Overflow Error Severity |
| 16 | UNEXPCESV | 0 | R/W | R/W | Unexpected Completion Error Severity |
| 15 | CMPLTAESV | 0 | R/W | R/W | Completer Abort Error Severity |
| 14 | CMPLTTESV | 0 | R/W | R/W | Completion Timeout Error Severity |
| 13 | FCPRTCESV | 1 | R/W | R/W | Flow Control Protocol Error Severity |
| 12 | POISTESV | 0 | R/W | R/W | Poisoned TLP Severity |
| 11 to 6 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5 | SUPDWNESV | 0 | R | R/W | Surprise Down Error Severity |
| 4 | DLPROTESV | 1 | R/W | R/W | Data Link Protocol Error Severity |
| 3 to 0 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

### 104.2.42 Advanced Error Reporting Capability Register 4 function *i* (AERCAP4F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Correctable Error Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HLOES | CIES | ANFES | RTTS | — | — | — | RNRS | BADLS | BADTS | — | — | — | — | — | RXERS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | RW1CS | RW1CS | RW1CS | RW1CS | R | R | R | RW1CS | RW1CS | RW1CS | R | R | R | R | R | RW1CS |
| Internal bus R/W | RW1CS | RW1CS | RW1CS | RW1CS | R | R | R | RW1CS | RW1CS | RW1CS | R | R | R | R | R | RW1CS |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | HLOES | 0 | RW1CS | RW1CS | Header Log Overflow Error Status |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 14 | CIES | 0 | RW1CS | RW1CS | Corrected Internal Error Status |
| 13 | ANFES | 0 | RW1CS | RW1CS | Advisory Non-Fatal Error Status |
| 12 | RTTS | 0 | RW1CS | RW1CS | Replay Timer Timeout Status |
| 11 to 9 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | RNRS | 0 | RW1CS | RW1CS | REPLAY_NUM Rollover Status |
| 7 | BADLS | 0 | RW1CS | RW1CS | Bad DLLP Status |
| 6 | BADTS | 0 | RW1CS | RW1CS | Bad TLP Status |
| 5 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | RXERS | 0 | RW1CS | RW1CS | Receiver Error Status |

### 104.2.43 Advanced Error Reporting Capability Register 5 function *i* (AERCAP5F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Correctable Error Mask Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HLOEM | CIEM | ANFEM | RTTM | — | — | — | RNRM | BADLM | BADTM | — | — | — | — | — | RXERM |
| Initial value: | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | HLOEM | 1 | R/W | R/W | Header Log Overflow Error Mask |
| 14 | CIEM | 1 | R/W | R/W | Corrected Internal Error Mask |
| 13 | ANFEM | 1 | R/W | R/W | Advisory Non-Fatal Error Mask |
| 12 | RTTM | 0 | R/W | R/W | Replay Timer Timeout Mask |
| 11 to 9 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | RNRM | 0 | R/W | R/W | REPLAY_NUM Rollover Mask |
| 7 | BADLM | 0 | R/W | R/W | Bad DLLP Mask |
| 6 | BADTM | 0 | R/W | R/W | Bad TLP Mask |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 5 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | RXERM | 0 | R/W | R/W | Receiver Error Mask |

### 104.2.44 Advanced Error Reporting Capability Register 6 function *i* (AERCAP6F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Advanced Error Capabilities and Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

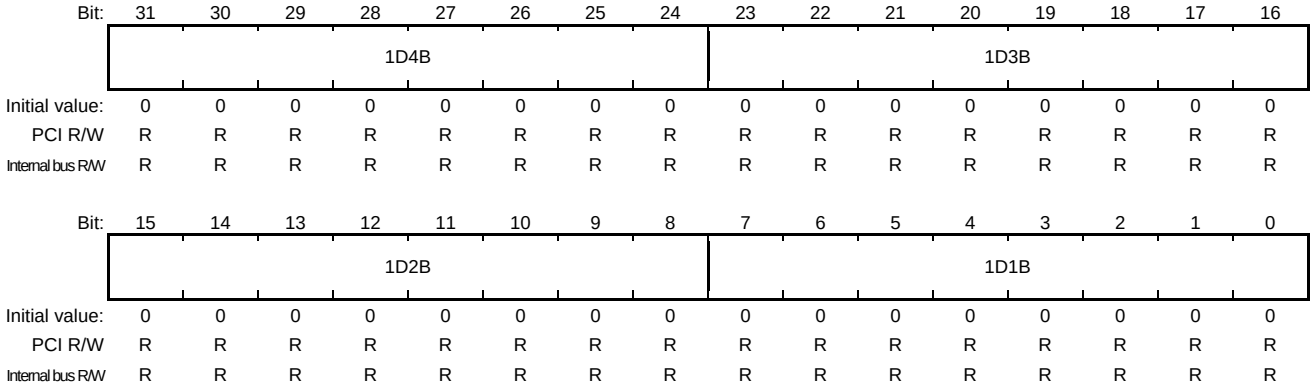
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | cfg_cto_pr | TPLP | MHE | MHC | ECRC_CHECK_EN | ECRC_CHECK_CAP | ECRC_GEN_EN | ECRC_GEN_CAP | FEP | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R/W | R | R/W | R | R/W | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R/W | R | R/W | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 13 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 | cfg_cto_prfx_hdr_log_cap | 0 | R | R | Completion Timeout Prefix/Header Log Capable bit. If set, this bit indicates that the function records the prefix/header of request TLPs that experience a completion timeout error. |
| 11 | TPLP | 0 | R | R | TLP Prefix Log Present |
| 10 | MHE | 0 | R/W | R | Multiple Header Recording Enable |
| 9 | MHC | 1 | R | R | Multiple Header Recording Capable |
| 8 | ECRC_CHECK_EN | 0 | R/W | R/W | ECRC Check Enable |
| 7 | ECRC_CHECK_CAP | 0 | R | R | ECRC Check Capable |
| 6 | ECRC_GEN_EN | 0 | R/W | R/W | ECRC Generation Enable |
| 5 | ECRC_GEN_CAP | 0 | R | R | ECRC Generation Capable |
| 4 to 0 | FEP | 0 | R | R | First Error Pointer |

### 104.2.45 Advanced Error Reporting Capability Register 7 function *i* (AERCAP7F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Header Log Register First

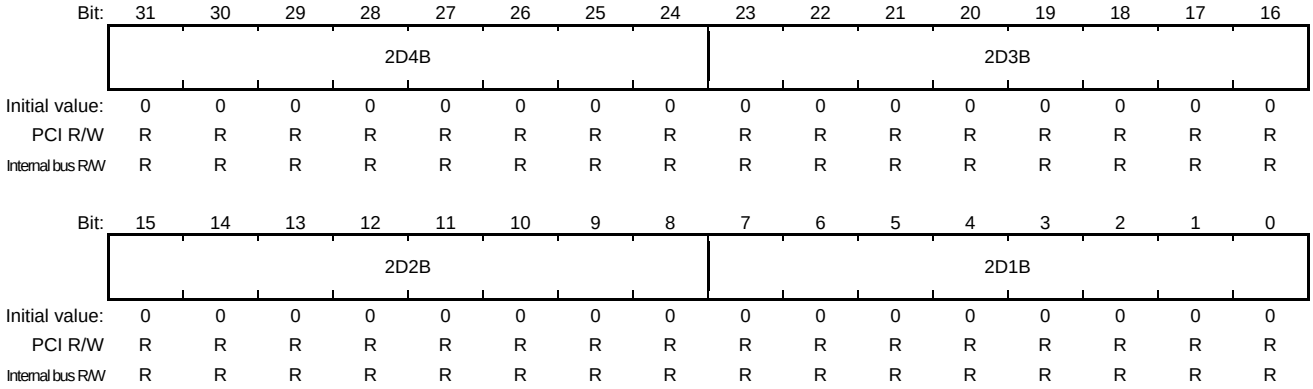



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | 1D4B | 0 | R | R | Byte 3 of Header log register of First 32 bit Data Word |
| 23 to 16 | 1D3B | 0 | R | R | Byte 2 of Header log register of First 32 bit Data Word |
| 15 to 8 | 1D2B | 0 | R | R | Byte 1 of Header log register of First 32 bit Data Word |
| 7 to 0 | 1D1B | 0 | R | R | Byte 0 of Header log register of First 32 bit Data Word |

### 104.2.46 Advanced Error Reporting Capability Register 8 function *i* (AERCAP8F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Header Log Register Second

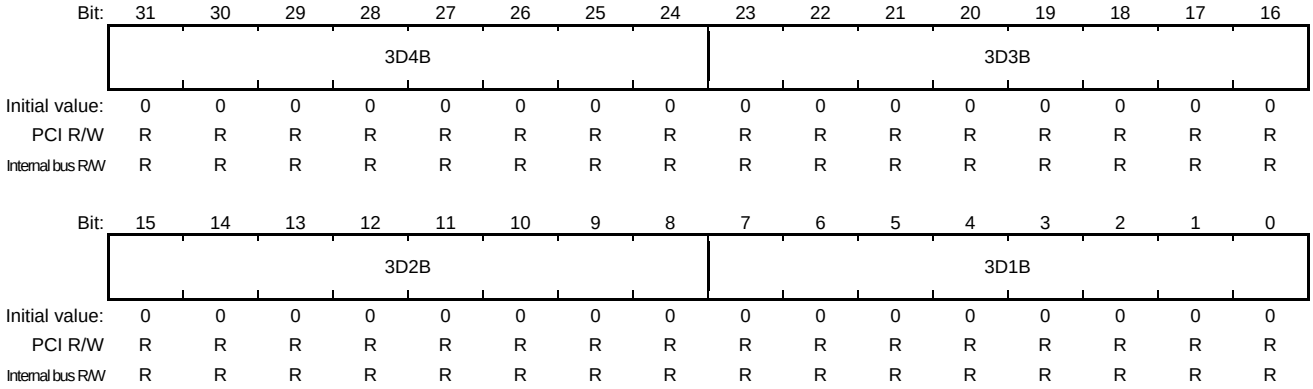



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | 2D4B | 0 | R | R | Byte 3 of Header log register of Second 32 bit Data Word |
| 23 to 16 | 2D3B | 0 | R | R | Byte 2 of Header log register of Second 32 bit Data Word |
| 15 to 8 | 2D2B | 0 | R | R | Byte 1 of Header log register of Second 32 bit Data Word |
| 7 to 0 | 2D1B | 0 | R | R | Byte 0 of Header log register of Second 32 bit Data Word |

### 104.2.47 Advanced Error Reporting Capability Register 9 function *i* (AERCAP9F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Header Log Register Third

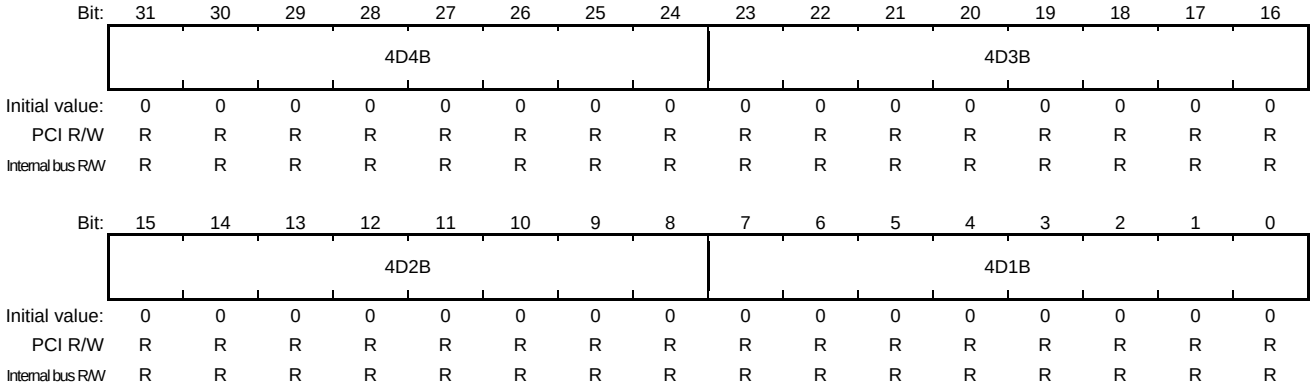



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | 3D4B | 0 | R | R | Byte 3 of Header log register of Third 32 bit Data Word |
| 23 to 16 | 3D3B | 0 | R | R | Byte 2 of Header log register of Third 32 bit Data Word |
| 15 to 8 | 3D2B | 0 | R | R | Byte 1 of Header log register of Third 32 bit Data Word |
| 7 to 0 | 3D1B | 0 | R | R | Byte 0 of Header log register of Third 32 bit Data Word |

### 104.2.48 Advanced Error Reporting Capability Register 10 function *i* (AERCAP10F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Header Log Register Fourth




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | 4D4B | 0 | R | R | Byte 3 of Header log register of Fourth 32 bit Data Word |
| 23 to 16 | 4D3B | 0 | R | R | Byte 2 of Header log register of Fourth 32 bit Data Word |
| 15 to 8 | 4D2B | 0 | R | R | Byte 1 of Header log register of Fourth 32 bit Data Word |
| 7 to 0 | 4D1B | 0 | R | R | Byte 0 of Header log register of Fourth 32 bit Data Word |

### 104.2.49 Advanced Error Reporting Capability Register 11 function *i* (AERCAP11F*i*) *i* = 0

**Header TYPE01 only**

- Root Error Command Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | FERE | NFERE | CERE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 3 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | FERE | 0 | R/W | R/W | Fatal Error Reporting Enable |
| 1 | NFERE | 0 | R/W | R/W | Non-Fatal Error Reporting Enable |
| 0 | CERE | 0 | R/W | R/W | Correctable Error Reporting Enable |

### 104.2.50 Advanced Error Reporting Capability Register 12 function *i* (AERCAP12F*i*) *i* = 0

**Header TYPE01 only**

- Root Error Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AEIMN | | | | | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | - | - | - | - | - | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | FEMR | NFEMR | FUF | MEFNF | EFNFR | MECR | ECR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 27 | AEIMN | 0 | — | R | Advanced Error Interrupt Message Number<br>This register indicates which MSI/MSI-X vector is used for the interrupt message generated in association with any of the status bits of this Capability. |
| 26 to 7 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 | FEMR | 0 | RW1C | RW1C | One or more Fatal Error Messages Received<br>Set when one or more Fatal Uncorrectable error Messages have been received. |
| 5 | NFEMR | 0 | RW1C | RW1C | One or more Non-Fatal Error Messages Received<br>Set when one or more Non-Fatal Uncorrectable error Messages have been received. |
| 4 | FUF | 0 | RW1C | RW1C | First Uncorrectable Error is Fatal<br>Set when the first Uncorrectable error Message received is for a Fatal error. |
| 3 | MEFNF | 0 | RW1C | RW1C | Multiple Fatal or Non-Fatal Errors Received<br>Set when either a Fatal or a Non-fatal error is received and ERR_FATAL/NONFATAL Received is already Set. |
| 2 | EFNFR | 0 | RW1C | RW1C | Fatal or Non-Fatal Error Received<br>Set when either a Fatal or a Non-fatal error Message is received and this bit is not already Set. |
| 1 | MECR | 0 | RW1C | RW1C | Multiple Correctable Errors Received<br>Set when a Correctable error Message is received and ERR_COR Received is already Set. |
| 0 | ECR | 0 | RW1C | RW1C | Correctable Error Received<br>Set when a Correctable error Message is received and this bit is not already Set. |

### 104.2.51 Advanced Error Reporting Capability Register 13 function *i* (AERCAP13F*i*) *i* = 0

**Header TYPE01 only**

- Error Source Identification Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFNFSI | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ECSI | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | EFNFSI | H'0 | R | R | Source of Fatal/Non-Fatal Error<br>Loaded with the Requester ID indicated in the received ERR_FATAL or ERR_NONFATAL Message when the ERR_FATAL/NONFATAL Received bit is not already set. |
| 15 to 0 | ECSI | H'0 | R | R | Source of Correctable Error<br>Loaded with the Requester ID indicated in the received ERR_COR Message when the ERR_COR Received bit is not already set. |

### 104.2.52 Device Serial Number Capability Register 0 function *i* (SERNUMCAP0F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Device serial number extended capability header

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Next Capability Offset | | | | | | | | | | | | Capability Version | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PCI Express Extended Capability ID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | Next Capability Offset | H'158 | R | R/W | Next Capability Offset<br>Pointer to the capability list. |
| 19 to 16 | Capability Version | H'1 | R | R | Capability Version<br>Indicates the version of this capability. |
| 15 to 0 | PCI Express Extended Capability ID | H'3 | R | R | PCI Express Extended Capability ID<br>Indicates the ID of this capability. |

### 104.2.53 Device Serial Number Capability Register 1 function *i* (SERNUMCAP1F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Serial Number Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Device Serial Number 1st | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Device Serial Number 1st | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | Device Serial Number 1st | All 0 | R | R | Device Serial Number 1st<br>Indicates the serial number of the device. |

### 104.2.54 Device Serial Number Capability Register 2 function *i* (SERNUMCAP2F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Serial Number Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Device Serial Number 2nd | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Device Serial Number 2nd | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | Device Serial Number 2nd | All 0 | R | R | Device Serial Number 2nd<br>Indicates the serial number of the device. |

### 104.2.55 Secondary PCI Express Extended Capability Register 0 function 0 (SEXPECAP0F0)

**Common to Header TYPE00/01**

- Secondary PCI Express Extended Capability Header

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NOFSET | | | | | | | | | | | | CVERSION | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EXTCI | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | NOFSET | H'178 | R | R | Next Capability Offset |
| 19 to 16 | CVERSION | H'1 | R | R | Capability Version<br>Indicates the version of the capability structure present. |
| 15 to 0 | EXTCI | H'19 | R | R | Secondary PCI Express Extended Capability ID<br>Indicates the nature and format of the extended capability. |

### 104.2.56 Secondary PCI Express Extended Capability Register 1 function 0 (SEXPECAP1F0)

**Common to Header TYPE00/01**

- Link Control 3 Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

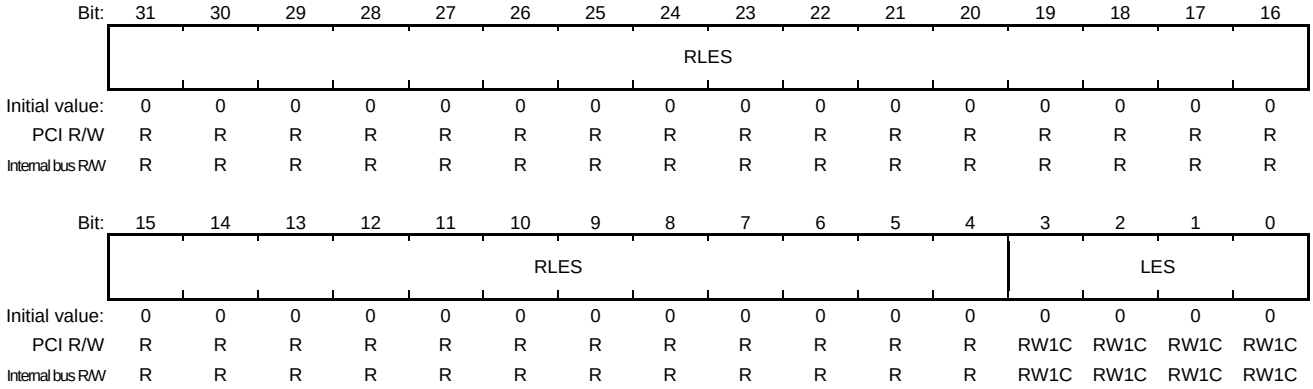
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EQRIE | PEQ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | R | Reserved<br>These bits are always read as 0.The write value should always be 0. |
| 1 | EQRIE | 0 | R/W | R/W | Link Equalization Request Interrupt Enable<br>When Set, this bit enables the generation of an interrupt to indicate that the Link Equalization 8.0 GT/s Request bit or the Link Equalization Request 16.0 GT/s bit has been set. |
| 0 | PEQ | 0 | R/W | R/W | Perform Equalization<br>When this bit is 1b and a 1b is written to the Retrain Link bit with the Target Link Speed field set to 8.0 GT/s or higher, the Downstream Port must perform Link Equalization. |

### 104.2.57 Secondary PCI Express Extended Capability Register 2 function 0 (SEXPECAP2F0)

**Common to Header TYPE00/01**

- Lane Error Status Register




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | RLES | All 0 | R | R | Reserved for future use.<br>These bits are always read as 0.The write value should always be 0. |
| 3 to 0 | LES | All 0 | RW1C | RW1C | Lane Error Status Bits per Lane.<br>Each bit indicates if the corresponding Lane detected a Lane-based error. A value of 1b indicates that a Lane based-error was detected on the corresponding Lane Number. |

### 104.2.58 Secondary PCI Express Extended Capability Register 3 function 0 (SEXPECAP3F0)

**Common to Header TYPE00/01**

- Lane Equalization Control Register for lanes 1 and 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | URPH1 | | | UTP1 | | | | — | DRPH1 | | | DTP1 | | | |
| Initial value: | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | URPH0 | | | UTP0 | | | | — | DRPH0 | | | DTP0 | | | |
| Initial value: | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 30 to 28 | URPH1 | H'7 | R | R | Upstream Port 8.0 GT/s Receiver Preset Hint 1<br>Field contains the Receiver preset hint value sent or received during 8.0 GT/s Link Equalization. |
| 27 to 24 | UTP1 | H'F | R | R | Upstream Port 8.0 GT/s Transmitter Preset 1<br>Field contains the Transmitter preset value sent or received during 8.0 GT/s Link Equalization. |
| 23 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 to 20 | DRPH1 | H'7 | R | R | Downstream Port 8.0 GT/s Receiver Preset Hint 1<br>Receiver preset hint value that may be used as a suggested setting for 8.0 GT/s receiver equalization by this Port when the Port is operating as a Downstream Port. |
| 19 to 16 | DTP1 | H'F | R | R | Downstream Port 8.0 GT/s Transmitter Preset 1<br>Transmitter preset value used for 8.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. |
| 15 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 14 to 12 | URPH0 | H'7 | R | R | Upstream Port 8.0 GT/s Receiver Preset Hint 0<br>Field contains the Receiver preset hint value sent or received during 8.0 GT/s Link Equalization. |
| 11 to 8 | UTP0 | H'F | R | R | Upstream Port 8.0 GT/s Transmitter Preset 0<br>Field contains the Transmitter preset value sent or received during 8.0 GT/s Link Equalization. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 7 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 to 4 | DRPH0 | H'7 | R | R | Downstream Port 8.0 GT/s Receiver Preset Hint 0<br>Receiver preset hint value that may be used as a suggested setting for 8.0 GT/s receiver equalization by this Port when the Port is operating as a Downstream Port. |
| 3 to 0 | DTP0 | H'F | R | R | Downstream Port 8.0 GT/s Transmitter Preset 0<br>Transmitter preset value used for 8.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. |

### 104.2.59 Secondary PCI Express Extended Capability Register 4 function 0 (SEXPECAP4F0)

**Common to Header TYPE00/01**

- Lane Equalization Control Register (LEC) or Lane Equalization Control Register 2(LEC2) #2.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | URPH3 | | | UTP3 | | | | — | DRPH3 | | | DTP3 | | | |
| Initial value: | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | URPH2 | | | UTP2 | | | | — | DRPH2 | | | DTP2 | | | |
| Initial value: | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 30 to 28 | URPH3 | H'7 | R | R | Upstream Port 8.0 GT/s Receiver Preset Hint 3<br>Field contains the Receiver preset hint value sent or received during 8.0 GT/s Link Equalization. |
| 27 to 24 | UTP3 | H'F | R | R | Upstream Port 8.0 GT/s Transmitter Preset 3<br>Field contains the Transmitter preset value sent or received during 8.0 GT/s Link Equalization. |
| 23 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 to 20 | DRPH3 | H'7 | R | R | Downstream Port 8.0 GT/s Receiver Preset Hint 3<br>Receiver preset hint value that may be used as a suggested setting for 8.0 GT/s receiver equalization by this Port when the Port is operating as a Downstream Port. |
| 19 to 16 | DTP3 | H'F | R | R | Downstream Port 8.0 GT/s Transmitter Preset 3<br>Transmitter preset value used for 8.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. |
| 15 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 14 to 12 | URPH2 | H'7 | R | R | Upstream Port 8.0 GT/s Receiver Preset Hint 2<br>Field contains the Receiver preset hint value sent or received during 8.0 GT/s Link Equalization. |
| 11 to 8 | UTP2 | H'F | R | R | Upstream Port 8.0 GT/s Transmitter Preset 2<br>Field contains the Transmitter preset value sent or received during 8.0 GT/s Link Equalization. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 7 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 to 4 | DRPH2 | H'7 | R | R | Downstream Port 8.0 GT/s Receiver Preset Hint 2<br>Receiver preset hint value that may be used as a suggested setting for 8.0 GT/s receiver equalization by this Port when the Port is operating as a Downstream Port. |
| 3 to 0 | DTP2 | H'F | R | R | Downstream Port 8.0 GT/s Transmitter Preset 2<br>Transmitter preset value used for 8.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. |

### 104.2.60 Physical Layer 16.0 GT/s Extended Capability Register 0 function 0 (PHY16ECAP0F0)

**Common to Header TYPE00/01**

- Physical Layer 16.0 GT/s Extended Capability Header

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NOFSET | | | | | | | | | | | | CVERSION | | | |
| Initial value: | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EXTCI | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | NOFSET | H'19C | R | R | Next Capability Offset |
| 19 to 16 | CVERSION | H'1 | R | R | Capability Version<br>Indicates the version of the capability structure present. |
| 15 to 0 | EXTCI | H'26 | R | R | PCI Express Extended Capability ID<br>Indicates the nature and format of the extended capability. |

### 104.2.61 Physical Layer 16.0 GT/s Extended Capability Register 1 function 0 (PHY16ECAP1F0)

**Common to Header TYPE00/01**

- 16.0 GT/s Capabilities Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | — | All 0 | R | R | Reserved<br>These bits are always read as 0.The write value should always be 0. |

### 104.2.62 Physical Layer 16.0 GT/s Extended Capability Register 2 function 0 (PHY16ECAP2F0)

**Common to Header TYPE00/01**

- 16.0 GT/s Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | — | All 0 | R | R | Reserved<br>These bits are always read as 0.The write value should always be 0. |

### 104.2.63 Physical Layer 16.0 GT/s Extended Capability Register 3 function 0 (PHY16ECAP3F0)

**Common to Header TYPE00/01**

- 16.0 GT/s Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | LE16R | E16CP3 | E16CP2 | E16CP1 | E16CP0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | RW1C | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | RW1C | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0.The write value should always be 0. |
| 4 | LE16R | 0 | RW1C | RW1C | Link Equalization Request 16.0GT/s<br>This bit is Set by hardware to request the 16.0 GT/s Link equalization process to be performed on the Link. |
| 3 | E16CP3 | 0 | R | R | Equalization 16.0GT/s Phase 3 Successful<br>When set to 1b, this bit indicates that Phase 3 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. |
| 2 | E16CP2 | 0 | R | R | Equalization 16.0GT/s Phase 2 Successful<br>When set to 1b, this bit indicates that Phase 2 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. |
| 1 | E16CP1 | 0 | R | R | Equalization 16.0GT/s Phase 1 Successful<br>When set to 1b, this bit indicates that Phase 1 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. |
| 0 | E16CP0 | 0 | R | R | Equalization 16.0GT/s Phase 0 Successful<br>When set to 1b, this bit indicates that Phase 0 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. |

### 104.2.64 Physical Layer 16.0 GT/s Extended Capability Register 4 function 0 (PHY16ECAP4F0)

**Common to Header TYPE00/01**

- 16.0 GT/s Local Data Parity Mismatch Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | LDS | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | R | Reserved for future use<br>These bits are always read as 0.The write value should always be 0. |
| 3 to 0 | LDS | All 0 | RW1C | RW1C | Local Data Parity Mismatch Status<br>Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. |

### 104.2.65 Physical Layer 16.0 GT/s Extended Capability Register 5 function 0 (PHY16ECAP5F0)

**Common to Header TYPE00/01**

- 16.0 GT/s First Retimer Data Parity Mismatch Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | FRDS | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | R | Reserved for future use<br>These bits are always read as 0.The write value should always be 0. |
| 3 to 0 | FRDS | All 0 | RW1C | RW1C | First Retimer Data Parity Mismatch Status<br>Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. |

### 104.2.66 Physical Layer 16.0 GT/s Extended Capability Register 6 function 0 (PHY16ECAP6F0)

**Common to Header TYPE00/01**

- 16.0 GT/s Second Retimer Data Parity Mismatch Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

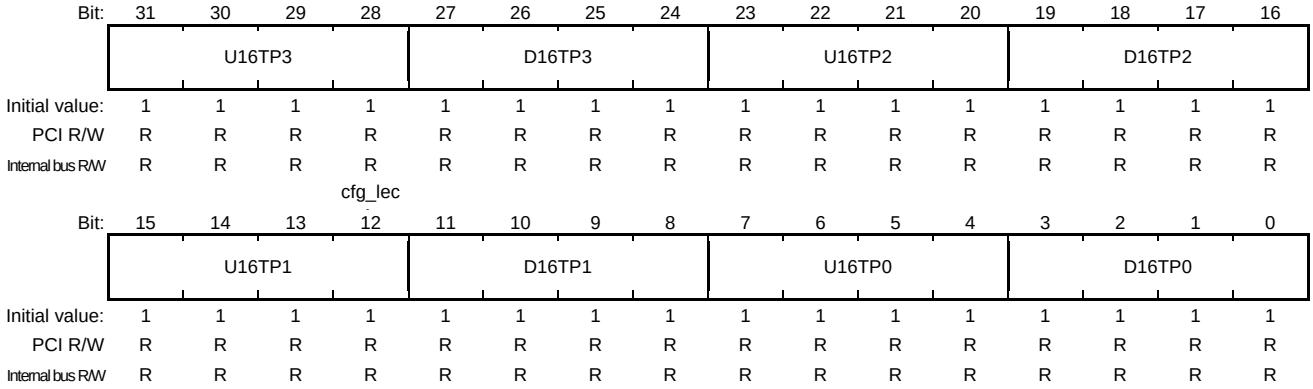
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | SRDS | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | R | Reserved for future use<br>These bits are always read as 0.The write value should always be 0. |
| 3 to 0 | SRDS | All 0 | RW1C | RW1C | Second Retimer Data Parity Mismatch Status<br>Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. |

### 104.2.67 Physical Layer 16.0 GT/s Extended Capability Register 7 function 0 (PHY16ECAP7F0)

**Common to Header TYPE00/01**

- 16.0 GT/s Lane Equalization Control Register for Lane 0-3

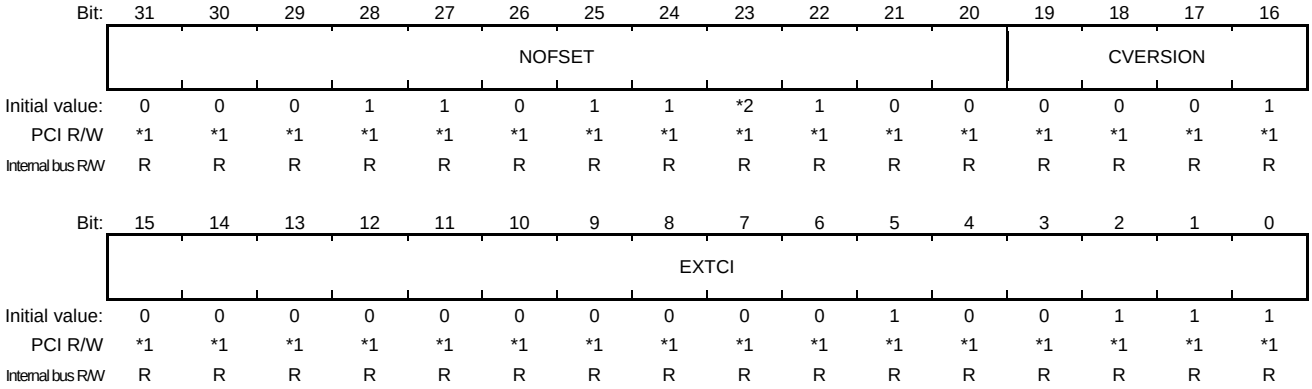



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 28 | U16TP3 | H'F | R | R | Upstream Port 16.0 GT/s Transmitter Preset3<br>Field contains the Transmit Preset value sent or received during 16.0 GT/s Link Equalization. |
| 27 to 24 | D16TP3 | H'F | R | R | Downstream Port 16.0 GT/s Transmitter Preset3<br>Transmitter Preset used for 16.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. |
| 23 to 20 | U16TP2 | H'F | R | R | Upstream Port 16.0 GT/s Transmitter Preset2<br>Field contains the Transmit Preset value sent or received during 16.0 GT/s Link Equalization. |
| 19 to 16 | D16TP2 | H'F | R | R | Downstream Port 16.0 GT/s Transmitter Preset2<br>Transmitter Preset used for 16.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. |
| 15 to 12 | U16TP1 | H'F | R | R | Upstream Port 16.0 GT/s Transmitter Preset1<br>Field contains the Transmit Preset value sent or received during 16.0 GT/s Link Equalization. |
| 11 to 8 | D16TP1 | H'F | R | R | Downstream Port 16.0 GT/s Transmitter Preset1<br>Transmitter Preset used for 16.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. |
| 7 to 4 | U16TP0 | H'F | R | R | Upstream Port 16.0 GT/s Transmitter Preset0<br>Field contains the Transmit Preset value sent or received during 16.0 GT/s Link Equalization. |
| 3 to 0 | D16TP0 | H'F | R | R | Downstream Port 16.0 GT/s Transmitter Preset0<br>Transmitter Preset used for 16.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. |

### 104.2.68 Lane Margining at the Receiver Extended Capability Register 0 function 0 (LNMGNCAP0F0)

**Common to Header TYPE00/01**

- Margining Extended Capability Header




Note: *1: R/W changes by the TYPE. TYPE00:R, TYPE01:-

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | NOFSET | H'1B<br>*2 | *1 | R | Next Capability Offset<br>*2Note: For initial value, in RC mode bit [23] is 1, in EP mode this bit is 0. Thus intial value can be H'1BC or H'1B4, depends on the device mode. |
| 19 to 16 | CVERSION | H'1 | *1 | R | Capability Version<br>Indicates the version of the capability structure present. |
| 15 to 0 | EXTCI | H'27 | *1 | R | PCI Express Extended Capability ID<br>Indicates the nature and format of the extended capability. |

### 104.2.69 Lane Margining at the Receiver Extended Capability Register 1 function 0 (LNMGNCAP1F0)

**Common to Header TYPE00/01**

- Margining Port Capabilities and Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MSR | MR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | MUDS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17 | MSR | 0 | R | R | Margining Software Ready<br>When Margining uses Driver Software is Set, then this bit, when Set, indicates that the required software has performed the required initialization. |
| 16 | MR | 0 | R | R | Margining Ready<br>Indicates when the Margining feature is ready to accept margining commands. |
| 15 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | MUDS | 0 | R | R | Margining uses Driver Software<br>If Set, indicates that Margining is partially implemented using Device Driver software. |

### 104.2.70 Lane Margining at the Receiver Extended Capability Register 2 function 0 (LNMGNCAP2F0)

**Common to Header TYPE00/01**

- Margining Lane Control and Status Register for Lane 0

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPS0 | | | | | | | | — | UMS 0 | MTS0 | | | RNS0 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MP0 | | | | | | | | — | US0 | MT0 | | | RN0 | | |
| Initial value: | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | MPS0 | H'0 | R | R | MarginPayload Status for Lane<br>This field is only meaningful, when the Margin Type is a defined encoding other than 'No Command'. |
| 23 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | UMS0 | 0 | R | R | Usage Model Status for Lane 0 |
| 21 to 19 | MTS0 | H'0 | R | R | Margin Type Status for Lane 0 |
| 18 to 16 | RNS0 | 0 | R | R | Receiver Number Status for Lane 0 |
| 15 to 8 | MP0 | H'9C | R/W | R/W | Margin Payload for Lane 0 |
| 7 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 | US0 | 0 | R/W | R/W | Usage Model for Lane 0 |
| 5 to 3 | MT0 | H'7 | R/W | R/W | Margin Type for Lane 0 |
| 2 to 0 | RN0 | H'0 | R/W | R/W | Receiver Number for Lane0 |

### 104.2.71 Lane Margining at the Receiver Extended Capability Register 3 function 0 (LNMGNCAP3F0)

**Common to Header TYPE00/01**

- Margining Lane Control and Status Register for Lane 1

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPS1 | | | | | | | | — | UMS 1 | MTS1 | | | RNS1 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MP1 | | | | | | | | — | US1 | MT1 | | | RN1 | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | MPS1 | H'0 | R | R | MarginPayload Status for Lane 1<br>This field is only meaningful, when the Margin Type is a defined encoding other than 'No Command'. |
| 23 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | UMS1 | 0 | R | R | Usage Model Status for Lane 1 |
| 21 to 19 | MTS1 | H'0 | R | R | Margin Type Status for Lane 1 |
| 18 to 16 | RNS1 | 0 | R | R | Receiver Number Status for Lane 1 |
| 15 to 8 | MP1 | H'9C | R/W | R/W | Margin Payload for Lane 1 |
| 7 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 | US1 | 0 | R/W | R/W | Usage Model for Lane 1 |
| 5 to 3 | MT1 | H'7 | R/W | R/W | Margin Type for Lane 1 |
| 2 to 0 | RN1 | H'0 | R/W | R/W | Receiver Number for Lane 1 |

### 104.2.72 Lane Margining at the Receiver Extended Capability Register 4 function 0 (LNMGNCAP4F0)

**Common to Header TYPE00/01**

- Margining Lane Control and Status Register for Lane 2

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPS2 | | | | | | | | — | UMS2 | MTS2 | | | RNS2 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MP2 | | | | | | | | — | US2 | MT2 | | | RN2 | | |
| Initial value: | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | MPS2 | H'0 | R | R | MarginPayload Status for Lane 2<br>This field is only meaningful, when the Margin Type is a defined encoding other than 'No Command'. |
| 23 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | UMS2 | 0 | R | R | Usage Model Status for Lane 2 |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 21 to 19 | MTS2 | H'0 | R | R | Margin Type Status for Lane 2 |
| 18 to 16 | RNS2 | 0 | R | R | Receiver Number Status for Lane 2 |
| 15 to 8 | MP2 | H'9C | R/W | R/W | Margin Payload for Lane 2 |
| 7 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 | US2 | 0 | R/W | R/W | Usage Model for Lane 2 |
| 5 to 3 | MT2 | H'7 | R/W | R/W | Margin Type for Lane 2 |
| 2 to 0 | RN2 | H'0 | R/W | R/W | Receiver Number for Lane 2 |

### 104.2.73 Lane Margining at the Receiver Extended Capability Register 5 function 0 (LNMGNCAP5F0)

**Common to Header TYPE00/01**

- Margining Lane Control and Status Register for Lane 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MPS3 | | | | | | | | — | UMS 3 | MTS3 | | | RNS3 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

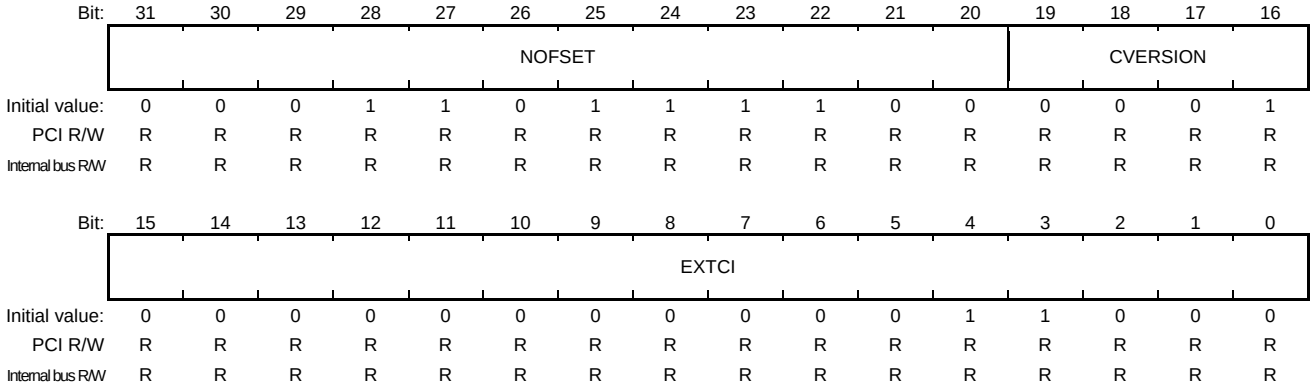
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MP3 | | | | | | | | — | US3 | MT3 | | | RN3 | | |
| Initial value: | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | MPS3 | H'0 | R | R | MarginPayload Status for Lane 3<br>This field is only meaningful, when the Margin Type is a defined encoding other than 'No Command'. |
| 23 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 | UMS3 | 0 | R | R | Usage Model Status for Lane 3 |
| 21 to 19 | MTS3 | H'0 | R | R | Margin Type Status for Lane 3 |
| 18 to 16 | RNS3 | 0 | R | R | Receiver Number Status for Lane 3 |
| 15 to 8 | MP3 | H'9C | R/W | R/W | Margin Payload for Lane 3 |
| 7 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 | US3 | 0 | R/W | R/W | Usage Model for Lane 3 |
| 5 to 3 | MT3 | H'7 | R/W | R/W | Margin Type for Lane 3 |
| 2 to 0 | RN3 | H'0 | R/W | R/W | Receiver Number for Lane 3 |

### 104.2.74 Latency Tolerance Reporting (LTR) Capability Register 0 function 0 (LTRCAP0F0)

**Header TYPE00 only**

- LTR Extended Capability Header




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | NOFSET | H'1BC | R | R | Next Capability Offset |
| 19 to 16 | CVERSION | H'1 | R | R | Capability Version<br>Indicates the version of the capability structure present. |
| 15 to 0 | EXTCI | H'18 | R | R | PCI Express Extended Capability ID<br>Indicates the nature and format of the extended capability. |

### 104.2.75 Latency Tolerance Reporting (LTR) Capability Register 1 function 0 (LTRCAP1F0)

**Header TYPE00 only**

- LTR Max Snoop and No-Snoop Latency Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | MNSLS | | | MNSL | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

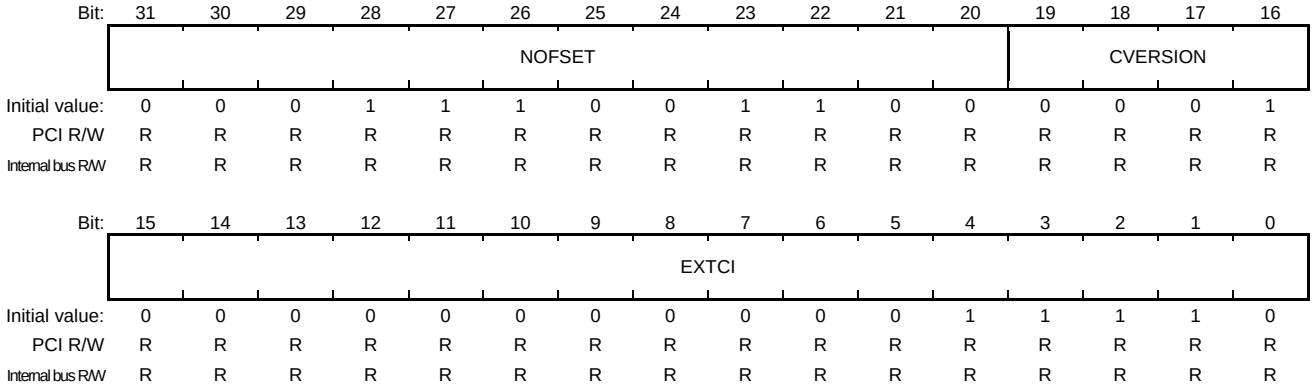
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | MSLS | | | MSL | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 26 | MNSLS | H'0 | R/W | R/W | Max No-Snoop Latency Scale<br>This register provides a scale for the value contained within the Max No-Snoop LatencyValue field. |
| 25 to 16 | MNSL | H'0 | R/W | R/W | Max No-Snoop Latency Value<br>Along with the Max No-Snoop LatencyScale field, this register specifies the maximum no-snoop latency that a device is permitted to request. |
| 15 to 13 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 to 10 | MSLS | H'0 | R/W | R/W | Max Snoop Latency Scale<br>This register provides a scale for the value contained within the Maximum Snoop LatencyValue field. |
| 9 to 0 | MSL | H'0 | R/W | R/W | Max Snoop Latency Value<br>Along with the Max Snoop LatencyScale field, this register specifies the maximum snoop latency that a device is permitted to request. |

### 104.2.76 L1 PM Substates Extended Capability Register 0 function 0 (L1PSCAP0F0)

**Common to Header TYPE00/01**

- L1 Substates Extended Capability Header

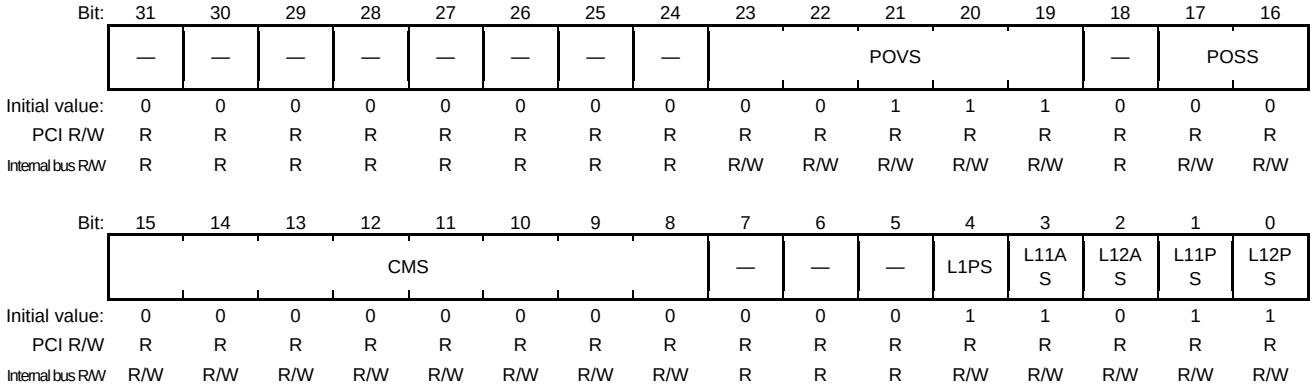



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | NOFSET | H'1CC | R | R | Next Capability Offset |
| 19 to 16 | CVERSION | H'1 | R | R | Capability Version<br>Indicates the version of the capability structure present. |
| 15 to 0 | EXTCI | H'1E | R | R | PCI Express Extended Capability ID<br>Indicates the nature and format of the extended capability. |

### 104.2.77 L1 PM Substates Extended Capability Register 1 function 0 (L1PSCAP1F0)

**Common to Header TYPE00/01**

- L1 Substates Capability Register



| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | POVS | | | | | — | POSS | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMS | | | | | | | | — | — | — | L1PS | L11AS | L12AS | L11PS | L12PS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 19 | POVS | H'7 | R | R/W | Port T_POWER_ON Value<br>Along with the Port T_POWER_ON Scale field in the L1 PM Substates Capabilities register sets the time (in µs) that this Port requires the port on the opposite side of Link to wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface. |
| 18 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 17, 16 | POSS | H'0 | R | R/W | Port T_POWER_ON Scale<br>Specifies the scale used for the Port T_POWER_ON Value field in the L1 PM Substates Capabilities register. |
| 15 to 8 | CMS | H'0 | R | R/W | Port Common Mode Restore Time<br>Time (in µs) required for this Port to re-establish common mode |
| 7 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | L1PS | 1 | R | R/W | L1 PM Substates Supported<br>When Set this bit indicates that this Port supports L1 PM Substates. |
| 3 | L11AS | 1 | R | R/W | ASPM L1.1 Supported<br>When Set this bit indicates that ASPM L1.1 is supported. |
| 2 | L12AS | 0 | R | R/W | ASPM L1.2 Supported<br>When Set this bit indicates that ASPM L1.2 is supported. |
| 1 | L11PS | 1 | R | R/W | PCI-PM L1.1 Supported<br>When Set this bit indicates that PCI-PM L1.1 is supported, and must be Set by all Ports implementing L1 PM Substates. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | L12PS | 1 | R | R/W | PCI-PM L1.2 Supported<br>When Set this bit indicates that PCI-PM L1.2 is supported. |

### 104.2.78 L1 PM Substates Extended Capability Register 2 function 0 (L1PSCAP2F0)

**Common to Header TYPE00/01**

- L1 Substates Control 1 Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | L12TS | | | — | — | — | L12TV | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | *1 | *1 | *1 | R | R | R | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 | *1 |
| Internal bus R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCM | | | | | | | | — | — | — | — | L11AE | L12AE | L11PE | L12PE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | — | — | — | — | — | — | — | — | R | R | R | R | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W |

Note: *1: R/W changes by the TYPE. TYPE00: R/W, TYPE01: -

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 29 | L12TS | H'0 | *1 | R/W | LTR L12 Threshold Scale<br>This field provides a scale for the value contained within the LTR_L1.2_THRESHOLD_Value.<br>Encoding is the same as the LatencyScale fields in the LTR Message. |
| 28 to 26 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 to 16 | L12TV | H'0 | *1 | R/W | LTR L12 Threshold Value<br>Along with the LTR_L1.2_THRESHOLD_Scale, this field indicates the LTR threshold used to determine if entry into L1 results in L1.1 (if enabled) or L1.2 (if enabled).<br>This field must only be modified when the ASPM L1.2 Enable bit is Clear. |
| 15 to 8 | TCM | H'0 | — | R/W | Common Mode Restore Time<br>Sets value of TCOMMONMODE (in µs), which must be used by the Downstream Port for timing the re-establishment of common mode.<br>This field must only be modified when the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are both Clear. |
| 7 to 4 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 3 | L11AE | 0 | R/W | R/W | ASPM L11 Enable<br>When Set this bit enables ASPM L1.1. |
| 2 | L12AE | 0 | R/W | R/W | ASPM L12 Enable<br>When Set this bit enables ASPM L1.2. |
| 1 | L11PE | 0 | R/W | R/W | PCI-PM L11 Enable<br>When Set this bit enables PCI-PM L1.1. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | L12PE | 0 | R/W | R/W | PCI-PM L12 Enable<br>When Set this bit enables PCI-PM L1.2. |

### 104.2.79 L1 PM Substates Extended Capability Register 3 function 0 (L1PSCAP3F0)

**Common to Header TYPE00/01**

- L1 Substates Control 2 Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TPOV | | | | | | TPOS | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | *1 | *1 | *1 | *1 | *1 | R | *1 | *1 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

Note: *1: R/W changes by the TYPE. TYPE00: R/W, TYPE01: -

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 3 | TPOV | H'5 | *1 | R/W | T Power On Value<br>Along with the T_POWER_ON Scale sets the minimum amount of time (in µs) that the Port must wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface.<br>T_POWER_ON is calculated by multiplying the value in this field by the value in the T_POWER_ON Scale field.<br>This field must only be modified when the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are both Clear. |
| 2 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 1, 0 | TPOS | H'0 | *1 | R/W | T Power On Scale<br>Specifies the scale used for T_POWER_ON Value.<br>H'0 = 2 µs<br>H'1 = 10 µs<br>H'2 = 100 µs<br>H'3 = Reserved |

### 104.2.80 Vendor-Specific Extended Capability Register 0 function *i* (VSECAP0F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Vendor-Specific Extended Capability Header

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NOFSET | | | | | | | | | | | | CVERSION | | | |
| Initial value: | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

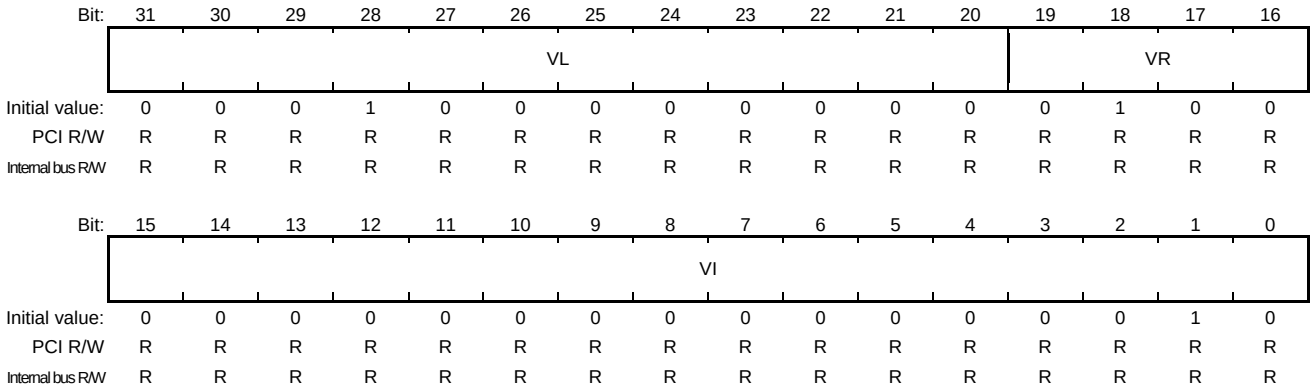
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EXTCI | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | NOFSET | H'2CC | R | R | Next Capability Offset |
| 19 to 16 | CVERSION | H'1 | R | R | Capability Version<br>Indicates the version of the capability structure present. |
| 15 to 0 | EXTCI | H'B | R | R | PCI Express Extended Capability ID<br>Indicates the nature and format of the extended capability. |

### 104.2.81 Vendor-Specific Extended Capability Register 1 function *i* (VSECAP1F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Vendor-Specific Header




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | VL | H'100 | R | R | VSEC Length<br>This field indicates the number of bytes in the entire VSEC structure, including the PCI Express Extended Capability header, the vendor-specific header, and the vendor-specific registers. |
| 19 to 16 | VR | H'4 | R | R | VSEC Rev<br>This field is a vendor-defined version number that indicates the version of the VSEC structure. |
| 15 to 0 | VI | H'2 | R | R | VSEC ID<br>This field is a vendor-defined ID number that indicates the nature and format of the VSEC structure. |

### 104.2.82 Vendor-Specific Extended Capability Register 2 function *i* (VSECAP2F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Event Counter Control (ECCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ECES | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ECLS | | | | ECS | — | — | ECE | | | ECC | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | W | W | W | W | W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | W | W | W | W | W |

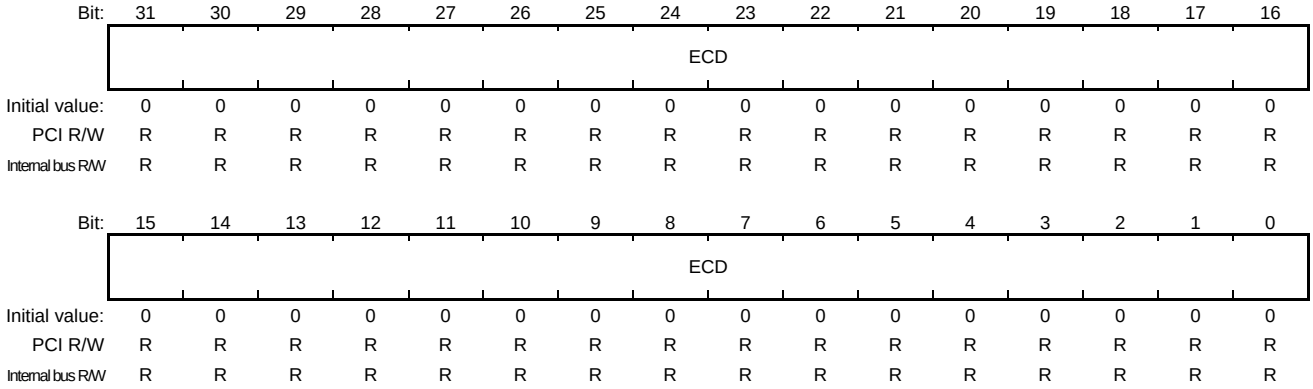
| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 16 | ECES | H'0 | R/W | R/W | Event Counter Data Select.<br>This field in conjunction with the ECLS field indexes the Event Counter data returned by the ECDR register.<br>■ 27-24: Group number(4-bit: 0..H'7)<br>■ 23-16: Event number(8-bit: 0..H'13) within the Group |
| 15 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 8 | ECLS | H'0 | R/W | R/W | Event Counter Lane Select.<br>This field in conjunction with ECES indexes the Event Counter data returned by the ECDR register.<br>■ H'0: Lane0<br>■ H'1: Lane1<br>■ H'2: Lane2<br>■ ..<br>■ H'F: Lane15 |
| 7 | ECS | 0 | R | R | Event Counter Status.<br>This register returns the current value of the Event Counter selected by the following fields:<br>■ ECES<br>■ ECLS |
| 6, 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 4 to 2 | ECE | H'0 | W | W | Event Counter Enable.<br>Enables/disables the Event Counter selected by the ECES and ECLS fields in this register.<br>■ H'0: no change<br>■ H'1: per event off<br>■ H'2: no change<br>■ H'3: per event on<br>■ H'4: no change<br>■ H'5: all off<br>■ H'6: no change<br>■ H'7: all on |
| 1, 0 | ECC | H'0 | W | W | Event Counter Clear.<br>Clears the Event Counter selected by the ECES and ECLS fields in this register.<br>■ H'0: no change<br>■ H'1: per clear<br>■ H'2: no change<br>■ H'3: all clear<br>■ Other: reserved |

### 104.2.83 Vendor-Specific Extended Capability Register 3 function *i* (VSECAP3F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Event Counter Data (ECDR)




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | ECD | H'0 | R | R | Event Counter Data.<br>This register returns the data selected by the following fields:<br>■ ECES in ECCR<br>■ ECLS in ECCR |

### 104.2.84 Vendor-Specific Extended Capability Register 4 function *i* (VSECAP4F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Time-based Analysis Control (TBACR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBRS | | | | | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBDS | | | | | | | | — | — | — | — | — | — | — | TST ART |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to24 | TBRS | H'00 | R/W | R/W | Time-based Report Select.<br>Selects what type of data is measured for the selected duration (TBDS), and returned in TBAD.<br>Each type of data is measured using one of three types of units:<br>■ Core_clk Cycles.<br>Total time in ps is [Value of TBAD returned when TBRS=0x00] * TBAD<br>■ Aux_clk Cycles.<br>Total time in ps is [Period of platform specific clock] * TBAD<br>■ Data Bytes.<br>Actual amount of bytes is 16 * TBAD<br>Core_clk Cycles<br>H'00: Duration of 1 cycle<br>H'01: TxL0s<br>H'02: RxL0s<br>H'03: L0<br>H'04: L1 (Rsvd when aux_clk is supplied from the platform specific clock during L1, L1.1 or L1.2)<br>H'07: Configuration/Recovery Aux_clk Cycles<br>H'05: L1.1<br>H'06: L1.2<br>Data Bytes<br>H'20: Tx TLP data payload Bytes<br>H'21: Rx TLP data payload Bytes<br>Else: Reserved |
| 23 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 15 to 8 | TBDS | H'1 | R/W | R/W | Time-based Duration Select.<br>Selects the duration of timebased analysis.<br>When "manual control" is selected and TSTART is set to '1', this analysis never stops until TSTOP is set to '0'.<br>H'0: Manual control<br>H'1: 1ms<br>H'2: 10ms<br>H'3: 100ms<br>H'4: 1s<br>H'5: 2s<br>H'6: 4s<br>Else: Reserved |
| 7 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | TSTART | 0 | R/W | R/W | Timer Start.<br>1: Start/Restart<br>0: Stop<br>This bit will be cleared automatically when the measurement is finished. |

### 104.2.85 Vendor-Specific Extended Capability Register 5 function *i* (VSECAP5F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Time-based Analysis Data

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBAD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBAD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | TBAD | H'0 | R | R | Time Based Analysis Data.<br>This register returns the data selected by the TBRS field in TBACR.<br>The results are cleared when next measurement starts. |

### 104.2.86 Vendor-Specific Extended Capability Register 6 function *i* (VSECAP6F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Enable

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | EI6E | EI5E | EI4E | EI3E | EI2E | EI1E | EI0E |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 7 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 | EI6E | 0 | R/W | R/W | Error Injection6 Enable (Specific TLP Error).<br>Enables insertion of errors into the packets that you select. |
| 5 | EI5E | 0 | R/W | R/W | Error Injection5 Enable (TLP Duplicate/Nullify Error).<br>Enables insertion of duplicate/nullified TLPs. |
| 4 | EI4E | 0 | R/W | R/W | Error Injection4 Enable (FC Credit Update Error).<br>Enables insertion of errors into UpdateFCs. |
| 3 | EI3E | 0 | R/W | R/W | Error Injection3 Enable (Symbol DataK Mask Error or Sync Header Error).<br>Enables DataK masking of special symbols or the breaking of the sync header. |
| 2 | EI2E | 0 | R/W | R/W | Error Injection2 Enable (DLLP Error).<br>Enables insertion of DLLP errors. |
| 1 | EI1E | 0 | R/W | R/W | Error Injection1 Enable (Sequence Number Error).<br>Enables insertion of errors into sequence numbers. |
| 0 | EI0E | 0 | R/W | R/W | Error Injection0 Enable (CRC Error).<br>Enables insertion of errors into various CRC. |

### 104.2.87 Vendor-Specific Extended Capability Register 7 function *i* (VSECAP7F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 0 (CRC Error).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | ECRCT | | | | ECNT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 8 | ECRCT | H'0 | R/W | R/W | Error injection type.<br>Selects the type of CRC error to be inserted.<br>■ Tx Path<br>H'0: New TLP's LCRC error injection<br>H'1: 16bCRC error injection of ACK/NAK DLLP<br>H'2: 16bCRC error injection of Update-FC DLLP<br>H'3: New TLP's ECRC error injection<br>H'4: TLP's FCRC error injection (128b/130b)<br>H'5: Parity error of TSOS (128b/130b)<br>H'6: Parity error of SKPOS (128b/130b)<br>■ Rx Path<br>H'8: LCRC error injection<br>H'B: ECRC error injection<br>Others: Reserved |
| 7 to 0 | ECNT | H'0 | R/W | R/W | Error injection count.<br>Indicates the number of errors. |

### 104.2.88 Vendor-Specific Extended Capability Register 8 function *i* (VSECAP8F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 1 (Sequence Number Error).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | E1BS | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | E1ST | E1CNT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 16 | E1BS | H'0 | R/W | R/W | Bad sequence number.<br>Indicates the value to add/subtract from the naturally-assigned sequence numbers.<br>This value is represented by two's complement.<br>H'0FFF: +4095<br>..<br>H'0002: +2<br>H'0001: +1<br>H'0000: 0<br>H'1FFF: -1<br>H'1FFE: -2<br>..<br>H'1001: -4095 |
| 15 to 9 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | E1ST | 0 | R/W | R/W | Sequence number type.<br>Selects the type of sequence number.<br>0: Insertion of New TLP's SEQ# error<br>1: Insertion of ACK/NAK DLLP's SEQ# Error |
| 7 to 0 | E1CNT | H'0 | R/W | R/W | Error injection count.<br>Indicates the number of errors. |

### 104.2.89 Vendor-Specific Extended Capability Register 9 function *i* (VSECAP9F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 2 (DLLP Error).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | E2DT | | E2CNT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 10 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9, 8 | E2DT | H'0 | R/W | R/W | DLLP Type.<br>Selects the type of DLLP errors to be inserted.<br>H'0: ACK/NAK DLLP's transmission block<br>H'1: Update FC DLLP's transmission block<br>H'2: Always Transmission for NAK DLLP<br>H'3: Reserved |
| 7 to 0 | E2CNT | H'0 | R/W | R/W | Error injection count.<br>Indicates the number of errors. |

### 104.2.90 Vendor-Specific Extended Capability Register 10 function *i* (VSECAP10F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 3 (Symbol Error).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | E3ST | | | E3CNT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 10 to 8 | E3ST | H'0 | R/W | R/W | Error Type.<br><br>8b/10b encoding - Mask K symbol.<br>H'0: Reserved<br>H'1: COM/PAD(TS1 Order set)<br>H'2: COM/PAD(TS2 Order set)<br>H'3: COM/FTS(FTS Order set)<br>H'4: COM/IDL(E-Idle Order set)<br>H'5: END/EDB Symbol<br>H'6: STP/SDP Symbol<br>H'7b: COM/SKP(SKP Order set)<br>128b/130b encoding - Change sync header.<br>H'0: Invert sync header<br>Others: Reserved |
| 7 to 0 | E3CNT | H'0 | R/W | R/W | Error injection count.<br>Indicates the number of errors. |

### 104.2.91 Vendor-Specific Extended Capability Register 11 function *i* (VSECAP11F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 4 (FC Credit Error)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | E4BUV | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | E4VN | | | — | E4UT | | | E4CNT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 16 | E4BUV | H'0 | R/W | R/W | Bad update-FC credit value.<br>Indicates the value to add/subtract from the UpdateFC credit.<br>H'0FFF: +4095<br>..<br>H'0002: +2<br>H'0001: +1<br>H'0000: 0<br>H'1FFF: -1<br>H'1FFE: -2<br>..<br>H'1001: -4095 |
| 15 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 14 to 12 | E4VN | H'0 | R/W | R/W | VC Number<br>Indicates target VC Number. |
| 11 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 10 to 8 | E4UT | H'0 | R/W | R/W | Update-FC type.<br>Selects the credit type.<br>H'0: Posted TLP Header Credit value control<br>H'1: Non-Posted TLP Header Credit value control<br>H'2: Completion TLP Header Credit value control<br>H'3: Reserved<br>H'4: Posted TLP Data Credit value control<br>H'5: Non-Posted TLP Data Credit value control<br>H'6: Completion TLP Data Credit value control<br>H'7: Reserved |
| 7 to 0 | E4CNT | H'0 | R/W | R/W | Error injection count.<br>Indicates the number of errors. |

### 104.2.92 Vendor-Specific Extended Capability Register 12 function *i* (VSECAP12F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 5 (Specific TLP Error)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | E5ST | E5CNT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | E5ST | 0 | R/W | R/W | Specified TLP.<br>Selects the specified TLP to be inserted.<br>0: Generates duplicate TLPs by handling ACK DLLP as NAK DLLP.<br>1: Generates Nullified TLP (Original TLP will be stored in retry buffer). |
| 7 to 0 | E5CNT | H'0 | R/W | R/W | Error injection count.<br>Indicates the number of errors. |

### 104.2.93 Vendor-Specific Extended Capability Register 13 function *i* (VSECAP13F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Compare Point Header DWORD #0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COPH0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COPH0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6COPH0 | H'0 | R/W | R/W | Packet Compare Point: 1st DWORD.<br>Specifies which Tx TLP header DWORD#0 bits to compare with the corresponding bits in the Packet Compare Value registers (E6CV*).<br>When all specified bits (in the Tx TLP header and E6CV*) match, the controller inserts errors into the TLP. |

### 104.2.94 Vendor-Specific Extended Capability Register 14 function *i* (VSECAP14F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Compare Point Header DWORD #1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COPH1 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

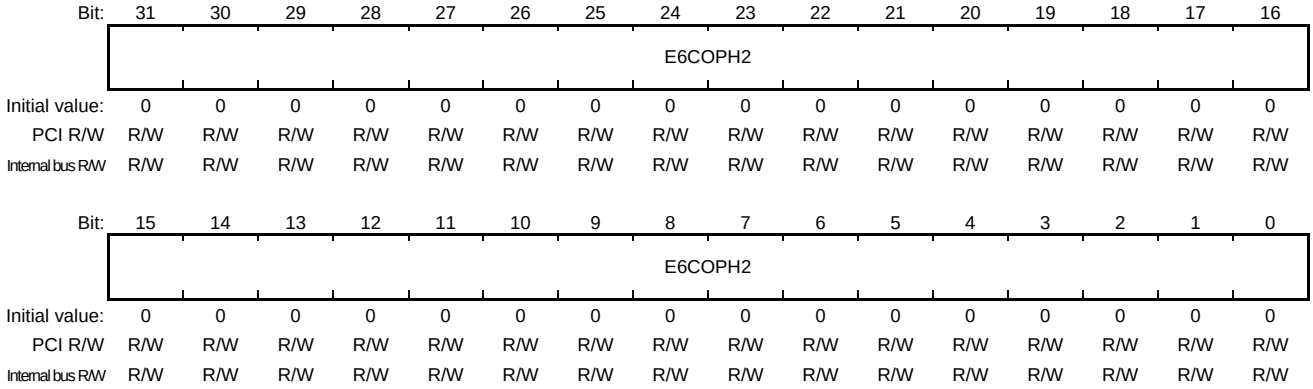
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COPH1 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6COPH1 | H'0 | R/W | R/W | Packet Compare Point: 2nd DWORD.<br>Specifies which Tx TLP header DWORD#1 bits to compare with the corresponding bits in the Packet Compare Value registers (E6CV*).<br>When all specified bits (in the Tx TLP header and E6CV*) match, the controller inserts errors into the TLP. |

### 104.2.95 Vendor-Specific Extended Capability Register 15 function *i* (VSECAP15F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Compare Point Header DWORD #2)




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6COPH2 | H'0 | R/W | R/W | Packet Compare Point: 3rd DWORD.<br>Specifies which Tx TLP header DWORD#2 bits to compare with the corresponding bits in the Packet Compare Value registers (E6CV*).<br>When all specified bits (in the Tx TLP header and E6CV*) match, the controller inserts errors into the TLP. |

### 104.2.96 Vendor-Specific Extended Capability Register 16 function *i* (VSECAP16F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Compare Point Header DWORD #3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COPH3 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COPH3 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6COPH3 | H'0 | R/W | R/W | Packet Compare Point: 4th DWORD.<br>Specifies which Tx TLP header DWORD#3 bits to compare with the corresponding bits in the Packet Compare Value registers (E6CV*).<br>When all specified bits (in the Tx TLP header and E6CV*) match, the controller inserts errors into the TLP. |

### 104.2.97 Vendor-Specific Extended Capability Register 17 function *i* (VSECAP17F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Compare Value Header DWORD #0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COVH0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COVH0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6COVH0 | H'0 | R/W | R/W | Packet Compare Value: 1st DWORD.<br>Specifies the value to compare against Tx the TLP header DWORD#0 bits specified in the Packet Compare Point registers (E6CP*). |

### 104.2.98 Vendor-Specific Extended Capability Register 18 function *i* (VSECAP18F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Compare Value Header DWORD #1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COVH1 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

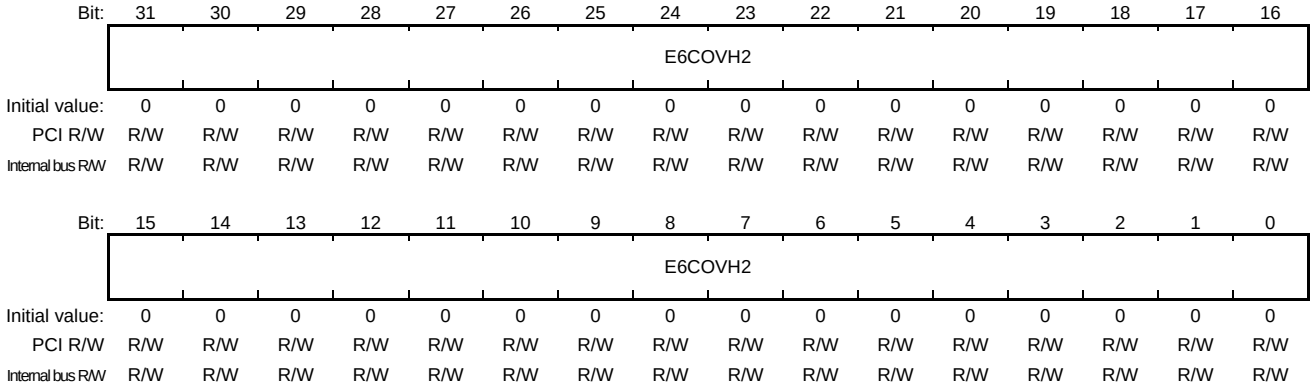
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COVH1 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6COVH1 | H'0 | R/W | R/W | Packet Compare Value: 2nd DWORD.<br>Specifies the value to compare against Tx the TLP header DWORD#1 bits specified in the Packet Compare Point registers (E6CP*). |

### 104.2.99 Vendor-Specific Extended Capability Register 19 function *i* (VSECAP19F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Compare Value Header DWORD #2)




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6COVH2 | H'0 | R/W | R/W | Packet Compare Value: 3rd DWORD.<br>Specifies the value to compare against Tx the TLP header DWORD#2 bits specified in the Packet Compare Point registers (E6CP*). |

### 104.2.100 Vendor-Specific Extended Capability Register 20 function *i* (VSECAP20F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Compare Value Header DWORD #3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COVH3 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6COVH3 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6COVH3 | H'0 | R/W | R/W | Packet Compare Value: 4th DWORD.<br>Specifies the value to compare against Tx the TLP header DWORD#3 bits specified in the Packet Compare Point registers (E6CP*). |

### 104.2.101 Vendor-Specific Extended Capability Register 21 function *i* (VSECAP21F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Change Point Header DWORD #0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6CHPH0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

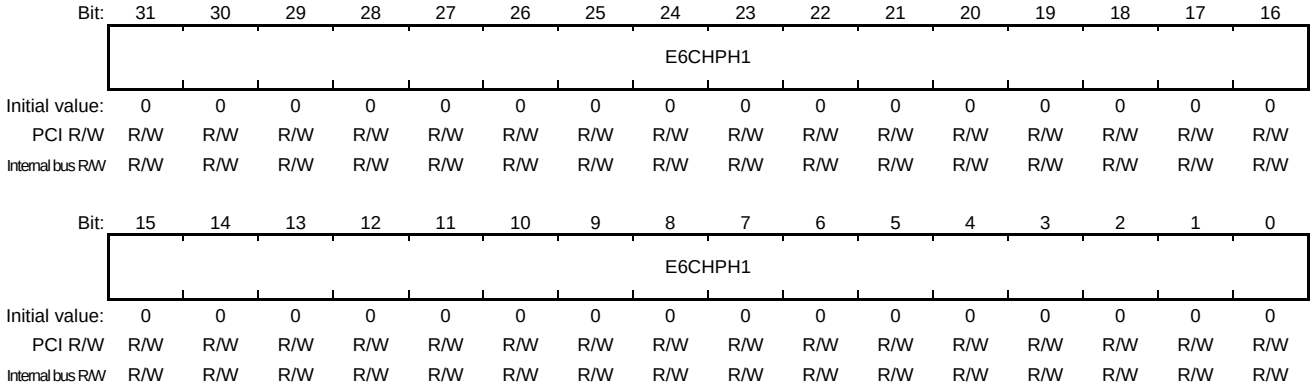
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6CHPH0 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6CHPH0 | H'0 | R/W | R/W | Packet Change Point: 1st DWORD.<br>Specifies which Tx TLP header DWORD#0 bits to replace with the corresponding bits in the Packet Change Value registers (E6CHV*). |

### 104.2.102 Vendor-Specific Extended Capability Register 22 function *i* (VSECAP22F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Change Point Header DWORD #1)

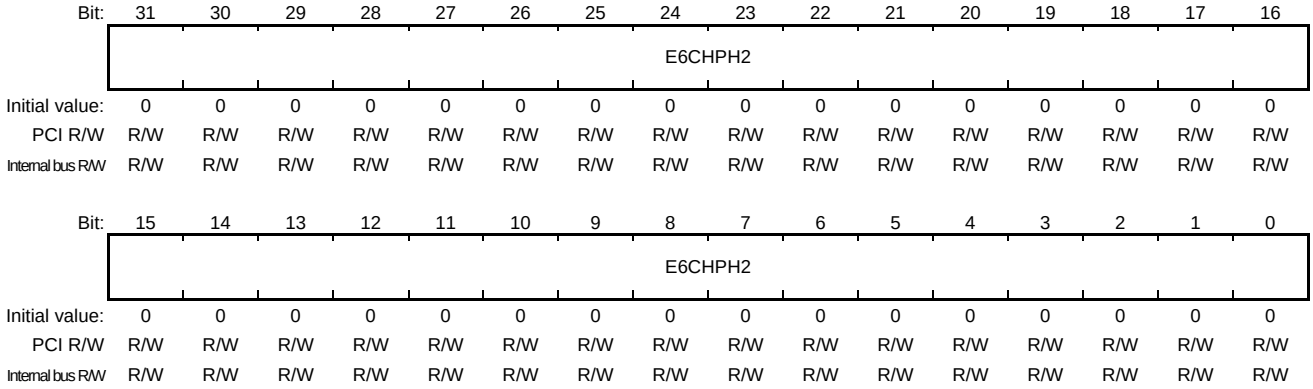



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6CHPH1 | H'0 | R/W | R/W | Packet Change Point: 2nd DWORD.<br>Specifies which Tx TLP header DWORD#2 bits to replace with the corresponding bits in the Packet Change Value registers (E6CHV*). |

### 104.2.103 Vendor-Specific Extended Capability Register 23 function *i* (VSECAP23F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Change Point Header DWORD #2)

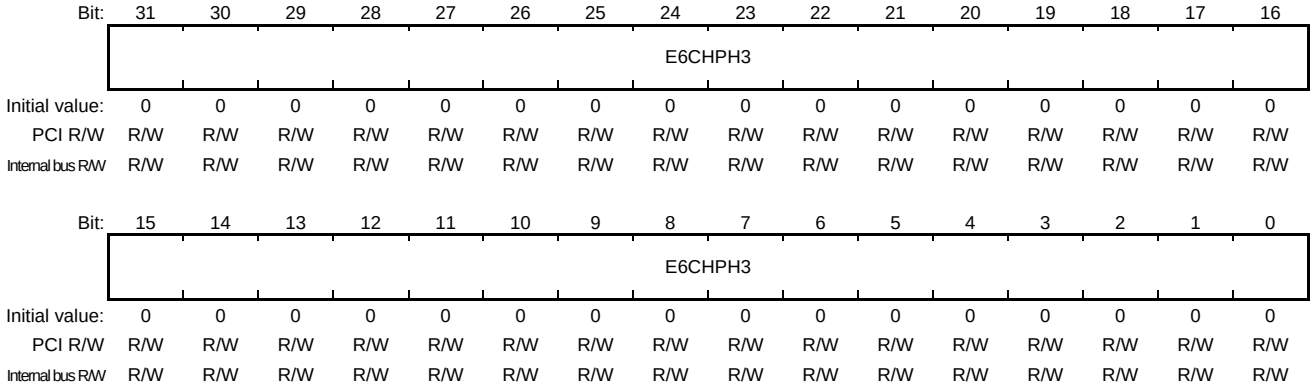



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6CHPH2 | H'0 | R/W | R/W | Packet Change Point: 3rd DWORD.<br>Specifies which Tx TLP header DWORD#2 bits to replace with the corresponding bits in the Packet Change Value registers (E6CHV*). |

### 104.2.104 Vendor-Specific Extended Capability Register 24 function *i* (VSECAP24F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Change Point Header DWORD #3)

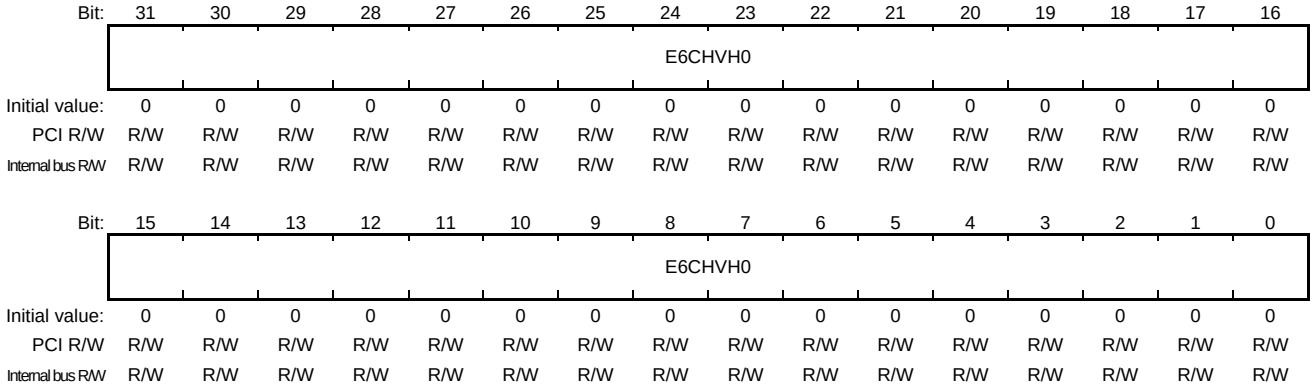



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6CHPH3 | H'0 | R/W | R/W | Packet Change Point: 4th DWORD.<br>Specifies which Tx TLP header DWORD#3 bits to replace with the corresponding bits in the Packet Change Value registers (E6CHV*). |

### 104.2.105 Vendor-Specific Extended Capability Register 25 function *i* (VSECAP25F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Change Value Header DWORD #0)

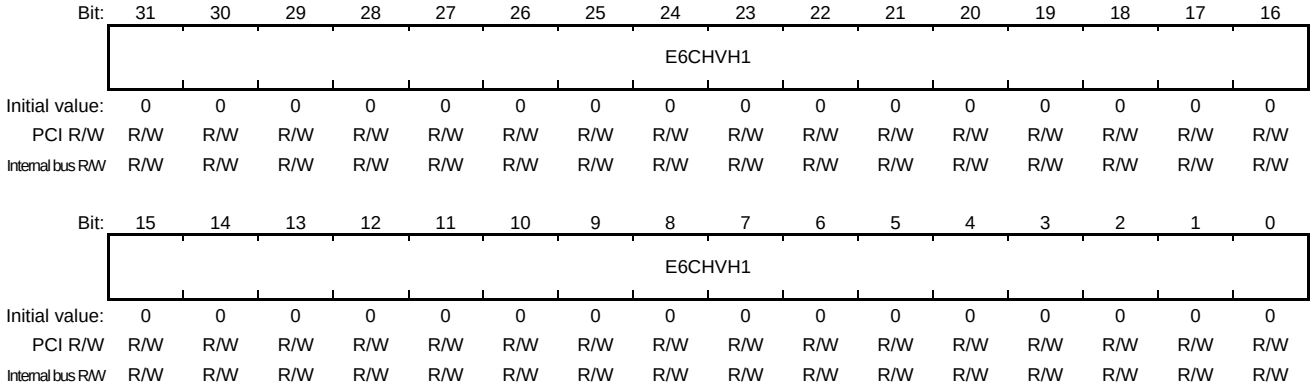



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6CHVH0 | H'0 | R/W | R/W | Packet Change Value: 1st DWORD.<br>Specifies replacement values for the Tx TLP header DWORD#0 bits defined in the Packet Change Point registers(E6CHP*).<br>Only applies when the E6IC field in E6TR is '0'. |

### 104.2.106 Vendor-Specific Extended Capability Register 26 function *i* (VSECAP26F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Change Value Header DWORD #1)

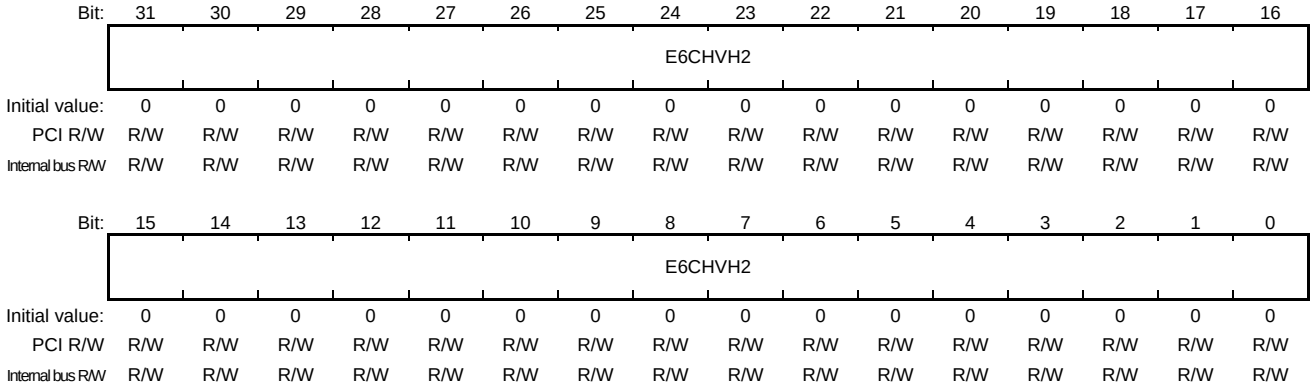



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6CHVH1 | H'0 | R/W | R/W | Packet Change Value: 2nd DWORD.<br>Specifies replacement values for the Tx TLP header DWORD#1 bits defined in the Packet Change Point registers(E6CHP*).<br>Only applies when the E6IC field in E6TR is '0'. |

### 104.2.107 Vendor-Specific Extended Capability Register 27 function *i* (VSECAP27F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Change Value Header DWORD #2)




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6CHVH2 | H'0 | R/W | R/W | Packet Change Value: 3rd DWORD.<br>Specifies replacement values for the Tx TLP header DWORD#2 bits defined in the Packet Change Point registers(E6CHP*).<br>Only applies when the E6IC field in E6TR is '0'. |

### 104.2.108 Vendor-Specific Extended Capability Register 28 function *i* (VSECAP28F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Change Value Header DWORD #3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6CHVH3 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | E6CHVH3 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | E6CHVH3 | H'0 | R/W | R/W | Packet Change Value: 4th DWORD.<br>Specifies replacement values for the Tx TLP header DWORD#3 bits defined in the Packet Change Point registers(E6CHP*).<br>Only applies when the E6IC field in E6TR is '0'. |

### 104.2.109 Vendor-Specific Extended Capability Register 29 function *i* (VSECAP29F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error Injection Control 6 (Packet Error)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | E6PT | | | E6IC | E6CNT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 9 | E6PT | H'0 | R/W | R/W | Packet type.<br>Selects the TLP packets to inject errors into.<br>H'0: TLP Header<br>H'1: TLP Prefix 1st 4-DWORDs<br>H'2: TLP Prefix 2nd -DWORDs<br>Else: Reserved |
| 8 | E6IC | 0 | R/W | R/W | Inverted Error Injection Control.<br>■ 0: E6CHVH[0/1/2/3] is used to replace bits specified by E6CHPH[0/1/2/3].<br>■ 1: E6CHVH[0/1/2/3] is ignored and inverts bits specified by E6CHPH[0/1/2/3]. |
| 7 to 0 | E6CNT | H'0 | R/W | R/W | Error Injection Count.<br>Indicates the number of errors to insert. |

### 104.2.110 Vendor-Specific Extended Capability Register 30 function *i* (VSECAP30F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug Control 1

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LPI | | TEN | | — | — | — | FDLE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FDL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23, 22 | LPI | H'0 | R/W | R/W | Low Power Entry Interval Time<br>Interval Time that the controller starts monitoring RXELECIDLE signal after L0s/L1/L2 entry.<br>H'0: 40ns<br>H'1: 160ns<br>H'2: 320ns<br>H'3: 640ns |
| 21, 20 | TEN | H'0 | R/W | R/W | Number of Tx EIOS<br>This register sets the number of transmit EIOS for L0s/L1 entry and Disable/Loopback/Hot-reset exit.<br>■ 2.5GT/s, 8.0GT/s or higher:<br>H'0: 1<br>H'1: 4<br>H'2: 8<br>H'3: 16<br>■ 5.0GT/s:<br>H'0: 2<br>H'1: 8<br>H'2: 16<br>H'3: 32 |
| 19 to 17 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | FDLE | 0 | R/W | R/W | Force Detect Lane Enable<br>When this bit is set, the controller ignores receiver detection from PHY during LTSSM Detect state and uses FDL |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 15 to 0 | FDL | H'0 | R/W | R/W | Force Detect Lane<br>When the FORCE_DETECT_LANE_EN field is set, the controller ignores receiver detection from PHY during LTSSM Detect state and uses this value instead.<br>0: Lane0<br>1: Lane1<br>2: Lane2<br>..<br>15: Lane15 |

### 104.2.111 Vendor-Specific Extended Capability Register 31 function *i* (VSECAP31F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug Control 2

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | FERD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | DLTE | DPTD | DRTC | — | — | — | — | — | NFL | RR | HL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | W | R/W |
| Internal bus R/W | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | FERD | 0 | R/W | R/W | Framing Error Recovery Disable<br>This bit disables a transition to Recovery state when a Framing Error is occurred. |
| 15 to 11 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 10 | DLTE | 0 | R/W | R/W | Direct Loopback Slave To Exit<br>When this bit is set and the LTSSM is in Loopback Slave Active State, the LTSSM transitions to Loopback Slave Exit state. |
| 9 | DPTD | 0 | R/W | R/W | Direct Polling.Compliance to Detect<br>When this bit is set and the LTSSM is in Polling Compliance State, the LTSSM transitions to Detect state. |
| 8 | DRTC | 0 | R/W | R/W | Direct Recovery.Idle to Configuration.<br>When this bit is set and the LTSSM is in Recovery Idle State, the LTSSM transitions to Configuration state. |
| 7 to 3 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | NFL | 0 | R/W | R/W | Force LinkDown<br>When this bit is set and the controller detects REPLY_NUM<br>rolling ver 4 times, the LTSSM transitions to Detect State. |
| 1 | RR | 0 | W | W | Recovery Request<br>When this bit is set to '1' in L0 or L0s, the LTSSM starts transitioning to Recovery State. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | HL | 0 | R/W | R/W | Hold and Release LTSSM.<br>For as long as this register is '1', the controller stays in the current LTSSM. |

### 104.2.112 Vendor-Specific Extended Capability Register 32 function *i* (VSECAP32F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug Status(Layer1 Per-lane).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSKPINTR | | | | | | | | — | — | — | PIPE TXE | PIPE RXE | PIPE RXV | PIPE DL | PIPE RXP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | LANESEL | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | DSKPINTR | 0 | R | R | Deskew Pointer<br>Indicates Deskew pointer of internal Deskew buffer of selected lane number(LANESEL). |
| 23 to 21 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 20 | PIPETXE | 1 | R | R | PIPE:TxElecIdle<br>Indicates PIPE TXELECIDLE signal of selected lane number(LANESEL). |
| 19 | PIPERXE | 1 | R | R | PIPE:RxElecIdle<br>Indicates PIPE RXELECIDLE signal of selected lane number(LANESEL). |
| 18 | PIPERXV | 0 | R | R | PIPE:RxValid<br>Indicates PIPE RXVALID signal of selected lane number(LANESEL). |
| 17 | PIPEDL | 0 | R | R | PIPE:Detect Lane<br>Indicates whether PHY indicates receiver detection or not on selected lane number(LANESEL). |
| 16 | PIPERXP | 0 | R | R | PIPE:RxPolarity<br>Indicates PIPE RXPOLARITY signal of selected lane number(LANESEL). |
| 15 to 4 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 3 to 0 | LANESEL | H'0 | R/W | R/W | Lane Select.<br>Lane Select register for Silicon Debug Status Register of Layer1-PerLane.<br>H'0: Lane0<br>H'1: Lane1<br>H'2: Lane2<br>..<br>H'F: Lane15 |

### 104.2.113 Vendor-Specific Extended Capability Register 33 function *i* (VSECAP33F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug Status(Layer1 LTSSM)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LTSSMV | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LANERVSL | — | — | — | — | PIPEPDWN | | | FRMERR | FRMERRPTR | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | RW1C | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | RW1C | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | LTSSMV | H'0 | R | R | LTSSM Variable<br>Indicates internal LTSSM variables defined in the PCI Express Base Specification.<br>● C-PCIe Mode:<br>0: directed_speed_change<br>1: changed_speed_recovery<br>2: successful_speed_negotiation<br>3: upconfigure_capable<br>Set to '1' if both ports advertised the UpConfigure capability in the last Config.Complete.<br>4: select_deemphasis<br>5: start_equalization_w_preset<br>6: equalization_done_8GT_data_rate<br>7: equalization_done_16GT_data_rate<br>15:8: idle_to_rlock_transitioned<br>● M-PCIe Mode:<br>0: idle_to_recovery<br>1: recovery_to_configuration |
| 15 | LANERVSL | 0 | R | R | Lane Reversal Operation<br>Receiver detected lane reversal.<br>This field is only valid in the L0 LTSSM state. |
| 14 to 11 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 10 to 8 | PIPEPDWN | H'2 | R | R | PIPE:PowerDown<br>Indicates PIPE PowerDown signal. |
| 7 | FRMERR | 0 | RW1C | RW1C | Framing Error<br>Indicates Framing Error detection status. |

| | | | | | |
|---|---|---|---|---|---|
| 6 to 0 | FRMERRPTR | 0 | R | R | First Framing Error Pointer.<br>Identifies the first Framing Error using the following encoding. The field contents are only valid value when FRMERR =1.<br>● Received Unexpected Framing Token<br>H'01: When non- STP/SDP/IDL Token was received and it was not in TLP/DLLP reception<br>H'02: When current token was not a valid EDB token and previous token was an EDB. (128/256 bit controller only)<br>H'03: When SDP token was received but not expected. (128 bit & (x8 \| x16) controller only)<br>H'04: When STP token was received but not expected. (128 bit & (x8 \| x16) controller only)<br>H'05: When EDS token was expected but not received or whenever an EDS token was received but not expected.<br>H'06: When a framing error was detected in the deskew block while a packet has been in progress in token_finder. Received Unexpected STP Token<br>H'11: When Framing CRC in STP token did not match<br>H'12: When Framing Parity in STP token did not match.<br>H'13: When Framing TLP Length in STP token was smaller than 5 DWORDs.<br>■ Received Unexpected Block<br>H'21: When Receiving an OS Block following SDS in Datastream state<br>H'22: When Data Block followed by OS Block different from SKP, EI, EIE in Datastream state<br>H'23: When Block with an undefined Block Type in Datastream state<br>H'24: When Data Stream without data over three cycles in Datastream state<br>H'25: When OS Block during Data Stream in Datastream state<br>H'26: When RxStatus Error was detected in Datastream state<br>H'27: When Not all active lanes receiving SKP OS starting at same cycle time in SKPOS state<br>H'28: When a 2-Block timeout occurs for SKP OS in SKPOS state<br>H'29: When Receiving consecutive OS Blocks within a Data Stream in SKPOS state<br>H'2A: When Phy status error was detected in SKPOS state<br>H'2B: When Not all active lanes receiving EIOS starting at same cycle time in EIOS state<br>H'2C: When At least one Symbol from the first 4 Symbols is not EIOS Symbol in EIOS state.<br>H'2D: When Not all active lanes receiving EIEOS starting at same cycle time in EIEOS state |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| | | | | | H'2E: When Not full 16 eieos symbols are received in EIEOS state<br>All other values not listed above are Reserved. |

### 104.2.114 Vendor-Specific Extended Capability Register 34 function *i* (VSECAP34F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug Status(PM)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LATNFTS | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | L1SBSTAT | | | PMER SFLG | INTPSSTAT | | | | — | — | — | INTPMSTAT | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | RW1C | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | RW1C | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | LATNFTS | H'0 | R | R | Latched N_FTS.<br>Indicates the value of N_FTS in the received TS Ordered Sets from the Link Partner. |
| 15 to 13 | L1SBSTAT | H'0 | R | R | L1Sub State.<br>Indicates internal state machine of L1Sub state.<br>H'0: S_L1_U : idle state<br>H'1: S_L1_0 : wait for aux_clk_active<br>H'2: S_L1_0_WAIT4_ACK : wait for pclkack<br>H'3: S_L1_0_WAIT4_CLKREQ : wait for clkreq<br>H'4: S_L1_N_ENTRY : check clkreq_in_n is de-asserted for t_power_off time (only for L1.2, reduces to one cycle for L1.1)<br>H'5: S_L1_N : L1 substate, turn off txcommonmode circuits (L1.2 only) and rx electrical idle detection circuits<br>H'6: S_L1_N_EXIT : locally/remotely initiated exit, assert pclkreq, wait for pclkack<br>H'7: S_L1_N_ABORT : wait for pclkack when aborting an attempt to enter L1_N |
| 12 | PMERSFLG | 0 | RW1C | RW1C | PME Re-send flag.<br>When the DUT sends a PM_PME message TLP, the DUT sets PME_Status bit. If host software does not clear PME_Status bit for 100ms(+50%/-5%), the DUT resends the PM_PME Message. This bit indicates that a PM_PME was resent. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 11 to 8 | INTPSSTAT | H'0 | R | R | Internal PM State(Slave).<br>Indicates internal state machine of Power Management Slave controller.<br>H'0: S_IDLE<br>H'1: S_RESPOND_NAK<br>H'2: S_BLOCK_TLP<br>H'3: S_WAIT_LAST_TLP_ACK<br>H'4: S_WAIT_EIDLE<br>H'5: S_LINK_ENTR_L1<br>H'6: S_L1<br>H'7: S_L1_EXIT<br>H'8: S_L23RDY<br>H'9: S_LINK_ENTR_L23<br>H'A: S_L23RDY_WAIT4ALIVE<br>H'B: S_ACK_WAIT4IDLE<br>H'C: S_WAIT_LAST_PMDLLP |
| 7 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 to 0 | INTPMSTAT | H'0 | R | R | Internal PM State(Master).<br>Indicates internal state machine of Power Management Master controller.<br>H'00: IDLE<br>H'01: L0<br>H'02: L0S<br>H'03: ENTER_L0S<br>H'04: L0S_EXIT<br>H'05: WAIT_PMCSR_CPL_SENT<br>H'08: L1<br>H'09: L1_BLOCK_TLP<br>H'0A: L1_WAIT_LAST_TLP_ACK<br>H'0B: L1_WAIT_PMDLLP_ACK<br>H'0C: L1_LINK_ENTR_L1<br>H'0D: L1_EXIT<br>H'0F: PREP_4L1<br>H'10: L23_BLOCK_TLP<br>H'11: L23_WAIT_LAST_TLP_ACK<br>H'12: L23_WAIT_PMDLLP_ACK<br>H'13: L23_ENTR_L23<br>H'14: L23RDY<br>H'15: PREP_4L23<br>H'16: L23RDY_WAIT4ALIVE<br>H'17: L0S_BLOCK_TLP<br>H'18: WAIT_LAST_PMDLLP<br>H'19: WAIT_DSTATE_UPDATE |

### 104.2.115 Vendor-Specific Extended Capability Register 35 function *i* (VSECAP35F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug Status(Layer2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | FCINIT2 | FCINIT1 | DLCMSM | | RXAKSQNO | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RXAKSQNO | | | | TXTLPSQNO | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 | FCINIT2 | 0 | R | R | FCINIT2.<br>Indicates the controller is in FCINIT2(VC0) state. |
| 26 | FCINIT1 | 0 | R | R | FCINIT1.<br>Indicates the controller is in FCINIT1(VC0) state. |
| 25, 24 | DLCMSM | H'0 | R | R | DLCMSM.<br>Indicates the current DLCMSM.<br>H'0: DL_INACTIVE<br>H'1: DL_FC_INIT<br>H'3: DL_ACTIVE |
| 23 to12 | RXAKSQNO | H'FFF | R | R | Tx Ack Sequence Number.<br>Indicates ACKDSEQ which is updated by receiving ACK/NAK DLLP. |
| 11 to 0 | TXTLPSQNO | H'0 | R | R | Tx Tlp Sequence Number.<br>Indicates next transmit sequence number for transmit TLP. |

### 104.2.116 Vendor-Specific Extended Capability Register 36 function *i* (VSECAP36F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug Status(Layer3 FC)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CREDITDT1 | | | | | | | | | | | | CREDITDT0 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CREDITDT0 | | | | | | | | — | CREDITSH | CREDITSTT | | CREDITSCT | CREDITSV | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | CREDITDT1 | 0 | R | R | Credit Data1<br>Current FC credit data selected by the CREDITSV, CREDITSCT, CREDITSTT, and CREDITSH viewport-select fields. |
| 19 to 8 | CREDITDT0 | 0 | R | R | Credit Data 0<br>Current FC credit data selected by the CREDITSV, CREDITSCT, CREDITSTT, and CREDITSH viewport-select fields. |
| 7 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 | CREDITSH | 0 | R/W | R/W | Credit Select (HeaderData).<br>This field in conjunction with the CREDITSV, CREDITSCT, and CREDITSTT viewport-select fields determines that data that is returned by the CREDITDT0 and CREDITDT1 data fields. |
| 5 to 4 | CREDITSTT | H'0 | R/W | R/W | Credit Select (TLP Type).<br>This field in conjunction with the CREDITSV, CREDITSCT, and CREDITSH viewport-select fields determines that data that is returned by the CREDITDT0 and CREDITDT1 data fields.<br>H'0: Posted<br>H'1: Non-Posted<br>H'2: Completion |
| 3 | CREDITSCT | 0 | R/W | R/W | Credit Select(Credit Type).<br>This field in conjunction with the CREDITSV, CREDITSTT, and CREDITSH viewportselect fields determines that data that is returned by the CREDITDT0 and CREDITDT1 data fields.<br>0: Rx<br>1: Tx |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 2 to 0 | CREDITSV | 0 | R/W | R/W | Credit Select(VC).<br>This field in conjunction with the CREDITSCT, CREDITSTT, and CREDITSH viewport-select fields determines that data that is returned by the CREDITDT0 and CREDITDT1 data fields.<br>H'0: VC0<br>H'1: VC1<br>H'2: VC2<br>..<br>H'7: VC7 |

### 104.2.117 Vendor-Specific Extended Capability Register 37 function *i* (VSECAP37F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug Status(Layer3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | MFTLP STATS | MFTLPPNTR | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | RW1C | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | RW1C | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | MFTLPSTATS | 0 | RW1C | RW1C | Malformed TLP Status.<br>Indicates malformed TLP has occurred. |
| 6 to 0 | MFTLPPNTR | H'0 | R | R | First Malformed TLP Error Pointer.<br>Indicates the element of the received first malformed TLP.<br>This pointer is validated by MFTLPSTATS.<br>H'01: AtomicOp address alignment<br>H'02: AtomicOp operand<br>H'03: AtomicOp byte enable<br>H'04: TLP length miss match<br>H'05: Max payload size<br>H'06: Message TLP without TC0<br>H'07: Invalid TC<br>H'08: Unexpected route bit in Message TLP<br>H'09: Unexpected CRS status in Completion TLP<br>H'0A: Byte enable<br>H'0B: Memory Address 4KB boundary<br>H'0C: TLP prefix rules<br>H'0D: Translation request rules<br>H'0E: Invalid TLP type<br>H'0F: Completion rules<br>H'7F: Application<br>Else: Reserved |

### 104.2.118 Vendor-Specific Extended Capability Register 38 function *i* (VSECAP38F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug EQ Control 1

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOMTGT | | | | | | | | FOMTGTEBL | — | — | — | — | — | EVALINTVLT | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | EXTEQTOUT | | — | — | — | EQRATESL | EQLANESL | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | FOMTGT | H'0 | R/W | R/W | FOM Target.<br>Indicates figure of merit target criteria value of EQ Master (DSP in EQ Phase3/USP in EQ Phase2).<br>This field is only valid when GEN3_EQ_FB_MODE is 0001b (Figure Of Merit). |
| 23 | FOMTGTEBL | 0 | R/W | R/W | FOM Target Enable.<br>Enables the FOM_TARGET fields. |
| 22 to 18 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17, 16 | EVALINTVLT | H'0 | R/W | R/W | Eval Interval Time.<br>Indicates interval time of RxEqEval assertion.<br>H'0: 500ns<br>H'1: 1us<br>H'2: 2us<br>H'3: 4us<br>This field is used for EQ Master(DSP in EQ Phase3/USP in EQ Phase2). |
| 15 to 10 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9, 8 | EXTEQTOUT | H'0 | R/W | R/W | Extends EQ Phase2/3 Timeout.<br>This field is used when the Ltssm is in Recovery.EQ2/3.<br>When this field is set, the value of EQ2/3 timeout is extended.<br>EQ Master(DSP in EQ Phase3/USP in EQ Phase2).<br>H'0: 24ms (default)<br>H'1: 48ms (x2)<br>H'2: 240ms (x10)<br>H'3: No timeout<br>EQ Slave(DSP in EQ Phase2/USP in EQ Phase3).<br>H'0: 32ms (default)<br>H'1: 56ms (32ms+24ms)<br>H'2: 248ms (32ms +9*24ms)<br>H'3: No timeout |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 7 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | EQRATESL | 0 | R/W | R/W | EQ Status Rate Select.<br>Setting this field in conjunction with the EQ_LANE_SEL field determines the per-lane Silicon Debug EQ Status data returned by the SD_EQ_CONTROL[2/3] and SD_EQ_STATUS[1/2/3] viewport registers.<br>0: 8.0GT/s Speed<br>1: 16.0GT/s Speed |
| 3 to 0 | EQLANESL | H'0 | R/W | R/W | EQ Status Lane Select.<br>Setting this field in conjunction with the EQ_RATE_SEL field determines the per-lane Silicon Debug EQ Status data returned by the SD_EQ_CONTROL[2/3] and SD_EQ_STATUS[1/2/3] viewport registers.<br>H'0: Lane0<br>H'1: Lane1<br>H'2: Lane2<br>..<br>H'F: Lane15 |

### 104.2.119 Vendor-Specific Extended Capability Register 39 function *i* (VSECAP39F*i*) *i* = 0, 1

### Common to Header TYPE00/01

- Silicon Debug EQ Control 2

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | FLTXPE | FLRXHE | FLTXCE | FLTXP | | | | — | — | — | FLRXH | | | FLTXPSTC | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLTXPSTC | | | | FLTXC | | | | | | FLTXPREC | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 30 | FLTXPE | 0 | R/W | R/W | Force Local Transmitter Preset Enable.<br>Enables the FLTXP field. |
| 29 | FLRXHE | 0 | R/W | R/W | Force Local Receiver Preset Hint Enable.<br>Enables the FLRXH field. |
| 28 | FLTXCE | 0 | R/W | R/W | Force Local Transmitter Coefficient Enable.<br>Enables the following fields:<br>■ FLTXPREC<br>■ FLTXC<br>■ FLTXPOSTC |
| 27 to 24 | FLTXP | H'0 | R/W | R/W | Force Local Transmitter Preset.<br>Indicates initial preset value of USP in EQ Slave (EQ Phase2) instead of receiving EQ TS2. |
| 23 to 21 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 20 to 18 | FLRXH | H'0 | R/W | R/W | Force Local Receiver Preset Hint.<br>Indicates the RxPresetHint value of EQ Slave (DSP in EQ Phase2/USP in EQ Phase3), instead of received or set value. |
| 17 to 12 | FLTXPSTC | H'0 | R/W | R/W | Force Local Transmitter Post-Cursor.<br>Indicates the coefficient value of EQ Slave (DSP in EQ Phase2/USP in EQ Phase3), instead of the value instructed from link partner. |
| 11 to 6 | FLTXC | H'0 | R/W | R/W | Force Local Transmitter Cursor.<br>Indicates the coefficient value of EQ Slave(DSP in EQ Phase2/USP in EQ Phase3), instead of the value instructed from link partner. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 5 to 0 | FLTXPREC | H'0 | R/W | R/W | Force Local Transmitter Pre-cursor.<br>Indicates the coefficient value of EQ Slave(DSP in EQ Phase2/USP in EQ Phase3), instead of the value instructed from link partner. |

### 104.2.120 Vendor-Specific Extended Capability Register 40 function *i* (VSECAP40F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug EQ Control 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | FRT XCE | — | — | — | — | — | — | — | — | — | — | FRTXPOSTC | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FRTXPOSTC | | | | FRTXC | | | | | | FRTXPREC | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 | FRTXCE | 0 | R/W | R/W | Force Remote Transmitter Coefficient Enable.<br>Enables the following fields:<br>■ FRTXPREC<br>■ FRTXC<br>■ FRTXPOSTC |
| 27 to 18 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17 to 12 | FRTXPOSTC | H'0 | R/W | R/W | Force Remote Transmitter Post-Cursor.<br>Indicates the coefficient value of EQ Master(DSP in EQ Phase3/USP in EQ Phase2), instead of the value instructed from link partner. |
| 11 to 6 | FRTXC | H'0 | R/W | R/W | Force Remote Transmitter Cursor.<br>Indicates the coefficient value of EQ Master(DSP in EQ Phase3/USP in EQ Phase2), instead of the value instructed from link partner. |
| 5 to 0 | FRTXPREC | H'0 | R/W | R/W | Force Remote Transmitter Pre-Cursor.<br>Indicates the coefficient value of EQ Master(DSP in EQ Phase3/USP in EQ Phase2), instead of the value instructed from link partner. |

### 104.2.121 Vendor-Specific Extended Capability Register 41 function *i* (VSECAP41F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug EQ Status 1

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | EQR JCT | EQR LCV | EQR LBV | EQR LAV | — | EQCNVGCI | | EQS EQ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | EQRJCTE | 0 | R | R | EQ Reject Event.<br>Indicates that the controller receives two consecutive TS1 OS w/Reject=1b during EQ Master phase(DSP in EQ Phase3/USP in EQ Phase2). This bit is automatically cleared when the controller starts EQ Master phase again. |
| 6 | EQRLCV | 0 | R | R | EQ Rule C Violation.<br>Indicates that coefficients rule C violation is detected in the values provided by PHY using direction change method during EQ Master phase(DSP in EQ Phase3/USP in EQ Phase2). The coefficients rule C correspond to the rules c) from section "Rules for Transmitter Coefficents" in the PCI Express Base Specification.<br>This bit is automatically cleared when the controller starts EQ Master phase again. |
| 5 | EQRLBV | 0 | R | R | EQ Rule B Violation.<br>Indicates that coefficients rule B violation is detected in the values provided by PHY using direction change method during EQ Master phase(DSP in EQ Phase3/USP in EQ Phase2).<br>The coefficients rules B correspond to the rules b) from section "Rules for Transmitter Coefficents" in the PCI Express Base Specification.<br>This bit is automatically cleared when the controller starts EQ Master phase again. |
| 4 | EQRLAV | 0 | R | R | EQ Rule A Violation.<br>Indicates that coefficients rule A violation is detected in the values provided by PHY using direction change method during EQ Master phase(DSP in EQ Phase3/USP in EQ Phase2). The coefficients rules A correspond to the rules a) from section "Rules for Transmitter Coefficents" in the PCI Express Base Specification.<br>This bit is automatically cleared when the controller starts EQ Master phase again. |
| 3 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 2, 1 | EQCNVGCI | H'0 | R | R | EQ Convergence Info.<br>Indicates equalization convergence information.<br>H'0: Equalization is not attempted<br>H'1: Equalization finished successfully<br>H'2: Equalization finished unsuccessfully<br>H'3: Reserved<br>This bit is automatically cleared when the controller starts EQ Master phase again. |
| 0 | EQSEQ | 0 | R | R | EQ Sequence.<br>Indicates that the controller is starting the equalization sequence. |

### 104.2.122 Vendor-Specific Extended Capability Register 42 function *i* (VSECAP42F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug EQ Status 2

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EQLFV | | | | | | | | — | — | — | EQLRXH | | | EQLPOSTC | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EQLPOSTC | | | | EQLC | | | | | | EQLPREC | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | EQLFV | H'0 | R | R | EQ Local Figure of Merit.<br>Indicates Local maximum Figure of Merit value. |
| 23 to 21 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 20 to 18 | EQLRXH | H'0 | R | R | EQ Local Receiver Preset Hint.<br>Indicates Local Receiver Preset Hint value. |
| 17 to 12 | EQLPOSTC | H'0 | R | R | EQ Local Post-Cursor.<br>Indicates Local post cursor coefficient value. |
| 11 to 6 | EQLC | H'0 | R | R | EQ Local Cursor.<br>Indicates Local cursor coefficient value. |
| 5 to 0 | EQLPREC | H'0 | R | R | EQ Local Pre-Cursor.<br>Indicates Local pre cursor coefficient value. |

### 104.2.123 Vendor-Specific Extended Capability Register 43 function *i* (VSECAP43F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Silicon Debug EQ Status 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | EQRMTFS | | | | | | EQRMTLF | | | | | | EQRMTPOSTC | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

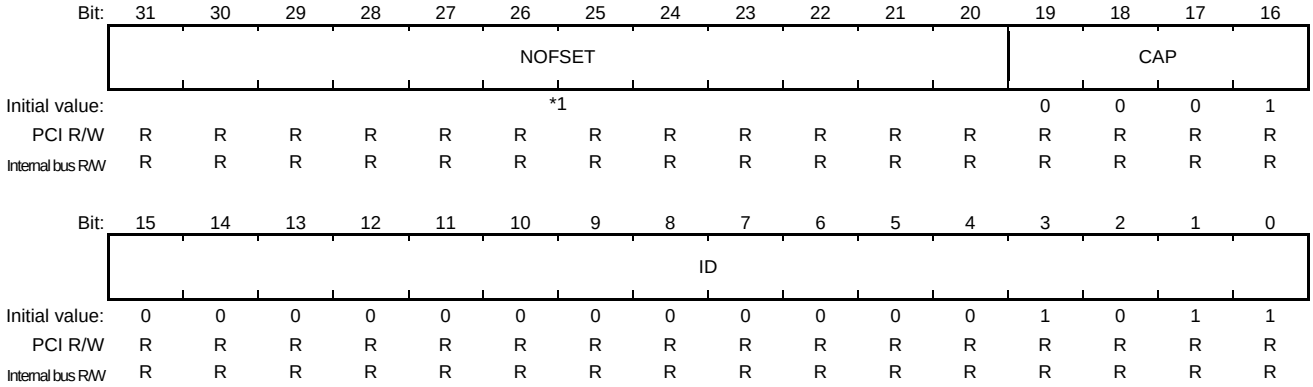
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EQRMTPOSTC | | | | EQRMTC | | | | | | EQRMTPREC | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31, 30 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29 to 24 | EQRMTFS | H'0 | R | R | EQ Remote FS.<br>Indicates Remote FS value. |
| 23 to 18 | EQRMTLF | H'0 | R | R | EQ Remote LF.Indicates Remote LF value. |
| 17 to 12 | EQRMTPOSTC | H'0 | R | R | EQ Remote Post-Cursor.<br>Indicates Remote post cursor coefficient value. |
| 11 to 6 | EQRMTC | H'0 | R | R | EQ Remote Cursor.<br>Indicates Remote cursor coefficient value. |
| 5 to 0 | EQRMTPREC | H'0 | R | R | Remote Pre-Cursor.<br>Indicates Remote pre cursor coefficient value. |

### 104.2.124 Vendor-Specific Extended 2 Capability Register 0 function i (VSE2CAP0F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- PF RAS Datapath Protection Capability Structure, Vendor-Specific Extended Capability Header

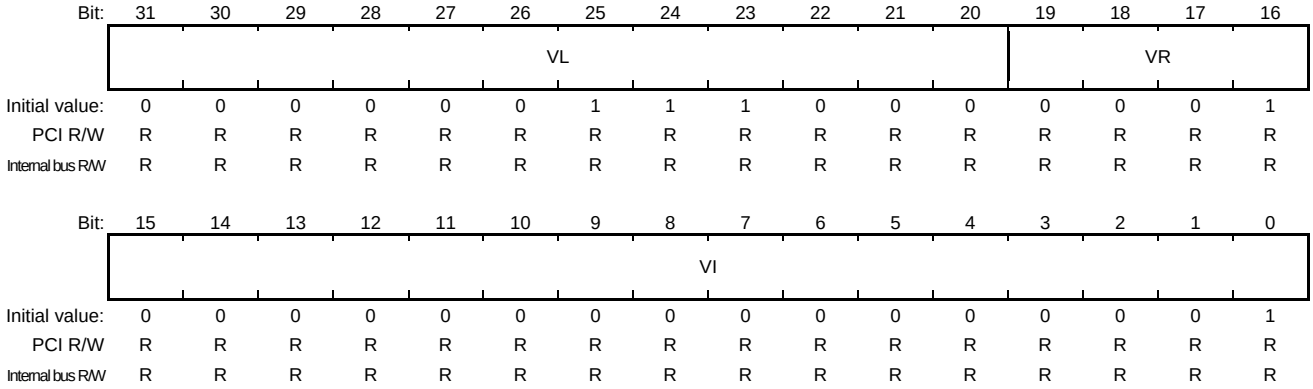



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | NOFSET | *1 | R | R | Next Capability Offset<br>*1: (*i*=0)H'304,(*i*=1)H'31C |
| 19 to 16 | CAP | H'1 | R | R | Capability Version<br>Indicates the version of the capability structure present. |
| 15 to 0 | ID | H'B | R | R | PCI Express Extended Capability ID<br>Indicates the nature and format of the extended capability. |

### 104.2.125 Vendor-Specific Extended 2 Capability Register 1 function i (VSE2CAP1F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Vendor-Specific Header




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | VL | H'38 | R | R | VSEC Length<br>This field indicates the number of bytes in the entire VSEC structure, including the PCI Express Extended Capability header, the vendor-specific header, and the vendor-specific registers. |
| 19 to 16 | VR | H'1 | R | R | VSEC Rev<br>This field is a vendor-defined version number that indicates the version of the VSEC structure. |
| 15 to 0 | VI | H'1 | R | R | VSEC ID<br>This field is a vendor-defined ID number that indicates the nature and format of the VSEC structure. |

### 104.2.126 Vendor-Specific Extended 2 Capability Register 2 function i (VSE2CAP2F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- ECC error correction control

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | EPD CRX | EPD ARX | EPDL 3RX | EPDL 2RX | EPD DR | EPD ABIR | EPD ABIC | EPD RX |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | EPD CTX | EPD ATX | EPD L3TX | EPD L2TX | EPD DW | EPD ABO | EPD ABM | EPD TX |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 | EPDCRX | 0 | R/W | R/W | Error correction disable for CXS Tx path (PCIe Rx path). |
| 22 | EPDARX | 0 | R/W | R/W | Error correction disable for ADM Rx path. |
| 21 | EPDL3RX | 0 | R/W | R/W | Error correction disable for layer 3 Rx path. |
| 20 | EPDL2RX | 0 | R/W | R/W | Error correction disable for layer 2 Rx path. |
| 19 | EPDDR | 0 | R/W | R/W | Error correction disable for DMA read engine. |
| 18 | EPDABIR | 0 | R/W | R/W | Error correction disable for AXI bridge inbound request path. |
| 17 | EPDABIC | 0 | R/W | R/W | Error correction disable for AXI bridge inbound completion composer. |
| 16 | EPDRX | 0 | R/W | R/W | Global error correction disable for all Rx layers. |
| 15 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | EPDCTX | 0 | R/W | R/W | Error correction disable for CXS Rx path (PCIe Tx path). |
| 6 | EPDATX | 0 | R/W | R/W | Error correction disable for Adm Tx path. |
| 5 | EPDL3TX | 0 | R/W | R/W | Error correction disable for layer 3 Tx path. |
| 4 | EPDL2TX | 0 | R/W | R/W | Error correction disable for layer 2 Tx path. |
| 3 | EPDDW | 0 | R/W | R/W | Error correction disable for DMA write engine. |
| 2 | EPDABO | 0 | R/W | R/W | Error correction disable for AXI bridge outbound request path. |
| 1 | EPDABM | 0 | R/W | R/W | Error correction disable for AXI bridge master completion buffer. |
| 0 | EPDTX | 0 | R/W | R/W | Global error correction disable for all Tx layers. |

### 104.2.127 Vendor-Specific Extended 2 Capability Register 3 function i (VSE2CAP3F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Corrected error (1-bit ECC) counter selection and control

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CORRCS | | | | | | | | CORRCSR | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | COR RENC | — | — | — | CORR CLRC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | CORRCS | H'0 | R/W | R/W | Counter selection.<br>This field selects the counter ID (within the region defined by CORR_COUNTER_SELECTION_REGION) whose contents can be read from the VSE2CAP4F*i* register. |
| 23 to 20 | CORRCSR | H'0 | R/W | R/W | Select correctable counter region:<br>H'0: Region select for Adm Rx path<br>H'1: Region select for layer 3 Rx path<br>H'2: Region select for layer 2 Rx path<br>H'3: Region select for DMA inbound path<br>H'4: Region select for AXI bridge inbound request path<br>H'5: Region select for AXI bridge inbound completion composer path<br>H'6: Region select for Adm Tx path<br>H'7: Region select for layer 3 Tx path<br>H'8: Region select for layer 2 Tx path<br>H'9: Region select for DMA outbound path<br>H'A: Region select for AXI bridge outbound request path<br>H'B: Region select for AXI bridge outbound master completion buffer path<br>H'C: Reserved<br>H'F: Reserved |
| 19 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | CORRENC | 1 | R/W | R/W | Enable correctable errors counters.<br>1: counters increment when the controller detects a correctable error<br>0: counters are frozen |
| 3 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | CORRCLRC | 0 | W | W | Clear all correctable error counters. |

### 104.2.128 Vendor-Specific Extended 2 Capability Register 4 function i (VSE2CAP4F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Corrected error (1-bit ECC) counter data

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CORRCNTSEL | | | | | | | | CORRCNTSELR | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CORRCNT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |
| Internal bus R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | CORRCNTSEL | H'0 | R/W | R/W | Counter selection.<br>Returns the value set in the CORRCNTSEL field of the RASDP_CORR_COUNTER_CTRL_OFF register. |
| 23 to 20 | CORRCNTSELR | H'0 | R/W | R/W | Selected correctable counter region:<br>H'0: Region select for Adm Rx path<br>H'1: Region select for layer 3 Rx path<br>H'2: Region select for layer 2 Rx path<br>H'3: Region select for DMA inbound path<br>H'4: Region select for AXI bridge inbound request path<br>H'5: Region select for AXI bridge inbound completion composer path<br>H'6: Region select for Adm Tx path<br>H'7: Region select for layer 3 Tx path<br>H'8: Region select for layer 2 Tx path<br>H'9: Region select for DMA outbound path<br>H'a: Region select for AXI bridge outbound request path<br>H'b: Region select for AXI bridge outbound master completion buffer path<br>H'c: Reserved<br>H'f: Reserved |
| 19 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | CORRCNT | H'0 | W | W | Current corrected error count for the selected counter.<br>Write only bit. |

### 104.2.129 Vendor-Specific Extended 2 Capability Register 5 function i (VSE2CAP5Fi) i = 0, 1

**Common to Header TYPE00/01**

- Uncorrected error (2-bit ECC and parity) counter selection and control

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UNCORRCS | | | | | | | | UNCORRCSR | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | UNCORREC | — | — | — | UNCORRCC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | UNCORRCS | H'0 | R/W | R/W | Counter selection.<br>This field selects the counter ID (within the region defined by UNCORR_COUNTER_SELECTION_REGION) whose contents can be read from the VSE2CAP6F*i* register. |
| 23 to 20 | UNCORRCSR | H'0 | R/W | R/W | Select uncorrectable counter region:<br>H'0: Region select for Adm Rx path<br>H'1: Region select for layer 3 Rx path<br>H'2: Region select for layer 2 Rx path<br>H'3: Region select for DMA inbound path<br>H'4: Region select for AXI bridge inbound request path<br>H'5: Region select for AXI bridge inbound completion composer path<br>H'6: Region select for Adm Tx path<br>H'7: Region select for layer 3 Tx path<br>H'8: Region select for layer 2 Tx path<br>H'9: Region select for DMA outbound path<br>H'a: Region select for AXI bridge outbound request path<br>H'b: Region select for AXI bridge outbound master completion buffer path<br>H'c: Reserved<br>H'f: Reserved |
| 19 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | UNCORREC | 1 | R/W | R/W | Enable uncorrectable errors counters.<br>1: enables the counters to increment on detected correctable errors<br>0: counters are frozen |
| 3 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | UNCORRCC | 0 | W | W | Clear uncorrectable errors counters.<br>When asserted causes all counters tracking the uncorrectable errors to be cleared |

### 104.2.130 Vendor-Specific Extended 2 Capability Register 6 function i (VSE2CAP6F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Uncorrected error (2-bit ECC and parity) counter data

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UNCORRCSD | | | | | | | | UNCORRCSDR | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | UNCORRC | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | UNCORRCSD | H’0 | R | R | Counter selection.<br>Returns the value set in the UNCORRCS field of the RASDP_UNCORR_COUNTER_CTRL_OFF register. |
| 23 to 20 | UNCORRCSDR | H’0 | R | R | Selected uncorrectable counter region:<br>H'0: Region select for Adm Rx path<br>H'1: Region select for layer 3 Rx path<br>H'2: Region select for layer 2 Rx path<br>H'3: Region select for DMA inbound path<br>H'4: Region select for AXI bridge inbound request path<br>H'5: Region select for AXI bridge inbound completion composer path<br>H'6: Region select for Adm Tx path<br>H'7: Region select for layer 3 Tx path<br>H'8: Region select for layer 2 Tx path<br>H'9: Region select for DMA outbound path<br>H'A: Region select for AXI bridge outbound request path<br>H'B: Region select for AXI bridge outbound master completion buffer path<br>H'C: Reserved<br>H'F: Reserved |
| 19 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | UNCORRC | H’0 | R | R | Current uncorrected error count for the selected counter |

### 104.2.131 Vendor-Specific Extended 2 Capability Register 7 function i (VSE2CAP7F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Error injection control for the following features: - 1-bit or 2-bit injection - Continuous or fixed-number (n) injection modes - Global enable/disable - Selectable location where injection occurs.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ERRIL | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ERRICC | | | | | | | | — | — | ERRJT | | — | — | — | ERRIE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R | R | R | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | ERRIL | H'0 | R/W | R/W | Error injection location.<br>Selects where error injection takes place. |
| 15 to 8 | ERRICC | H'0 | R/W | R/W | Error injection count.<br>H'0: errors are inserted in every TLP until you clear ERRIE.<br>H'1: one errors injected<br>H'2: two errors injected<br>H'n: amount of errors injected |
| 7, 6 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5, 4 | ERRJT | H'0 | R/W | R/W | Error injection type:<br>H'0: none<br>H'1: 1-bit<br>H'2: 2-bit |
| 3 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | ERRIE | 0 | R/W | R/W | Error injection global enable.<br>When set enables the error insertion logic. |

### 104.2.132 Vendor-Specific Extended 2 Capability Register 8 function i (VSE2CAP8F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Corrected errors locations

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LLCE | | | | | | | | RLCE | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LFCE | | | | | | | | RFCE | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | LLCE | H'0 | R | R | Location/ID of the last corrected error within the region defined by RLCE. |
| 23 to 20 | RLCE | H'E | R | R | Region of the last corrected error:<br>H'0: Region select for Adm Rx path<br>H'1: Region select for layer 3 Rx path<br>H'2: Region select for layer 2 Rx path<br>H'3: Region select for DMA inbound path<br>H'4: Region select for AXI bridge inbound request path<br>H'5: Region select for AXI bridge inbound completion composer path<br>H'6: Region select for Adm Tx path<br>H'7: Region select for layer 3 Tx path<br>H'8: Region select for layer 2 Tx path<br>H'9: Region select for DMA outbound path<br>H'A: Region select for AXI bridge outbound request path<br>H'B: Region select for AXI bridge outbound master completion buffer path<br>H'C: Reserved<br>H'D: Reserved<br>H'E: Reserved<br>H'F: Reserved |
| 19 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | LFCE | H'0 | R | R | Location/ID of the first corrected error within the region defined by RFCE. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 7 to 4 | RFCE | H'E | R | R | Region of the first corrected error:<br>H'0: Region select for Adm Rx path<br>H'1: Region select for layer 3 Rx path<br>H'2: Region select for layer 2 Rx path<br>H'3: Region select for DMA inbound path<br>H'4: Region select for AXI bridge inbound request path<br>H'5: Region select for AXI bridge inbound completion composer path<br>H'6: Region select for Adm Tx path<br>H'7: Region select for layer 3 Tx path<br>H'8: Region select for layer 2 Tx path<br>H'9: Region select for DMA outbound path<br>H'A: Region select for AXI bridge outbound request path<br>H'B: Region select for AXI bridge outbound master completion buffer path<br>H'C: Reserved<br>H'D: Reserved<br>H'E: Reserved<br>H'F: Reserved |
| 3 to 0 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.133 Vendor-Specific Extended 2 Capability Register 9 function i (VSE2CAP9F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Uncorrected errors locations

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LLUE | | | | | | | | RLUE | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LFUE | | | | | | | | RFUE | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | LLUE | H'0 | R | R | Location/ID of the last uncorrected error within the region defined by RLUE. |
| 23 to 20 | RLUE | H'E | R | R | Region of the last uncorrected error:<br>H'0: Region select for Adm Rx path<br>H'1: Region select for layer 3 Rx path<br>H'2: Region select for layer 2 Rx path<br>H'3: Region select for DMA inbound path<br>H'4: Region select for AXI bridge inbound request path<br>H'5: Region select for AXI bridge inbound completion composer path<br>H'6: Region select for Adm Tx path<br>H'7: Region select for layer 3 Tx path<br>H'8: Region select for layer 2 Tx path<br>H'9: Region select for DMA outbound path<br>H'A: Region select for AXI bridge outbound request path<br>H'B: Region select for AXI bridge outbound master completion buffer path<br>H'C: Reserved<br>H'D: Reserved<br>H'E: Reserved<br>H'F: Reserved |
| 19 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | LFUE | H'0 | R | R | Location/ID of the first uncorrected error within the region defined by RFUE. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 7 to 4 | RFUE | H'E | R | R | Region of the first uncorrected error:<br>H'0: Region select for Adm Rx path<br>H'1: Region select for layer 3 Rx path<br>H'2: Region select for layer 2 Rx path<br>H'3: Region select for DMA inbound path<br>H'4: Region select for AXI bridge inbound request path<br>H'5: Region select for AXI bridge inbound completion composer path<br>H'6: Region select for Adm Tx path<br>H'7: Region select for layer 3 Tx path<br>H'8: Region select for layer 2 Tx path<br>H'9: Region select for DMA outbound path<br>H'A: Region select for AXI bridge outbound request path<br>H'b: Region select for AXI bridge outbound master completion buffer path<br>H'c: Reserved<br>H'd: Reserved<br>H'e: Reserved<br>H'f: Reserved |
| 3 to 0 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.134 Vendor-Specific Extended 2 Capability Register 10 function i (VSE2CAP10F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- RASDP error mode enable

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ALDEN | ERRME |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | ALDEN | 0 | R/W | R/W | Write '1' to enable the controller to bring the link down when the controller enters RASDP error mode. |
| 0 | ERRME | 0 | R/W | R/W | Write '1' to enable the controller enter RASDP error mode when it detects an uncorrectable error. |

### 104.2.135 Vendor-Specific Extended 2 Capability Register 11 function i (VSE2CAP11F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- Exit RASDP error mode

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

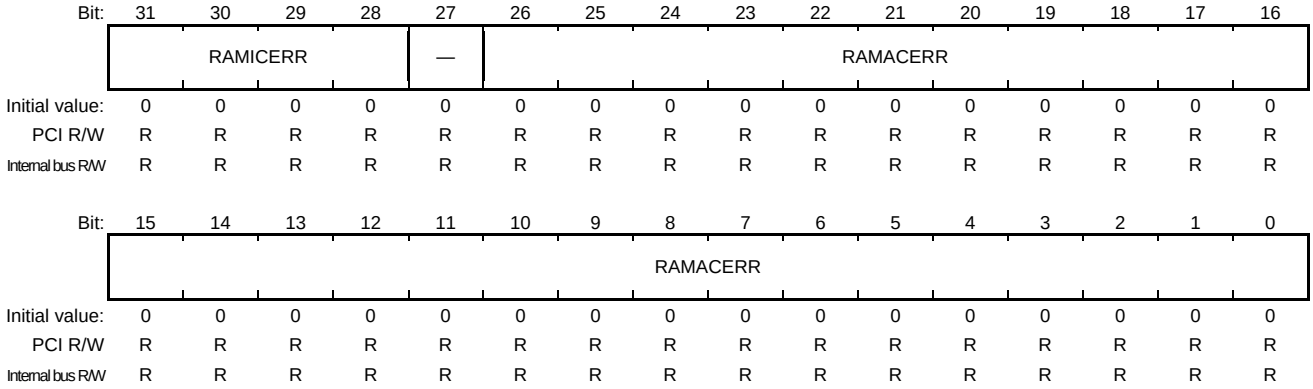
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | EMC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | EMC | 0 | W | W | Write '1' to take the controller out of RASDP error mode.<br>The controller will then report uncorrectable errors (through AER internal error reporting) and also stop nullifying/discarding TLPs. |

### 104.2.136 Vendor-Specific Extended 2 Capability Register 12 function i (VSE2CAP12F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- RAM Address where a corrected error (1-bit ECC) has been detected.




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 28 | RAMICERR | H'0 | R | R | RAM index where a corrected error (1-bit ECC) has been detected. |
| 27 | — | 0 | R | R | Reserved<br>Theis bit is always read as 0. The write value should always be 0. |
| 26 to 0 | RAMACERR | H'0 | R | R | RAM Address where a corrected error (1-bit ECC) has been detected. |

### 104.2.137 Vendor-Specific Extended 2 Capability Register 13 function i (VSE2CAP13F*i*) *i* = 0, 1

**Common to Header TYPE00/01**

- RAM Address where an uncorrected error (2-bit ECC) has been detected

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RAMIUERR | | | | — | RAMAUERR | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

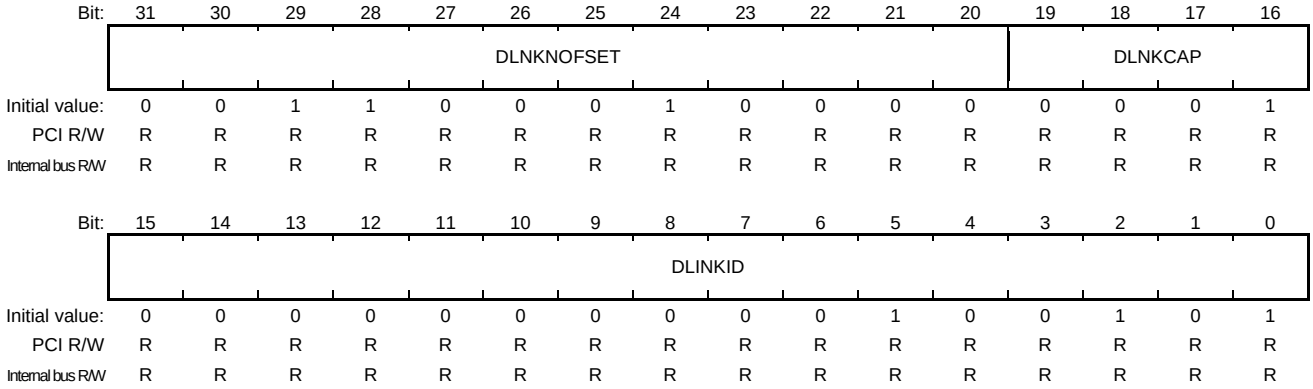
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RAMAUERR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 28 | RAMIUERR | H'0 | R | R | RAM index where an uncorrected error (2-bit ECC) has been detected. |
| 27 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 26 to 0 | RAMAUERR | H'0 | R | R | RAM Address where an uncorrected error (2-bit ECC) has been detected. |

### 104.2.138 Data Link Feature Extended Capability Register 0 function 0 (DLFECAP0F0)

### Common to Header TYPE00/01

- Data Link Feature Extended Capability Header

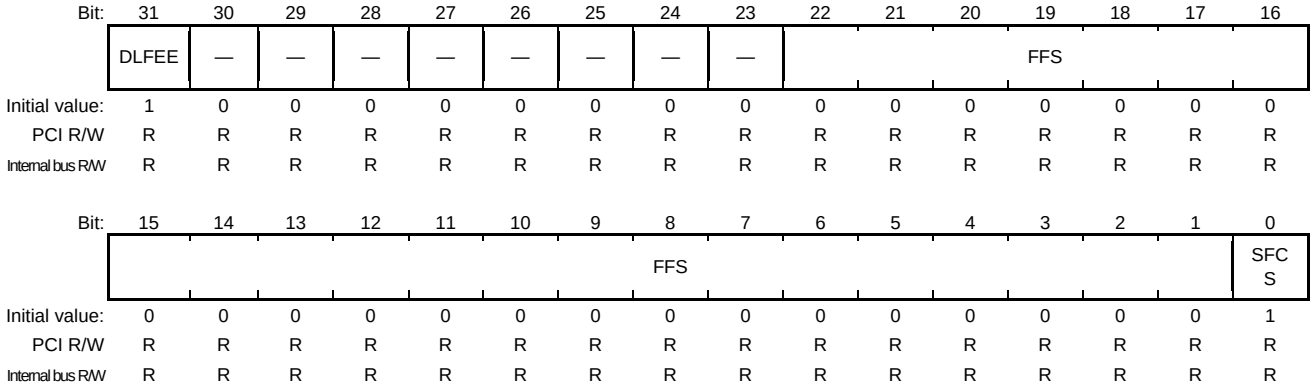



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | DLNKNOFSET | H'310 | R | R | Next Capability Offset |
| 19 to 16 | DLNKCAP | H'1 | R | R | Capability Version<br>Indicates the version of the capability structure present. |
| 15 to 0 | DLINKID | H'25 | R | R | PCI Express Extended Capability ID<br>Indicates the nature and format of the extended capability. |

### 104.2.139 Data Link Feature Extended Capability Register 1 function 0 (DLFECAP1F0)

### Common to Header TYPE00/01

- Data Link Feature Capabilities

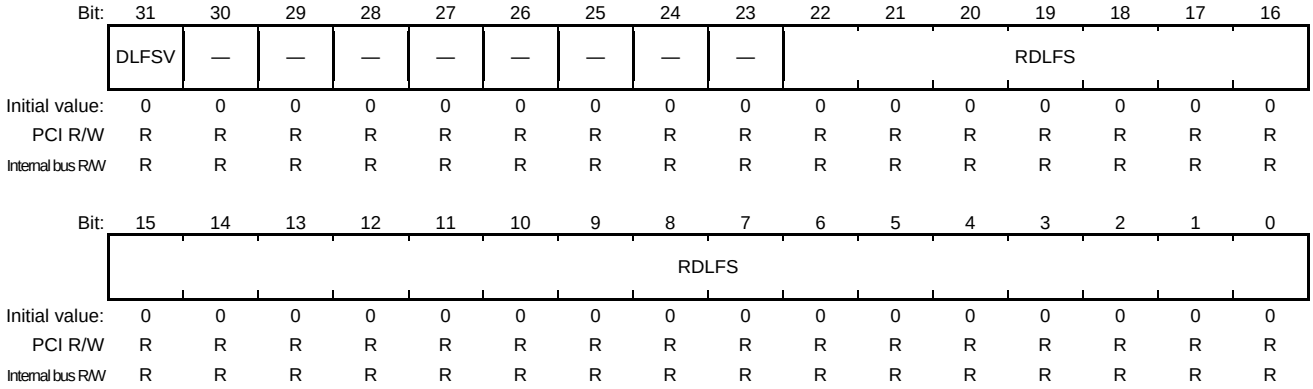



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | DLFEE | 1 | R | R | Data Link Feature Exchange Enable<br>If Set, this bit indicates that this Port will enter the DL_Feature negotiation state. |
| 30 to 23 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 22 to 1 | FFS | H'0 | R | R | Local Data Link Feature Supported<br>This field contains the Feature Supported value used when this Port sends a Data Link Feature DLLP. |
| 0 | SFCS | 1 | R | R | Local Scaled Flow Control Supported<br>This bit indicates that this Port supports the Scaled Flow Control Feature. |

### 104.2.140 Data Link Feature Extended Capability Register 2 function 0 (DLFECAP2F0)

### Common to Header TYPE00/01

- Data Link Feature Status Register

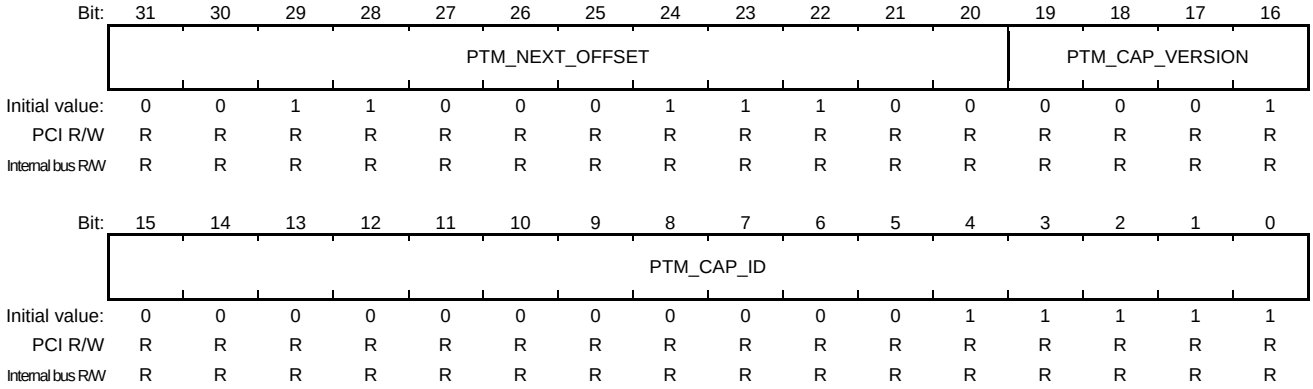



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | DLFSV | 0 | R | R | Remote Data Link Feature Supported Valid<br>This bit indicates that the Port has received a Data Link Feature DLLP in state DL_Feature and that the Remote Data Link Feature Supported field is meaningful. |
| 30 to 23 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 22 to 0 | RDLFS | H'0 | R | R | Remote Data Link Feature Supported<br>These bits indicate that the Remote Port supports the corresponding Data Link Feature. |

### 104.2.141 Precision Time Management (PTM) Capability Register 0 function 0 (PTMCAP0F0)

**Common to Header TYPE00/01**

- Precision Time Measurement Capability Header.




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | PTM_NEXT_OFFSET | H'31C | R | R | Precision Time Measurement PCI Express Extended Capability Next Offset. |
| 19 to 16 | PTM_CAP_VERSION | H'1 | R | R | Precision Time Measurement PCI Express Extended<br>Capability Version. |
| 15 to 0 | PTM_CAP_ID | H'1F | R | R | Precision Time Measurement PCI Express Extended Capability ID. |

### 104.2.142 Precision Time Management (PTM) Capability Register 1 function 0 (PTMCAP1F0)

### Common to Header TYPE00/01

- PTM Capability Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_CLK_GRAN | | | | | | | | — | — | — | — | — | PTM_ROOT _CAPABLE | PTM_RES_ CAPABLE | PTM_REQ_ CAPABLE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | RSVDP | RSVDP | RSVDP | RSVDP | RSVDP | RSVDP | RSVDP | RSVDP | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | PTM_CLK_GRAN | H'0 | RSVDP | R/W | PTM Local Clock Granularity<br>0000 0001b – 1111 1110b: Indicates the period of this Time Source's local clock in ns |
| 7 to 3 | — | All 0 | R | R | Reserved |
| 2 | PTM_ROOT_ CAPABLE | 0 | R | R/W | PTM Root Capable<br>Root Ports, RCRB's and Switches are permitted to set this bit to 1b, if they implement a PTM Time Source Role and are capable of serving as the PTM Root. |
| 1 | PTM_RES_CAPABLE | 0 | R | R/W | PTM Responder Capable<br>Root Ports and RCRB's are permitted to, and Switches supporting PTM must, set this bit to 1b to indicate they implement the PTM Responder role.<br>If PTM Root Capable is Set, this bit must be Set to 1b. |
| 0 | PTM_REQ_CAPABLE | 0 | R | R/W | PTM Requester Capable<br>Endpoints and RCiEPs are permitted to, and Switches supporting PTM must, set this bit to 1b to indicate they implement the PTM Requester role. |

### 104.2.143 Precision Time Management (PTM) Capability Register 2 function 0 (PTMCAP2F0)

**Common to Header TYPE00/01**

- PTM Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

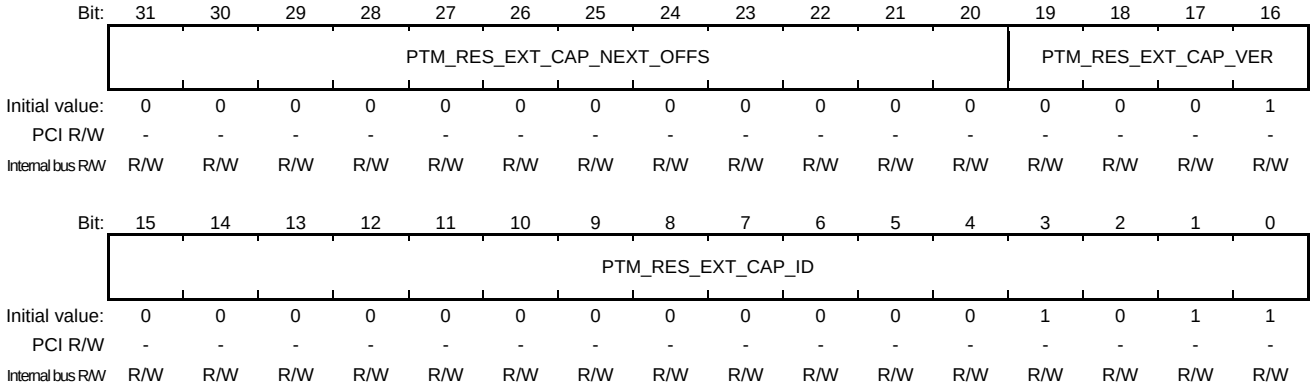
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EFF_GRAN | | | | | | | | — | — | — | — | — | — | ROOT_SELECT | PTM_ENABLE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | EFF_GRAN | H'0 | R/W | R/W | PTM Effective Granularity<br>For Functions implementing the PTM Requester Role, this field provides information relating to the expected accuracy of the PTM clock, but does not otherwise affect the PTM mechanism.<br>B'0000 0000: Unknown PTM granularity – one or more Switches between this Function and the PTM Root reported a Local Clock Granularity value of B'0000 0000.<br>B'0000 0001 – 1111 1110b: Indicates the effective PTM granularity in ns.<br>B'1111 1111: Indicates the effective PTM granularity is greater than 254 ns. |
| 7 to 2 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 1 | ROOT_SELECT | 0 | R/W | R/W | Root Select<br>When Set, if the PTM Enable bit is also Set, this Time Source is the PTM Root. |
| 0 | PTM_ENABLE | 0 | R/W | R/W | PTM Enable<br>When Set, this Function is permitted to participate in the PTM mechanism according to its selected role(s). |

### 104.2.144 PTM Responder Capability Structure (VSEC) Register 0 function 0 (PTMRSCAP0F0)

**Header TYPE01 only**

- Precision Time Measurement Responder Capability Header (VSEC)

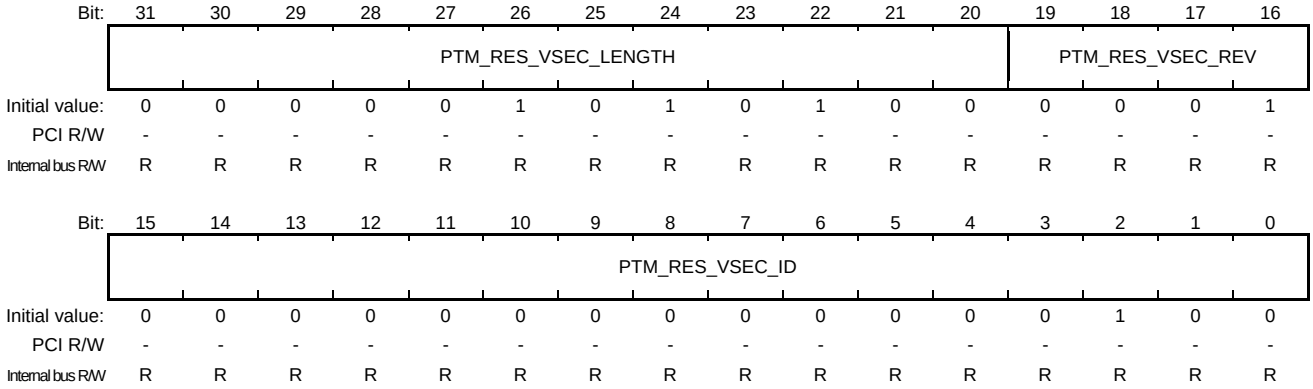



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | PTM_RES_EXT_CAP_NEXT_OFFS | H'000 | — | R/W | Precision Time Measurement Responder VSEC Next Pointer |
| 19 to 16 | PTM_RES_EXT_CAP_VER | H'1 | — | R/W | Precision Time Measurement Responder VSEC Version |
| 15 to 0 | PTM_RES_EXT_CAP_ID | H'B | — | R/W | Precision Time Measurement Responder VSEC ID |

### 104.2.145 PTM Responder Capability Structure (VSEC) Register 1 function 0 (PTMRSCAP1F0)

**Header TYPE01 only**

- Precision Time Measurement Responder Vendor Specific Header




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | PTM_RES_ VSEC_LENGTH | H'54 | — | R | PTM Responder VSEC Length |
| 19 to 16 | PTM_RES_ VSEC_REV | H'1 | — | R | PTM Responder VSEC Revision |
| 15 to 0 | PTM_RES_ VSEC_ID | H'4 | — | R | PTM Responder VSEC ID |

### 104.2.146 PTM Responder Capability Structure (VSEC) Register 2 function 0 (PTMRSCAP2F0)

**Header TYPE01 only**

- PTM Responder Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PTM_RE | PTM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | - |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | PTM_RES_PROP_DELAY_BREV | 0 | R/W | R/W | PTM Requester Propagation Delay Byte Reversal - Reverse<br>the order of bytes in the PTM Propagation Delay data word<br>of the PTM ResponseD Message, for compatibility with<br>previous revisions |
| 0 | PTM_RES_<br>CCONTEXT_VALID | 0 | — | R | PTM Responder Control Context Valid<br>PTM Local Timing is valid.<br>This bit is set over the DBI. A speed change or aux_clk_active will set this bit low. |

### 104.2.147 PTM Responder Capability Structure (VSEC) Register 3 function 0 (PTMRSCAP3F0)

**Header TYPE01 only**

- PTM Responder Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

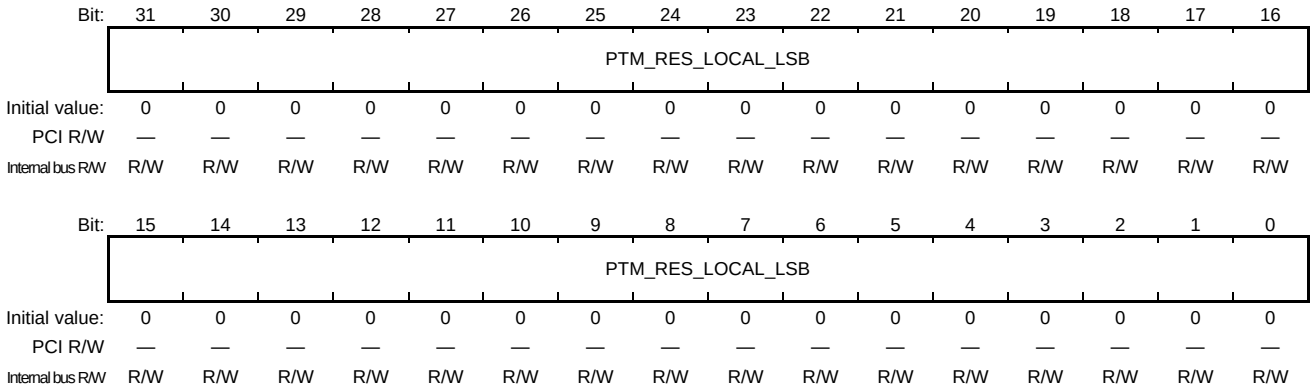
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PTM_RES_REQUEST_RECEIVED | PTM_RES_CONTEXT_VALID |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | PTM_RES_REQUEST_RECEIVED | 0 | R | R | PTM 1st Request Received<br>PTM Responder has received the first PTM Request Message. Upon receipt of a second PTM Request Message a ResponseD message with timing information will be sent from the Responder, if the context is valid. |
| 0 | PTM_RES_CONTEXT_VALID | 0 | R | R | PTM Responder Status Context Valid<br>PTM Local Timing Context is Valid. Value set from upstream port Requester in a Switch. Shadows the same in the control register in all other products. |

### 104.2.148 PTM Responder Capability Structure (VSEC) Register 4 function 0 (PTMRSCAP4F0)

**Header TYPE01 only**

- PTM Responder Local Clock LSB

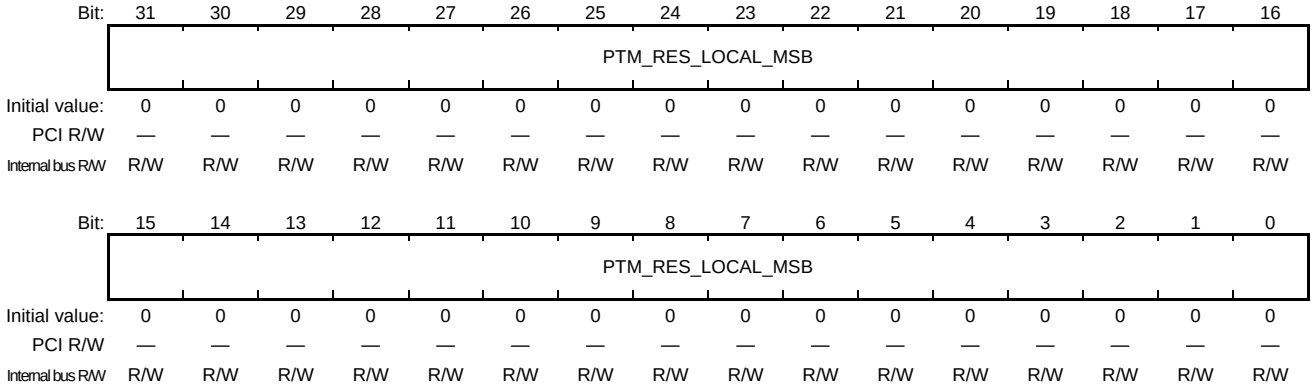



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_ LOCAL_LSB | H'0 | — | R/W | PTM Responder Local Clock LSB<br>Lower 32 bits of local timer value. |

### 104.2.149 PTM Responder Capability Structure (VSEC) Register 5 function 0 (PTMRSCAP5F0)

**Header TYPE01 only**

- PTM Responder Local Clock MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_LOCAL_MSB | H'0 | — | R/W | PTM Responder Local Clock MSB<br>Upper 32 bits of local timer value. |

### 104.2.150 PTM Responder Capability Structure (VSEC) Register 6 function 0 (PTMRSCAP6F0)

**Header TYPE01 only**

- PTM Responder T2 Timestamp LSB

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_RES_T2_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

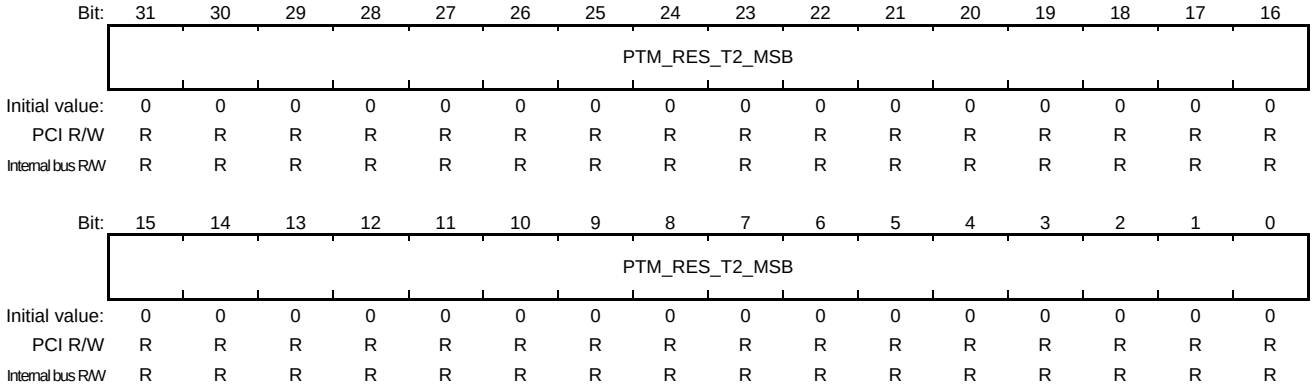
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_RES_T2_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_T2_LSB | H'0 | R | R | PTM Responder T2 Timestamp LSB<br>Lower 32 bits of the T2 Timestamp value. |

### 104.2.151 PTM Responder Capability Structure (VSEC) Register 7 function 0 (PTMRSCAP7F0)

**Header TYPE01 only**

- PTM Responder T2 Timestamp MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_T2_MSB | H'0 | R | R | PTM Responder T2 Timestamp MSB<br>Upper 32 bits of the T2 Timestamp value. |

### 104.2.152 PTM Responder Capability Structure (VSEC) Register 8 function 0 (PTMRSCAP8F0)

**Header TYPE01 only**

- PTM Responder T2 Previous Timestamp LSB

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_RES_T2P_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

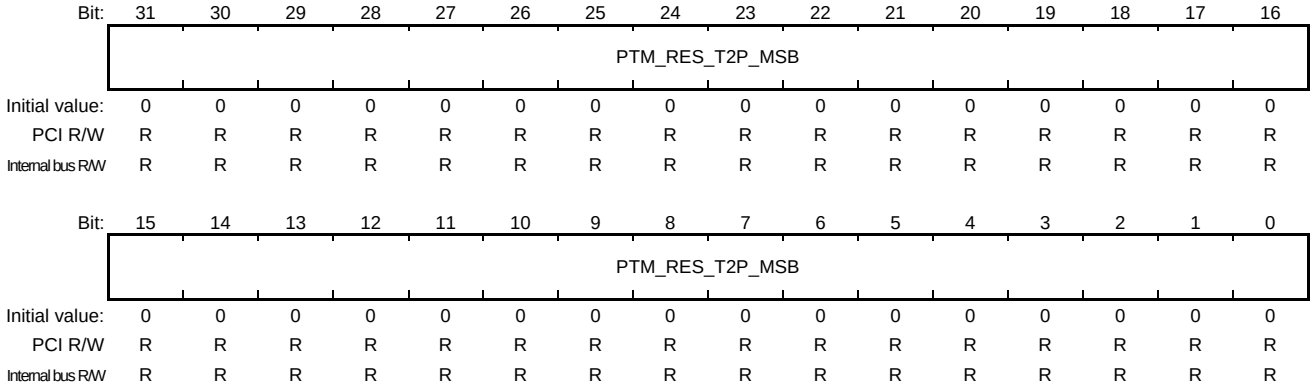
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_RES_T2P_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_T2P_LSB | H'0 | R | R | PTM Responder T2 Previous Timestamp LSB<br>Lower 32 bits of the previously stored T2 Timestamp value. |

### 104.2.153 PTM Responder Capability Structure (VSEC) Register 9 function 0 (PTMRSCAP9F0)

**Header TYPE01 only**

- PTM Responder T2 Previous Timestamp MSB

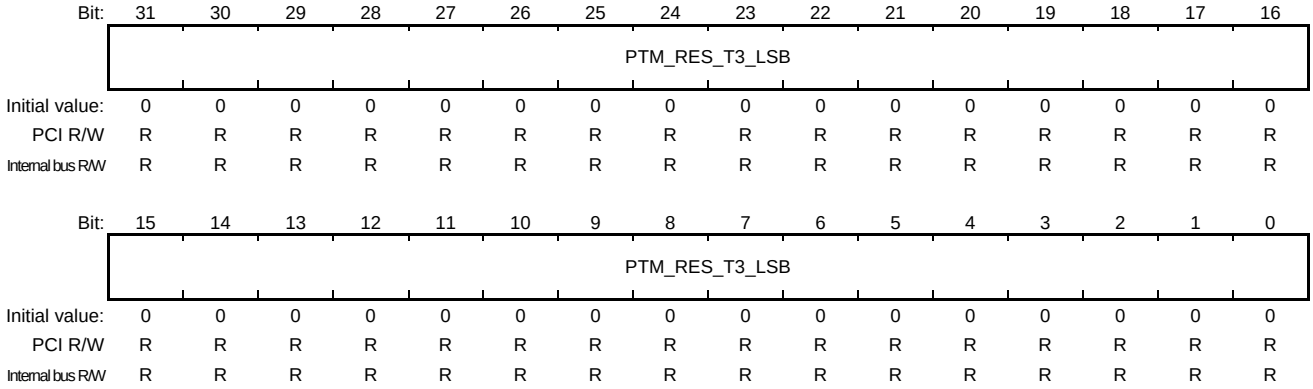



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_T2P_MSB | H'0 | R | R | PTM Responder T2 Previous Timestamp LSB<br>Lower 32 bits of the previously stored T2 Timestamp value. |

### 104.2.154 PTM Responder Capability Structure (VSEC) Register 10 function 0 (PTMRSCAP10F0)

**Header TYPE01 only**

- PTM Responder T3 Timestamp LSB

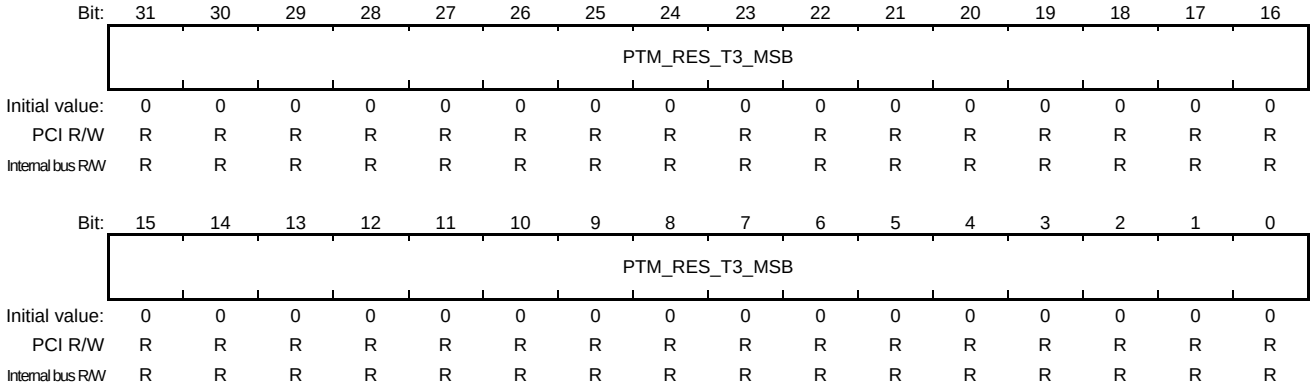



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_T3_LSB | H'0 | R | R | PTM Responder T3 Timestamp LSB<br>Lower 32 bits of the T3 Timestamp value. |

### 104.2.155 PTM Responder Capability Structure (VSEC) Register 11 function 0 (PTMRSCAP11F0)

**Header TYPE01 only**

- PTM Responder T3 Timestamp MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_T3_MSB | H'0 | R | R | PTM Responder T3 Timestamp MSB<br>Upper 32 bits of the T3 Timestamp value. |

### 104.2.156 PTM Responder Capability Structure (VSEC) Register 12 function 0 (PTMRSCAP12F0)

**Header TYPE01 only**

- PTM Responder T3 Previous Timestamp LSB

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_RES_T3P_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

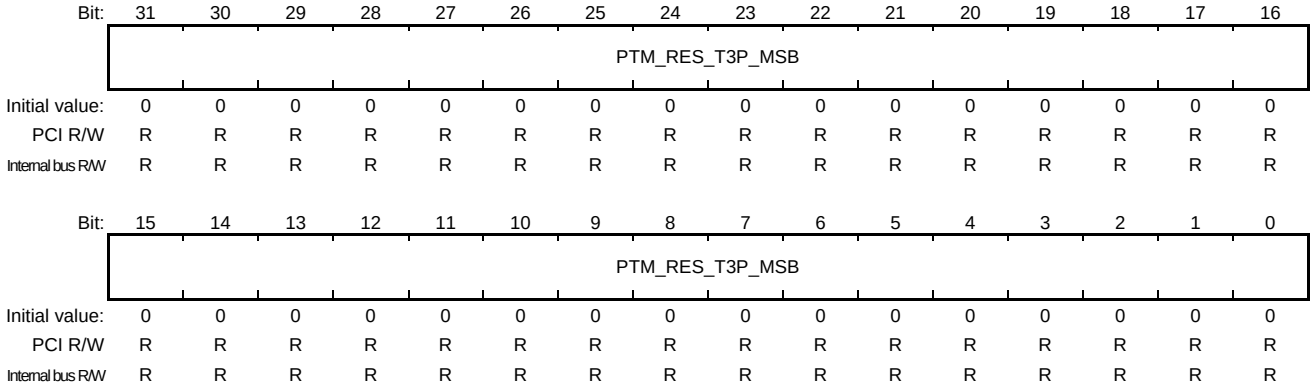
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_RES_T3P_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_T3P_LSB | H'0 | R | R | PTM Responder T3 Previous Timestamp LSB<br>Lower 32 bits of the previously stored T3 Timestamp value. |

### 104.2.157 PTM Responder Capability Structure (VSEC) Register 13 function 0 (PTMRSCAP13F0)

**Header TYPE01 only**

- PTM Responder T3 Previous Timestamp MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_RES_T3P_MSB | H'0 | R | R | PTM Responder T3 Previous Timestamp MSB<br>Upper 32 bits of the previously stored T3 Timestamp value. |

### 104.2.158 PTM Responder Capability Structure (VSEC) Register 14 function 0 (PTMRSCAP14F0)

**Header TYPE01 only**

- PTM Responder TX Latency

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PTM_RES_TX_LATENCY | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | PTM_RES_<br>TX_LATENCY | H'0 | R/W | R/W | PTM Responder TX Latency<br>Responder Transmit path latency (12 bit wide) |

### 104.2.159 PTM Responder Capability Structure (VSEC) Register 15 function 0 (PTMRSCAP15F0)

**Header TYPE01 only**

- PTM Responder RX Latency

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PTM_RES_RX_LATENCY | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | PTM_RES_ RX_LATENCY | H'0 | R/W | R/W | PTM Responder RX Latency<br>Responder Receive path latency (12 bit wide). |

### 104.2.160 PTM Requestor Capability Structure (VSEC) Register 0 function *i* (PTMRQCAP0F*i*) *i* = 0, 1

**Header TYPE00 only**

- Precision Time Measurement Requester Capability Header (VSEC).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_EXT_CAP_NEXT_OFFS | | | | | | | | | | | | PTM_REQ_EXT_CAP_VER | | | |
| Initial value: | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_EXT_CAP_ID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | PTM_REQ_EXT_CAP _NEXT_OFFS | H'380 | R | R | Precision Time Measurement Requester VSEC Next Pointer |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 19 to 16 | PTM_REQ_EXT_CAP_VER | H'1 | R | R | Precision Time Measurement Requester VSEC Version |
| 15 to 0 | PTM_REQ_EXT_CAP_ID | H'B | R | R | Precision Time Measurement Requester VSEC ID |

### 104.2.161 PTM Requestor Capability Structure (VSEC) Register 1 function *i* (PTMRQCAP1F*i*) *i* = 0, 1

**Header TYPE00 only**

- Precision Time Measurement Requester Vendor Specific Header

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_VSEC_LENGTH | | | | | | | | | | | | PTM_REQ_VSEC_REV | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

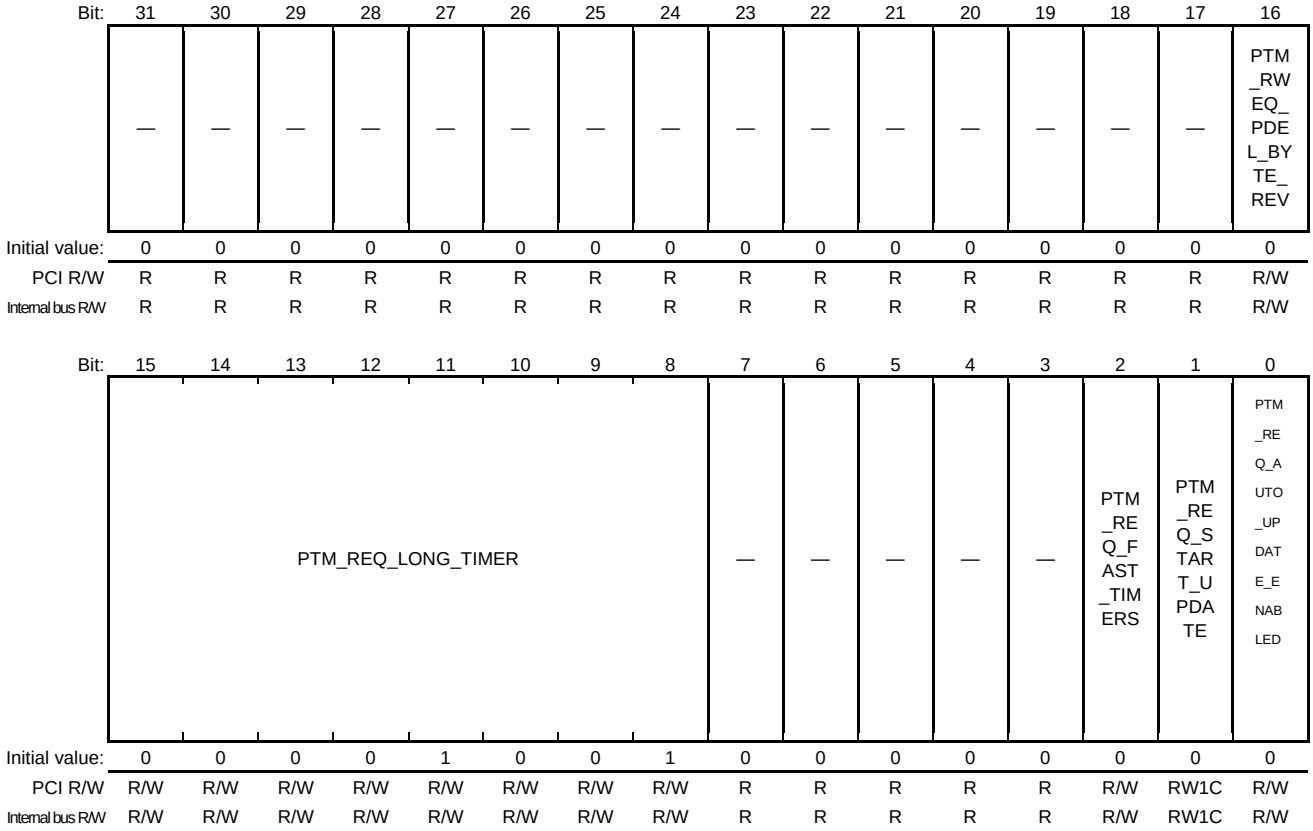
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_VSEC_ID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | PTM_REQ_VSEC_LENGTH | H'54 | R | R/W | PTM Requester VSEC Length. |
| 19 to 16 | PTM_REQ_VSEC_REV | H'1 | R | R/W | PTM Requester VSEC Length. |
| 15 to 0 | PTM_REQ_VSEC_ID | H'3 | R | R/W | PTM Requester VSEC ID |

### 104.2.162 PTM Requestor Capability Structure (VSEC) Register 2 function *i* (PTMRQCAP2F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Vendor Specific Control Register




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | PTM_RWEQ_PDEL_BYTE_REV | 0 | R/W | R/W | PTM Requester Propagation Delay Byte Reversal - Reverse<br>the order of bytes in the PTM Propagation Delay data word<br>of the PTM ResponseD Message, for compatibility with<br>previous revisions. |
| 15 to 8 | PTM_REQ_ LONG_TIMER | H'9 | R/W | R/W | PTM Requester Long Timer<br>Determines the period between each auto update PTM Dialogue in miliseconds.<br>Update period is the register value +1 milisecond.<br>For the Switch product this value must not be set larger than 0x9 for spec compliance. |
| 7 to 3 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 2 | PTM_REQ_FAST_TIMERS | 0 | R/W | R/W | PTM Fast Timers<br>Debug mode for PTM Timers.<br>The 100us timer output will go high at 30us and the 10ms timer output will go high at 100us (The Long Timer Value is ignored). |
| 1 | PTM_REQ_ START_UPDATE | 0 | RW1C | RW1C | PTM Requester Start Update<br>When set the PTM Requester will attempt a PTM Dialogue to update it's context; This bit is self clearing. |
| 0 | PTM_REQ_AUTO_ UPDATE_ENABLED | 0 | R/W | R/W | PTM Requester Auto Update Enabled<br>When enabled PTM Requester will automatically atempt to update it's context every 10ms. |

### 104.2.163 PTM Requestor Capability Structure (VSEC) Register 3 function *i* (PTMRQCAP3F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Vendor Specific Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

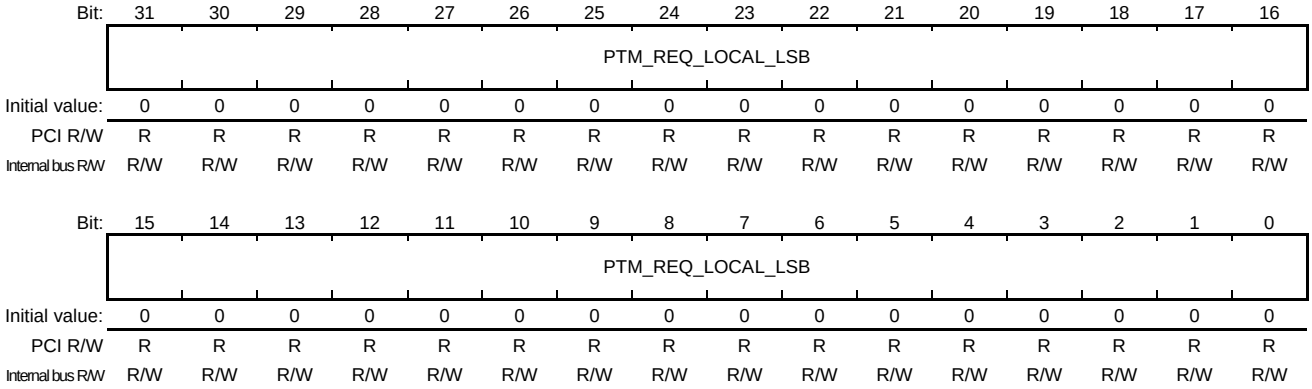
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PTM_ | PTM_ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | PTM_REQ_MANUAL_UPDATE_ALLOWED | 0 | R | R | PTM Requester Manual Update Allowed<br>Indicates whether or not a Manual Update can be signaled. |
| 0 | PTM_REQ_CONTEXT_VALID | 0 | R | R | PTM Requester Context Valid<br>Indicate that the Timing Context is valid. |

### 104.2.164 PTM Requestor Capability Structure (VSEC) Register 4 function *i* (PTMRQCAP4F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Local Clock LSB

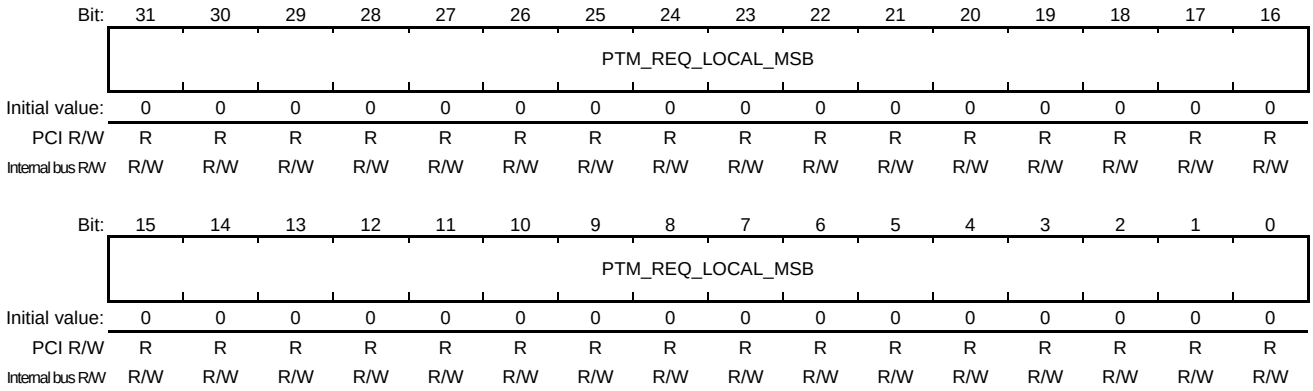



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_ LOCAL_LSB | H'0 | R | R/W | PTM Requester Local Clock LSB<br>Lower 32 bits of local timer value. |

### 104.2.165 PTM Requestor Capability Structure (VSEC) Register 5 function *i* (PTMRQCAP5F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Local Clock MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_ LOCAL_MSB | H'0 | R | R/W | PTM Requester Local Clock MSB<br>Upper 32 bits of local timer value. |

### 104.2.166 PTM Requestor Capability Structure (VSEC) Register 6 function *i* (PTMRQCAP6F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester T1 Timestamp LSB.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_T1_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

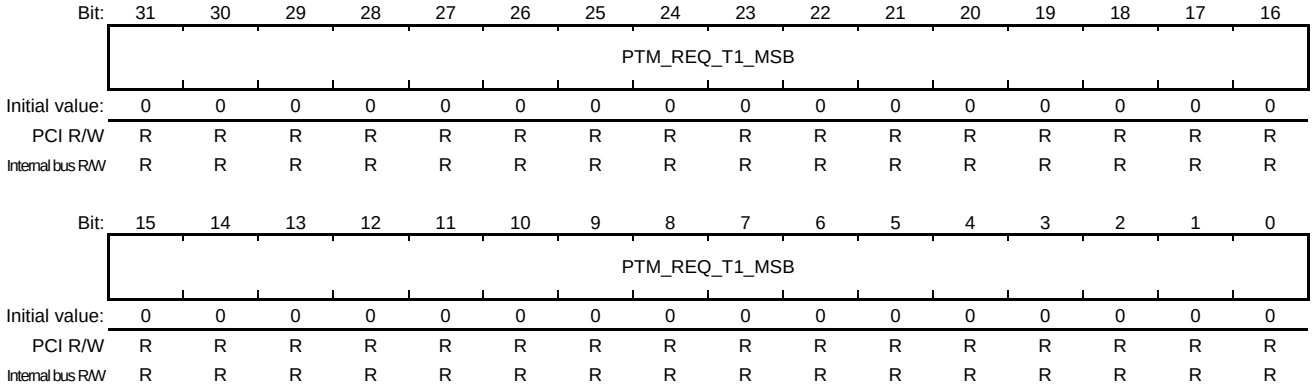
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_T1_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_T1_LSB | H'0 | R | R | PTM Requester T1 Timestamp LSB<br>Lower 32 bits of the T1 Timestamp value. |

### 104.2.167 PTM Requestor Capability Structure (VSEC) Register 7 function *i* (PTMRQCAP7F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester T1 Timestamp MSB

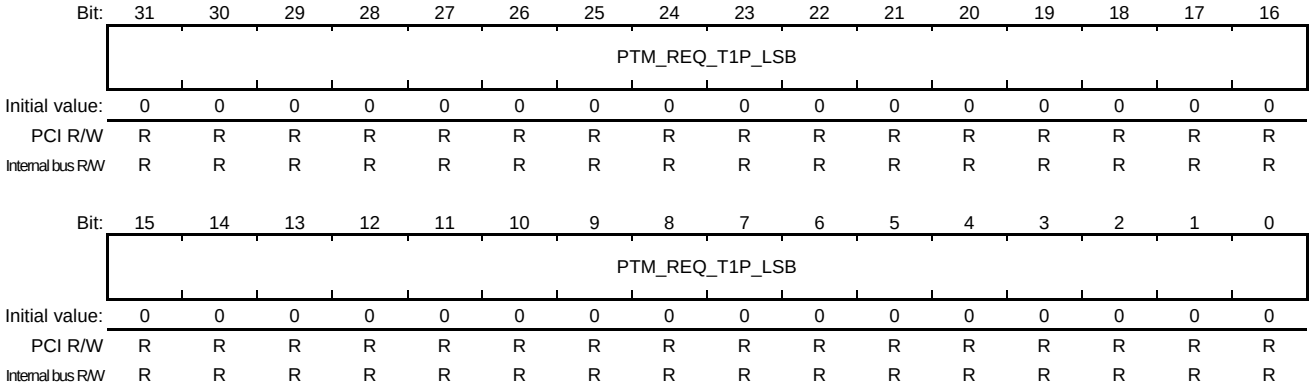



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_T1_MSB | H'0 | R | R | PTM Requester T1 Timestamp MSB<br>Upper 32 bits of the T1 Timestamp value. |

### 104.2.168 PTM Requestor Capability Structure (VSEC) Register 8 function *i* (PTMRQCAP8F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester T1 Previous Timestamp LSB

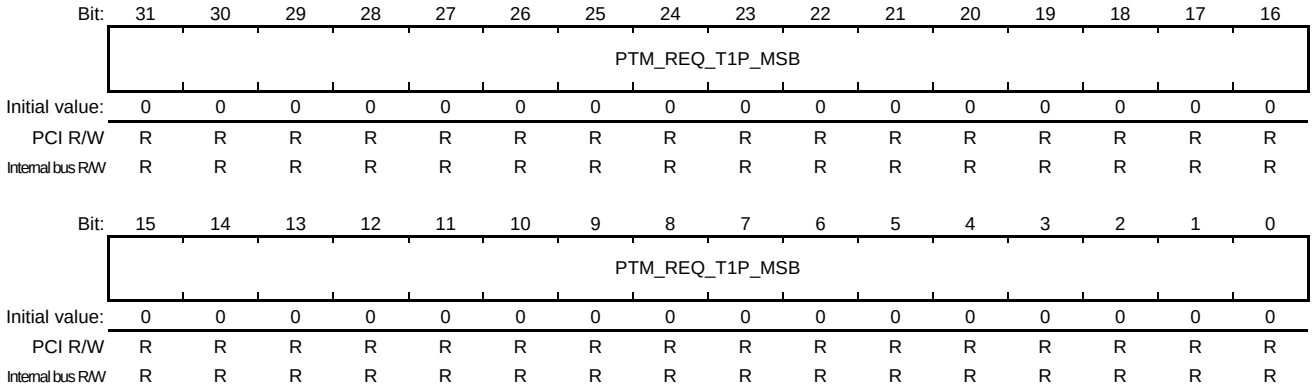



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_T1P_LSB | All 0 | R | R | PTM Requester T1 Previous Timestamp LSB<br>Lower 32 bits of the previously stored T1 Timestamp value. |

### 104.2.169 PTM Requestor Capability Structure (VSEC) Register 9 function *i* (PTMRQCAP9F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester T1 Previous Timestamp MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_ T1P_MSB | All 0 | R | R | PTM Requester T1 Previous Timestamp MSB<br>Upper 32 bits of the previously stored T1 Timestamp value. |

### 104.2.170 PTM Requestor Capability Structure (VSEC) Register 10 function *i* (PTMRQCAP10F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester T4 Timestamp LSB. For more details

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_T4_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_T4_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_T4_LSB | All 0 | R | R | PTM Requester T4 Timestamp LSB<br>Lower 32 bits of the T4 Timestamp value. |

### 104.2.171 PTM Requestor Capability Structure (VSEC) Register 11 function *i* (PTMRQCAP11F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester T4 Timestamp MSB

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_T4_MSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

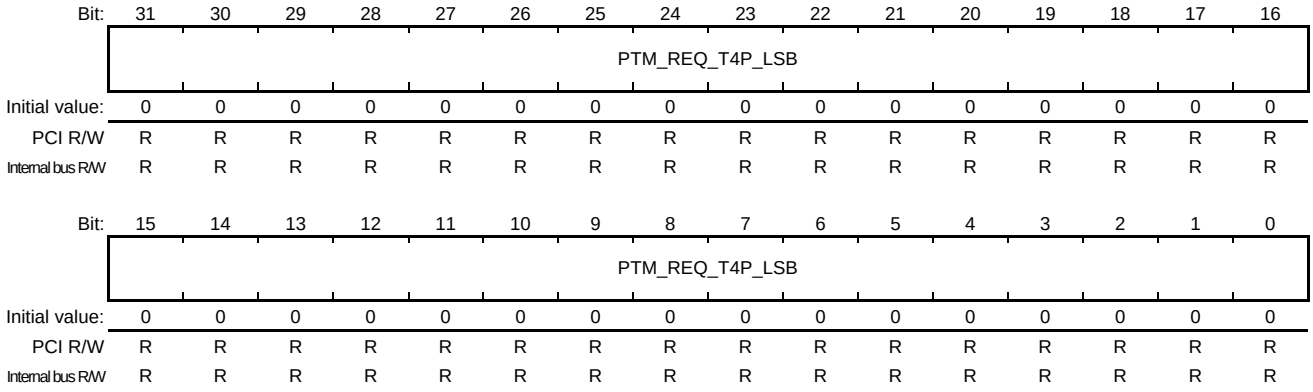
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_T4_MSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_T4_MSB | All 0 | R | R | PTM Requester T4 Timestamp MSB<br>Upper 32 bits of the T4 Timestamp value |

### 104.2.172 PTM Requestor Capability Structure (VSEC) Register 12 function *i* (PTMRQCAP12F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester T4 Previous Timestamp LSB

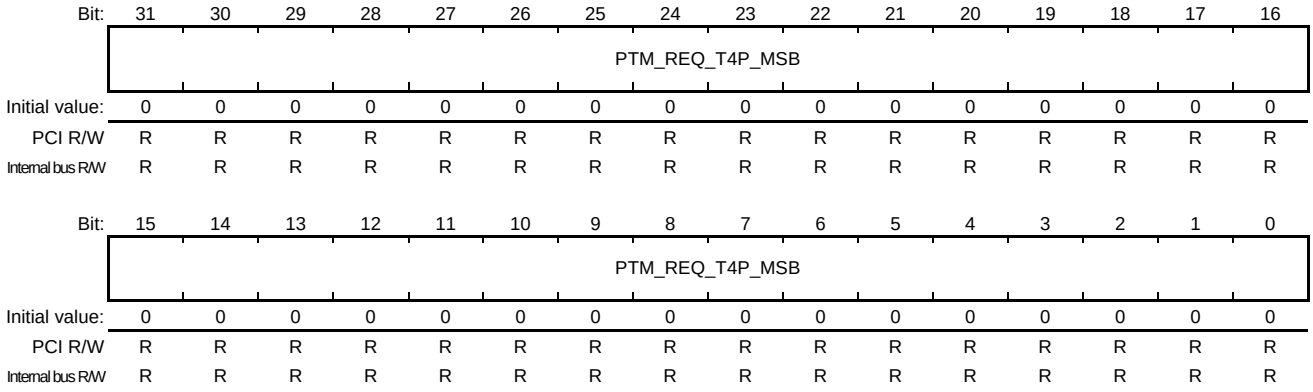



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_ T4P_LSB | All 0 | R | R | PTM Requester T4 Previous Timestamp LSB<br>Lower 32 bits of the previously stored T4 Timestamp value. |

### 104.2.173 PTM Requestor Capability Structure (VSEC) Register 13 function *i* (PTMRQCAP13F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester T4 Previous Timestamp MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_T4P_MSB | All 0 | R | R | PTM Requester T4 Previous Timestamp MSB<br>Upper 32 bits of the previously stored T4 Timestamp value. |

### 104.2.174 PTM Requestor Capability Structure (VSEC) Register 14 function *i* (PTMRQCAP14F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Master Time LSB

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_MASTER_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

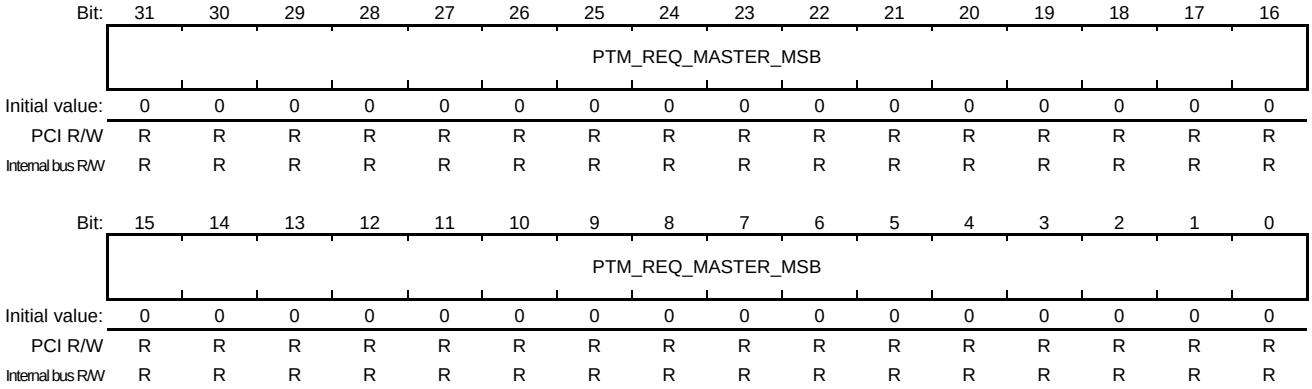
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_MASTER_LSB | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_MASTER_LSB | All 0 | R | R | PTM Requester Master Time LSB<br>Lower 32 bits of Master Time value sent from PTM Responder. |

### 104.2.175 PTM Requestor Capability Structure (VSEC) Register 15 function *i* (PTMRQCAP15F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Master Time MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_MASTER_MSB | All 0 | R | R | PTM Requester Master Time MSB<br>Upper 32 bits of Master Time value sent from PTM Responder. |

### 104.2.176 PTM Requestor Capability Structure (VSEC) Register 16 function *i* (PTMRQCAP16F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Propagation Delay

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_PROP_DELAY | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

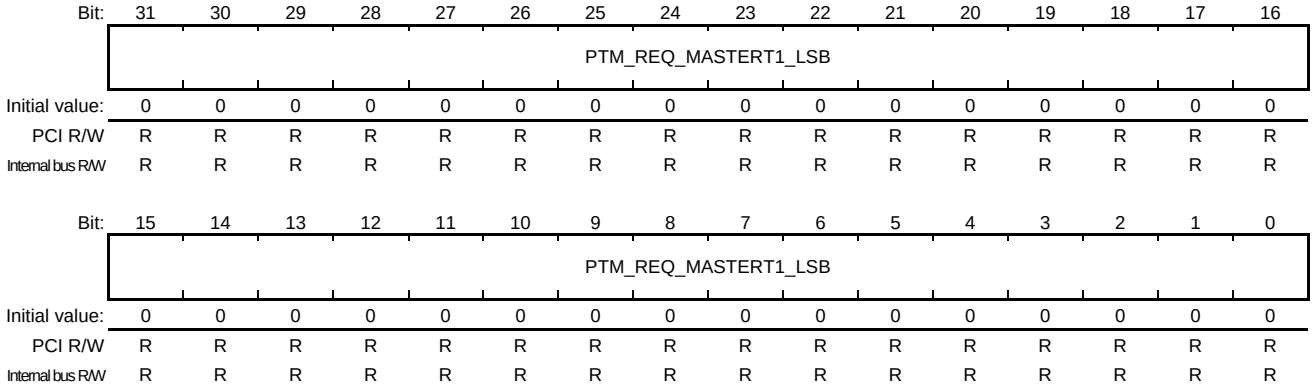
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTM_REQ_PROP_DELAY | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_PROP_DELAY | All 0 | R | R | PTM Requester Propagation Delay<br>Propagation Delay sent from the PTM Responder |

### 104.2.177 PTM Requestor Capability Structure (VSEC) Register 17 function *i* (PTMRQCAP17F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Master Time at T1 LSB

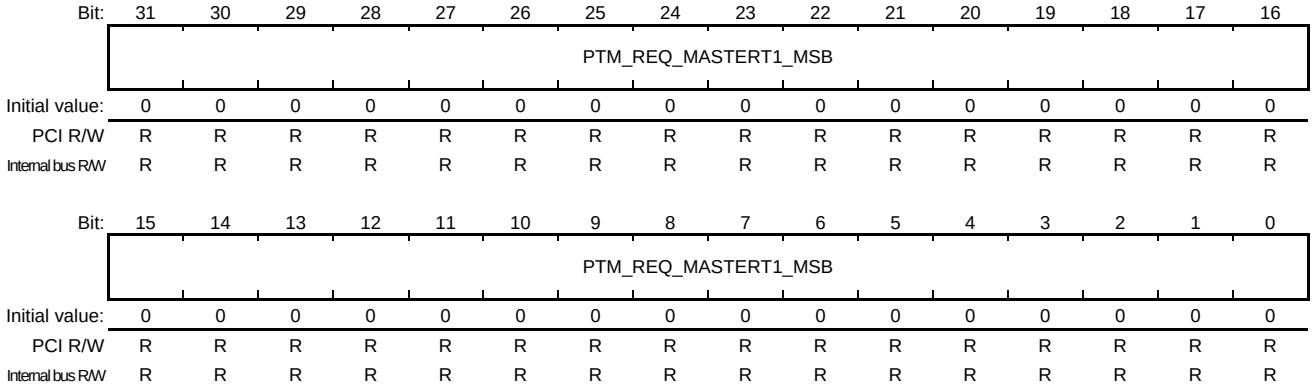



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_MASTERT1_LSB | All 0 | R | R | PTM Requester Master Time at T1 LSB<br>Lower 32 bits of the calculated Master Time at T1 prime. |

### 104.2.178 PTM Requestor Capability Structure (VSEC) Register 18 function *i* (PTMRQCAP18F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester Master Time at T1 MSB




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PTM_REQ_MASTERT1_MSB | All 0 | R | R | PTM Requester Master Time at T1 MSB<br>Upper 32 bits of the calculated Master Time at T1 prime. |

### 104.2.179 PTM Requestor Capability Structure (VSEC) Register 19 function *i* (PTMRQCAP19F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester TX Latency

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PTM_REQ_TX_LATENCY | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | PTM_REQ_TX_LATENCY | H'0 | R/W | R/W | PTM Requester TX Latency<br>Requester Transmit path latency (12 bit wide). |

### 104.2.180 PTM Requestor Capability Structure (VSEC) Register 20 function *i* (PTMRQCAP20F*i*) *i* = 0, 1

**Header TYPE00 only**

- PTM Requester RX Latency

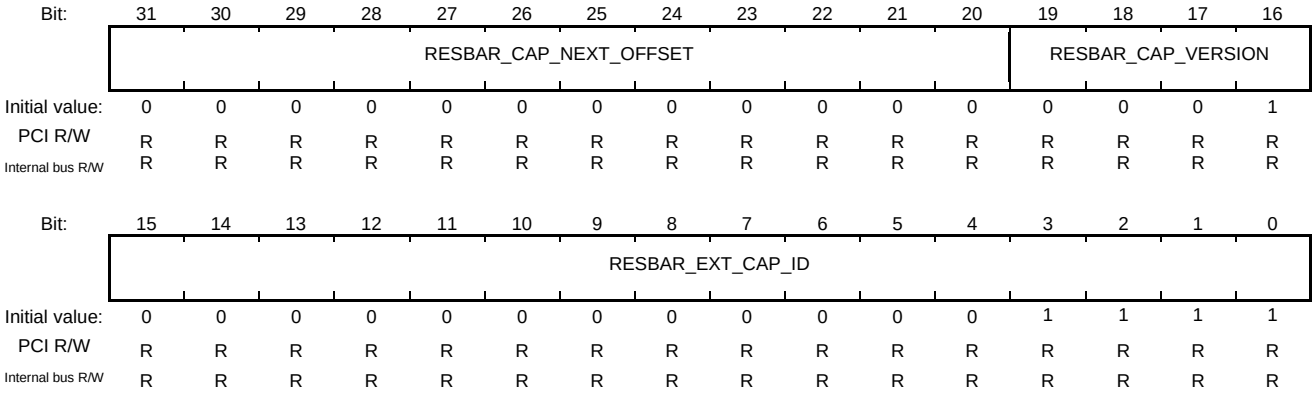
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | PTM_REQ_RX_LATENCY | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 0 | PTM_REQ_ RX_LATENCY | 0 | R/W | R/W | PTM Requester RX Latency<br>Requester Receive path latency (12 bit wide). |

### 104.2.181 Resizable BAR Capability Header function *i* (RESBAR_CAP_HDR_REG_F*i*) *i* = 0, 1

**Header TYPE00 only**




| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | RESBAR_CAP_NEXT_OFFSET | H'0 | R | R | Next Capability Offset.<br>This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.<br>Note: This register field is sticky.<br>0x0 MIN_VAL: Min value<br>0xfff MAX_VAL: Max value |
| 19 to 16 | RESBAR_CAP_VERSION | H'1 | R | R | Capability Version.<br>This field is a PCI-SIG defined version number that indicates the version of the Capability structure present.<br>This bit depends on the version of the specification.<br>Note: This register field is sticky.<br>0x0 MIN_VAL: Min value<br>0xf MAX_VAL: Max value |
| 15 to 0 | RESBAR_EXT_CAP_ID | H'15 | R | R | Resizable BAR Extended Capability ID.<br>This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability.<br>The Extended Capability ID for the Resizable BAR Capability is 0015h.<br>Note: This register field is sticky.<br>0x0 MIN_VAL: Min value<br>0xffff MAX_VAL: Max value |

### 104.2.182 Resizable BAR0 Capability Register function *i* (RESBAR_CAP_REG_0_REG_F*i*) *i* = 0, 1

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CAP_REG_0_128TB | RESBAR_CA P_REG_0_64TB | RESBAR_CA P_REG_0_32TB | RESBAR_CA P_REG_0_16TB | RESBAR_CA P_REG_0_8TB | RESBAR_CA P_REG_0_4TB | RESBAR_CA P_REG_0_2TB | RESBAR_CA P_REG_0_1TB | RESBAR_CA P_REG_0_51 2GB | RESBAR_CA P_REG_0_25 6GB | RESBAR_CA P_REG_0_12 8GB | RESBAR_CA P_REG_0_64 GB | RESBAR_CA P_REG_0_32 GB | RESBAR_CA P_REG_0_16 GB | RESBAR_CA P_REG_0_8G B | RESBAR_CA P_REG_0_4G B |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CAP_REG_0_2GB | RESBAR_CA P_REG_0_1GB | RESBAR_CA P_REG_0_51 2MB | RESBAR_CA P_REG_0_25 6MB | RESBAR_CA P_REG_0_12 8MB | RESBAR_CA P_REG_0_64 MB | RESBAR_CA P_REG_0_32 MB | RESBAR_CA P_REG_0_16 MB | RESBAR_CA P_REG_0_8 MB | RESBAR_CA P_REG_0_4 MB | RESBAR_CA P_REG_0_2 MB | RESBAR_CA P_REG_0_1 MB | — | | | |
| Initial value: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 31 | RESBAR_CAP_REG_0_128TB | H'0 | R | R | Up to 128TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 128 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 TB. |
| 30 | RESBAR_CAP_REG_0_64TB | H'0 | R | R | Up to 64TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 64 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 TB. |
| 29 | RESBAR_CAP_REG_0_32TB | H'0 | R | R | Up to 32TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 32 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 TB. |
| 28 | RESBAR_CAP_REG_0_16TB | H'0 | R | R | Up to 16TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 16 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 TB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 27 | RESBAR_CAP_REG_0_8TB | H'0 | R | R | Up to 8TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 8 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 TB. |
| 26 | RESBAR_CAP_REG_0_4TB | H'0 | R | R | Up to 4TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 4 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 TB. |
| 25 | RESBAR_CAP_REG_0_2TB | H'0 | R | R | Up to 2TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 2 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 TB. |
| 24 | RESBAR_CAP_REG_0_1TB | H'0 | R | R | Up to 1TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 1 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1TB. |
| 23 | RESBAR_CAP_REG_0_512GB | H'0 | R | R | Up to 512GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 512 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 GB. |
| 22 | RESBAR_CAP_REG_0_256GB | H'0 | R | R | Up to 256GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 256 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 GB. |
| 21 | RESBAR_CAP_REG_0_128GB | H'0 | R | R | Up to 128GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 128 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 GB. |
| 20 | RESBAR_CAP_REG_0_64GB | H'0 | R | R | Up to 64GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 64 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 GB. |
| 19 | RESBAR_CAP_REG_0_32GB | H'0 | R | R | Up to 32GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 32 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 GB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 18 | RESBAR_CAP_REG_0_16GB | H'0 | R | R | Up to 16GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 16 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 GB. |
| 17 | RESBAR_CAP_REG_0_8GB | H'0 | R | R | Up to 8GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 8 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 GB. |
| 16 | RESBAR_CAP_REG_0_4GB | H'0 | R | R | Up to 4GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 4 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 GB. |
| 15 | RESBAR_CAP_REG_0_2GB | H'0 | R | R | Up to 2GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 2 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 GB. |
| 14 | RESBAR_CAP_REG_0_1GB | H'0 | R | R | Up to 1GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 1 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 GB. |
| 13 | RESBAR_CAP_REG_0_512MB | H'0 | R | R | Up to 512MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 512 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 MB. |
| 12 | RESBAR_CAP_REG_0_256MB | H'1 | R | R | Up to 256MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 256 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 MB. |
| 11 | RESBAR_CAP_REG_0_128MB | H'1 | R | R | Up to 128MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 128 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 MB. |
| 10 | RESBAR_CAP_REG_0_64MB | H'1 | R | R | Up to 64MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 64 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 MB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 9 | RESBAR_CAP_REG_0_32MB | H'1 | R | R | Up to 32MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 32 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 MB. |
| 8 | RESBAR_CAP_REG_0_16MB | H'1 | R | R | Up to 16MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 16 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 MB. |
| 7 | RESBAR_CAP_REG_0_8MB | H'1 | R | R | Up to 8MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 8 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 MB. |
| 6 | RESBAR_CAP_REG_0_4MB | H'1 | R | R | Up to 4MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 4 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 MB. |
| 5 | RESBAR_CAP_REG_0_2MB | H'1 | R | R | Up to 2MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 2 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 MB. |
| 4 | RESBAR_CAP_REG_0_1MB | H'1 | R | R | Up to 1MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 1 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 MB. |
| 3 to 0 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.183 Resizable BAR0 Control Register function *i* (RESBAR_CTRL_REG_0_REG_F*i*) *i* = 0, 1

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CTRL_REG_0_8EB | RES BAR _CT RL_ REG _0_4 EB | RES BAR _CT RL_ REG _0_2 EB | RES BAR _CT RL_ REG _0_1 EB | RES BAR _CT RL_ REG _0_5 12PB | RES BAR _CT RL_ REG _0_2 56PB | RES BAR _CT RL_ REG _0_1 28PB | RES BAR _CT RL_ REG _0_6 4PB | RES BAR _CT RL_ REG _0_3 2PB | RES BAR _CT RL_ REG _0_1 6PB | RES BAR _CT RL_ REG _0_8 PB | RES BAR _CT RL_ REG _0_4 PB | RES BAR _CT RL_ REG _0_2 PB | RES BAR _CT RL_ REG _0_1 PB | RES BAR _CT RL_ REG _0_5 12TB | RES BAR _CT RL_ REG _0_2 56TB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | RESBAR_CTRL_REG_BAR_SIZE | | | | | | RESBAR_CTRL_REG _NUM_BARS | | | — | | RESBAR_CTRL_REG _IDX_0 | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 31 | RESBAR_CTRL_REG_0_8EB | H'0 | R | R | Up to 8EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 8 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 EB. |
| 30 | RESBAR_CTRL_REG_0_4EB | H'0 | R | R | Up to 4EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 4 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 EB. |
| 29 | RESBAR_CTRL_REG_0_2EB | H'0 | R | R | Up to 2EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 2 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 EB. |
| 28 | RESBAR_CTRL_REG_0_1EB | H'0 | R | R | Up to 1EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 1 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 EB. |
| 27 | RESBAR_CTRL_REG_0_512PB | H'0 | R | R | Up to 512PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 512 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 PB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 26 | RESBAR_CTRL_REG_0_256PB | H'0 | R | R | Up to 256PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 256 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 PB. |
| 25 | RESBAR_CTRL_REG_0_128PB | H'0 | R | R | Up to 128PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 128 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 PB. |
| 24 | RESBAR_CTRL_REG_0_64PB | H'0 | R | R | Up to 64PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 64 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 PB. |
| 23 | RESBAR_CTRL_REG_0_32PB | H'0 | R | R | Up to 32PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 32 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 PB. |
| 22 | RESBAR_CTRL_REG_0_16PB | H'0 | R | R | Up to 16PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 16 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 PB. |
| 21 | RESBAR_CTRL_REG_0_8PB | H'0 | R | R | Up to 8PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 8 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 PB. |
| 20 | RESBAR_CTRL_REG_0_4PB | H'0 | R | R | Up to 4PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 4 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 PB. |
| 19 | RESBAR_CTRL_REG_0_2PB | H'0 | R | R | Up to 2PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 2 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 PB. |
| 18 | RESBAR_CTRL_REG_0_1PB | H'0 | R | R | Up to 1PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 1 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 PB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 17 | RESBAR_CTRL_REG_0_512TB | H'0 | R | R | Up to 512TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 512 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 TB. |
| 16 | RESBAR_CTRL_REG_0_256TB | H'0 | R | R | Up to 256TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR0 sized to 256 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 TB. |
| 15 to 14 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 8 | RESBAR_CTRL_REG_BAR_SIZE | H'8 | R/W | R/W | BAR0 Size.<br>The default value of this field is equal to the default size of the address space that the BAR0 resource is requesting via the BAR's read-only bits.<br>For backward compatibility with software, the default value must be in the range from 0 to 19.<br>Note:<br>0x0 MIN_VAL: 1MB(2^20 bytes)<br>0x2b MAX_VAL: 8EB(2^63 bytes) |
| 7 to 5 | RESBAR_CTRL_REG_NUM_BARS | H'3 | R | R | Number of Resizeable BARs.<br>This field indicates the total number of resizable BARs in the capability structure for the Function.<br>The value of this field must be in the range of 01h to 06h. The field is valid in Resizable BAR0 Control register (0) (at offset 008h), and is RsvdP for all others.<br>Note: This register field is sticky.<br>0x1 BAR1: one resizable BAR for the function<br>0x2 BAR2: two resizable BARs for the function.<br>0x3 BAR3: three resizable BARs for the function.<br>0x4 BAR4: four resizable BARs for the function.<br>0x5 BAR5: five resizable BARs for the function.<br>0x6 BAR6: six resizable BARs for the function. |
| 4 to 3 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | RESBAR_CTRL_REG_IDX_0 | H'0 | R | R | BAR Index.<br>This encoded value points to the beginning of the BAR0.<br>For a 64-bit Base Address register, the BAR Index indicates the lower DWORD. This value indicates which BAR supports a negotiable size.<br>Note: This register field is sticky.<br>0x0 BAR_10h: BAR located at offset 10h.<br>0x1 BAR_14h: BAR located at offset 14h.<br>0x2 BAR_18h: BAR located at offset 18h.<br>0x3 BAR_1Ch: BAR located at offset 1Ch.<br>0x4 BAR_20h: BAR located at offset 20h.<br>0x5 BAR_24h: BAR located at offset 24h. |

### 104.2.184 Resizable BAR2 Capability Register function *i* (RESBAR_CAP_REG_2_REG_F*i*) *i* = 0, 1

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CAP_REG_2_128TB | RESBAR_CA P_REG_2_64TB | RESBAR_CA P_REG_2_32TB | RESBAR_CA P_REG_2_16TB | RESBAR_CA P_REG_2_8TB | RESBAR_CA P_REG_2_4TB | RESBAR_CA P_REG_2_2TB | RESBAR_CA P_REG_2_1TB | RESBAR_CA P_REG_2_51 2GB | RESBAR_CA P_REG_2_25 6GB | RESBAR_CA P_REG_2_12 8GB | RESBAR_CA P_REG_2_64 GB | RESBAR_CA P_REG_2_32 GB | RESBAR_CA P_REG_2_16 GB | RESBAR_CA P_REG_2_8GB | RESBAR_CA P_REG_2_4GB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CAP_REG_2_2GB | RESBAR_CA P_REG_2_1GB | RESBAR_CA P_REG_2_51 2MB | RESBAR_CA P_REG_2_25 6MB | RESBAR_CA P_REG_2_12 8MB | RESBAR_CA P_REG_2_64 MB | RESBAR_CA P_REG_2_32 MB | RESBAR_CA P_REG_2_16 MB | RESBAR_CA P_REG_2_8 MB | RESBAR_CA P_REG_2_4 MB | RESBAR_CA P_REG_2_2 MB | RESBAR_CA P_REG_2_1 MB | — | | | |
| Initial value: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 31 | RESBAR_CAP_REG_2_128TB | H'0 | R | R | Up to 128TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 128 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 TB. |
| 30 | RESBAR_CAP_REG_2_64TB | H'0 | R | R | Up to 64TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 64 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 TB. |
| 29 | RESBAR_CAP_REG_2_32TB | H'0 | R | R | Up to 32TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 32 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 TB. |
| 28 | RESBAR_CAP_REG_2_16TB | H'0 | R | R | Up to 16TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 16 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 TB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 27 | RESBAR_CAP_REG_2_8TB | H'0 | R | R | Up to 8TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 8 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 TB. |
| 26 | RESBAR_CAP_REG_2_4TB | H'0 | R | R | Up to 4TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 4 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 TB. |
| 25 | RESBAR_CAP_REG_2_2TB | H'0 | R | R | Up to 2TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 2 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 TB. |
| 24 | RESBAR_CAP_REG_2_1TB | H'0 | R | R | Up to 1TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 1 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1TB. |
| 23 | RESBAR_CAP_REG_2_512GB | H'0 | R | R | Up to 512GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 512 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 GB. |
| 22 | RESBAR_CAP_REG_2_256GB | H'0 | R | R | Up to 256GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 256 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 GB. |
| 21 | RESBAR_CAP_REG_2_128GB | H'0 | R | R | Up to 128GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 128 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 GB. |
| 20 | RESBAR_CAP_REG_2_64GB | H'0 | R | R | Up to 64GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 64 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 GB. |
| 19 | RESBAR_CAP_REG_2_32GB | H'0 | R | R | Up to 32GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 32 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 GB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 18 | RESBAR_CAP_REG_2_16GB | H'0 | R | R | Up to 16GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 16 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 GB. |
| 17 | RESBAR_CAP_REG_2_8GB | H'0 | R | R | Up to 8GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 8 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 GB. |
| 16 | RESBAR_CAP_REG_2_4GB | H'0 | R | R | Up to 4GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 4 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 GB. |
| 15 | RESBAR_CAP_REG_2_2GB | H'0 | R | R | Up to 2GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 2 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 GB. |
| 14 | RESBAR_CAP_REG_2_1GB | H'0 | R | R | Up to 1GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 1 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 GB. |
| 13 | RESBAR_CAP_REG_2_512MB | H'0 | R | R | Up to 512MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 512 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 MB. |
| 12 | RESBAR_CAP_REG_2_256MB | H'1 | R | R | Up to 256MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 256 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 MB. |
| 11 | RESBAR_CAP_REG_2_128MB | H'1 | R | R | Up to 128MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 128 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 MB. |
| 10 | RESBAR_CAP_REG_2_64MB | H'1 | R | R | Up to 64MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 64 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 MB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 9 | RESBAR_CAP_REG_2_32MB | H'1 | R | R | Up to 32MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 32 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 MB. |
| 8 | RESBAR_CAP_REG_2_16MB | H'1 | R | R | Up to 16MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 16 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 MB. |
| 7 | RESBAR_CAP_REG_2_8MB | H'1 | R | R | Up to 8MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 8 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 MB. |
| 6 | RESBAR_CAP_REG_2_4MB | H'1 | R | R | Up to 4MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 4 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 MB. |
| 5 | RESBAR_CAP_REG_2_2MB | H'1 | R | R | Up to 2MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 2 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 MB. |
| 4 | RESBAR_CAP_REG_2_1MB | H'1 | R | R | Up to 1MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 1 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 MB. |
| 3 to 0 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.185 Resizable BAR2 Control Register function *i* (RESBAR_CTRL_REG_2_REG_F*i*) *i* = 0, 1

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CTRL_REG_2_8EB | RESBAR_CT RL_REG _2_4EB | RESBAR_CT RL_REG _2_2EB | RESBAR_CT RL_REG _2_1EB | RESBAR_CT RL_REG _2_5 12PB | RESBAR_CT RL_REG _2_2 56PB | RESBAR_CT RL_REG _2_1 28PB | RESBAR_CT RL_REG _2_6 4PB | RESBAR_CT RL_REG _2_3 2PB | RESBAR_CT RL_REG _2_1 6PB | RESBAR_CT RL_REG _2_8 PB | RESBAR_CT RL_REG _2_4 PB | RESBAR_CT RL_REG _2_2 PB | RESBAR_CT RL_REG _2_1 PB | RESBAR_CT RL_REG _2_5 12TB | RESBAR_CT RL_REG _2_2 56TB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | RESBAR_CTRL_REG_BAR_SIZE | | | | | | RESBAR_CTRL_REG _NUM_BARS | | | — | | RESBAR_CTRL_REG _IDX_2 | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| PCI R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 31 | RESBAR_CTRL_REG_2_8EB | H'0 | R | R | Up to 8EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 8 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 EB. |
| 30 | RESBAR_CTRL_REG_2_4EB | H'0 | R | R | Up to 4EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 4 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 EB. |
| 29 | RESBAR_CTRL_REG_2_2EB | H'0 | R | R | Up to 2EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 2 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 EB. |
| 28 | RESBAR_CTRL_REG_2_1EB | H'0 | R | R | Up to 1EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 1 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 EB. |
| 27 | RESBAR_CTRL_REG_2_512PB | H'0 | R | R | Up to 512PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 512 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 PB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 26 | RESBAR_CTRL_REG_2_256PB | H'0 | R | R | Up to 256PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 256 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 PB. |
| 25 | RESBAR_CTRL_REG_2_128PB | H'0 | R | R | Up to 128PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 128 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 PB. |
| 24 | RESBAR_CTRL_REG_2_64PB | H'0 | R | R | Up to 64PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 64 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 PB. |
| 23 | RESBAR_CTRL_REG_2_32PB | H'0 | R | R | Up to 32PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 32 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 PB. |
| 22 | RESBAR_CTRL_REG_2_16PB | H'0 | R | R | Up to 16PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 16 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 PB. |
| 21 | RESBAR_CTRL_REG_2_8PB | H'0 | R | R | Up to 8PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 8 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 PB. |
| 20 | RESBAR_CTRL_REG_2_4PB | H'0 | R | R | Up to 4PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 4 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 PB. |
| 19 | RESBAR_CTRL_REG_2_2PB | H'0 | R | R | Up to 2PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 2 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 PB. |
| 18 | RESBAR_CTRL_REG_2_1PB | H'0 | R | R | Up to 1PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 1 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 PB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 17 | RESBAR_CTRL_REG_2_512TB | H'0 | R | R | Up to 512TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 512 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 TB. |
| 16 | RESBAR_CTRL_REG_2_256TB | H'0 | R | R | Up to 256TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR2 sized to 256 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 TB. |
| 15 to 14 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 8 | RESBAR_CTRL_REG_BAR_SIZE | H'8 | R/W | R/W | BAR2 Size.<br>The default value of this field is equal to the default size of the address space that the BAR2 resource is requesting via the BAR's read-only bits.<br>For backward compatibility with software, the default value must be in the range from 0 to 19.<br>Note:<br>0x0 MIN_VAL: 1MB(2^20 bytes)<br>0x2b MAX_VAL: 8EB(2^63 bytes) |
| 7 to 5 | RESBAR_CTRL_REG_NUM_BARS | H'0 | R | R | Number of Resizeable BARs.<br>This field indicates the total number of resizable BARs in the capability structure for the Function.<br>The value of this field must be in the range of 01h to 06h. The field is valid in Resizable BAR2 Control register (0) (at offset 008h), and is RsvdP for all others.<br>Note: This register field is sticky.<br>0x1 BAR1: one resizable BAR for the function<br>0x2 BAR2: two resizable BARs for the function.<br>0x3 BAR3: three resizable BARs for the function.<br>0x4 BAR4: four resizable BARs for the function.<br>0x5 BAR5: five resizable BARs for the function.<br>0x6 BAR6: six resizable BARs for the function. |
| 4 to 3 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | RESBAR_CTRL_REG_IDX_2 | H'2 | R | R | BAR Index.<br>This encoded value points to the beginning of the BAR2.<br>For a 64-bit Base Address register, the BAR Index indicates the lower DWORD. This value indicates which BAR supports a negotiable size.<br>Note: This register field is sticky.<br>0x0 BAR_10h: BAR located at offset 10h.<br>0x1 BAR_14h: BAR located at offset 14h.<br>0x2 BAR_18h: BAR located at offset 18h.<br>0x3 BAR_1Ch: BAR located at offset 1Ch.<br>0x4 BAR_20h: BAR located at offset 20h.<br>0x5 BAR_24h: BAR located at offset 24h. |

### 104.2.186 Resizable BAR5 Capability Register function *i* (RESBAR_CAP_REG_5_REG_F*i*) *i* = 0, 1

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CAP_REG_5_128TB | RESBAR_CAP_REG_5_64TB | RESBAR_CAP_REG_5_32TB | RESBAR_CAP_REG_5_16TB | RESBAR_CAP_REG_5_8TB | RESBAR_CAP_REG_5_4TB | RESBAR_CAP_REG_5_2TB | RESBAR_CAP_REG_5_1TB | RESBAR_CAP_REG_5_512GB | RESBAR_CAP_REG_5_256GB | RESBAR_CAP_REG_5_128GB | RESBAR_CAP_REG_5_64GB | RESBAR_CAP_REG_5_32GB | RESBAR_CAP_REG_5_16GB | RESBAR_CAP_REG_5_8GB | RESBAR_CAP_REG_5_4GB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CAP_REG_5_2GB | RESBAR_CAP_REG_5_1GB | RESBAR_CAP_REG_5_512MB | RESBAR_CAP_REG_5_256MB | RESBAR_CAP_REG_5_128MB | RESBAR_CAP_REG_5_64MB | RESBAR_CAP_REG_5_32MB | RESBAR_CAP_REG_5_16MB | RESBAR_CAP_REG_5_8MB | RESBAR_CAP_REG_5_4MB | RESBAR_CAP_REG_5_2MB | RESBAR_CAP_REG_5_1MB | — | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 31 | RESBAR_CAP_REG_5_128TB | H'0 | R | R | Up to 128TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 128 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 TB. |
| 30 | RESBAR_CAP_REG_5_64TB | H'0 | R | R | Up to 64TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 64 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 TB. |
| 29 | RESBAR_CAP_REG_5_32TB | H'0 | R | R | Up to 32TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 32 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 TB. |
| 28 | RESBAR_CAP_REG_5_16TB | H'0 | R | R | Up to 16TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 16 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 TB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 27 | RESBAR_CAP_REG_5_8TB | H'0 | R | R | Up to 8TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 8 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 TB. |
| 26 | RESBAR_CAP_REG_5_4TB | H'0 | R | R | Up to 4TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 4 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 TB. |
| 25 | RESBAR_CAP_REG_5_2TB | H'0 | R | R | Up to 2TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 2 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 TB. |
| 24 | RESBAR_CAP_REG_5_1TB | H'0 | R | R | Up to 1TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 1 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1TB. |
| 23 | RESBAR_CAP_REG_5_512GB | H'0 | R | R | Up to 512GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 512 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 GB. |
| 22 | RESBAR_CAP_REG_5_256GB | H'0 | R | R | Up to 256GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 256 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 GB. |
| 21 | RESBAR_CAP_REG_5_128GB | H'0 | R | R | Up to 128GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 128 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 GB. |
| 20 | RESBAR_CAP_REG_5_64GB | H'0 | R | R | Up to 64GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 64 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 GB. |
| 19 | RESBAR_CAP_REG_5_32GB | H'0 | R | R | Up to 32GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 32 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 GB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 18 | RESBAR_CAP_REG_5_16GB | H'0 | R | R | Up to 16GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 16 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 GB. |
| 17 | RESBAR_CAP_REG_5_8GB | H'0 | R | R | Up to 8GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 8 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 GB. |
| 16 | RESBAR_CAP_REG_5_4GB | H'0 | R | R | Up to 4GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 4 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 GB. |
| 15 | RESBAR_CAP_REG_5_2GB | H'0 | R | R | Up to 2GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 2 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 GB. |
| 14 | RESBAR_CAP_REG_5_1GB | H'0 | R | R | Up to 1GB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 1 GB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 GB. |
| 13 | RESBAR_CAP_REG_5_512MB | H'0 | R | R | Up to 512MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 512 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 MB. |
| 12 | RESBAR_CAP_REG_5_256MB | H'0 | R | R | Up to 256MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 256 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 MB. |
| 11 | RESBAR_CAP_REG_5_128MB | H'0 | R | R | Up to 128MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 128 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 MB. |
| 10 | RESBAR_CAP_REG_5_64MB | H'0 | R | R | Up to 64MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 64 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 MB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 9 | RESBAR_CAP_REG_5_32MB | H'0 | R | R | Up to 32MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 32 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 MB. |
| 8 | RESBAR_CAP_REG_5_16MB | H'0 | R | R | Up to 16MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 16 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 MB. |
| 7 | RESBAR_CAP_REG_5_8MB | H'0 | R | R | Up to 8MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 8 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 MB. |
| 6 | RESBAR_CAP_REG_5_4MB | H'0 | R | R | Up to 4MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 4 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 MB. |
| 5 | RESBAR_CAP_REG_5_2MB | H'0 | R | R | Up to 2MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 2 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 MB. |
| 4 | RESBAR_CAP_REG_5_1MB | H'1 | R | R | Up to 1MB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 1 MB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 MB. |
| 3 to 0 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.187 Resizable BAR5 Control Register function *i* (RESBAR_CTRL_REG_5_REG_F*i*) *i* = 0, 1

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RESBAR_CTRL_REG_5_8EB | RESBAR_CT RL_REG _5_4EB | RESBAR_CT RL_REG _5_2EB | RESBAR_CT RL_REG _5_1EB | RESBAR_CT RL_REG _5_512PB | RESBAR_CT RL_REG _5_256PB | RESBAR_CT RL_REG _5_128PB | RESBAR_CT RL_REG _5_64PB | RESBAR_CT RL_REG _5_32PB | RESBAR_CT RL_REG _5_16PB | RESBAR_CT RL_REG _5_8PB | RESBAR_CT RL_REG _5_4PB | RESBAR_CT RL_REG _5_2PB | RESBAR_CT RL_REG _5_1PB | RESBAR_CT RL_REG _5_512TB | RESBAR_CT RL_REG _5_256TB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | RESBAR_CTRL_REG_BAR_SIZE | | | | | | RESBAR_CTRL_REG_NUM_BARS | | | — | | RESBAR_CTRL_REG_IDX_5 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| PCI R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 31 | RESBAR_CTRL_REG_5_8EB | H'0 | R | R | Up to 8EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 8 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 EB. |
| 30 | RESBAR_CTRL_REG_5_4EB | H'0 | R | R | Up to 4EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 4 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 EB. |
| 29 | RESBAR_CTRL_REG_5_2EB | H'0 | R | R | Up to 2EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 2 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 EB. |
| 28 | RESBAR_CTRL_REG_5_1EB | H'0 | R | R | Up to 1EB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 1 EB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 EB. |
| 27 | RESBAR_CTRL_REG_5_512PB | H'0 | R | R | Up to 512PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 512 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 PB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 26 | RESBAR_CTRL_REG_5_256PB | H'0 | R | R | Up to 256PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 256 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 PB. |
| 25 | RESBAR_CTRL_REG_5_128PB | H'0 | R | R | Up to 128PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 128 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 128 PB. |
| 24 | RESBAR_CTRL_REG_5_64PB | H'0 | R | R | Up to 64PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 64 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 64 PB. |
| 23 | RESBAR_CTRL_REG_5_32PB | H'0 | R | R | Up to 32PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 32 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 32 PB. |
| 22 | RESBAR_CTRL_REG_5_16PB | H'0 | R | R | Up to 16PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 16 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 16 PB. |
| 21 | RESBAR_CTRL_REG_5_8PB | H'0 | R | R | Up to 8PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 8 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 8 PB. |
| 20 | RESBAR_CTRL_REG_5_4PB | H'0 | R | R | Up to 4PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 4 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 4 PB. |
| 19 | RESBAR_CTRL_REG_5_2PB | H'0 | R | R | Up to 2PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 2 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 2 PB. |
| 18 | RESBAR_CTRL_REG_5_1PB | H'0 | R | R | Up to 1PB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 1 PB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 1 PB. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal bus R/W | Description |
|---|---|---|---|---|---|
| 17 | RESBAR_CTRL_REG_5_512TB | H'0 | R | R | Up to 512TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 512 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 512 TB. |
| 16 | RESBAR_CTRL_REG_5_256TB | H'0 | R | R | Up to 256TB BAR Supported.<br>When Set, indicates that the Function supports operating with the BAR5 sized to 256 TB.<br>Note: This register field is sticky.<br>0x1 SET: If Set, function supports operating with BAR sized to 256 TB. |
| 15 to 14 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 8 | RESBAR_CTRL_REG_BAR_SIZE | H'0 | R/W | R/W | BAR5 Size.<br>The default value of this field is equal to the default size of the address space that the BAR5 resource is requesting via the BAR's read-only bits.<br>For backward compatibility with software, the default value must be in the range from 0 to 19.<br>Note:<br>0x0 MIN_VAL: 1MB(2^20 bytes)<br>0x2b MAX_VAL: 8EB(2^63 bytes) |
| 7 to 5 | RESBAR_CTRL_REG_NUM_BARS | H'0 | R | R | Number of Resizeable BARs.<br>This field indicates the total number of resizable BARs in the capability structure for the Function.<br>The value of this field must be in the range of 01h to 06h. The field is valid in Resizable BAR5 Control register (0) (at offset 008h), and is RsvdP for all others.<br>Note: This register field is sticky.<br>0x1 BAR1: one resizable BAR for the function<br>0x2 BAR2: two resizable BARs for the function.<br>0x3 BAR3: three resizable BARs for the function.<br>0x4 BAR4: four resizable BARs for the function.<br>0x5 BAR5: five resizable BARs for the function.<br>0x6 BAR6: six resizable BARs for the function. |
| 4 to 3 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | RESBAR_CTRL_REG_IDX_5 | H'5 | R | R | BAR Index.<br>This encoded value points to the beginning of the BAR0. For a 64-bit Base Address register, the BAR Index indicates the lower DWORD. This value indicates which BAR supports a negotiable size.<br>Note: This register field is sticky.<br>0x0 BAR_10h: BAR located at offset 10h.<br>0x1 BAR_14h: BAR located at offset 14h.<br>0x2 BAR_18h: BAR located at offset 18h.<br>0x3 BAR_1Ch: BAR located at offset 1Ch.<br>0x4 BAR_20h: BAR located at offset 20h.<br>0x5 BAR_24h: BAR located at offset 24h. |

### 104.2.188 Port Logic Registers 0 (PRTLGC0)

**Common to Header TYPE00/01**

- Ack Latency Timer and Replay Timer Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REPLAY_TIME_LIMIT | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ROUND_TRIP_LATENCY_TIME_LIMIT | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | REPLAY_TIME_LIMIT | H'308D | R/W | R/W | Replay Timer Limit.<br>The replay timer expires when it reaches this limit. The controller initiates a replay upon reception of a NAK or when the replay timer expires. |
| 15 to 0 | ROUND_TRIP_LATENCY_TIME_LIMIT | H'102F | R/W | R/W | Ack Latency Timer Limit.<br>The Ack latency timer expires when it reaches this limit. After reset, the controller updates the default according to the Negotiated Link Width, Max_Payload_Size, and speed. |

### 104.2.189 Port Logic Registers 1 (PRTLGC1)

**Common to Header TYPE00/01**

- Vendor Specific DLLP Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VENDOR_SPEC_DLLP | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

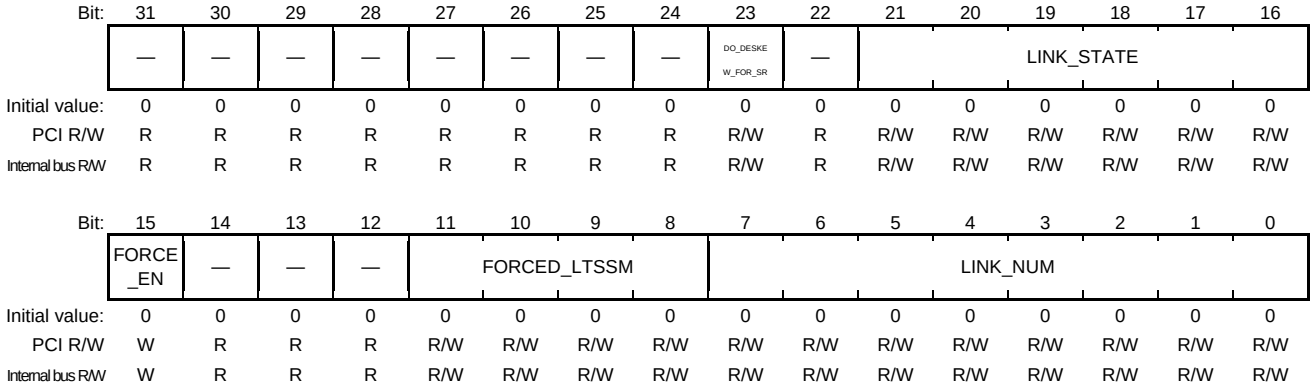
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VENDOR_SPEC_DLLP | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | VENDOR_SPEC_ DLLP | H'FFF FFFFF | R/W | R/W | Vendor Specific DLLP Register.<br>Used to send a specific PCI Express DLLP.<br>Your application writes the 8-bit DLLP Type and 24-bits of Payload data into this register, then sets the field VENDOR_SPECIFIC_DLLP_REQ of PRTLGC4 to send the DLLP.<br>■ [7:0] = Type<br>■ [31:8] = Payload (24 bits)<br>The dllp type is in bits [7:0] while the remainder is the vendor defined payload. |

### 104.2.190 Port Logic Registers 2 (PRTLGC2)

### Common to Header TYPE00/01

- Port Force Link Register




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 | DO_DESKEW_FOR_ SRIS | 0 | R/W | R/W | Use the transitions from TS2 to Logical Idle Symbol, SKP OS to Logical Idle Symbol, and FTS Sequence to SKP OS to do deskew for SRIS instead of using received SKP OS if DO_DESKEW_FOR_SRIS is set to 1. |
| 22 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 21 to 16 | LINK_STATE | H'0 | R/W | R/W | Forced LTSSM State.<br>The LTSSM state that the controller is forced to when you set the FORCE_EN bit (Force Link). |
| 15 | FORCE_EN | 0 | W | W | Force Link.<br>The controller supports a testing and debug capability to allow your software to force the LTSSM state machine into a specific state, and to force the controller to transmit a specific Link Command.<br>Asserting this bit triggers the following actions:<br>■ Forces the LTSSM to the state specified by the Forced LTSSM State field.<br>■ Forces the controller to transmit the command specified by the Forced Link Command field.<br>This is a self-clearing register field. Reading from this register field always returns a "0". |
| 14 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 11 to 8 | FORCED_LTSSM | H'0 | R/W | R/W | Forced Link Command.<br>The link command that the controller is forced to transmit when you set FORCE_EN bit (Force Link). |
| 7 to 0 | LINK_NUM | H'0 | R/W | R/W | Link Number.<br>Not used for endpoint. Not used for M-PCIe. |

### 104.2.191 Port Logic Registers 3 (PRTLGC3)

### Common to Header TYPE00/01

- Ack Frequency and L0-L1 ASPM Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | ENTER_ASPM | L1_ENTRANCE_LATENCY | | | L0S_ENTRANCE_LATENCY | | | COMMON_CLK_N_FTS | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ACK_N_FTS | | | | | | | | ACK_FREQ | | | | | | | |
| Initial value: | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 30 | ENTER_ASPM | 0 | R/W | R/W | ASPM L1 Entry Control.<br>1: Core enters ASPM L1 after a period in which it has been idle.<br>0: Core enters ASPM L1 only after idle period during which both receive and transmit are in L0s. |
| 29 to 27 | L1_ENTRANCE_LATENCY | H'0 | R/W | R/W | L1 Entrance Latency.<br>Value range is:<br>000: 1 us<br>001: 2 us<br>010: 4 us<br>011: 8 us<br>100: 16 us<br>101: 32 us<br>110 or 111: 64 us<br>Note: Programming this timer with a value greater that 32us has no effect unless extended sync is used, or all of the credits are infinite. |
| 26 to 24 | L0S_ENTRANCE_LATENCY | H'0 | R/W | R/W | L0s Entrance Latency.<br>Values correspond to:<br>000: 1 us<br>001: 2 us<br>010: 3 us<br>011: 4 us<br>000: 5 us<br>101: 6 us<br>110 or 111: 7 us<br>This field is applicable to STALL while in L0 for M-PCIe. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 23 to 16 | COMMON_CLK_N_FTS | H'0 | R | R | Common Clock N_FTS.<br>This is the N_FTS when common clock is used.<br>Define by parameter CX_COMM_NFTS. |
| 15 to 8 | ACK_N_FTS | H'BE | R/W | R/W | N_FTS.<br>The number of Fast Training Sequence ordered sets to be transmitted when transitioning from L0s to L0.<br>The maximum number of FTS ordered-sets that a component can request is 255.<br>The controller does not support a value of zero; a value of zero can cause the LTSSM to go into the recovery state when exiting from L0s.<br>This field is reserved (fixed to '0') for M-PCIe. |
| 7 to 0 | ACK_FREQ | H'0 | R/W | R/W | Ack Frequency. The controller accumulates the number of pending ACKs specified here (up to 255) before scheduling an ACK DLLP.<br>■ 0: Indicates that this Ack Frequency Counter feature is turned off. The controller generates a low-priority ACK request for every TLP that it receives. The controller waits until the ACK Latency Timer expires, then converts the current low-priority ACK request to a high-priority ACK request and schedules the DLLP for transmission to the remote link partner.<br>■ 1-255: Indicates that the controller will schedule a highpriority ACK after receiving this number of TLPs. It might schedule the ACK before receiving this number of TLPs if the ACK Latency Timer expires, but never later.<br>For a typical system, you do not have to modify the default setting. |

### 104.2.192 Port Logic Registers 4 (PRTLGC4)

### Common to Header TYPE00/01

- Port Link Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TRANSMIT_L | EXTENDED_SYN | CORRUPT_LCRC_ENABLE | BEACON_ENABL | CROSS_LINK_ACTIVE | CROSS_LINK_EN | LINK_CAPABLE | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | LINK_RATE | | | | FAST_LINK_MODE | LINK_DISABLE | DLL_LINK_EN | — | RESET_ASSERT | LOOPBACK_ENABLE | SCRAMBLE_DISABLE | VENDOR_SPECIFIC_DLLP_REQ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | RW1C |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | RW1C |

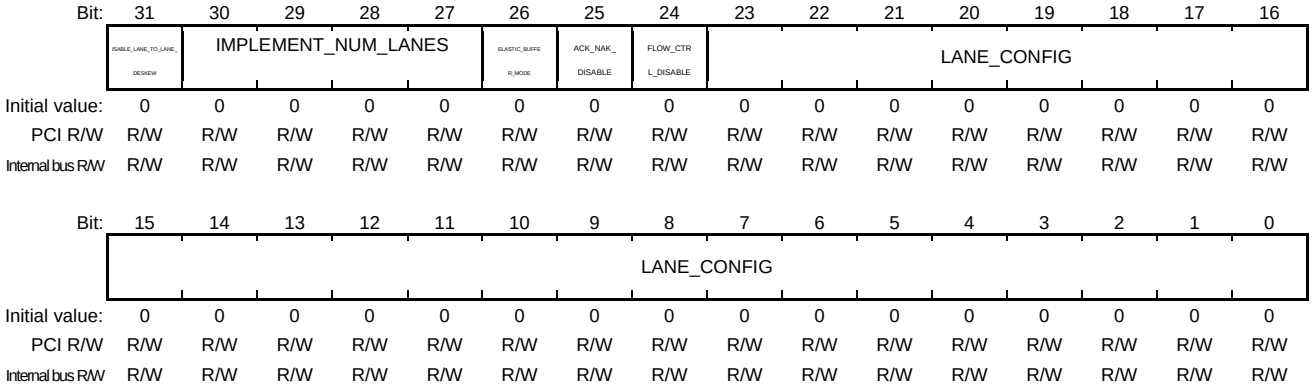
| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 28 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 | TRANSMIT_LANE_REVERSALE_ENABLE | 0 | R | R | TRANSMIT_LANE_REVERSALE_ENABLE is an internally reserved field. Do not use |
| 26 | EXTENDED_SYNCH | 0 | R | R | EXTENDED_SYNCH is an internally reserved field. Do not use. |
| 25 | CORRUPT_LCRC_ENABLE | 0 | R | R | CORRUPT_LCRC_ENABLE is an internally reserved field.Do not use. |
| 24 | BEACON_ENABLE | 0 | R | R | BEACON_ENABLE is an internally reserved field. Do not use. |
| 23 | CROSS_LINK_ACTIVE | 0 | R | R | Crosslink Active. Indicates a change from upstream to downstream or downstream to upstream. This field is reserved (fixed to '0') for M-PCIe. |
| 22 | CROSS_LINK_EN | 1 | R/W | R/W | Crosslink Enable.<br>This field is reserved (fixed to '0') for M-PCIe. |
| 21 to 16 | LINK_CAPABLE | H'0 | R/W | R/W | Link Mode Enable.<br>Sets the number of lanes in the link that you want to connect to the link partner.<br>000001: x1<br>000011: x2<br>000111: x4<br>001111: x8<br>011111: x16<br>111111: x32 (not supported)<br>This field is reserved (fixed to '0') for M-PCIe. |
| 15 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 11 to 8 | LINK_RATE | H'1 | R/W | R/W | LINK_RATE is an internally reserved field. Do not use. |
| 7 | FAST_LINK_MODE | 0 | R/W | R/W | Fast Link Mode.<br>Sets all internal LTSSM millisecond timers to Fast Mode for speeding up simulation. |
| 6 | LINK_DISABLE | 0 | R/W | R/W | LINK_DISABLE is an internally reserved field. Do not use. |
| 5 | DLL_LINK_EN | 1 | R/W | R/W | DLL Link Enable.<br>Enables link initialization. When DLL Link Enable =0, the controller does not transmit InitFC DLLPs and does not establish a link. |
| 4 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 3 | RESET_ASSERT | 0 | R/W | R/W | Reset Assert.<br>Triggers a recovery and forces the LTSSM to the hot reset state (downstream port only). |
| 2 | LOOPBACK_ENABLE | 0 | R/W | R/W | Loopback Enable.<br>Turns on loopback. For more details, see "Loopback". |
| 1 | SCRAMBLE_DISABLE | 0 | R/W | R/W | Scramble Disable. Turns off data scrambling. |
| 0 | VENDOR_SPECIFIC_ DLLP_REQ | 0 | RW1C | RW1C | Vendor Specific DLLP Request.<br>When software writes a '1' to this bit, the controller transmits the DLLP contained in the VENDOR_SPEC_DLLP field of PRTLGC1. |

### 104.2.193 Port Logic Registers 5 (PRTLGC5)

### Common to Header TYPE00/01

- Lane Skew Register




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | ISABLE_LANE_TO_<br>LANE_DESKEW | 0 | R/W | R/W | Disable Lane-to-Lane Deskew.<br>Causes the controller to disable the internal Lane-to-Lane deskew logic. |
| 30 to 27 | IMPLEMENT_NUM_<br>LANES | H'0 | R/W | R/W | Implementation-specific Number of Lanes.<br>Set the implementation-specific number of lanes.<br>Allowed values are:<br>H'0: 1 lane<br>H'1: 2 lanes<br>H'3: 4 lanes<br>H'7: 8 lanes<br>H'F: 16 lanes |
| 26 | ELASTIC_BUFFER_<br>MODE | 0 | R/W | R/W | Selects Elasticity Buffer operating mode:<br>0: Nominal Half Full Buffer mode<br>1: Nominal Empty Buffer Mode |
| 25 | ACK_NAK_DISABLE | 0 | R/W | R/W | Ack/Nak Disable.<br>Prevents the controller from sending ACK and NAK DLLPs. |
| 24 | FLOW_CTRL_<br>DISABLE | 0 | R/W | R/W | Flow Control Disable.<br>Prevents the controller from sending FC DLLPs. |
| 23 to 0 | LANE_CONFIG | H'0 | R/W | R/W | LANE_CONFIG field<br>H'00_0000: Quad lane mode<br>H'00_0040: Dual lane mode<br>Others: Setting prohibited |

### 104.2.194 Port Logic Registers 6 (PRTLGC6)

### Common to Header TYPE00/01

- Timer Control and Max Function Number Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | FAST_LINK_SCALING_FACTOR | | UPDATE_FREQ_TIMER | | | | | TIMER_MOD_ACK_NAK | | | | | TIMER_MOD_REPLAY_TIMER | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TIMER_MOD_REPLAY_TIMER | | — | — | — | — | — | — | MAX_FUNC_NUM | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 30 to 29 | FAST_LINK_SCALING_FACTOR | H'0 | R/W | R/W | Fast Link Timer Scaling Factor.<br>Sets the scaling factor of LTSSM timer when FAST_LINK_MODE field in PRTLGC4 is set to '1'.<br>H'0: Scaling Factor is 1024 (1ms is 1us *a)<br>H'1: Scaling Factor is 256 (1ms is 4us)<br>H'2: Scaling Factor is 64 (1ms is 16us)<br>H'3: Scaling Factor is 16 (1ms is 64us)<br>Default is set by the hidden configuration parameter DEFAULT_FAST_LINK_SCALING_FACTOR which defaults to '0'.<br>*a. When the LTSSM is in Config or L12 Entry State, 1ms timer is 2us, 2ms timer is 4us and 3ms timer is 6us. Not used for M-PCIe. |
| 28 to 24 | UPDATE_FREQ_TIMER | H'0 | R/W | R/W | UPDATE_FREQ_TIMER is an internally reserved field. Do not use. |
| 23 to 19 | TIMER_MOD_ACK_NAK | H'0 | R/W | R/W | Ack Latency Timer Modifier.<br>Increases the timer value for the Ack latency timer in increments of 64 clock cycles. A value of "0" represents no modification to the timer value. |
| 18 to 14 | TIMER_MOD_REPLAY_TIMER | H'0 | R/W | R/W | Replay Timer Limit Modifier.<br>Increases the time-out value for the replay timer in increments of 64 clock cycles at Gen1 or Gen2 speed, and in increments of 256 clock cycles at Gen3 speed. |
| 13 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | MAX_FUNC_NUM | H'0 | R/W | R/W | Maximum function number that can be used in a request.<br>Configuration requests targeted at function numbers above this value are returned with UR (unsupported request). |

### 104.2.195 Port Logic Registers 7 (PRTLGC7)

### Common to Header TYPE00/01

- Symbol Timer Register and Filter Mask 1 Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CX_FLT_MASK_RC_CFG_DISCARD | CX_FLT_MASK_RC_IO_DISCARD | CX_FLT_MASK_MSG_DROP | CX_FLT_MASK_CPL_ECRC_DISCA | CX_FLT_MASK_ECRC_DISCARD | CX_FLT_MASK_CPL_LEN_MATCH | CX_FLT_MASK_CPL_ATTR_MATCH | CX_FLT_MASK_CPL_TC_MATCH | CX_FLT_MASK_CPL_FUNC_MATC | CX_FLT_MASK_CPL_REQID_MATC | CX_FLT_MASK_CPL_TAGERR_MAT | CX_FLT_MASK_LOCKED_RD_AS_ | CX_FLT_MASK_CFG_TYPE1_REQ_ | CX_FLT_MASK_UR_OUTSIDE_BAR | CX_FLT_MASK_UR_POIS | CX_FLT_MASK_UR_FUNC_MISMAT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DISABLE_FC_WD_TIMER | EIDLE_TIMER | | | | SKP_INT_VAL | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | CX_FLT_MASK_RC_CFG_DISCARD | 0 | R/W | R/W | 0: For RADM RC filter to not allow CFG transaction being received<br>1: For RADM RC filter to allow CFG transaction being received |
| 30 | CX_FLT_MASK_RC_IO_DISCARD | 0 | R/W | R/W | 0: For RADM RC filter to not allow IO transaction being received<br>1: For RADM RC filter to allow IO transaction being received |
| 29 | CX_FLT_MASK_MSG_DROP | 0 | R/W | R/W | 0: Drop MSG TLP (except for Vendor MSG). Send decoded message on the SII.<br>1: Do not Drop MSG (except for Vendor MSG). Send message TLPs to your application on TRGT1 and send decoded message on the SII. |
| 28 | CX_FLT_MASK_CPL_ECRC_DISCARD | 0 | R/W | R/W | Only used when completion queue is advertised with infinite credits and is in store-and-forward mode.<br>0: Discard completions with ECRC errors<br>1: Allow completions with ECRC errors to be passed up<br>Reserved field for SW. |
| 27 | CX_FLT_MASK_ECRC_DISCARD | 0 | R/W | R/W | 0: Discard TLPs with ECRC errors<br>1: Allow TLPs with ECRC errors to be passed up |
| 26 | CX_FLT_MASK_CPL_LEN_MATCH | 0 | R/W | R/W | 0: Enforce length match for completions; a violation results in cpl_abort, and possibly AER of unexp_cpl_err<br>1: MASK length match for completions |
| 25 | CX_FLT_MASK_CPL_ATTR_MATCH | 0 | R/W | R/W | 0: Enforce attribute match for completions; a violation results in a malformed TLP error, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca<br>1: Mask attribute match for completions |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 24 | CX_FLT_MASK_CPL_ TC_MATCH | 0 | R/W | R/W | 0: Enforce Traffic Class match for completions;<br>a violation results in a malformed TLP error, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca<br>1: Mask Traffic Class match for completions |
| 23 | CX_FLT_MASK_CPL_ FUNC_MATCH | 0 | R/W | R/W | 0: Enforce function match for completions;<br>a violation results in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca<br>1: Mask function match for completions |
| 22 | CX_FLT_MASK_CPL_ REQID_MATCH | 0 | R/W | R/W | 0: Enforce Req. Id match for completions;<br>a violation result in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca<br>1: Mask Req. Id match for completions |
| 21 | CX_FLT_MASK_CPL_ TAGERR_MATCH | 0 | R/W | R/W | 0: Enforce Tag Error Rules for completions;<br>a violation result in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca<br>1: Mask Tag Error Rules for completions |
| 20 | CX_FLT_MASK_ LOCKED_RD_AS_UR | 0 | R/W | R/W | 0: Treat locked Read TLPs as UR for EP;<br>Supported for RC<br>1: Treat locked Read TLPs as Supported for EP;<br>UR for RC |
| 19 | CX_FLT_MASK_CFG_ TYPE1_REQ_AS_UR | 0 | R/W | R/W | 0: Treat CFG type1 TLPs as UR for EP;<br>Supported for RC<br>1: Treat CFG type1 TLPs as Supported for EP;<br>UR for RC |
| 18 | CX_FLT_MASK_ UR_OUTSIDE_BAR | 0 | R/W | R/W | 0: Treat out-of-bar TLPs as UR<br>1: Do not treat out-of-bar TLPs as UR |
| 17 | CX_FLT_MASK_ UR_POIS | 0 | R/W | R/W | 0: Treat poisoned request TLPs as UR<br>1: Do not treat poisoned request TLPs as UR |
| 16 | CX_FLT_MASK_UR_ FUNC_MISMATCH | 0 | R/W | R/W | 0: Treat Function MisMatched TLPs as UR<br>1: Do not treat Function MisMatched TLPs as UR |
| 15 | DISABLE_FC_WD_ TIMER | 0 | R/W | R/W | Disable FC Watchdog Timer |
| 14 to 11 | EIDLE_TIMER | H'0 | R/W | R/W | EIDLE_TIMER is an internally reserved field. Do not use. |
| 10 to 0 | SKP_INT_VAL | H'0 | R/W | R/W | SKP Interval Value.<br>The number of symbol times to wait between transmitting SKP ordered sets.<br>Note: This value is not used at Gen3 speed; the skip interval is hardcoded to 370 blocks.<br>For M-PCIe configurations, if the 2K_PPM_DISABLED field in the M-PCIe Configuration Attribute is changed, then this field is changed automatically as follows.<br>■ 2K_PPM_DISABLED=1: 1280 / CX_NB<br>■ 2K_PPM_DISABLED=0: 228/CX_NB<br>You need to set this field again if necessary when 2K_PPM_DISABLED is changed. |

### 104.2.196 Port Logic Registers 8 (PRTLGC8)

### Common to Header TYPE00/01

- Filter Mask 2 Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | CX_FLT_MASK_POIS_ERROR_REP | CX_FLT_MASK_PRS_DROP | CX_FLT_UNMASK_TD | CX_FLT_UNMASK_UR_POIS_TRGT | CX_FLT_MASK_LN_VENMSG1_DR | CX_FLT_MASK_HANDLE_FLUSH | CX_FLT_MASK_DABORT_4UCPL | CX_FLT_MASK_VENMSG1_DROP | CX_FLT_MASK_VENMSG0_DROP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CX_FLT_MASK_POIS_ERROR_REPORTING | 0 | R/W | R/W | 0: Disable masking of error reporting for Poisoned TLPs<br>1: Enable masking of error reporting for Poisoned TLPs |
| 7 | CX_FLT_MASK_PRS_DROP | 0 | R/W | R/W | 0: Allow PRS message to pass through<br>1: Drop PRS Messages silently<br>This bit is ignored when the CX_FLT_MASK_MSG_DROP bit in the PRTLGC7 register is set to '1'. |
| 6 | CX_FLT_UNMASK_TD | 0 | R/W | R/W | 0: Disable unmask TD bit if CX_STRIP_ECRC_ENABLE<br>1: Enable unmask TD bit if CX_STRIP_ECRC_ENABLE |
| 5 | CX_FLT_UNMASK_UR_POIS_TRGT0 | 0 | R/W | R/W | 0: Disable unmask CX_FLT_MASK_UR_POIS with TRGT0 destination<br>1: Enable unmask CX_FLT_MASK_UR_POIS with TRGT0 destination |
| 4 | CX_FLT_MASK_LN_VENMSG1_DROP | 0 | R/W | R/W | 0: Allow LN message to pass through<br>1: Drop LN Messages silently |
| 3 | CX_FLT_MASK_HANDLE_FLUSH | 0 | R/W | R/W | 0: Disable controller Filter to handle flush request<br>1: Enable controller Filter to handle flush request |
| 2 | CX_FLT_MASK_DABORT_4UCPL | 0 | R/W | R/W | 0: Enable DLLP abort for unexpected completion<br>1: Do not enable DLLP abort for unexpected completion |
| 1 | CX_FLT_MASK_VENMSG1_DROP | 0 | R/W | R/W | 0: Vendor MSG Type 1 dropped silently<br>1: Vendor MSG Type 1 not dropped |
| 0 | CX_FLT_MASK_VENMSG0_DROP | 0 | R/W | R/W | 0: Vendor MSG Type 0 dropped with UR error reporting<br>1: Vendor MSG Type 0 not dropped |

### 104.2.197 Port Logic Registers 9 (PRTLGC9)

### Common to Header TYPE00/01

- AMBA Multiple Outbound Decomposed NP SubRequests Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

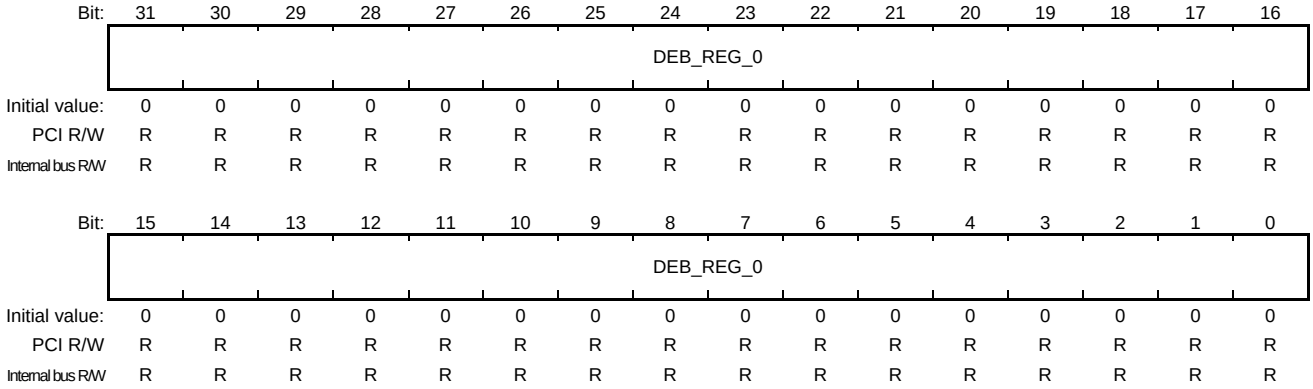
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OB_RD_SPLIT_BURST_EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | OB_RD_SPLIT_BURST_EN | 1 | R/W | R/W | Enable AMBA Multiple Outbound Decomposed NP SubRequests.<br>This bit when set to "0" disables the possibility of having multiple outstanding non-posted requests that were derived from decomposition of an outbound AMBA request. |

### 104.2.198 Port Logic Registers 10 (PRTLGC10)

### Common to Header TYPE00/01

- Debug Register 0




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | DEB_REG_0 | All 0 | R | R | The value on cxpl_debug_info[31:0]. |

### 104.2.199 Port Logic Registers 11 (PRTLGC11)

### Common to Header TYPE00/01

- Debug Register 1

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEB_REG_1 | | | | | | | | | | | | | | | |
| Initial value: | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEB_REG_1 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | DEB_REG_1 | H'8200_0000 | R | R | The value on cxpl_debug_info[63:32]. |

### 104.2.200 Port Logic Registers 12 (PRTLGC12)

### Common to Header TYPE00/01

- Transmit Posted FC Credit Status

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TX_P_HEADER_FC_CREDIT | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TX_P_HEADER_FC_CREDIT | | | | TX_P_DATA_FC_CREDIT | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 12 | TX_P_HEADER_ FC_CREDIT | H'0 | R | R | Transmit Posted Header FC Credits.<br>The posted Header credits advertised by the receiver at the other end of the link, updated with each UpdateFC DLLP. |
| 11 to 0 | TX_P_DATA_ FC_CREDIT | H'0 | R | R | Transmit Posted Data FC Credits.<br>The posted Data credits advertised by the receiver at the other end of the link, updated with each UpdateFC DLLP. |

### 104.2.201 Port Logic Registers 13 (PRTLGC13)

### Common to Header TYPE00/01

- Transmit Non-Posted FC Credit Status

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TX_NP_HEADER_FC_CREDIT | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TX_NP_HEADER_FC_CREDIT | | | | TX_NP_DATA_FC_CREDIT | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 12 | TX_NP_HEADER_FC_CREDIT | H'0 | R | R | Transmit Non-Posted Header FC Credits.<br>The non-posted Header credits advertised by the receiver at the other end of the link, updated with each UpdateFC DLLP. |
| 11 to 0 | TX_NP_DATA_FC_CREDIT | H'0 | R | R | Transmit Non-Posted Data FC Credits.<br>The non-posted Data credits advertised by the receiver at the other end of the link, updated with each UpdateFC DLLP. |

### 104.2.202 Port Logic Registers 14 (PRTLGC14)

### Common to Header TYPE00/01

- Transmit Completion FC Credit Status

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TX_CPL_HEADER_FC_CREDIT | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

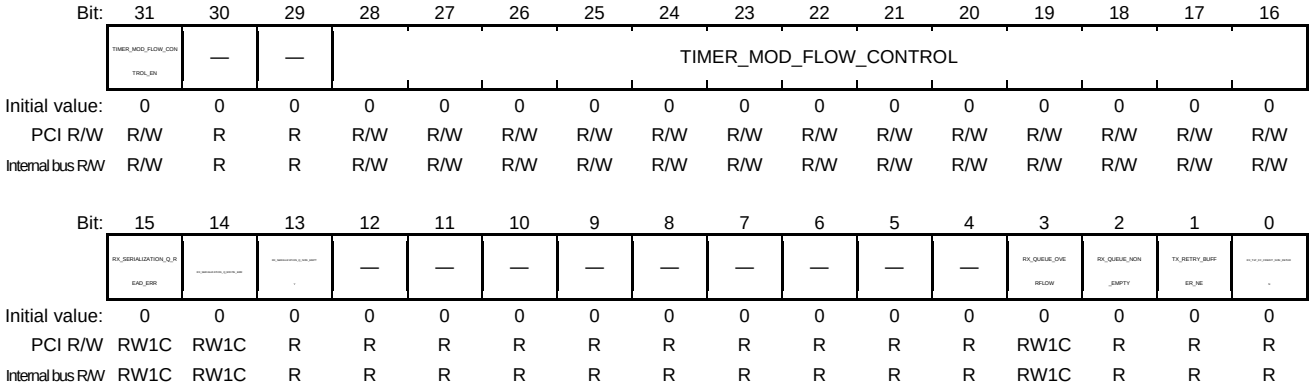
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TX_CPL_HEADER_FC_CREDIT | | | | TX_CPL_DATA_FC_CREDIT | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 12 | TX_CPL_HEADER_FC_CREDIT | H'0 | R | R | Transmit Completion Header FC Credits.<br>The Completion Header credits advertised by the receiver at the other end of the link, updated with each UpdateFC DLLP. |
| 11 to 0 | TX_CPL_DATA_FC_CREDIT | H'0 | R | R | Transmit Completion Data FC Credits.<br>The Completion Data credits advertised by the receiver at the other end of the link, updated with each UpdateFC DLLP. |

### 104.2.203 Port Logic Registers 15 (PRTLGC15)

### Common to Header TYPE00/01

- Queue Status



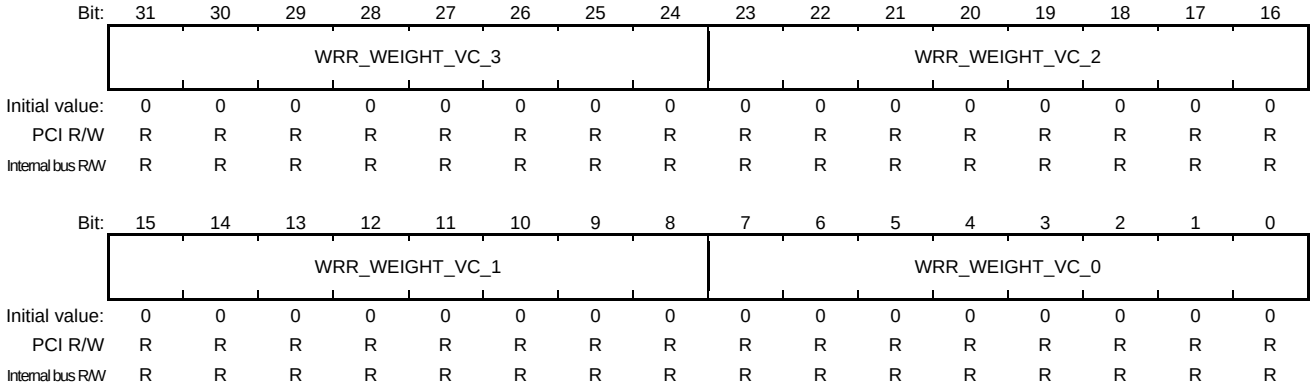

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | TIMER_MOD_FLOW_CONTROL_EN | 0 | R/W | R/W | FC Latency Timer Override Enable.<br>When this bit is set, the value from the "FC Latency Timer Override Value" field in this register will override the FC latency timer value that the controller calculates according to the PCIe specification. |
| 30 to 29 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 16 | TIMER_MOD_FLOW_CONTROL | H'0 | R/W | R/W | FC Latency Timer Override Value.<br>When you set the "FC Latency Timer Override Enable" in this register, the value in this field will override the FC latency timer value that the controller calculates according to the PCIe specification. |
| 15 | RX_SERIALIZATION_Q_READ_ERR | 0 | RW1C | RW1C | Receive Serialization Read Error.<br>Indicates the serialization queue has attempted to read an incorrectly formatted TLP. |
| 14 | RX_SERIALIZATION_Q_WRITE_ERR | 0 | RW1C | RW1C | Receive Serialization Queue Write Error.<br>Indicates insufficient buffer space available to write to the serialization queue. |
| 13 | RX_SERIALIZATION_Q_NON_EMPTY | 0 | R | R | Receive Serialization Queue Not Empty.<br>Indicates there is data in the serialization queue. |
| 12 to 4 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 3 | RX_QUEUE_OVERFLOW | 0 | RW1C | RW1C | Receive Credit Queue Overflow.<br>Indicates insufficient buffer space available to write to the P/NP/CPL credit queue. |
| 2 | RX_QUEUE_NON_EMPTY | 0 | R | R | Receive Credit Queue Not Empty.<br>Indicates there is data in one or more of the receive buffers. |
| 1 | TX_RETRY_BUFFER_NE | 0 | R | R | Transmit Retry Buffer Not Empty.<br>Indicates that there is data in the transmit retry buffer. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | RX_TLP_FC_CREDIT_NON_RETURN | 0 | R | R | Received TLP FC Credits Not Returned.<br>Indicates that the controller has received a TLP but has not yet sent an UpdateFC DLLP indicating that the credits for that TLP have been restored by the receiver at the other end of the link. |

### 104.2.204 Port Logic Registers 16 (PRTLGC16)

### Common to Header TYPE00/01

- VC Transmit Arbitration Register 1

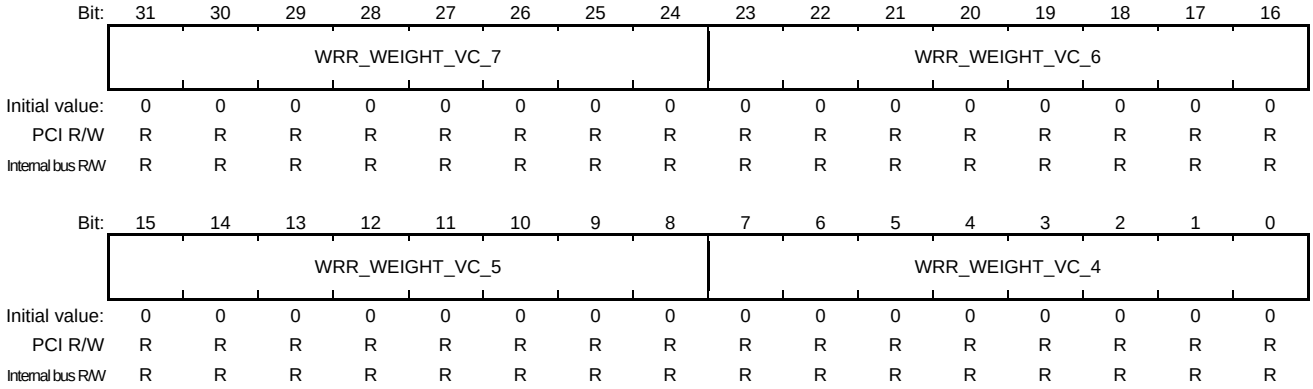



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | WRR_WEIGHT_VC_3 | H'0 | R | R | WRR Weight for VC3 |
| 23 to 16 | WRR_WEIGHT_VC_2 | H'0 | R | R | WRR Weight for VC2 |
| 15 to 8 | WRR_WEIGHT_VC_1 | H'0 | R | R | WRR Weight for VC1 |
| 7 to 0 | WRR_WEIGHT_VC_0 | H'0 | R | R | WRR Weight for VC0 |

### 104.2.205 Port Logic Registers 17 (PRTLGC17)

### Common to Header TYPE00/01

- VC Transmit Arbitration Register 2

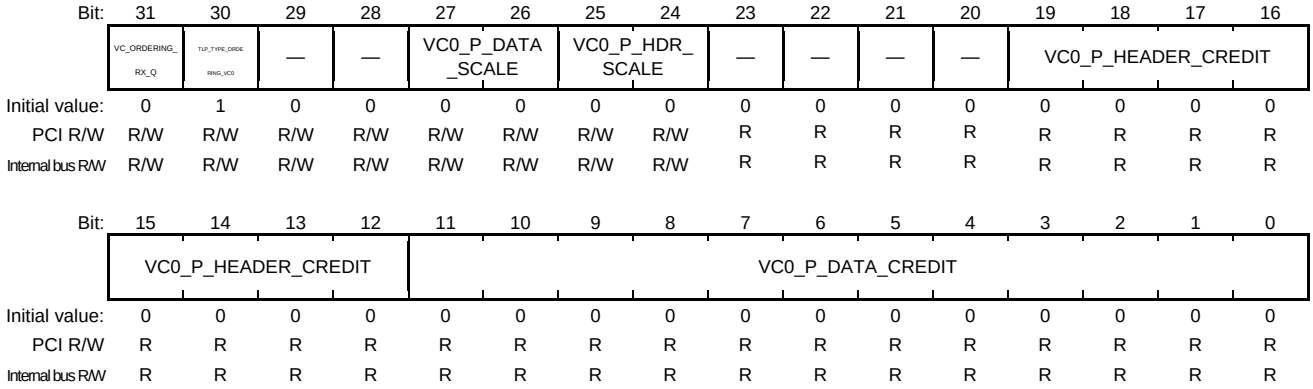



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | WRR_WEIGHT_VC_7 | H'0 | R | R | WRR Weight for VC7 |
| 23 to 16 | WRR_WEIGHT_VC_6 | H'0 | R | R | WRR Weight for VC6 |
| 15 to 8 | WRR_WEIGHT_VC_5 | H'0 | R | R | WRR Weight for VC5 |
| 7 to 0 | WRR_WEIGHT_VC_4 | H'0 | R | R | WRR Weight for VC4 |

### 104.2.206 Port Logic Registers 18 (PRTLGC18)

### Common to Header TYPE00/01

- Segmented-Buffer VC0 Posted Receive Queue Control




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | VC_ORDERING_ RX_Q | 0 | R/W | R/W | VC Ordering for Receive Queues.<br>Determines the VC ordering rule for the receive queues, used only in the segmented-buffer configuration:<br>1: Strict ordering, higher numbered VCs have higher priority<br>0: Round robin |
| 30 | TLP_TYPE_ ORDERING_VC0 | 1 | R/W | R/W | TLP Type Ordering for VC0.<br>Determines the TLP type ordering rule for VC0 receive queues, used only in the segmented-buffer configuration:<br>1: PCIe ordering rules (recommended)<br>0: Strict ordering: posted, completion, then non-posted |
| 29 to 28 | — | All 0 | R/W | R/W | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 26 | VC0_P_DATA_SCALE | H'0 | R/W | R/W | VC0 Scale Posted Data Credites. |
| 25 to 24 | VC0_P_HDR_SCALE | H'0 | R/W | R/W | VC0 Scale Posted Header Credites. |
| 23 to 20 | — | All 0 | R | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 19 to 12 | VC0_P_HEADER_ CREDIT | H'0 | R | R | VC0 Posted Header Credits.<br>The number of initial posted header credits for VC0, used only in the segmented-buffer configuration. |
| 11 to 0 | VC0_P_DATA_CREDIT | H'0 | R | R | VC0 Posted Data Credits.<br>The number of initial posted data credits for VC0, used only in the segmented-buffer configuration. |

### 104.2.207 Port Logic Registers 19 (PRTLGC19)

### Common to Header TYPE00/01

- Segmented-Buffer VC0 Non-Posted Receive Queue Control

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VC0_NP_DATA_SCALE | | VC0_NP_HDR_SCALE | | — | — | — | — | VC0_NP_HEADER_CREDIT | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VC0_NP_HEADER_CREDIT | | | | VC0_NP_DATA_CREDIT | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 28 | — | All 0 | R/W | R/W | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27to 26 | VC0_NP_DATA_SCALE | H'0 | R/W | R/W | VC0 Scale Non-Posted Data Credites. |
| 25 to 24 | VC0_NP_HDR_SCALE | H'0 | R/W | R/W | VC0 Scale Non-Posted Header Credites. |
| 23 to 20 | — | All 0 | R | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 19 to 12 | VC0_NP_HEADER_CREDIT | H'0 | R | R | VC0 Non-Posted Header Credits.<br>The number of initial nonposted header credits for VC0, used only in the segmentedbuffer configuration. |
| 11 to 0 | VC0_NP_DATA_CREDIT | H'0 | R | R | VC0 Non-Posted Data Credits.<br>The number of initial nonposted data credits for VC0, used only in the segmented-buffer configuration. |

### 104.2.208 Port Logic Registers 20 (PRTLGC20)

### Common to Header TYPE00/01

- Segmented-Buffer VC0 Completion Receive Queue Control.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | VC0_CPL_DATA_SCALE | | VC0_CPL_HDR_SCALE | | — | — | — | — | VC0_CPL_HEADER_CREDIT | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VC0_CPL_HEADER_CREDIT | | | | VC0_CPL_DATA_CREDIT | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 28 | — | All 0 | R/W | R/W | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 27 to 26 | VC0_CPL_ DATA_SCALE | H'0 | R/W | R/W | VC0 Scale CPL Data Credites. |
| 25 to 24 | VC0_CPL_HDR_ SCALE | H'0 | R/W | R/W | VC0 Scale CPL Header Credites. |
| 23 to 20 | — | All 0 | R | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 19 to 12 | VC0_CPL_HEADER_ CREDIT | H'0 | R | R | VC0 Completion Header Credits.<br>The number of initial Completion header credits for VC0, used only in the segmented-buffer configuration. |
| 11 to 0 | VC0_CPL_DATA_ CREDIT | H'0 | R | R | VC0 Completion Data Credits.<br>The number of initial Completion data credits for VC0, used only in the segmented-buffer configuration. |

### 104.2.209 Port Logic Registers 21 (PRTLGC21)

### Common to Header TYPE00/01

- Link Width and Speed Change Control Register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | GEN1_EI_INFERENCE | SEL_DEEMPHASIS | CONFIG_TX_COMP_RX | CONFIG_PHY_TX_CHANGE | DIRECT_SPEED_CHANGE | AUTO_LANE_FLIP_CTRL_EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PRE_DET_LANE | | | NUM_OF_LANES | | | | | FAST_TRAINING_SEQ | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 22 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 21 | GEN1_EI_INFERENCE | 0 | R/W | R/W | Electrical Idle Inference Mode at Gen1 Rate.<br>Programmable mode to determine inferred electrical idle (EI) in Recovery.<br>0: Use RxElecIdle signal to infer Electrical Idle<br>1: Use RxValid signal to infer Electrical Idle |
| 20 | SEL_DEEMPHASIS | 0 | R/W | R/W | Used to set the de-emphasis level for upstream ports.<br>This bit selects the level of de-emphasis the link operates at.<br>0: -6 dB<br>1: -3.5 dB<br>This field is reserved (fixed to '0') for M-PCIe. |
| 19 | CONFIG_TX_COMP_<br>RX | 0 | R/W | R/W | Config Tx Compliance Receive Bit.<br>When set to 1, signals LTSSM to transmit TS ordered sets with the compliance receive bit assert (equal to "1"). This field is reserved (fixed to '0') for M-PCIe. |
| 18 | CONFIG_PHY_TX_<br>CHANGE | 0 | R/W | R/W | Config PHY Tx Swing.<br>Controls the PHY transmitter voltage swing level. The controller drives the mac_phy_txswing output from this register bit field.<br>0: Full Swing<br>1: Low Swing<br>This field is reserved (fixed to '0') for M-PCIe. |
| 17 | DIRECT_SPEED_<br>CHANGE | 0 | R/W | R/W | Directed Speed Change.<br>Writing "1" to this field instructs the LTSSM to initiate a speed change to Gen2 or Gen3 after the link is initialized at Gen1 speed. |
| 16 | AUTO_LANE_<br>FLIP_CTRL_EN | 0 | R/W | R/W | Enable Auto flipping of the lanes.<br>You must set the CX_AUTO_LANE_FLIP_CTRL_EN configuration parameter to include the hardware for this feature in the controller. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 15 to 13 | PRE_DET_LANE | H'0 | R/W | R/W | Predetermined Lane for Auto Flip.<br>This field defines which physical lane is connected to logical Lane0 by the flip operation performed in Detect.<br>H'0: Connect logical Lane0 to physical lane 0 or<br>CX_NL-1 or CX_NL/2-1 or CX_NL/4-1 or CX_NL/8-1, depending on which lane is detected<br>H'1: Connect logical Lane0 to physical lane 1<br>H'2: Connect logical Lane0 to physical lane 3<br>H'3: Connect logical Lane0 to physical lane 7<br>H'4: Connect logical Lane0 to physical lane 15 |
| 12 to 8 | NUM_OF_LANES | H'0 | R/W | R/W | Predetermined Number of Lanes.<br>Defines the number of lanes which are connected and not bad.<br>H'1: 1 lane<br>H'2: 2 lanes<br>H'3: 3 lanes |
| 7 to 0 | FAST_TRAINING_SEQ | H'0 | R/W | R/W | Sets the Number of Fast Training Sequences (N_FTS) that the controller advertises as its N_FTS during Gen2 or Gen3 link training. |

### 104.2.210 Port Logic Registers 22 (PRTLGC22)

### Common to Header TYPE00/01

- PHY Status Register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PHY_STATUS | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

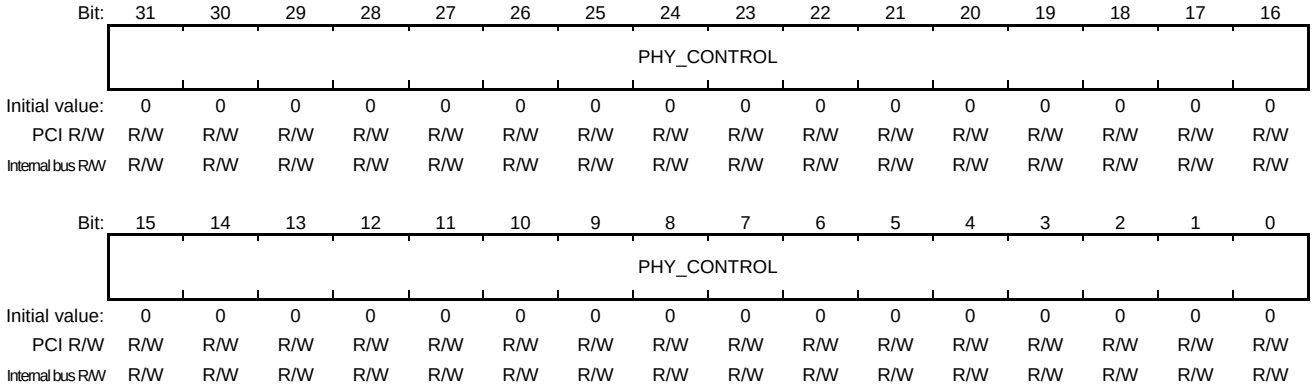
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PHY_STATUS | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PHY_STATUS | H'0 | R | R | PHY Status.<br>Data received directly from the phy_cfg_status bus. |

### 104.2.211 Port Logic Registers 23 (PRTLGC23)

### Common to Header TYPE00/01

- PHY Control Register




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | PHY_CONTROL | H'0 | R/W | R/W | PHY Control. Data sent directly to the cfg_phy_control bus.<br>These is a GPIO register driving the values on the static cfg_phy_control output signals. |

### 104.2.212 Port Logic Registers 24 (PRTLGC24)

### Common to Header TYPE00/01

- Programmable Target Map Control Register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | TARGET_MAP_INDEX | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

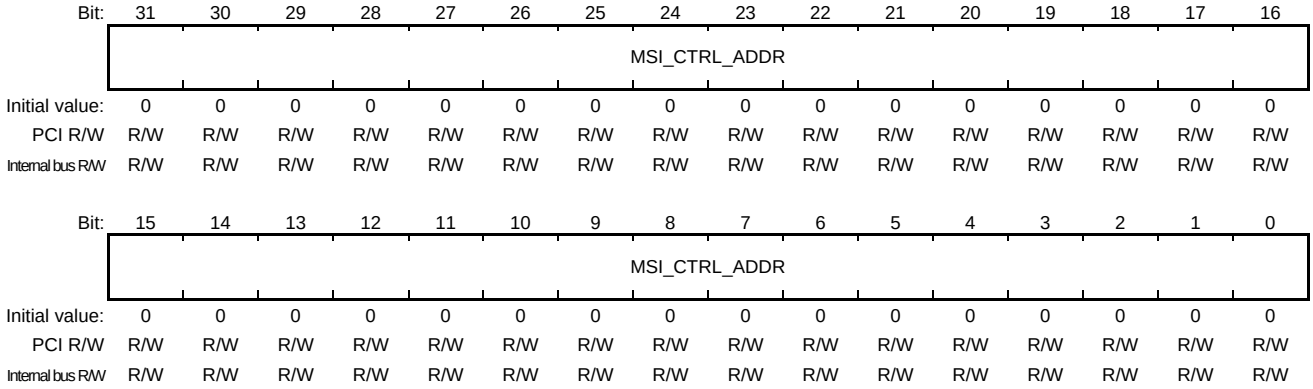
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | TARGET_MAP_VF | | | | | | TARGET_MAP_ROM | TARGET_MAP_PF | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 20 to 16 | TARGET_MAP_INDEX | H'0 | R/W | R/W | The number of the PF Function on which the Target Values are set.<br>This register does not respect the Byte Enable setting. any write will affect all register bits. |
| 15 to 13 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 to 7 | TARGET_MAP_VF | H'0 | R | R | Target Values for each BAR on the VF Function selected by the index number.<br>This register does not respect the Byte Enable setting. any write will affect all register bits.<br>Not support virtual channel capability structure. Read value must be H'0. |
| 6 | TARGET_MAP_ROM | 0 | R/W | R/W | Target Value for the ROM page of the PF Function selected by the index number. This register does not respect the Byte Enable setting. any write will affect all register bits. |
| 5 to 0 | TARGET_MAP_PF | H'0 | R/W | R/W | Target Values for each BAR on the PF Function selected by the index number.<br>This register does not respect the Byte Enable setting. any write will affect all register bits. |

### 104.2.213 Port Logic Registers 25 (PRTLGC25)

**Header TYPE00 Only**

- Integrated MSI Reception Module (iMRM) Address Register.

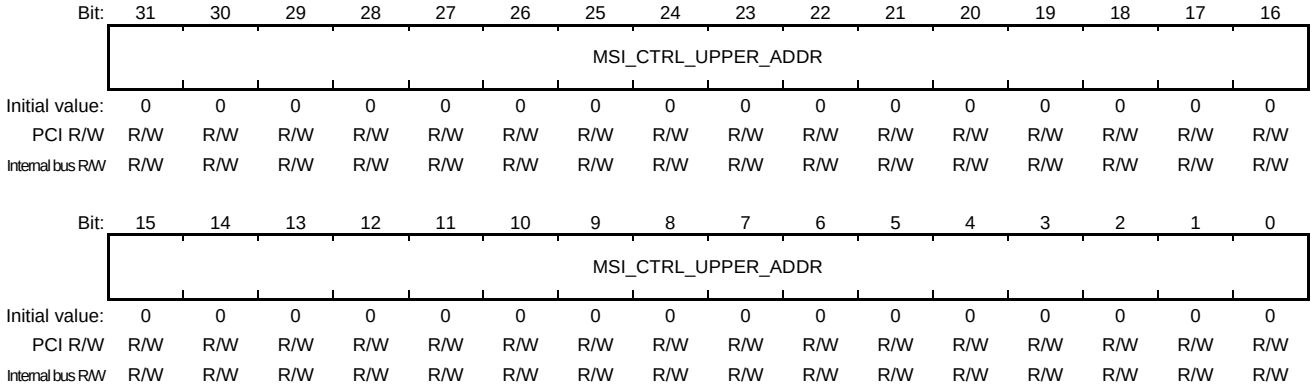



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | MSI_CTRL_ADDR | H'0 | R/W | R/W | Integrated MSI Reception Module Address.<br>System specified address for MSI memory write transaction termination. |

### 104.2.214 Port Logic Registers 26 (PRTLGC26)

### Header TYPE00 Only

- Integrated MSI Reception Module Upper Address Register




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | MSI_CTRL_UPPER_ ADDR | H'0 | R/W | R/W | Integrated MSI Reception Module Upper Address.<br>System specified upper address for MSI memory write transaction termination. |

### 104.2.215 Port Logic Registers 27+3**i* (PRTLGC27+3**i*) (*i* = 0 to 7)

**Header TYPE00 Only**

- Integrated MSI Reception Module Interrupt#i Enable Register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSI_CTRL_INT_0_EN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSI_CTRL_INT_0_EN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | MSI_CTRL_INT_0_EN | H'0 | R/W | R/W | MSI Interrupt#i Enable.<br>Specifies which interrupts are enabled.<br>Note: i = 0 to 7 |

### 104.2.216 Port Logic Registers 28+3*i* (PRTLGC28+3*i*) (*i* = 0 to 7)

**Header TYPE00 Only**

- Integrated MSI Reception Module Interrupt#i Mask Register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSI_CTRL_INT_0_MASK | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSI_CTRL_INT_0_MASK | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | MSI_CTRL_INT_0_ MASK | H'0 | R/W | R/W | MSI Interrupt#i Mask.<br>Allows enabled interrupts to be masked.<br>Note: i = 0 to 7 |

### 104.2.217 Port Logic Registers 29+3*i* (PRTLGC29+3*i*) (*i* = 0 to 7)

**Header TYPE00 Only**

- Integrated MSI Reception Module Interrupt#i Status Register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSI_CTRL_INT_0_STATUS | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |
| Internal bus R/W | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |

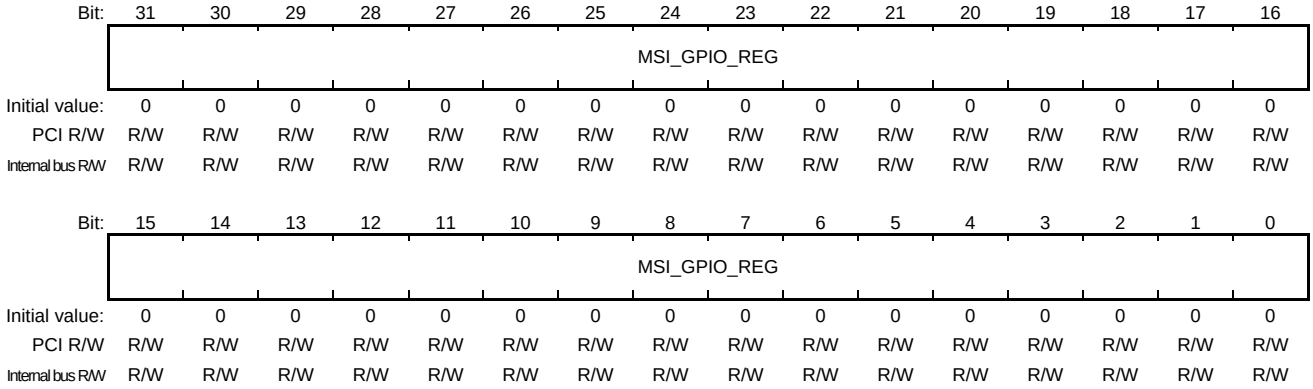
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSI_CTRL_INT_0_STATUS | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |
| Internal bus R/W | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | MSI_CTRL_INT_0_STATUS | H'0 | RW1C | RW1C | MSI Interrupt#i Status.<br>When an MSI is detected for EP#i, one bit in this register is set.<br>Note: i = 0 to 7 |

### 104.2.218 Port Logic Registers 51 (PRTLGC51)

### Header TYPE00 Only

- Integrated MSI Reception Module General Purpose IO Register




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | MSI_GPIO_REG | H'0 | R/W | R/W | MSI GPIO Register.<br>The contents of this register drives the top-level GPIO msi_ctrl_io[31:0] |

### 104.2.219 Port Logic Registers 52 (PRTLGC52)

### Common to Header TYPE00/01

- RADM clock gating enable control register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | R_AXI_CLK_GATING_EN | RADM_CLK_GATING_EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | R_AXI_CLK_GATING_EN | 1 | R/W | R/W | AXI Clock Gating Enable.<br>This register enables the AXI Bridge to autonomously enable and disable the AXI Master clock, the AXI Slave clock and the AXI DBI slave clock.<br>0: Disable<br>1: Enable (default) |
| 0 | RADM_CLK_ GATING_EN | 1 | R/W | R/W | Enable Radm clock gating feature.<br>0: Disable<br>1: Enable(default) |

### 104.2.220 Port Logic Registers 53 (PRTLGC53)

### Common to Header TYPE00/01

- Gen3 Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | RATE_SHADOW_SEL | | GEN3_EQ_INVREQ_EVAL_DIFF_DISABLE | USP_SEND_8GT_EQ_TS2_DISABLE | AUTO_EQ_DISABLE | — | — | GEN3_DC_BALANCE_DISABLE | GEN3_DLLP_XMT_DELAY_DISABLE | GEN3_EQUALIZATION_DISABLE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R/W | R/W | R | *1 | *1 | R | R | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | RXEQ_RGRDLESS_RXTS | RXEQ_PH01_EN | EQ_REDO | EQ_EIEOS_CNT | EQ_PHASE_2_3 | DISABLE_SCRAMBLER_GEN_3 | — | — | — | — | — | — | — | GEN3_ZRXDC_NONCOMPL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | *1 | *1 | R/W | R/W | *1 | R/W | R | R | R | R | R | R | R | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R | R/W | R | R | R | R | R | R | R | R/W |

Note: *1: R/W changes by the TYPE. TYPE00: R/W, TYPE01: -

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 to 24 | RATE_SHADOW_SEL | H'0 | R/W | R/W | Rate Shadow Select.<br>This register value decide the Data Rate of shadow register.<br>H'0: Gen3 Data Rate is selected for shadow register.<br>H'1: Gen4 Data Rate is selected for shadow register.<br>H'2: Gen5 Data Rate is selected for shadow register.<br>H'3: Reserved. |
| 23 | GEN3_EQ_INVREQ_EVAL_DIFF_DISABLE | 0 | R/W | R/W | Eq InvalidRequest and RxEqEval Different Time Assertion Disable.<br>Disable the assertion of Eq InvalidRequest and RxEqEval at different time. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 22 | USP_SEND_8GT_EQ_TS2_DISABLE | 1 | *1 | R/W | Upstream Port Send 8GT/s or 16GT/s EQ TS2 Disable.<br>The base spec defines that USP can optionally send 8GT or 16GT EQ TS2 and it means USP can set DSP TxPreset value in Gen4 or Gen5 Data Rate. If this register set to 0, USP sends 8GT or 16GT EQ TS2. If this register set to 1, USP does not send 8GT or 16GT EQ TS2. This applies to upstream ports only.<br>If RATE_SHADOW_SEL==00b, this register is RSVD.<br>If RATE_SHADOW_SEL==01b, this register is for Gen4 data rate.<br>If RATE_SHADOW_SEL==10b, this register is for Gen5 data rate. |
| 21 | AUTO_EQ_DISABLE | 0 | *1 | R* | Autonomous Equalization Disable.<br>When the controller is in L0 state at Gen3 data rate and equalization was completed successfully in Autonomous EQ Mechanism, setting this bit in DSP will not direct the controller to Recovery state to perform Gen4 equalization.<br>Note *:<br>If RATE_SHADOW_SEL==00b, this register is RSVD.<br>If RATE_SHADOW_SEL==01b, this register is for Gen4 data rate.<br>If RATE_SHADOW_SEL==10b, this register is for Gen5 data rate. |
| 20 to 19 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 18 | GEN3_DC_BALANCE_DISABLE | 0 | R/W | R/W | DC Balance Disable. Disable DC Balance feature. |
| 17 | GEN3_DLLP_XMT_DELAY_DISABLE | 0 | R/W | R/W | DLLP Transmission Delay Disable.<br>Disable delay transmission of DLLPs before equalization. |
| 16 | GEN3_EQUALIZATION_DISABLE | 0 | R/W | R/W | Equalization Disable. Disable equalization feature.<br>This bit cannot be changed once the LTSSM starts link training. |
| 15 to 14 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 | RXEQ_RGRDLESS_RXTS | 0 | *1 | R/W | When set to '1', the controller as Gen3 EQ master asserts RxEqEval to instruct the PHY to do Rx adaptation and evaluation after a 500ns timeout from a new preset request.<br>0: mac_phy_rxeqeval asserts after 1us and 2 TS1 received from remote partner.<br>1: mac_phy_rxeqeval asserts after 500ns regardless of TS's received or not |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 12 | RXEQ_PH01_EN | 0 | *1 | R/W | Rx Equalization Phase 0/Phase 1 Hold Enable.<br>When this bit is set the upstream port holds phase 0 (the downstream port holds phase 1) for 10ms.<br>The programmable bits [RXEQ_PH01_EN, EQ_PHASE_2_3] can be used to obtain the following variations of the equalization procedure:<br>00: Tx equalization only in phase 2/3<br>01: No Tx equalization, no Rx equalization<br>10: Tx equalization in phase 2/3, Rx equalization in phase 0/1<br>11: No Tx equalization, Rx equalization in phase 0/1<br>If RATE_SHADOW_SEL==01b, this register is for Gen4 data rate.<br>If RATE_SHADOW_SEL==10b, this register is for Gen5 data rate. |
| 11 | EQ_REDO | 0 | R/W | R/W | Equalization Redo Disable.<br>Disable autonomous mechanism for requesting to redo the equalization process. |
| 10 | EQ_EIEOS_CNT | 0 | R/W | R/W | Equalization EIEOS Count Reset Disable.<br>Disable requesting reset of EIEOS count during equalization. |
| 9 | EQ_PHASE_2_3 | 0 | *1 | R/W | Equalization Phase 2 and Phase 3 Disable.<br>This applies to downstream ports only.<br>If RATE_SHADOW_SEL==00b, this register is for Gen3<br>data rate.<br>If RATE_SHADOW_SEL==01b, this register is for<br>Gen4 data rate.<br>If RATE_SHADOW_SEL==10b, this register<br>is for Gen5 data rate. |
| 8 | DISABLE_<br>SCRAMBLER_GEN_3 | 0 | R/W | R/W | Disable Scrambler for Gen3 and Gen4 Data Rate.<br>The Gen3 and Gen4 scrambler/descrambler within the controller needs to be disabled when the scrambling function is implemented outside of the controller (for example within the PHY). |
| 7 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | GEN3_ZRXDC_ NONCOMPL | 1 | R/W | R/W | Gen3 Receiver Impedance ZRX-DC Not Compliant.<br>Receivers that operate at 8.0 GT/s with an impedance otherthan the range defined by the ZRX-DC parameter for 2.5GT/s (40-60 Ohms) must meet additional behavior requirements in the following LTSSM states: Polling, Rx_L0s, L1, L2, and Disabled.<br>0: The receiver complies with the ZRX-DC parameter for<br>2.5 GT/s when operating at 8 GT/s or higher.<br>1: The receiver does not comply with the ZRX-DC parameter<br>for 2.5 GT/s when operating at 8 GT/s or higher. |

### 104.2.221 Port Logic Registers 54 (PRTLGC54)

### Common to Header TYPE00/01

- Gen3 EQ Control Register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | | — | | GEN3_EQ_PSET_REQ_VEC | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GEN3_EQ_PSET_REQ_VEC | | | | | | | | — | | | | GEN3_EQ_FB_MODE | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 26 | GEN3_REQ_SEND _CONSEC_EIEOS_ FOR_PSET_MAP | 1 | R/W | R/W | Request controller to send back-to-back EIEOS in Recovery.RcvrLock state until presets to coefficients mapping is complete.<br>■ 0: Do not request<br>■ 1: request<br>Note: Gen3 and Gen4 share the same register bit and have the same feature. |
| 25 | — | 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

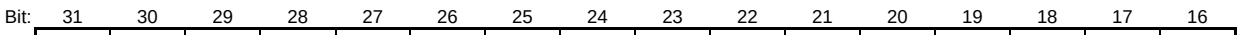
| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 24 | GEN3_EQ_FOM_INC_INITIAL_EVAL | 0 | R/W | R/W | Include Initial FOM. Include or not the FOM feedback from the initial preset evaluation performed in the EQ Master, when finding the highest FOM among all preset evaluations.<br>■ 0: Do not include<br>■ 1: Include |
| 23 to 8 | GEN3_EQ_PSET_REQ_VEC | All 0 | R/W | R/W | Preset Request Vector. Requesting of Presets during the initial part of the EQ Master Phase. Encoding scheme is as follows:<br>Bit [15:0] =H'0000: No preset is requested and evaluated in EQ Master Phase.<br>Bit [i] =1: "Preset=i" is requested and evaluated in EQ Master Phase.<br>■ 0000000000000000: No preset be requested and evaluated in EQ Master Phase<br>■ 000000xxxxxxxxx1: Preset 0 is requested and evaluated in EQ Master Phase<br>■ 000000xxxxxxxx1x: Preset 1 is requested and evaluated in EQ Master Phase<br>■ 000000xxxxxxx1xx: Preset 2 is requested and evaluated in EQ Master Phase<br>■ 000000xxxxxx1xxx: Preset 3 is requested and evaluated in EQ Master Phase<br>■ 000000xxxxx1xxxx: Preset 4 is requested and evaluated in EQ Master Phase<br>■ 000000xxxx1xxxxx: Preset 5 is requested and evaluated in EQ Master Phase<br>■ 000000xxx1xxxxxx: Preset 6 is requested and evaluated in EQ Master Phase<br>■ 000000xx1xxxxxxx: Preset 7 is requested and evaluated in EQ Master Phase<br>■ 000000x1xxxxxxxx: Preset 8 is requested and evaluated in EQ Master Phase<br>■ 00000x1xxxxxxxxx: Preset 9 is requested and evaluated in EQ Master Phase<br>■ 000001xxxxxxxxxx: Preset 10 is requested and evaluated in EQ Master Phase<br>■ All other encodings: Reserved |
| 7 | — | 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 | GEN3_LOWER_RATE_EQ_REDO_ENABLE | 0 | R/W | R/W | Support EQ redo and lower rate change:<br>■ 0: not support<br>■ 1: support<br>Note: Gen3 and Gen4 share the same register bit and have the same feature. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 5 | GEN3_EQ_EVAL_2MS_DISABLE | 0 | R/W | R/W | Phase2_3 2 ms Timeout Disable. Determine behavior in Phase2 for USP (Phase3 if DSP) when the PHY does not respond within 2ms to the assertion of RxEqEval:<br>■ 0: abort the current evaluation, stop any attempt to modify the remote transmitter settings, Phase2 is terminated by the 24ms timeout<br>■ 1: ignore the 2ms timeout and continue as normal. This is used to support PHYs that require more than 2ms to respond to the assertion of RxEqEval. |
| 4 | GEN3_EQ_PHASE23_EXIT_MODE | 0 | R/W | R/W | Behavior After 24 ms Timeout (when optimal settings are not found).<br>For a USP: Determine next LTSSM state from Phase2 after 24ms Timeout<br>■ 0: Recovery.Speed<br>■ 1: Recovery.Equalization.Phase3<br>When optimal settings are not found then:<br>■ Equalization Phase 2 Successful status bit is not set in the "Link Status Register 2" when<br>GEN3_EQ_PHASE23_EXIT_MODE = 0<br>■ Equalization Phase 2 Successful status bit is set in the "Link Status Register 2" when<br>GEN3_EQ_PHASE23_EXIT_MODE = 1<br>■ Equalization Phase 2 Complete status bit is set in the "Link Status Register 2"<br>For a DSP: Determine next LTSSM state from Phase3 after 24ms Timeout<br>■ 0: Recovery.Speed<br>■ 1: Recovery.Equalization.RcvrLock<br>When optimal settings are not found then:<br>■ Equalization Phase 3 Successful status bit is not set in the "Link Status Register 2" when<br>GEN3_EQ_PHASE23_EXIT_MODE = 0<br>■ Equalization Phase 3 Successful status bit is set in the "Link Status Register 2" when<br>GEN3_EQ_PHASE23_EXIT_MODE = 1<br>■ Equalization Phase 3 Complete status bit is set in the "Link Status Register 2" |
| 3 to 0 | GEN3_EQ_FB_MODE | All 0 | R/W | R/W | Feedback Mode.<br>■ 0000b: Direction Change<br>■ 0001b: Figure Of Merit<br>■ 0010b – 1111b: Reserved |

### 104.2.222 Port Logic Registers 55 (PRTLGC55)

### Common to Header TYPE00/01

- Gen3 EQ Direction Change Feedback Mode Control Register.



| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | GEN3_EQ_FMDC_MAX_P | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GEN3_EQ_FMDC_MAX_POST_CUSROR_DELTA | | GEN3_EQ_FMDC_MAX_PRE_CUSROR_DELTA | | | | GEN3_EQ_FMDC_N_EVALS | | | | | GEN3_EQ_FMDC_T_MIN_PHASE23 | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 17 to 14 | GEN3_EQ_FMDC_MAX_POST_CUSROR_DELTA | H'0 | R/W | R/W | Convergence Window Aperture for C+1. Post-cursor coefficients maximum delta within the convergence window depth. Allowed range: 0,1,2,..15. |
| 13 to 10 | GEN3_EQ_FMDC_MAX_PRE_CUSROR_DELTA | H'0 | R/W | R/W | Convergence Window Aperture for C-1. Pre-cursor coefficients maximum delta within the convergence window depth. Allowed range: 0,1,2,..15. |
| 9 to 5 | GEN3_EQ_FMDC_N_EVALS | H'2 | R/W | R/W | Convergence Window Depth. Number of consecutive evaluations considered in Phase 2/3 when determining if optimal coefficients have been found. Allowed range: 0,1,2,..16 up to a maximum of CX_GEN3_EQ_COEFQ_DEPTH.<br>When set to 0, EQ Master is performed without sending any requests to the remote partner in Phase 2 for USP and Phase 3 for DSP. Therefore, the remote partner will not change its transmitter coefficients and will move to the next state. |
| 4 to 0 | GEN3_EQ_FMDC_T_MIN_PHASE23 | H'0 | R/W | R/W | Minimum Time (in ms) To Remain in EQ Master Phase. The LTSSM stays in EQ Master phase for at least this amount of time, before starting to check for convergence of the coefficients. Allowed values 0,1,...,24. |

### 104.2.223 Port Logic Registers 56 (PRTLGC56)

### Common to Header TYPE00/01

- Order Rule Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CPL_PASS_P | | | | | | | | NP_PASS_P | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | CPL_PASS_P | H'00 | R/W | R/W | Completion Passing Posted Ordering Rule Control.<br>Determines if CPL can pass halted P queue.<br>■ 0: CPL can not pass P (recommended)<br>■ 1: CPL can pass P |
| 7 to 0 | NP_PASS_P | H'00 | R/W | R/W | Non-Posted Passing Posted Ordering Rule Control.<br>Determines if NP can pass halted P queue.<br>■ 0 : NP can not pass P (recommended).<br>■ 1 : NP can pass P |

### 104.2.224 Port Logic Registers 57 (PRTLGC57)

### Common to Header TYPE00/01

- PIPE Loopback Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIPE_LOOPBACK | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | PIPE_LOOPBACK | 0 | R/W | R/W | PIPE Loopback Enable. |
| 30 to 27 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 26 to 0 | — | All 0 | R | R | Reserved<br>The Writing to this bit is forbidden. |

### 104.2.225 Port Logic Registers 58 (PRTLGC58)

### Common to Header TYPE00/01

- DBI Read-Only Write Enable Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CPLQ_MNG_EN | ARI_DEVICE_NUM | DISABLE_AUTO_LTR_CLR_MSG | SIMPLIFIED_REPLAY_TIMER | UR_CA_MASK_4_TRGT1 | DEFAULT_TARGET | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 7 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 | CPLQ_MNG_EN | 1 | R/W | R/W | This field enables the Completion Queue Management feature. Default value is 1. |
| 5 | ARI_DEVICE_NUM | 0 | R/W | R/W | When ARI is enabled, this field enables use of the device ID. |
| 4 | DISABLE_AUTO_LTR_CLR_MSG | 0 | R/W | R/W | Disable the autonomous generation of LTR clear message in upstream port.<br>■ 0: Allow the autonomous generation of LTR clear message.<br>■ 1: Disable the autonomous generation of LTR clear message. |
| 3 | SIMPLIFIED_REPLAY_TIMER | 1 | R/W | R/W | Enables Simplified Replay Timer (Gen4). For more details, see "Transmit Replay" in the Controller Operations chapter. Simplified Replay Timer Values are:<br>■ A value from 24,000 to 31,000 Symbol Times when Extended Synch is 0b.<br>■ A value from 80,000 to 100,000 Symbol Times when Extended Synch is 1b.<br>Must not be changed while link is in use. |
| 2 | UR_CA_MASK_4_TRGT1 | 0 | R/W | R/W | This field only applies to request TLPs (with UR filtering status) that you have chosen to forward to the application (when you set DEFAULT_TARGET in this register).<br>■ When you set this field to '1', the core suppresses error logging, Error Message generation, and CPL generation (for non-posted requests).<br>■ You should set this if you have set the Default Target port logic register to '1'. |
| 1 | DEFAULT_TARGET | 0 | R/W | R/W | Default target a received IO or MEM request with UR/CA/CRS is sent to by the controller.<br>■ 0: The controller drops all incoming I/O or MEM requests (after corresponding error reporting). A completion with UR status will be generated for non-posted requests.<br>■ 1: The controller forwards all incoming I/O or MEM requests with UR/CA/CRS status to your application. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

### 104.2.226 Port Logic Registers 59 (PRTLGC59)

### Common to Header TYPE00/01

- UpConfigure Multi-lane Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | UPCONFIGURE_SUPPORT | DIRECT_LINK_WIDTH_CHANGE | TARGET_LINK_WIDTH | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | UPCONFIGURE_SUPPORT | 1 | R/W | R/W | Upconfigure Support.<br>The controller sends this value as the Link Upconfigure Capability in TS2 Ordered Sets in Configuration.Complete state. |
| 6 | DIRECT_LINK_WIDTH_CHANGE | 0 | R/W | R/W | Directed Link Width Change.<br>The controller always moves to Configuration state through<br>Recovery state when this bit is set to '1'.<br>■ If the upconfigure_capable variable is '1' and the HWAUTOWDIS bit in EXPCAP4F*i* is '0', the controller starts upconfigure or autonomous width downsizing (to the TARGET_LINK_WIDTH value) in the Configuration state.<br>■ If TARGET_LINK_WIDTH value is 0x0, the controller does not start upconfigure or autonomous width downsizing in the Configuration state.<br>The controller self-clears this field when the controller accepts this request. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 5 to 0 | TARGET_LINK_WIDTH | H'00 | R/W | R/W | Target Link Width.<br>Values correspond to:<br>■ 6'b000000: Core does not start upconfigure or autonomous width downsizing in the Configuration state.<br>■ 6'b000001: x1<br>■ 6'b000010: x2<br>■ 6'b000100: x4<br>■ 6'b001000: x8<br>■ 6'b010000: x16<br>■ 6'b100000: x32 |

### 104.2.227 Port Logic Registers 60 (PRTLGC60)

### Common to Header TYPE00/01

- PHY Interoperability Control Register. This register is reserved for internal use. You should not write to this register and change the default unless specifically instructed by vendor support.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | P2NOBEACON_ENABL | L1_CLK_SEL | L1_NOWAIT_P1 | L1SUB_EXIT_MODE | — | RXSTANDBY_CONTROL | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | P2NOBEACON_ENABLE | 1 | R/W | R/W | P2.NoBeacon Enable bit. |
| 10 | L1_CLK_SEL | 0 | R/W | R/W | L1 Clock control bit.<br>■ 1: Controller does not request aux_clk switch and core_clk gating in L1.<br>■ 0: Controller requests aux_clk switch and core_clk gating in L1. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 9 | L1_NOWAIT_P1 | 1 | R/W | R/W | L1 entry control bit.<br>■ 1: Core does not wait for PHY to acknowledge transition to P1 before entering L1.<br>■ 0: Core waits for the PHY to acknowledge transition to P1 before entering L1.<br>Note: The access attributes of this field are as follows:<br>■ Wire: No access.<br>■ Dbi: R/W (sticky) |
| 8 | L1SUB_EXIT_MODE | 1 | R/W | R/W | L1 Exit Control Using phy_mac_pclkack_n.<br>■ 1: Core exits L1 without waiting for the PHY to assert phy_mac_pclkack_n.<br>■ 0: Core waits for the PHY to assert phy_mac_pclkack_n before exiting L1. |
| 7 | — | 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 to 0 | RXSTANDBY_CONTROL | All 0 | R/W | R/W | Rxstandby Control. Bits 0..5 determine if the controller asserts the RxStandby signal (mac_phy_rxstandby) in the indicated condition. Bit 6 enables the controller to perform the RxStandby/RxStandbyStatus handshake.<br>■ [0]: Rx EIOS and subsequent T TX-IDLE-MIN<br>■ [1]: Rate Change<br>■ [2]: Inactive lane for upconfigure/downconfigure<br>■ [3]: PowerDown=P1orP2<br>■ [4]: RxL0s.Idle<br>■ [5]: EI Infer in L0<br>■ [6]: Execute RxStandby/RxStandbyStatus Handshake |

### 104.2.228 Port Logic Registers 61 (PRTLGC61)

### Common to Header TYPE00/01

- The target completion LUT(TRGT_CPL_LUT) Delete Entry Control register. Using this register you can delete one entry in the TRGT_CPL_LUT. You should only use this register when you know that your application will never send the completion because of an FLR or any other reason.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DELETE_EN | LOOK_UP_ID | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LOOK_UP_ID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | DELETE_EN | 0 | W | W | This is a one-shot bit. A '1' write to this bit triggers the deletion of the target completion LUT entry that is specified in the LOOK_UP_ID field. This is a self-clearing register field. Reading from this register field always returns a '0'. |
| 30 to 0 | LOOK_UP_ID | All 0 | R/W | R/W | This number selects one entry to delete of the TRGT_CPL_LUT. |

### 104.2.229 Port Logic Registers 62 (PRTLGC62)

### Common to Header TYPE00/01

- Link Reset Request Flush Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | AUTO_FLUSH_EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 24 | — | H'FF | R | R | Reserved<br>These bits are always read as 0. The write value should always be H'FF. |
| 23 to 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | AUTO_FLUSH_EN | 1 | R/W | R/W | Enables automatic flushing of pending requests before sending the reset request to the application logic to reset PCIEC. The flushing process is initiated if any of the following events occur:<br>■ Hot reset request. A downstream port (DSP) can "hot reset" an upstream port (USP) by sending two consecutive TS1 ordered sets with the hot reset bit asserted.<br>■ Warm (Soft) reset request. Generated when exiting from D3 to D0.<br>■ Link down reset request. The link has gone down and the controller is requesting a reset.<br>If you disable automatic flushing, you are responsible for resetting PCIEC. |

### 104.2.230 Port Logic Registers 63 (PRTLGC63)

### Common to Header TYPE00/01

- AXI Bridge Slave Error Response Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AMBA_ERROR_RESPONSE_MAP | | | | | | — | — | — | — | — | AMBA_ERROR_RESPONSE_CRS | | | — | |
| Initial value: | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W | R | R/W |

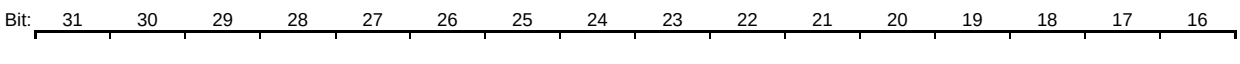
| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 10 | AMBA_ERROR_RESPONSE_MAP | H'27 | R/W | R/W | AXI Slave Response Error Map. Allows you to selectively map the errors received from the PCIe completion (for nonposted requests) to the AXI slave responses, slv_rresp or slv_bresp. The recommended setting is SLVERR. CRS is always mapped to OKAY.<br>■ [0]<br>- 0: UR (unsupported request) -> DECERR<br>- 1: UR (unsupported request) -> SLVERR<br>■ [1]<br>- 0: CRS (configuration retry status) -> DECERR<br>- 1: CRS (configuration retry status) -> SLVERR<br>■ [2]<br>- 0: CA (completer abort) -> DECERR<br>- 1: CA (completer abort) -> SLVERR<br>■ [3]: Reserved<br>■ [4]: Reserved<br>■ [5]:<br>- 0: Completion Timeout -> DECERR<br>- 1: Completion Timeout -> SLVERR<br>The AXI bridge internally drops (processes internally but not passed to your application) a completion that has been marked by the Rx filter as UC or MLF, and does not pass its status directly down to the slave interface. It waits for a timeout and then signals "Completion Timeout" to the slave interface.<br>The controller sets the AXI slave read data bus to 0xFFFF for all error responses. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 9 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 to 3 | AMBA_ERROR_RESPONSE_CRS | All 0 | R/W | R/W | CRS Slave Error Response Mapping. Determines the AXI slave response for CRS completions. For more details see "Error Handling" in the “AXI Bridge Module” chapter of the Databook.<br>AXI:<br>■ 00: OKAY<br>■ 01: OKAY with all FFFF_FFFF data for all CRS completions<br>■ 10: OKAY with FFFF_0001 data for CRS completions to vendor ID read requests, OKAY with FFFF_FFFF data for all other CRS completions<br>■ 11: SLVERR/DECERR (the<br>AXI_ERROR_RESPONSE_MAP field determines the PCIe-to-AXI Slave error response mapping) |
| 2 | AMBA_ERROR_RESPONSE_VENDORID | 0 | R/W | R/W | Vendor ID Non-existent Slave Error Response Mapping.<br>Determines the AXI slave response for errors on reads to non-existent Vendor ID register. For more details see "Error Handling" in the “AXI Bridge Module” chapter of the Databook.<br>■ 1: ERROR<br>AXI:<br>■ 0: OKAY (with FFFF data).<br>■ 1: SLVERR/DECERR (the<br>AXI_ERROR_RESPONSE_MAP field determines the PCIe-to-AXI Slave error response mapping) |
| 1 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 0 | AMBA_ERROR_RESPONSE_GLOBAL | 0 | R/W | R/W | Global Slave Error Response Mapping. Determines the AXI slave response for all error scenarios on non-posted requests. For more details see "Error Handling" in the "AXI Bridge Module” chapter of the Databook.<br>■ 1: ERROR for normal link (data) accesses and look at bit [2] for other scenarios.<br>AXI:<br>■ 0: OKAY (with FFFF data for non-posted requests)<br>■ 1: SLVERR/DECERR (the AXI_ERROR_RESPONSE_MAP field determines the PCIe-to-AXI Slave error response mapping)<br>The error response mapping is not applicable to Nonexistent Vendor ID register reads. |

### 104.2.231 Port Logic Registers 64 (PRTLGC64)

### Common to Header TYPE00/01

- Link Down AXI Bridge Slave Timeout Register



| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | LINK_TIMEOUT_ENABLE_DEFAULT | LINK_TIMEOUT_PERIOD_DEFAULT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | LINK_TIMEOUT_ENABLE_DEFAULT | 1 | R/W | R/W | Disable Flush. You can disable the flush feature by setting this field to "1". |
| 7 to 0 | LINK_TIMEOUT_PERIOD_DEFAULT | H'32 | R/W | R/W | Timeout Value (ms). The timer will timeout and then flush the bridge TX request queues after this amount of time. The timer counts when there are pending outbound AXI slave interface requests and the PCIe TX link is not transmitting any of these requests. The timer is clocked by PIPE clock. |

### 104.2.232 Port Logic Registers 65 (PRTLGC65)

### Common to Header TYPE00/01

- AMBA Ordering Control

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | AX_MSTR_ZEROL_READ_FW | — | — | AX_MSTR_ORDR_P_EVENT_SEL | | — | AX_SNP_EN | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R | R/W | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R | R/W | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

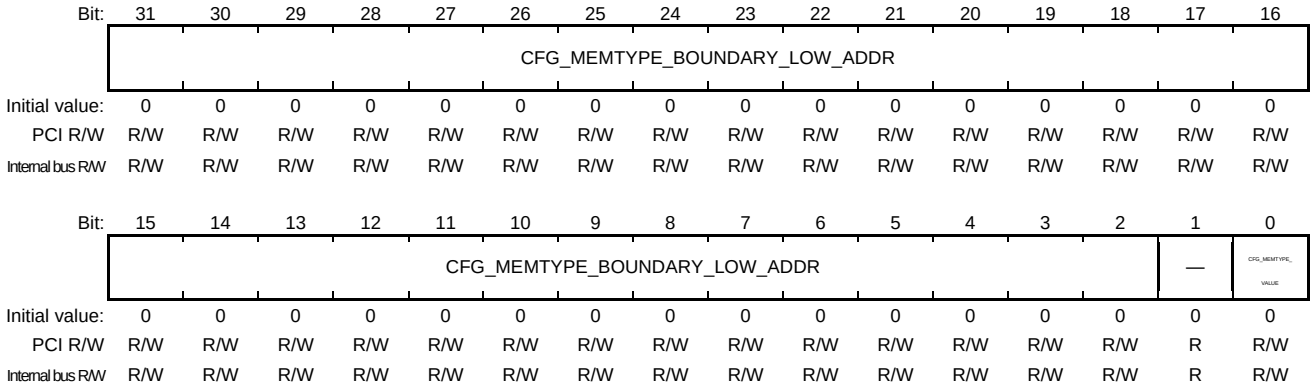
| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 7 | AX_MSTR_ZEROL READ_FW | 0 | R/W | R/W | AXI Master Zero Length Read Forward to the application.<br>The PCIEC AXI bridge is able to terminate in order with the Posted transactions the zero length read, implementing the PCIe express flush semantics of the Posted transactions.<br>■ 0x0: The zero length Read is terminated at the PCIe AXI bridge master<br>■ 0x1: The zero length Read is forward to the internal bus. |
| 6 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 to 3 | AX_MSTR_ORDR_ P_EVENT_SEL | All 0 | R/W | R/W | AXI Master Posted Ordering Event Selector.<br>This field selects how the master interface determines when a P write is completed when enforcing the PCIe ordering rule, "NP must not pass P" at the AXI Master Interface. The AXI protocol does not support ordering between channels.<br>Therefore, NP reads can pass P on your AXI bus fabric. This can result in an ordering violation when the read overtakes a P that is going to the same address. Therefore, the bridge master does not issue any NP requests until all outstanding P writes reach their destination. It does this by waiting for the all of the write responses on the B channel. This can affect the performance of the master read channel. For scenarios where the interconnect serializes the AXI master "AW", "W" and "AR" channels,you can increase the performance by reducing the need to wait until the complete Posted transaction has effectively reached the application slave.<br>■ 00: B'last event: wait for the all of the write responses on the B channel thereby ensuring that the complete Posted transaction has effectively reached the application slave (default).<br>■ 01: AW'last event: wait until the complete Posted transaction has left the AXI address channel at the bridge master.<br>■ 10: W'last event: wait until the complete Posted transaction has left the AXI data channel at the bridge master.<br>■ 11: Reserved<br>Note 2: This setting will not affect:<br>■ MSI interrupt catcher and P data ordering. This is always driven by the B'last event.<br>■ DMA read engine TLP ordering. This is always driven by the B'last event.<br>■ NP write transactions which are always serialized with P write transactions. |
| 2 | — | 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | AX_SNP_EN | 0 | R/W | R/W | AXI Serialize Non-Posted Requests Enable. This field enables the AXI Bridge to serialize same ID Non-Posted Read/Write Requests on the wire. Serialization implies one outstanding same ID NP Read or Write on the wire and used to avoid AXI RAR and WAW hazards at the remote link partner. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 0 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

### 104.2.233 Port Logic Registers 66 (PRTLGC66)

#### Common to Header TYPE00/01

- ACE Cache Coherency Control Register 1

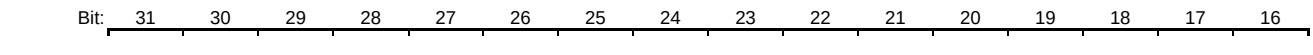


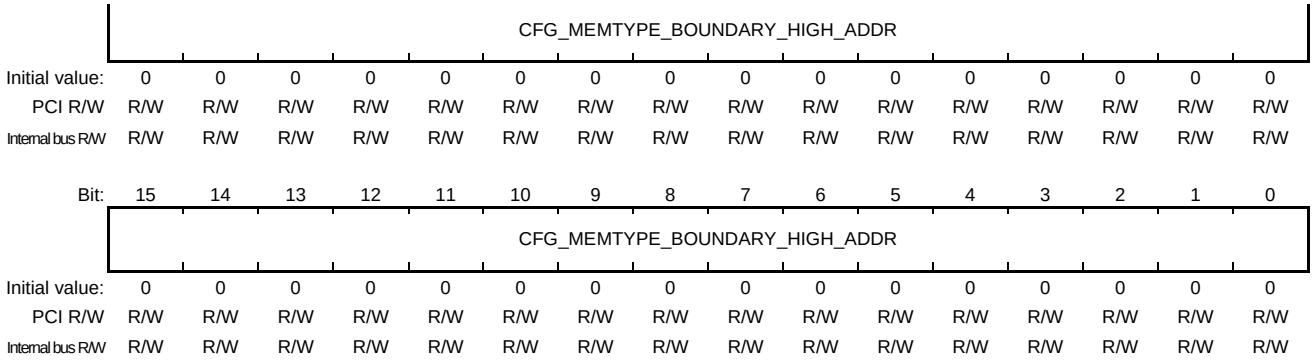

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 2 | CFG_MEMTYPE_BOUNDARY_LOW_ADDR | All 0 | R/W | R/W | Boundary Lower Address For Memory Type. Bits [31:0] of dword-aligned address of the boundary for Memory type.<br>The two lower address LSBs are "00". Addresses up to but not including this value are in the lower address space region; addresses equal or greater than this value are in the upper address space region. |
| 1 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | CFG_MEMTYPE_VALUE | 0 | R/W | R/W | Sets the memory type for the lower and upper parts of the address space:<br>■ 0: lower = Peripheral; upper = Memory<br>■ 1: lower = Memory type; upper = Peripheral |

### 104.2.234 Port Logic Registers 67 (PRTLGC67)

#### Header TYPE01 Only

- ACE Cache Coherency Control Register 2



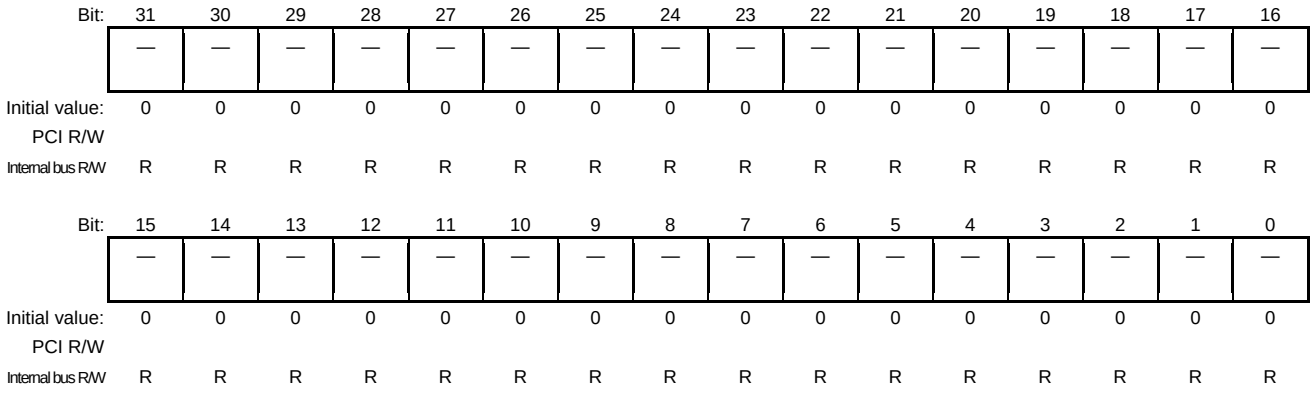



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | CFG_MEMTYPE_BOUNDARY_HIGH_ADDR | All 0 | R/W | R/W | Boundary Upper Address For Memory Type.<br>Bits [63:32] of the 64-bit DWORD-aligned address of the boundary for Memory type. |

### 104.2.235 Port Logic Registers 68 (PRTLGC68)

**Common to Header TYPE00/01**

- ACE Cache Coherency Control Register 3 (V4H/V4M does not support ACE Lite)

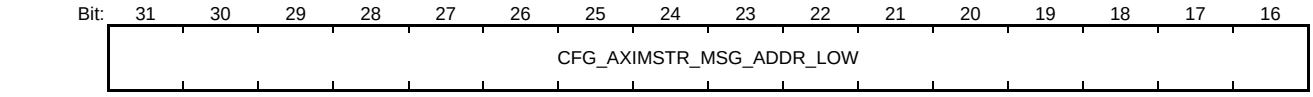



| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | — | All 0 | | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.236 Port Logic Registers 69 (PRTLGC69)

**Common to Header TYPE00/01**

- Lower 20 bits of the programmable AXI address where Messages coming from wire are mapped to.



| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFG_AXIMSTR_MSG_ADDR_LOW | | | | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFG_AXIMSTR_MSG_ADDR_LOW | | | | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | CFG_AXIMSTR_MSG_ADDR_LOW | All 0 | R/W | R/W | Lower 20 bits of the programmable AXI address for Messages. |
| 11 to 0 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

### 104.2.237 Port Logic Registers 70 (PRTLGC70)

### Common to Header TYPE00/01

- Upper 32 bits of the programmable AXI address where Messages coming from wire are mapped to.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFG_AXIMSTR_MSG_ADDR_HIGH | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CFG_AXIMSTR_MSG_ADDR_HIGH | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | CFG_AXIMSTR_MSG_ADDR_HIGH | All 0 | R/W | R/W | Upper 32 bits of the programmable AXI address for Messages. |

### 104.2.238 Port Logic Registers 73 (PRTLGC73)

### Common to Header TYPE00/01

- Interface Timer Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | FORCE_ | INTERFACE_TIMER_SCAL | | INTERFA | INTERFA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | FORCE_PENDING | 0 | R/W | R/W | Writing to this bit forces the value of the pending flags. |
| 3 to 2 | INTERFACE_TIMER_SCALING | All 0 | R/W | R/W | Interface timer scaling. This field can be used to reduce the timer duration for verification purpose. This field should only be programmed when the INTERFACE_TIMER_EN bit I set to 1'b0. |
| 1 | INTERFACE_TIMER_AER_EN | 0 | R/W | R/W | Interface timer AER generation enable. |
| 0 | INTERFACE_TIMER_EN | 0 | R/W | R/W | Interface timer enable. |

### 104.2.239 Port Logic Registers 74 (PRTLGC74)

### Common to Header TYPE00/01

- Interface Timer Target Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTERFACE_TIMER_TARGET | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 15 to 0 | INTERFACE_TIMER_TARGET | H'32 | R/W | R/W | Interface timer target value. This field should only be programmed when the INTERFACE_TIMER_EN bit is set to 1'b0. |

### 104.2.240 Port Logic Registers 75 (PRTLGC75)

### Common to Header TYPE00/01

- Interface Timer Status Register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | SLAVE_RD_ADD_TIMEOUT | SLAVE_WR_DATA_TIMEOUT | SLAVE_WR_ADD_TIMEOUT | MASTERH_RD_DATA_TIMEOUT | MASTERH_WR_RES_TIMEOUT | MASTER_RD_DATA_TIMEOUT | MASTER_WR_RES_TIMEOUT | CLIENT2_INTERFACE_TIMEOUT | CLIENT1_INTERFACE_TIMEOUT | CLIENT0_INTERFACE_TIMEOUT | CPL_INTERFACE_TIMEOUT | MESSAGE_INTERFACE_TIMEOUT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C |
| Internal bus R/W | R | R | R | R | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | SLAVE_RD_ADD_TIMEOUT | 0 | R/W1C | R/W1C | Slave read address channel timeout. |
| 10 | SLAVE_WR_DATA_TIMEOUT | 0 | R/W1C | R/W1C | Slave write data channel timeout. |
| 9 | SLAVE_WR_ADD_TIMEOUT | 0 | R/W1C | R/W1C | Slave write address channel timeout. |
| 8 | MASTERH_RD_DATA_TIMEOUT | 0 | R/W1C | R/W1C | High priority Master read data channel timeout. |
| 7 | MASTERH_WR_RES_TIMEOUT | 0 | R/W1C | R/W1C | High priority Master write response channel timeout. |
| 6 | MASTER_RD_DATA_TIMEOUT | 0 | R/W1C | R/W1C | Master read data channel timeout. |
| 5 | MASTER_WR_RES_TIMEOUT | 0 | R/W1C | R/W1C | Master write response channel timeout. |
| 4 | CLIENT2_INTERFACE_TIMEOUT | 0 | R/W1C | R/W1C | Client2 interface timeout. |
| 3 | CLIENT1_INTERFACE_TIMEOUT | 0 | R/W1C | R/W1C | Client1 interface timeout. |
| 2 | CLIENT0_INTERFACE_TIMEOUT | 0 | R/W1C | R/W1C | Client0 interface timeout. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 1 | CPL_INTERFACE_TIMEOUT | 0 | R/W1C | R/W1C | CPL interface timeout. |
| 0 | MESSAGE_INTERFACE_TIMEOUT | 0 | R/W1C | R/W1C | Message interface timeout. |

### 104.2.241 Port Logic Registers 80 (PRTLGC80)

**Common to Header TYPE00/01**

- Masks for functional safety interrupt events

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | masked_s | masked_s | masked_s | masked_s | masked_s | masked_s |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 6 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 | masked_safety_int[RASDP_CORR_INDEX] | 0 | R/W | R/W | Mask for functional safety interrupt event 5 (RASDP correctable). |
| 4 | masked_safety_int[PCIE_CORR_INDEX] | 0 | R/W | R/W | Mask for functional safety interrupt event 4 (PCIe correctable). |
| 3 | masked_safety_int[PCIE_UNCORR_INDEX] | 0 | R/W | R/W | Mask for functional safety interrupt event 3 (PCIe uncorrectable). |
| 2 | masked_safety_int[IF_TIMER_SE_INDEX] | 0 | R/W | R/W | Mask for functional safety interrupt event 2 (Interface timers). |
| 1 | masked_safety_int[CDM_CHK_SE_INDEX] | 0 | R/W | R/W | Mask for functional safety interrupt event 1 (CDM register checker). |
| 0 | masked_safety_int[RASDP_SE_INDEX] | 0 | R/W | R/W | Mask for functional safety interrupt event 0 (RASDP). |

### 104.2.242 Port Logic Registers 81 (PRTLGC81)

**Common to Header TYPE00/01**

- Status for functional safety interrupt events

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | SAFETY | SAFETY | SAFETY | SAFETY | SAFETY | SAFETY |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 6 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 | SAFETY_INT_STATUS_5 | 0 | R/W1C | R/W1C | Status for functional safety interrupt event 5 (RASDP correctable). |
| 4 | SAFETY_INT_STATUS_4 | 0 | R/W1C | R/W1C | Status for functional safety interrupt event 4 (PCIe correctable). |
| 3 | SAFETY_INT_STATUS_3 | 0 | R/W1C | R/W1C | Status for functional safety interrupt event 3 (PCIe uncorrectable). |
| 2 | SAFETY_INT_STATUS_2 | 0 | R/W1C | R/W1C | Status for functional safety interrupt event 2 (Interface timers). |
| 1 | SAFETY_INT_STATUS_1 | 0 | R/W1C | R/W1C | Status for functional safety interrupt event 1 (CDM register checker). |
| 0 | SAFETY_INT_STATUS_0 | 0 | R/W1C | R/W1C | Status for functional safety interrupt event 0 (RASDP). |

### 104.2.243 Port Logic Registers 82 (PRTLGC82)

### Common to Header TYPE00/01

- CDM Register Checking Control and Status Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | CHK_RE | CHK_RE | CHK_RE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W1C | R/W1C | R/W1C |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W1C | R/W1C | R/W1C |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | CHK_RE | CHK_RE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 19 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 18 | CHK_REG_COMPLETE | 0 | R/W1C | R/W1C | The system has completed a checking cycle. |
| 17 | CHK_REG_LOGIC_ERROR | 0 | R/W1C | R/W1C | The system has detected an error in its own checking logic. |
| 16 | CHK_REG_COMPARISON_ERROR | 0 | R/W1C | R/W1C | The system has detected that there is a bit error in the CDM Register Data. |
| 15 to 2 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | CHK_REG_CONTINUOUS | 0 | R/W | R/W | Set Continuous Checking Sequence. |
| 0 | CHK_REG_START | 0 | R/W | R/W | Begins a checking sequence. |

### 104.2.244 Port Logic Registers 83 (PRTLGC83)

**Common to Header TYPE00/01**

- CDM Register Checking First and Last address to check

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHK_REG_END_ADDR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHK_REG_START_ADDR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 16 | CHK_REG_END_ADDR | H'BFF | R/W | R/W | The last address that is checked by the system. |
| 15 to 0 | CHK_REG_START_ADDR | All 0 | R/W | R/W | The first address that is checked by the system. |

### 104.2.245 Port Logic Registers 84 (PRTLGC84)

**Common to Header TYPE00/01**

- CDM Register Checking Error Address

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHK_REG_ERR_ADDR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CHK_REG_ERR_ADDR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 0 | CHK_REG_ERR_ADDR | All 0 | R | R | The address at which an error has been detected. Valid only when the CDM Register Checker Comparison Error bit is set in the status register. |

### 104.2.246 Port Logic Registers 85 (PRTLGC85)

**Common to Header TYPE00/01**

- CDM Register Checking error PF Numbers

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | CHK_REG_PF_ERR_NUMBER | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 5 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 to 0 | CHK_REG_PF_ERR_NUMBER | All 0 | R | R | The PF number at which the error was detected. Valid only when the CDM Register Checker Comparison Error bit is set in the status register. |

### 104.2.247 Port Logic Registers 86 (PRTLGC86)

**Common to Header TYPE00/01**

- LTR Latency Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NO_SNOOP_L | _ | _ | NO_SNOOP_LATENCY_SCALE | | | NO_SNOOP_LATENCY_VALUE | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SNOOP_LAT | _ | _ | SNOOP_LATENCY_SCALE | | | SNOOP_LATENCY_VALUE | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | NO_SNOOP_LATENCY_REQUIRE | 0 | R | R/W | No Snoop Latency Requirement. |
| 30 to 29 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 26 | NO_SNOOP_LATENCY_SCALE | All 0 | R | R/W | No Snoop Latency Scale. |
| 25 to 16 | NO_SNOOP_LATENCY_VALUE | All 0 | R | R/W | No Snoop Latency Value. |
| 15 | SNOOP_LATENCY_REQUIRE | 0 | R | R/W | Snoop Latency Requirement. |
| 14 to 13 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 to 10 | SNOOP_LATENCY_SCALE | All 0 | R | R/W | Snoop Latency Scale. |
| 9 to 0 | SNOOP_LATENCY_VALUE | All 0 | R | R/W | Snoop Latency Value. |

### 104.2.248 Port Logic Registers 87 (PRTLGC87)

### Common to Header TYPE00/01

- Auxiliary Clock Frequency Control Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ | _ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | _ | _ | _ | _ | _ | _ | AUX_CLK_FREQ | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| PCI R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 10 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 to 0 | AUX_CLK_FREQ | H'00A | R/W | R/W | The aux_clk frequency in MHz. This value is used to provide a 1 us reference for counting time during low-power states with aux_clk when the PHY has removed the pipe_clk.<br>Frequencies lower than 1 MHz are possible but with a loss of accuracy in the time counted. If the actual frequency (f) of aux_clk does not exactly match the programmed frequency (f_prog), then there is an error in the time counted by the controller that can be expressed in percentage as: err% = (f_prog/f-1)*100. For example if f=2.5 MHz and f_prog=3 MHz, then err% =(3/2.5-1)*100 =20%, meaning that the time counted by the controller on aux_clk will be 20% greater than the time in us programmed in the corresponding time register (for example T_POWER_ON). |

### 104.2.249 Port Logic Registers 88 (PRTLGC88)

### Common to Header TYPE00/01

- L1 Substates Timing Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | L1SUB_ | L1SUB_T_PCLKACK | | L1SUB_T_L1_2 | | | | L1SUB_T_POWER_OFF | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| PCI R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | L1SUB_CLOCK_SWITCH | 0 | R/W | R/W | If this bit is set to 1'b1 the reference clock will be running regardless of the CLKREQ# setting. If this bit is set to 1'b0 the reference clock may be gated off when CLKREQ# is deasserted. If the bit is set to 1'b1 the controller will delay the switching of aux_clk to the slow platform clock until it detects that the link partner has de-asserted CLKREQ#. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 7 to 6 | L1SUB_T_PCLKACK | H'3 | R/W | R/W | Max delay (in 1us units) between a MAC request to remove the clock on mac_phy_pclkreq_n and a PHY response on phy_mac_pclkack_n. If the PHY does not respond within this time the request is aborted. Range is 0..3 |
| 5 to 2 | L1SUB_T_L1_2 | H'4 | R/W | R/W | Duration (in 1us units) of L1.2. Range is 0.15.<br>Note: The timeout value can vary by 50%. |
| 1 to 0 | L1SUB_T_POWER_OFF | H'2 | R/W | R/W | Duration (in 1us units) of L1.2.Entry. Range is 0.3.<br>Note: The timeout value can vary by 50%. |

### 104.2.250 Port Logic Registers 91 (PRTLGC91)

### Common to Header TYPE00/01

- Gen4 Lane Margining 1 Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | MARGINING_MAX_VOLTAGE_OFFSET | | | | | | – | MARGINING_NUM_VOLTAGE_STEPS | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | MARGINING_MAX_TIMING_OFFSET | | | | | | – | – | MARGINING_NUM_TIMING_STEPS | | | | | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| PCI R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 30 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29 to 24 | MARGINING_MAX_VOLTAGE_OFFSET | H'5 | R/W | R/W | M(MaxVoltageOffset) for Lane Margining at the Receiver. This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |
| 23 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 22 to 16 | MARGINING_NUM_VOLTAGE_STEPS | H'20 | R/W | R/W | M(NumVoltageSteps) for Lane Margining at the Receiver. This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |
| 15 to 14 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 13 to 8 | MARGINING_MAX_TIMING_OFFSET | H'14 | R/W | R/W | M(MaxTimingOffset) for Lane Margining at the Receiver. This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |
| 7 to 6 | — | All 0 | R | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | MARGINING_NUM_TIMING_STEPS | H'9 | R/W | R/W | M(NumTimingSteps) for Lane Margining at the Receiver. This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |

### 104.2.251 Port Logic Registers 92 (PRTLGC92)

### Common to Header TYPE00/01

- Gen4 Lane Margining 2 Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MARGINING_ | – | – | MARGINI | MARGINI | MARGINI | MARGINI | MARGINI | – | – | – | MARGINING_MAXLANES | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| PCI R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

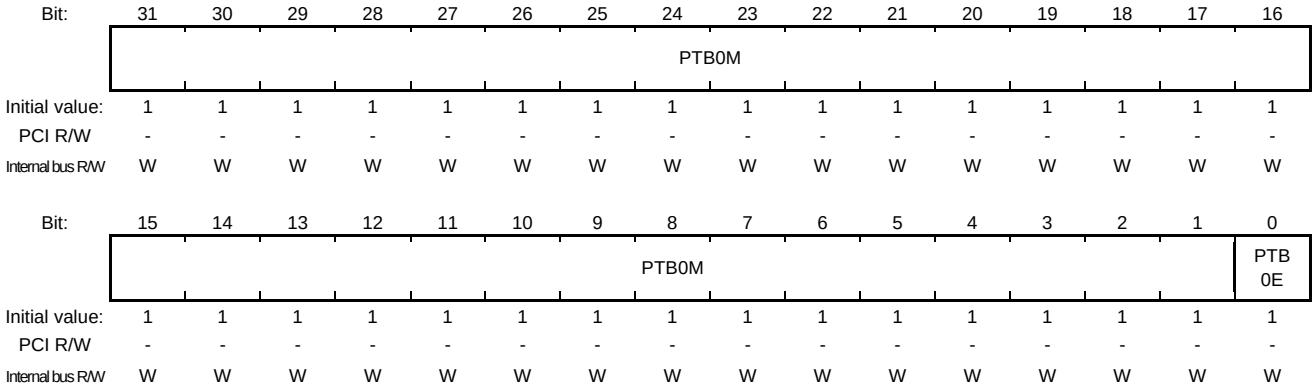
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | MARGINING_SAMPLE_RATE_TIMING | | | | | | – | – | MARGINING_SAMPLE_RATE_VOLTAGE | | | | | |
| Initial value: | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PCI R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Internal bus R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 | MARGINING_RATE_MODE | 0 | R/W | R/W | Margining Rate Mode.<br>■ 0: Base Spec compliant mode: Lane Margining at Receiver is available at Gen4 data rate only.<br>■ 1: Special PHY Support mode: Lane Margining at Receiver is available at Gen4 data rate and at Gen1,Gen2 and Gen3 data rate for debug. |
| 30 to 29 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 28 | MARGINING_IND_ERROR_SAMPLER | 0 | R/W | R/W | M(IndErrorSampler) for Lane Margining at the Receiver. This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 27 | MARGINING_SAMPLE_REPORTING_METHOD | 1 | R/W | R/W | M(SampleReportingMethod) for Lane Margining at the Receiver. This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |
| 26 | MARGINING_IND_LEFT_RIGHT_TIMING | 1 | R/W | R/W | M(IndLeftRightTiming) for Lane Margining at the Receiver.<br>This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |
| 25 | MARGINING_IND_UP_DOWN_VOLTAGE | 0 | R/W | R/W | M(IndUpDownVoltage) for Lane Margining at the Receiver.<br>This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |
| 24 | MARGINING_VOLTAGE_SUPPORTED | 0 | R/W | R/W | M(VoltageSupported) for Lane Margining at the Receiver.<br>This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |
| 23 to 21 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 20 to 16 | MARGINING_MAXLANES | H'4 | R/W | R/W | M(MaxLanes) for Lane Margining at the Receiver. This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0. |
| 15 to 14 | — | All 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 13 to 8 | MARGINING_SAMPLE_RATE_TIMING | H'3F | R/W | R/W | M(SamplingRateTiming) for Lane Margining at the Receiver.<br>This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter , see the PCI Express Base Specification 4.0.Note: This value is not used when MARGINING_IND_ERROR_SAMPLER is 0b. The M(SamplingRateTiming) is fixed to 63 internally. |
| 7 to 6 | — | 0 | R | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5 to 0 | MARGINING_SAMPLE_RATE_VOLTAGE | All 0 | R/W | R/W | M(SamplingRateVoltage) for Lane Margining at the Receiver.<br>This parameter is defined in 9 Electrical Sub-Block of PCI Express Base Specification 4.0. For a description of this standard PCIe parameter, see the PCI Express Base Specification 4.0.Note: This value is not used when MARGINING_IND_ERROR_SAMPLER is 0b. The M(SamplingRateVoltage) is fixed to 63 internally. |

### 104.2.252 BAR0 MASK Register function *i* (BAR0MASKF*i*) (*i* = 0, 1)

**Header TYPE00 only**




| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | PTB0M | H'7FFF FFFF | — | W | BAR0 MASK |
| 0 | PTB0E | 1 | — | W | BAR0 MASK Enabled |

### 104.2.253 BAR1 MASK Register function *i* (BAR1MASKF*i*) (*i* = 0, 1)

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB1M | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB1M | | | | | | | | | | | | | | | PTB 1E |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | PTB1M | H'7FFF FFFF | — | W | BAR1 MASK |
| 0 | PTB1E | 0 | — | W | BAR1 MASK Enabled |

### 104.2.254 BAR2 MASK Register function *i* (BAR2MASKF*i*) (*i* = 0, 1)

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB2M | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB2M | | | | | | | | | | | | | | | PTB 2E |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | PTB2M | H'7FFF FFFF | — | W | BAR2 MASK |
| 0 | PTB2E | 1 | — | W | BAR2 MASK Enabled |

### 104.2.255 BAR3 MASK Register function *i* (BAR3MASKF*i*) (*i* = 0, 1)

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB3M | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB3M | | | | | | | | | | | | | | | PTB 3E |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | PTB3M | H'7FFF FFFF | — | W | BAR3 MASK<br>*a: PTB3M >> H'1 |
| 0 | PTB3E | 0 | — | W | BAR3 MASK Enabled |

### 104.2.256 BAR4 MASK Register function *i* (BAR4MASKF*i*) (*i* = 0, 1)

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB4M | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB4M | | | | | | | | | | | | | | | PTB 4E |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | PTB4M | H'7FFF FFFF | — | W | BAR4 MASK |
| 0 | PTB4E | 1 | — | W | BAR4 MASK Enabled |

### 104.2.257 BAR5 MASK Register function *i* (BAR5MASKF*i*) (*i* = 0, 1)

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB5M | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTB5M | | | | | | | | | | | | | | | PTB 5E |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | PTB5M | H'7FFF FFFF | — | W | BAR5 MASK |
| 0 | PTB5E | 1 | — | W | BAR5 MASK Enabled |

### 104.2.258 EXP ROM MASK Register function *i* (EROMMASKF*i*) (*i* = 0, 1)

**Header TYPE00 only**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ROMMSK | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ROMMSK | | | | | | | | | | | | | | | ROM MSKE |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PCI R/W | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Internal bus R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | PCI R/W | Internal Bus R/W | Description |
|---|---|---|---|---|---|
| 31 to 1 | ROMMSK | All 1 | — | W | Expansion ROM MASK |
| 0 | ROMMSKE | 1 | — | W | Expansion ROM Bar Mask Register Enabled |

### 104.2.259 ATU Capabilities Registers [0+*i**18] (ATUCAP[0+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Region Control 1 Register for outbound

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | – | CTRL_1_FUNC_NUM | | | – | – | – | – |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | – | – | INCR EAS | – | – | ATTR | | TD | TC | | | TYPE | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 22 to 20 | CTRL_1_FUNC_NUM | All 0 | R/W | Function Number.<br>■ When the address of an outbound TLP is matched to this region and the FUNC_BYPASS field in the "ATU Capabilities Registers [1+i*18]" is '0', then the function number used in generating the function part of the requester ID (RID) field of the TLP is taken from this 5-bit register.<br>■This field is swapped before AXI decomposition occurs so that the correct "Max_Read_Request_Size" and "Max_Payload_Size" values are used. |
| 19 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 | INCREASE_REGION_SIZE | 0 | R/W | Increase the maximum ATU Region size. When set, the maximum ATU Region size is determined by CX_ATU_MAX_REGION_SIZE. When clear, the maximum ATU Region size is 4 GB (default). |
| 12 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 10 to 9 | ATTR | All 0 | R/W | When the address of an outbound TLP is matched to this region, then the ATTR field of the TLP is changed to the value in this register. |
| 8 | TD | 0 | R/W | This is a reserved field. Do not use. |
| 7 to 5 | TC | All 0 | R/W | When the address of an outbound TLP is matched to this region, then the TC field of the TLP is changed to the value in this register. |
| 4 to 0 | TYPE | All 0 | R/W | When the address of an outbound TLP is matched to this region, then the TYPE field of the TLP is changed to the value in this register. |

### 104.2.260 ATU Capabilities Registers [1+*i**18] (ATUCAP[1+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Region Control 2 Register for outbound

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REGION_EN | _ | INVERT_ | CFG_SHI | DMA_BY | _ | _ | _ | HEADER | INHIBIT_ | TLP_HEA | SNP | FUNC_B | MSB2BITS_TAG | | TAG_SU |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAG | | | | | | | | MSG_CODE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

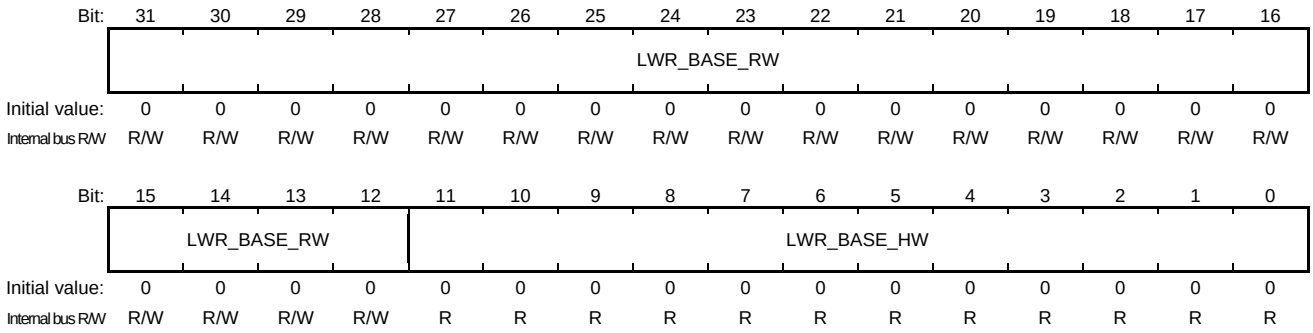
| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | REGION_EN | 0 | R/W | Region Enable. This bit must be set to '1' for address translation to take place. |
| 30 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29 | INVERT_MODE | 0 | R/W | Invert Mode. When set the address matching region is inverted. Therefore, an address match occurs when the untranslated address is in the region outside the defined range (Base Address to Limit Address). |
| 28 | CFG_SHIFT_MODE | 0 | R/W | CFG Shift Mode.<br>The ATU uses bits [27:12] of the untranslated address (on the AXI slave interface address) to form the BDF number of the outgoing CFG TLP.<br>This supports the Enhanced Configuration Address Mapping (ECAM) mechanism (Section 7.2.2 of the PCI Express Base 4.0 Specification, revision 1.0) by allowing all outgoing I/O and MEM TLPs (that have been translated to CFG) to be mapped from memory space into any 256 MB region of the PCIe configuration space. |
| 27 | DMA_BYPASS | 0 | R/W | DMA Bypass Mode. Allows request TLPs which are initiated by the DMA controller to pass through the ATU untranslated. |
| 26 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 | HEADER_SUBSTITUTE_EN | 0 | R/W | Header Substitute Enable.<br>When enabled and region address is matched, the ATU fully substitutes bytes 8-11 (for 3 DWORD header) or bytes 12-15 (for 4 DWORD header) of the outbound TLP header with the contents of the LWR_TARGET_RW field in ATU Capabilities Registers [5+i*18].<br>■ 1: LWR_TARGET_RW in the ATU Capabilities Registers [5+i*18] is used to fill bytes 8-to-11 (for 3 DWORD header) or bytes 12-to-15 (for 4 DWORD header) of the translated TLP header.<br>■ 0: LWR_TARGET_RW in the ATU Capabilities Registers [5+i*18] forms the new address of the translated region. |
| 22 | INHIBIT_PAYLOAD | 0 | R/W | Inhibit TLP Payload Data for TLP's in Matched Region;<br>assign ATU region to be TLP without data. When enabled and region address is matched, the ATU marks all TLPs as having no payload data by forcing the TLP header Fmt[1] bit =0.<br>■ 1: Fmt[1] =0 so that only TLP type without data is sent.<br>For example, a Msg instead of MsgD will be sent.<br>■ 0: Fmt[1] =0/1 so that TLPs with or without data can be sent. |
| 21 | TLP_HEADER_FIELDS_BYPASS | 0 | R/W | TLP Header Fields Translation Bypass. In this mode header fields of the translated TLP is taken from your application transmit interface. |
| 20 | SNP | 0 | R/W | Serialize Non-Posted Requests. In this mode, when the AXI Bridge is populated, same AXI ID Non-Posted Read/Write Requests are transmitted on the wire if there are no other same ID Non-Posted Requests outstanding. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 19 | FUNC_BYPASS | 0 | R/W | Function Number Translation Bypass. In this mode, the function number of the translated TLP is taken you're your application transmit interface and not from the CTRL_1_FUNC_NUM field of the "ATU Capabilities Registers [0+i*18]". |
| 18 to 17 | MSB2BITS_TAG | All 0 | R/W | TAG.<br>Two most significant bits of the substituted TAG field in the outgoing TLP header when TAG_SUBSTITUTE_EN is set. |
| 16 | TAG_SUBSTITUTE_EN | 0 | R/W | TAG Substitute Enable.<br>When enabled and region address is matched, the ATU substitutes the TAG field of the outbound TLP header with the contents of the TAG field in this register. The expected usage scenario is translation from AXI MWr to Vendor Defined Msg/MsgD. |
| 15 to 8 | TAG | All 0 | R/W | TAG.<br>The substituted TAG field (byte 6) in the outgoing TLP header when TAG_SUBSTITUTE_EN is set. |
| 7 to 0 | MSG_CODE | All 0 | R/W | MSG TLPs (Message Code). When the address of an outbound TLP is matched to this region, and the translated TLP TYPE field is Msg or MsgD; then the message field of the TLP is changed to the value in this register. |

### 104.2.261 ATU Capabilities Registers [2+*i**18] (ATUCAP[2+*i**18]) (*i* = 0 to 15)

### Common to Header TYPE00/01

- ATU Lower Base Address Register for outbound

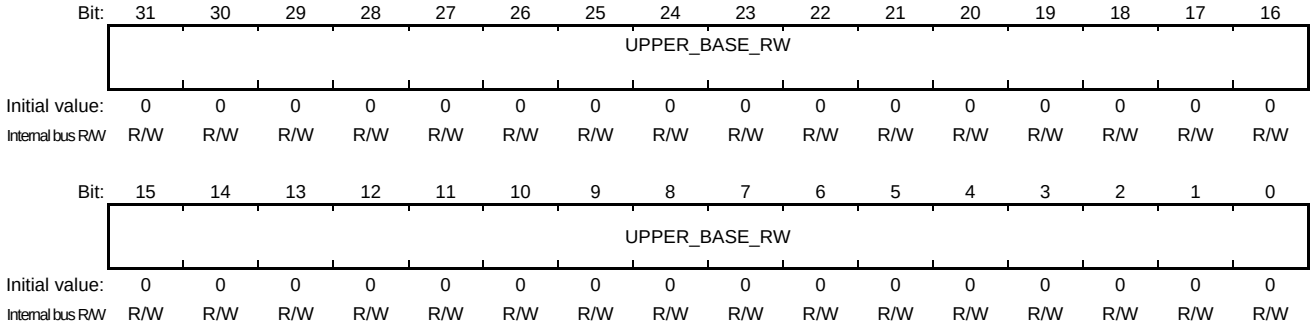



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 12 | LWR_BASE_RW | All 0 | R/W | Forms bits [31:12] of the start address of the address region to be translated. |
| 11 to 0 | LWR_BASE_HW | All 0 | R | Forms bits [11:0] of the start address of the address region to be translated. The start address must be aligned to a 4 kB boundary, so these bits are always 0. A write to this location is ignored by the PCIe controller. |

### 104.2.262 ATU Capabilities Registers [3+*i**18] (ATUCAP[3+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Upper Base Address Register for outbound

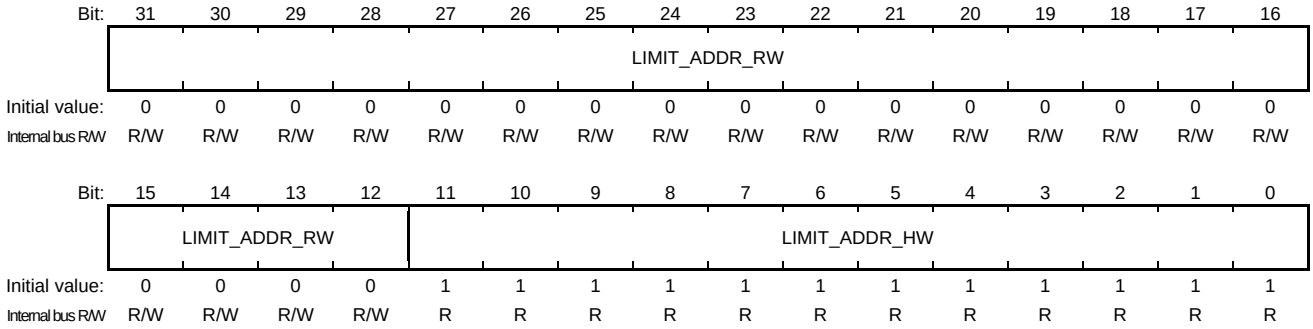



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | UPPER_BASE_RW | All 0 | R/W | Forms bits [63:32] of the start (and end) address of the address region to be translated. In systems with a 32-bit address space, this register is not used and therefore writing to this register has no effect. |

### 104.2.263 ATU Capabilities Registers [4+*i**18] (ATUCAP[4+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Limit Address Register for outbound

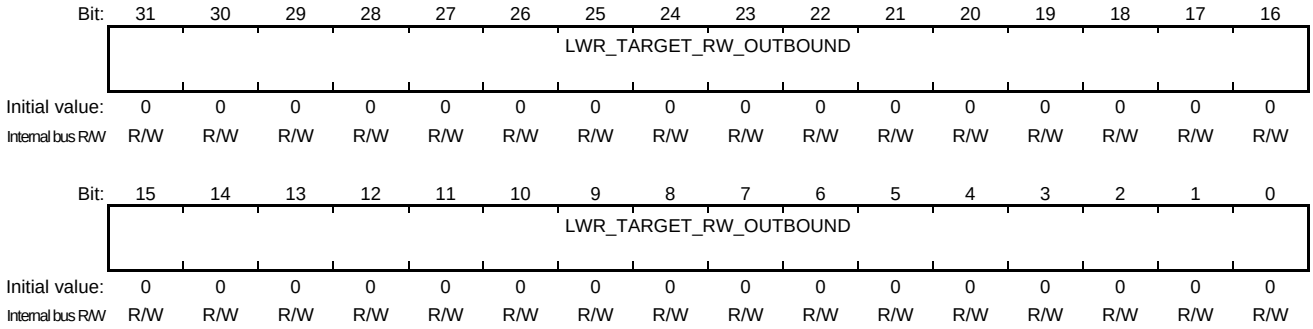



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 12 | LIMIT_ADDR_RW | All 0 | R/W | Forms upper bits of the end address of the address region to be translated. |
| 11 to 0 | LIMIT_ADDR_HW | All H'FFF | R | Forms lower bits of the end address of the address region to be translated. The end address must be aligned to a 4 kB boundary, so these bits are always all ones. A write to this location is ignored by the PCIEC. |

### 104.2.264 ATU Capabilities Registers [5+*i**18] (ATUCAP[5+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Lower Target Address Register for outbound

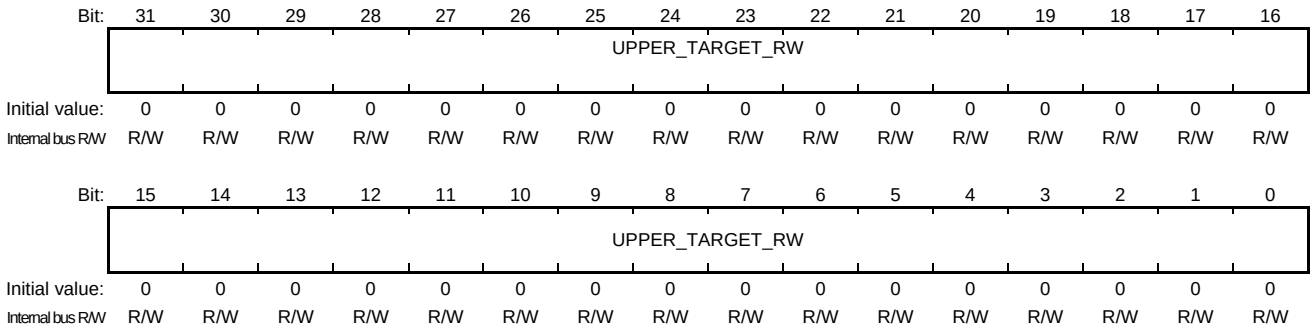



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LWR_TARGET_RW_OUTBOUND | All 0 | R/W | When HEADER_SUBSTITUTE_EN in ATU Capabilities Registers [1+i*18] is '0 (normal operation):<br>■ LWR_TARGET_RW[31:12] forms MSB's of the Lower Target part of the new address of the translated region;<br>■ LWR_TARGET_RW[11:0] are not used. (The start address must be aligned to a 4 kB boundary, so the lower bits of the start address of the new address of the translated region (bits 11:0) are always '0').<br>When HEADER_SUBSTITUTE_EN in ATU Capabilities Registers [1+i*18] is '1':<br>■ LWR_TARGET_RW[31:0] forms bytes 8-11 (for 3 DWORD header) or bytes 12-15 (for 4 DWORD header) of the outbound TLP header. Usage scenarios include the transmission of Vendor Defined Messages where the controller determines the content of bytes 12 to 15 of the TLP header. |

### 104.2.265 ATU Capabilities Registers [6+*i**18] (ATUCAP[6+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Upper Target Address Register for outbound




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | UPPER_TARGET_RW | All 0 | R/W | Forms bits [63:32] of the start address (Upper Target part) of the new address of the translated region. |

### 104.2.266 ATU Capabilities Registers [7+*i**18] (ATUCAP[7+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Region Control 3 Register for outbound (setting for virtual function)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.267 ATU Capabilities Registers [8+*i**18] (ATUCAP[8+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Upper Limit Address Register for outbound

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UPPR_LIMIT_ADDR_HW | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

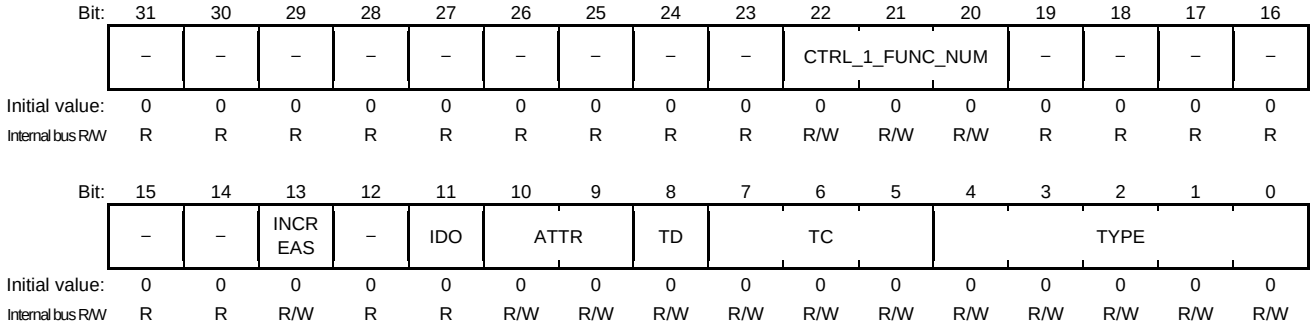
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UPPR_LIMIT_ADDR_HW | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | UPPR_LIMIT_ADDR_HW | All 0 | R | Forms MSB's of the Upper Limit part of the region "end address" to be translated. Only applies to 64-bit systems and when the INCREASE_REGION_SIZE field in ATU Capabilities Registers [0+i*18] is '1'. These bits are always '0'. |

### 104.2.268 ATU Capabilities Registers [9+*i**18] (ATUCAP[9+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Region Control 1 Register for inbound




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 22 to 20 | CTRL_1_FUNC_NUM | All 0 | R/W | Function Number.<br>■ MEM-I/O: When the Address and BAR matching logic in the controller indicate that a MEM-I/O transaction matches a BAR in the function corresponding to this value, then address translation proceeds. This check is only performed if the "Function Number Match Enable" bit of the "ATU Capabilities Registers [10+i*18]" is set.<br>■ CFG0/CFG1: When the destination function number as specified in the routing ID of the TLP header matches the function, then address translation proceeds. This check is only performed if the "Function Number Match Enable" bit of the "ATU Capabilities Registers [10+i*18]" is set. |
| 19 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 | INCREASE_REGION_SIZE | 0 | R/W | Increase the maximum ATU Region size. When set, the maximum ATU Region size is determined by CX_ATU_MAX_REGION_SIZE. When clear, the maximum ATU Region size is 4 GB (default). |
| 12 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | IDO | 0 | R/W | When the IDO field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "ATTR Match Enable" bit of the "ATU Capabilities Registers [10+i*18]" is set.<br>Note: This function is not supported. |
| 10 to 9 | ATTR | All 0 | R/W | When the ATTR field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "ATTR Match Enable" bit of the "ATU Capabilities Registers [10+i*18]" is set. |
| 8 | TD | 0 | R/W | When the TD field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "TD Match Enable" bit of the "ATU Capabilities Registers [10+i*18]" is set. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 7 to 5 | TC | All 0 | R/W | When the TC field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "TC Match Enable" bit of the "ATU Capabilities Registers [10+i*18]" is set. |
| 4 to 0 | TYPE | All 0 | R/W | When the TYPE field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful). |

### 104.2.269 ATU Capabilities Registers [10+*i**18] (ATUCAP[10+*i**18]) (*i* = 0 to 15)

### Common to Header TYPE00/01

- ATU Region Control 2 Register for inbound

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REGION_EN | MATCH_MODE | INVERT_MODE | CFG_SHIFT_MODE | FUZZY_TYPE_MATCH_CODE | – | RESPONSE_CODE | | SINGLE_ADDR_LOC_TRANS_EN | – | MSG_CODE_MATCH_EN | – | FUNC_NUM_MATCH_EN | – | – | ATTR_MATCH_EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R | R/W | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TD_MATCH_EN | TC_MATCH_EN | MSG_TYPE_MATCH_MODE | – | – | BAR_NUM | | | MSG_CODE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | REGION_EN | 0 | R/W | Region Enable. This bit must be set to '1' for address translation to take place. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 30 | MATCH_MODE | 0 | R/W | Match Mode. Determines Inbound matching mode for TLPs.<br>The mode depends on the type of TLP that is received as follows:<br>For MEM-I/O TLPs, this field is interpreted as follows:<br>■ 0: Address Match Mode. The ATU operates using addresses as in the outbound direction. The Region Base and Limit Registers must be setup.<br>■ 1:BAR Match Mode. BAR matching is used. The "BAR Number" field is relevant. Not used for RC.<br>For CFG0 TLPs, this field is interpreted as follows:<br>■ 0: Routing ID Match Mode. The ATU interprets the Routing ID (Bytes 8 to 11 of TLP header) as an address.<br>This corresponds to the upper 16 bits of the address in MEM-I/O transactions. The Routing ID of the TLP must be within the base and limit of the ATU region for matching to proceed.<br>■ 1: Accept Mode. The ATU accepts all CFG0 transactions as address matches. The routing ID in the CFG0 TLP is ignored. This is useful as all received CFG0 TLPs should be processed regardless of the Bus number.<br>For MSG/MSGD TLPs, this field is interpreted as follows:<br>■ 0: Address Match Mode. The ATU treats the third DWORD and fourth DWORD of the inbound MSG/MSGD TLP as an address and it is matched against the Region Base and Limit Registers.<br>■ 1: Vendor ID Match Mode. This mode is relevant for ID routed Vendor Defined Messages. The ATU ignores the Routing ID (Bus, Device, Function) in bits [31:16] of the third DWORD of the TLP header, but matches against the Vendor ID in bits [15:0] of the third DWORD of the TLP header. Bits [15:0] of the Region Upper Base register should be programmed with the required Vendor ID. The lower Base and Limit Register should be programmed to translate TLPs based on vendor specific information in the fourth DWORD of the TLP header.<br>-If SINGLE_ADDRESS_LOCATION_TRANSLATE_EN = 1<br>AND MSG_TYPE_MATCH_MODE = 1, then Match Mode is ignored. |
| 29 | INVERT_MODE | 0 | R/W | Invert Mode. When set the address matching region is inverted. Therefore, an address match occurs when the untranslated address is in the region outside the defined range (Base Address to Limit Address). |
| 28 | CFG_SHIFT_MODE | 0 | R/W | CFG Shift Mode. This is useful for CFG transactions where the PCIe configuration mechanism maps bits [27:12] of the address to the bus/device and function number. This allows a CFG configuration space to be located in any 256MB window of your application memory space using a 28-bit effective address. Shifts bits [31:16] of the untranslated address to form bits [27:12] of the translated address. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 27 | FUZZY_TYPE_MATCH_CODE | 0 | R/W | Fuzzy Type Match Mode. When enabled, the ATU relaxes the matching of the TLP TYPE field against the expected TYPE field so that<br>■ CfgRd0 and CfgRd1 TLPs are seen as identical. Similarly with CfgWr0 and CfgWr1.<br>■ MWr, MRd and MRdLk TLPs are seen as identical<br>■ The Routing field of Msg/MsgD TLPs is ignored<br>■ FetchAdd, Swap and CAS are seen as identical.<br>For example, CFG0 in the TYPE field in the "ATU Capabilities Registers [9+i*18]" matches against an inbound CfgRd0, CfgRd1, CfgWr0 or CfgWr1 TLP. |
| 26 | — | All 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 25 to 24 | RESPONSE_CODE | All 0 | R/W | Response Code. Defines the type of response to give for accesses matching this region. This overrides the normal RADM filter response. Note that this feature is not available for any region where Single Address Location Translate is enabled. |
| 23 | SINGLE_ADDR_LOC_TRANS_EN | 0 | R/W | Single Address Location Translate Enable.<br>When enabled, Rx TLPs can be translated to a single address location as determined by the target address register of the ATU region. The main usage scenario is translation of Messages (such as Vendor Defined or ATS Messages) to MWr TLPs. |
| 22 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

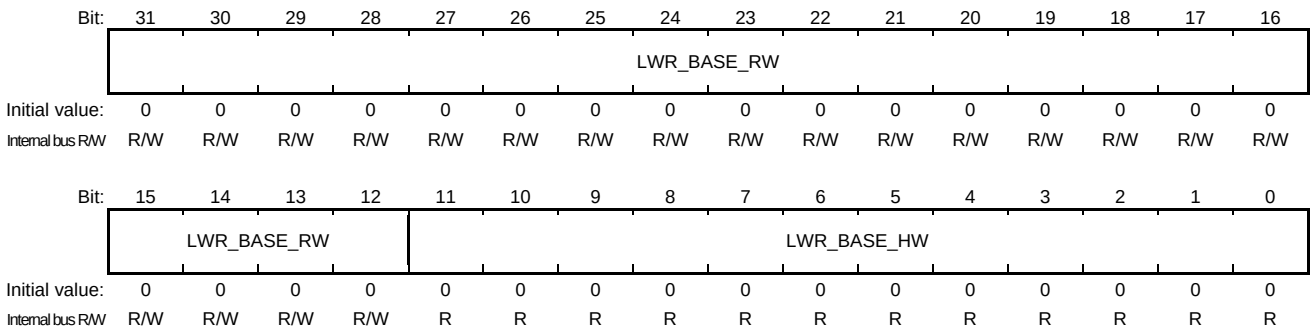
| | | | | |
|---|---|---|---|---|
| 21 | MSG_CODE_MATCH_EN | 0 | R/W | Message Code Match Enable (Msg TLPS). Ensures that a successful message Code TLP field comparison match (see Message Code field of the "ATU Capabilities Registers [10+i*18]") occurs (in MSG transactions) for address translation to proceed. |
| 20 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 19 | FUNC_NUM_MATCH_EN | 0 | R/W | Function Number Match Enable. Ensures that a successful Function Number TLP field comparison match (see Function Number field of the "ATU Capabilities Registers [9+i*18]") occurs (in MEM-I/O and CFG0/CFG1 transactions) for address translation to proceed. |
| 18 to 17 | — | All 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 16 | ATTR_MATCH_EN | 0 | R/W | ATTR Match Enable. Ensures that a successful ATTR TLP field comparison match (see ATTR field of the "ATU Capabilities Registers [9+i*18]") occurs for address translation to proceed. |
| 15 | TD_MATCH_EN | 0 | R/W | TD Match Enable. Ensures that a successful TD TLP field comparison match (see TD field of the "ATU Capabilities Registers [9+i*18]") occurs for address translation to proceed. |

| | | | | |
|---|---|---|---|---|
| 14 | TC_MATCH_EN | 0 | R/W | TC Match Enable. Ensures that a successful TC TLP field comparison match (see TC field of the "ATU Capabilities Registers [9+i*18]") occurs for address translation to proceed. |
| 13 | MSG_TYPE_MATCH_ MODE | 0 | R/W | Message Type Match Mode.<br>When enabled, and if single address location translate enable is set, then inbound TLPs of type MSG/MSGd which match the type field of the ATU Capabilities Registers [9+i*18] register (=>TYPE[4:3]=2'b10) will be translated. Message type match mode overrides any value of MATCH_MODE field in this register. Usage scenarios for this are translation of VDM. |
| 12 to 11 | — | All 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 10 to 8 | BAR_NUM | 0 | R/W | BAR Number. When the BAR number of an inbound MEM or IO TLP " that is matched by the normal internal BAR address matching mechanism " is the same as this field, address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "Match Mode" bit of the "ATU Capabilities Registers [10+i*18]" is set.<br>■ 000b - BAR0<br>■ 001b - BAR1<br>■ 010b - BAR2<br>■ 011b - BAR3<br>■ 100b - BAR4<br>■ 101b - BAR5<br>■ 110b - ROM<br>■ 111b - reserved<br>■ IO translation would require either 00100b or 00101b in the inbound TLP TYPE; the BAR Number set in the range 000b - 101b and that BAR configured as an IO BAR. |
| 7 to 0 | MSG_CODE | 0 | R/W | MSG TLPs: (Message Code). When the TYPE field of an inbound Msg/MsgD TLP is matched to this value, then address translation proceeds (when all other enabled fieldmatches are successful). This check is only performed if the "Message Code Match Enable" bit of the "ATU Capabilities Registers [10+i*18]" is set. |

### 104.2.270 ATU Capabilities Registers [11+*i**18] (ATUCAP[11+*i**18]) (*i* = 0 to 15)

### Common to Header TYPE00/01

- ATU Lower Base Address Register for inbound



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 12 | LWR_BASE_RW | All 0 | R/W | Forms bits [31:12] of the start address of the address region to be translated. |
| 11 to 0 | LWR_BASE_HW | All 0 | R | Forms bits [11:0] of the start address of the address region to be translated. The start address must be aligned to a 4 kB boundary, so these bits are always 0. A write to this location is ignored by the PCIEC. |

### 104.2.271 ATU Capabilities Registers [12+*i**18] (ATUCAP[12+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Upper Base Address Register for inbound

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UPPER_BASE_RW | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UPPER_BASE_RW | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | UPPER_BASE_RW | All 0 | R/W | Forms bits [63:32] of the start (and end) address of the address region to be translated. |

### 104.2.272 ATU Capabilities Registers [13+*i**18] (ATUCAP[13+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Limit Address Register for inbound

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LIMIT_ADDR_RW | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

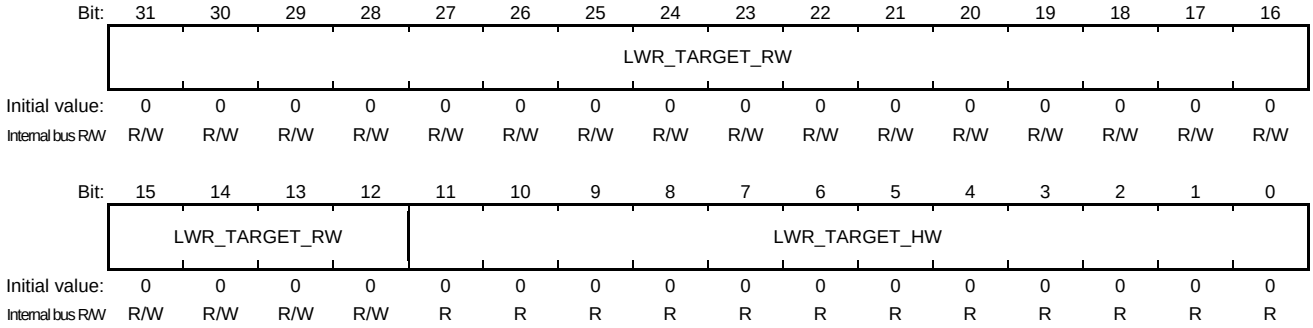
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LIMIT_ADDR_RW | | | | LIMIT_ADDR_HW | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 12 | LIMIT_ADDR_RW | All 0 | R/W | Forms upper bits of the end address of the address region to be translated. |
| 11 to 0 | LIMIT_ADDR_HW | H'FFF | R | Forms lower bits of the end address of the address region to be translated. The end address must be aligned to a 4 kB boundary, so these bits are always all ones. A write to this location is ignored by the PCIEC. |

### 104.2.273 ATU Capabilities Registers [14+*i**18] (ATUCAP[14+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Lower Target Address Register for inbound

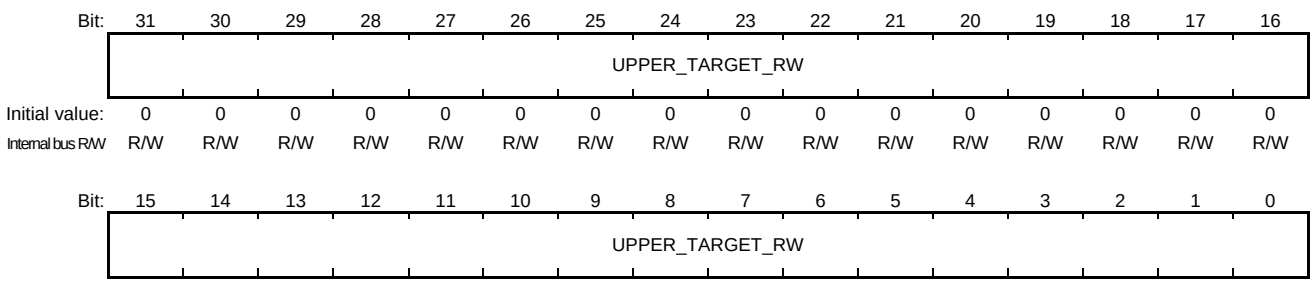



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 12 | LWR_TARGET_RW | All 0 | R/W | Forms MSB's of the Lower Target part of the new address of the translated region. These bits are always '0'. |
| 11 to 0 | LWR_TARGET_HW | All 0 | R | Forms the LSB's of the Lower Target part of the new address of the translated region. The start address must be aligned to a 4 kB boundary (in address match mode); and to the Bar size boundary (in BAR match mode) so that these bits are always '0'. If the BAR is smaller than the ATU region size, then the ATU target address must align to the ATU region size; otherwise it must align to the BAR size.<br>A write to this location is ignored by the PCIEC. |

### 104.2.274 ATU Capabilities Registers [15+*i**18] (ATUCAP[15+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Upper Target Address Register for inbound



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | UPPER_TARGET_RW | All 0 | R/W | Forms bits [63:32] of the start address (Upper Target part) of the new address of the translated region. In systems with a 32-bit address space, this register is not used and therefore writing to this register has no effect. |

### 104.2.275 ATU Capabilities Registers [16+*i**18] (ATUCAP[16+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Region Control 3 Register for inbound (Setting for VF)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.276 ATU Capabilities Registers [17+*i**18] (ATUCAP[17+*i**18]) (*i* = 0 to 15)

**Common to Header TYPE00/01**

- ATU Upper Limit Address Register for inbound

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UPPR_LIMIT_ADDR_HW | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | UPPR_LIMIT_ADDR_HW | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | UPPR_LIMIT_ADDR_HW | 0 | R | Forms MSB's of the Upper Limit part of the region "end address" to be translated. Only applies to 64-bit systems and when the INCREASE_REGION_SIZE field in ATU Capabilities Registers 9 is '1'. These bits are always '0'. |

### 104.2.277 DMA Capabilities Registers 0 (DMACAP0)

**Common to Header TYPE00/01**

- DMA Arbitration Scheme for TRGT1 Interface
- This register is used to control traffic priorities among various sources that are delivered to internal bus where H’0 represents the highest priority. Concurrent traffic from channels with same priority are sorted according to Round-Robin arbitration rules. The arbitration priority defaults to Non-DMA requests (highest), Write Channel MRd, Read Channel MRd, Read Channel MWr.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | RDBUFF_TRGT_WEIGHT | | | RD_CTRL_TRGT_WEIGHT | | | WR_CTRL_TRGT_WEIGHT | | | RTRGT1_WEIGHT | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

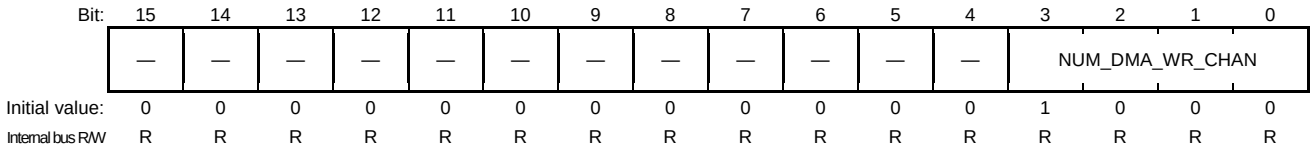
| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 9 | RDBUFF_TRGT_WEIGHT | H’3 | R/W | DMA Read Channel MWr Requests. |
| 8 to 6 | RD_CTRL_TRGT_WEIGHT | H’2 | R/W | DMA Read Channel MRd Requests. For LL element/descriptor access. |
| 5 to 3 | WR_CTRL_TRGT_WEIGHT | H’1 | R/W | DMA Write Channel MRd Requests. For DMA data requests and LL element/descriptor access. |
| 2 to 0 | RTRGT1_WEIGHT | All 0 | R/W | Non-DMA Rx Requests. |

### 104.2.278 DMA Capabilities Registers 1 (DMACAP1)

**Common to Header TYPE00/01**

- DMA Number of Channels Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | DIS_C2W_CACHE_RD | DIS_C2W_CACHE_WR | — | — | — | — | NUM_DMA_RD_CHAN | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R | R | R | R |

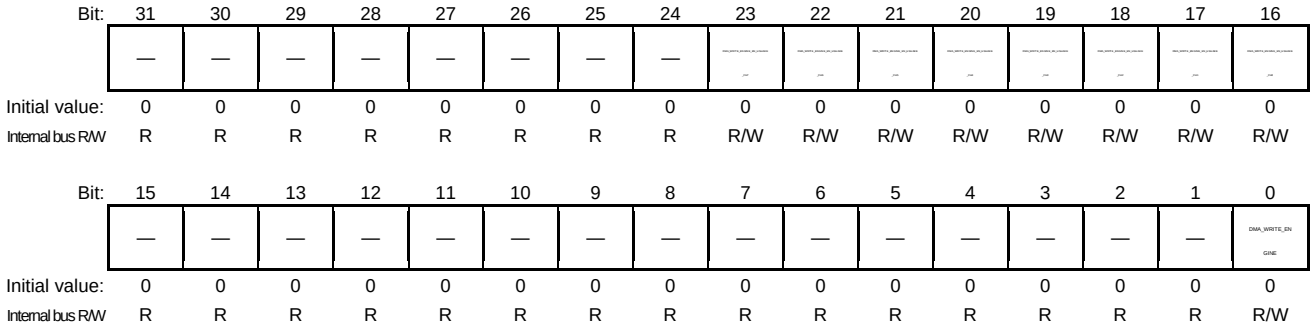



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 | DIS_C2W_CACHE_RD | 0 | R/W | Disable DMA Read Channels "completion to memory write" context cache pre-fetch function.<br>Note: For internal debugging only. |
| 24 | DIS_C2W_CACHE_WR | 0 | R/W | Disable DMA Write Channels "completion to memory write" context cache pre-fetch function.<br>Note: For internal debugging only. |
| 23 to 20 | — | All 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 19 to 16 | NUM_DMA_RD_CHAN | H'8 | R | Number of Read Channels. You can read this register to determine the number of read channels the DMA controller has been configured to support. |
| 15 to 4 | — | All 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 3 to 0 | NUM_DMA_WR_CHAN | H'8 | R | Number of Write Channels. You can read this register to determine the number of write channels the DMA controller has been configured to support. |

### 104.2.279 DMA Capabilities Registers 2 (DMACAP2)

### Common to Header TYPE00/01

- DMA Write Engine Enable Register



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 | DMA_WRITE_ENGINE_EN_HSHAKE_CH7 | 0 | R/W | Enable Handshake for DMA Write Engine Channel 7. |
| 22 | DMA_WRITE_ENGINE_EN_HSHAKE_CH6 | 0 | R/W | Enable Handshake for DMA Write Engine Channel 6. |
| 21 | DMA_WRITE_ENGINE_EN_HSHAKE_CH5 | 0 | R/W | Enable Handshake for DMA Write Engine Channel 5. |
| 20 | DMA_WRITE_ENGINE_EN_HSHAKE_CH4 | 0 | R/W | Enable Handshake for DMA Write Engine Channel 4. |
| 19 | DMA_WRITE_ENGINE_EN_HSHAKE_CH3 | 0 | R/W | Enable Handshake for DMA Write Engine Channel 3. |
| 18 | DMA_WRITE_ENGINE_EN_HSHAKE_CH2 | 0 | R/W | Enable Handshake for DMA Write Engine Channel 2. |
| 17 | DMA_WRITE_ENGINE_EN_HSHAKE_CH1 | 0 | R/W | Enable Handshake for DMA Write Engine Channel 1. |
| 16 | DMA_WRITE_ENGINE_EN_HSHAKE_CH0 | 0 | R/W | Enable Handshake for DMA Write Engine Channel 0. |
| 15 to 1 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 0 | DMA_WRITE_ENGINE | 0 | R/W | DMA Write Engine Enable.<br>■ 1: Enable<br>■ 0: Disable (Soft Reset)<br>For normal operation, you must initially set this bit to "1", before any other software setup actions. You do not need to toggle or rewrite to this bit during normal operation. You should set this bit to "0" when you want to "Soft Reset" the DMA controller write logic. There are three possible reasons for resetting the DMA controller write logic:<br>■ The "Abort Interrupt Status" bit is set in the "DMA Capabilities Registers 10" (DMACAP10), and any of the bits is in the " DMA Capabilities Registers 13" (DMACAP13) are set. Resetting the DMA controller write logic re-initializes the control logic ensuring that the next DMA write transfer is executed successfully.<br>■ You have executed the procedure outlined in "Stop Bit", after which, the "Abort Interrupt Status" bit is set and the Channel Status field (CS) of the DMA write "DMA Capabilities Registers [37+i*16]" (DMACAP[37+i*16]) is set to "Stopped." Resetting the DMA controller write logic reinitializes the control logic ensuring that the next DMA write transfer is executed successfully.<br>■ During software development, when you incorrectly program the DMA write engine.<br>To "Soft Reset" the DMA controller write logic, you must:<br>■ De-assert the DMA write engine enable bit.<br>■ Wait for the DMA to complete any in-progress TLP transfer, by waiting until a read on the DMA write engine enable bit returns a "0".<br>■ Assert the DMA write engine enable bit.<br>This "Soft Reset" does not clear the DMA configuration registers. The DMA write transfer does not start until you write to the " DMA Capabilities Registers 3" (DMACAP3). |

### 104.2.280 DMA Capabilities Registers 3 (DMACAP3)

### Common to Header TYPE00/01

- DMA Write Doorbell Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WR_STOP | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

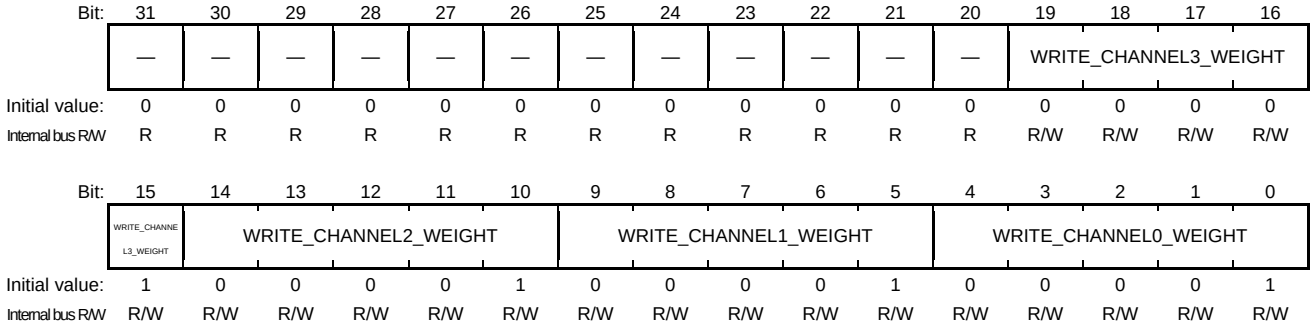
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | WR_DOORBELL_NUM | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | WR_STOP | 0 | R/W | Stop. Set in conjunction with the Doorbell Number field. The DMA write channel stops issuing requests, sets the channel status to "Stopped", and asserts the "Abort" interrupt if it is enabled. Before setting the Stop bit, you must read the channel Status field (CS) of the " DMA Capabilities Registers 59" (DMACAP59) to ensure that the write channel is "Running" (transferring data). For more information, see "Stopping the DMA Transfer (Software Stop)." |
| 30 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | WR_DOORBELL_NUM | All 0 | R/W | Doorbell Number. You must write the channel number to this register to start the DMA write transfer for that channel. The DMA detects a write to this register field even if the value of this field does not change. You do not need to toggle or write any other value to this register to start a new transfer. The range of this field is 0x0 to 0x7, and 0x0 corresponds to channel 0. Also note that a write to this field triggers the controller to exit L1 substates. |

### 104.2.281 DMA Capabilities Registers 4 (DMACAP4)

### Common to Header TYPE00/01

- DMA Write Engine Channel Arbitration Weight Low Register




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 15 | WRITE_CHANNEL3_WEIGHT | H'1 | R/W | Channel 3 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel write request. A value of '0' means that one TLP is issued before moving to the next channel. |
| 14 to 10 | WRITE_CHANNEL2_WEIGHT | H'1 | R/W | Channel 2 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel write request. A value of '0' means that one TLP is issued before moving to the next channel. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 9 to 5 | WRITE_CHANNEL1_WEIGHT | H'1 | R/W | Channel 1 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel write request. A value of '0' means that one TLP is issued before moving to the next channel. |
| 4 to 0 | WRITE_CHANNEL0_WEIGHT | H'1 | R/W | Channel 0 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel write request. A value of '0' means that one TLP is issued before moving to the next channel. |

### 104.2.282 DMA Capabilities Registers 5 (DMACAP5)

### Common to Header TYPE00/01

- DMA Write Engine Channel Arbitration Weight High Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | WRITE_CHANNEL7_WEIGHT | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WRITE_CHAN | WRITE_CHANNEL6_WEIGHT | | | | | WRITE_CHANNEL5_WEIGHT | | | | | WRITE_CHANNEL4_WEIGHT | | | | |
| Initial value: | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 15 | WRITE_CHANNEL7_WEIGHT | H'1 | R/W | Channel 7 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel write request. A value of '0' means that one TLP is issued before moving to the next channel. |
| 14 to 10 | WRITE_CHANNEL6_WEIGHT | H'1 | R/W | Channel 6 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel write request. A value of '0' means that one TLP is issued before moving to the next channel. |
| 9 to 5 | WRITE_CHANNEL5_WEIGHT | H'1 | R/W | Channel 5 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel write request. A value of '0' means that one TLP is issued before moving to the next channel. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 4 to 0 | WRITE_CHANNEL4_WEIGHT | H'1 | R/W | Channel 4 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel write request. A value of '0' means that one TLP is issued before moving to the next channel. |

### 104.2.283 DMA Capabilities Registers 6 (DMACAP6)

**Common to Header TYPE00/01**

- DMA Read Engine Enable Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DMA_READ_ENGINE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 | DMA_READ_ENGINE_EN_HSHAKE_CH7 | 0 | R/W | Enable Handshake for DMA Read Engine Channel 7. |
| 22 | DMA_READ_ENGINE_EN_HSHAKE_CH6 | 0 | R/W | Enable Handshake for DMA Read Engine Channel 6. |
| 21 | DMA_READ_ENGINE_EN_HSHAKE_CH5 | 0 | R/W | Enable Handshake for DMA Read Engine Channel 5. |
| 20 | DMA_READ_ENGINE_EN_HSHAKE_CH4 | 0 | R/W | Enable Handshake for DMA Read Engine Channel 4. |
| 19 | DMA_READ_ENGINE_EN_HSHAKE_CH3 | 0 | R/W | Enable Handshake for DMA Read Engine Channel 3. |
| 18 | DMA_READ_ENGINE_EN_HSHAKE_CH2 | 0 | R/W | Enable Handshake for DMA Read Engine Channel 2. |
| 17 | DMA_READ_ENGINE_EN_HSHAKE_CH1 | 0 | R/W | Enable Handshake for DMA Read Engine Channel 1. |
| 16 | DMA_READ_ENGINE_EN_HSHAKE_CH0 | 0 | R/W | Enable Handshake for DMA Read Engine Channel 0. |
| 15 to 1 | — | All 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 0 | DMA_READ_ENGINE | 0 | R/W | DMA Read Engine Enable.<br>■ 1: Enable<br>■ 0: Disable (Soft Reset)<br>For normal operation, you must initially set this bit to "1", before any other software setup actions. You do not need to toggle or rewrite to this bit during normal operation. You should set this field to "0" when you want to "Soft Reset" the DMA controller read logic. There are three possible reasons for resetting the DMA controller read logic:<br>■ The "Abort Interrupt Status" bit is set (in the "DMA Capabilities Registers 23" (DMACAP23), and any of the bits in the ""DMA Capabilities Registers 26" (DMACAP26) is set. Resetting the DMA controller read logic re-initializes the control logic, ensuring that the next DMA read transfer is executed successfully.<br>■ You have executed the procedure outlined in "Stop Bit", after which, the "Abort Interrupt Status" bit is set and the channel Status field (CS) of the DMA read "DMA Capabilities Registers [45+i*16]" (DMACAP[45+i*16]) is set to "Stopped". Resetting the DMA controller read logic reinitializes the control logic ensuring that the next DMA read transfer is executed successfully.<br>■ During software development, when you incorrectly program the DMA read engine. To "Soft Reset" the DMA controller read logic, you must:<br>■ De-assert the DMA read engine enable bit.<br>■ Wait for the DMA to complete any in-progress TLP transfer, by waiting until a read on the DMA read engine enable bit returns a "0".<br>■ Assert the DMA read engine enable bit. This "Soft Reset" does not clear the DMA configuration registers. The DMA read transfer does not start until you write to the " DMA Capabilities Registers 7" (DMACAP7). |

### 104.2.284 DMA Capabilities Registers 7 (DMACAP7)

### Common to Header TYPE00/01

- DMA Read Doorbell Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RD_STOP | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

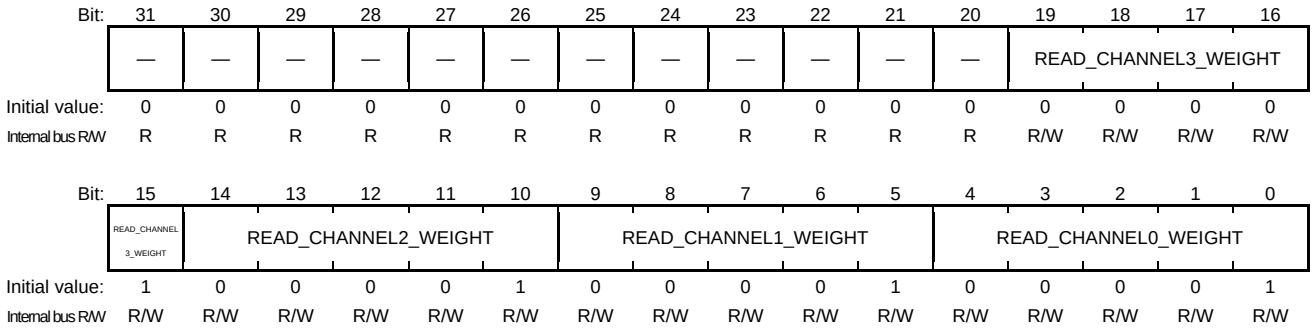
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | RD_DOORBELL_NUM | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | RD_STOP | 0 | R/W | Stop. Set in conjunction with the Doorbell Number field. The DMA read channel stops issuing requests, sets the channel status to "Stopped", and asserts the "Abort" interrupt if it is enabled. Before setting the Stop bit, you must read the channel Status field (CS) of the "DMA Capabilities Registers 68" (DMACAP68) to ensure that the read channel is "Running" (transferring data).<br>For more information, see "Stopping the DMA Transfer (Software Stop)". |
| 30 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | RD_DOORBELL_NUM | All 0 | R/W | Doorbell Number. You must write 0x0 to this register to start the DMA read transfer for that channel. The DMA detects a write to this register field even if the value of this field does not change. The range of this field is 0x0 to 0x7, and 0x0 corresponds to channel 0. Also note that a write to this field triggers the controller to exit L1 substates. |

### 104.2.285 DMA Capabilities Registers 8 (DMACAP8)

### Common to Header TYPE00/01

- Description: DMA Read Engine Channel Arbitration Weight Low Register




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 15 | READ_CHANNEL3_WEIGHT | H'1 | R/W | Channel 3 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel read request. |
| 14 to 10 | READ_CHANNEL2_WEIGHT | H'1 | R/W | Channel 2 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel read request. |
| 9 to 5 | READ_CHANNEL1_WEIGHT | H'1 | R/W | Channel 1 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel read request. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 4 to 0 | READ_CHANNEL0_WEIGHT | H'1 | R/W | Channel 0 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel read request. |

### 104.2.286 DMA Capabilities Registers 9 (DMACAP9)

**Common to Header TYPE00/01**

- DMA Read Engine Channel Arbitration Weight High Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | READ_CHANNEL7_WEIGHT | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

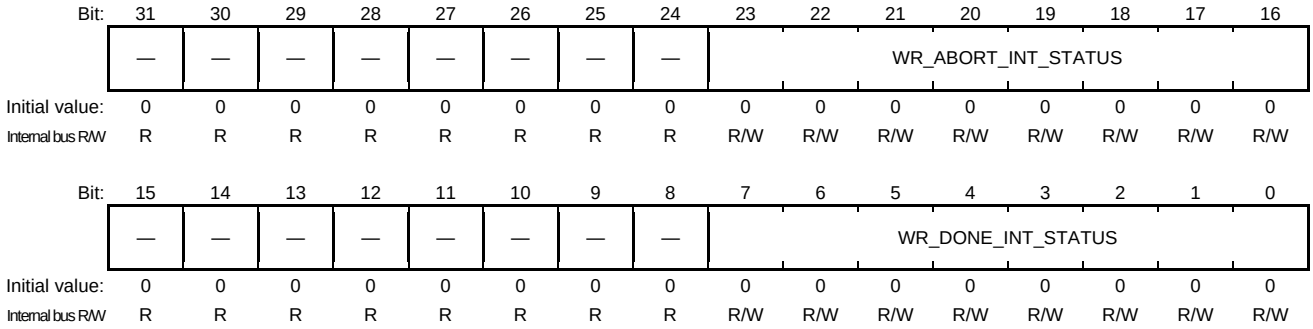
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | READ_CHANNEL7_WEIGHT | READ_CHANNEL6_WEIGHT | | | | | READ_CHANNEL5_WEIGHT | | | | | READ_CHANNEL4_WEIGHT | | | | |
| Initial value: | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 15 | READ_CHANNEL7_WEIGHT | H'1 | R/W | Channel 7 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel read request. |
| 14 to 10 | READ_CHANNEL6_WEIGHT | H'1 | R/W | Channel 6 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel read request. |
| 9 to 5 | READ_CHANNEL5_WEIGHT | H'1 | R/W | Channel 5 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel read request. |
| 4 to 0 | READ_CHANNEL4_WEIGHT | H'1 | R/W | Channel 4 Weight. The weight is initialized by software before ringing the doorbell. The value is used by the channel weighted round robin arbiter to select the next channel read request. |

### 104.2.287 DMA Capabilities Registers 10 (DMACAP10)

**Common to Header TYPE00/01**

- DMA Write Interrupt Status Register




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | WR_ABORT_INT_STATUS | All 0 | R/W | Abort Interrupt Status. The DMA write channel has detected an error, or you manually stopped the transfer as described in "Error Handling Assistance by Remote Software". Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA write interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the DMA write interrupt Clear register to clear this interrupt bit.<br>Note: You can write to this register to emulate interrupt generation, during software or hardware testing. A write to the address triggers an interrupt, but the DMA does not set the Done or Abort bits in this register. |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | WR_DONE_INT_STATUS | All 0 | R/W | Done Interrupt Status. The DMA write channel has successfully completed the DMA transfer. For more details, see "Interrupts and Error Handling". Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA write interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the DMA write interrupt Clear register to clear this interrupt bit.<br>Note: You can write to this register to emulate interrupt generation, during software or hardware testing. A write to the address triggers an interrupt, but the DMA does not set the Done or Abort bits in this register. |

### 104.2.288 DMA Capabilities Registers 11 (DMACAP11)

**Common to Header TYPE00/01**

- DMA Write Interrupt Mask Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | WR_ABORT_INT_MASK | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | WR_DONE_INT_MASK | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | WR_ABORT_INT_MASK | H'FF | R/W | Abort Interrupt Mask. Prevents the Abort interrupt status field in the DMA write interrupt status register from asserting the intreq_pci*m*_dma_p output. Each bit corresponds to a DMA channel. (*m* = 0, 1, 2, 3)<br>Bit [0] corresponds to channel 0. |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | WR_DONE_INT_MASK | H'FF | R/W | Done Interrupt Mask. Prevents the Done interrupt status field in the DMA write interrupt status register from asserting the intreq_pci*m*_dma_p output. Each bit corresponds to a DMA channel. (*m* = 0, 1, 2, 3)<br>Bit [0] corresponds to channel 0. |

### 104.2.289 DMA Capabilities Registers 12 (DMACAP12)

**Common to Header TYPE00/01**

- DMA Write Interrupt Clear Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | WR_ABORT_INT_CLEAR | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | WR_DONE_INT_CLEAR | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

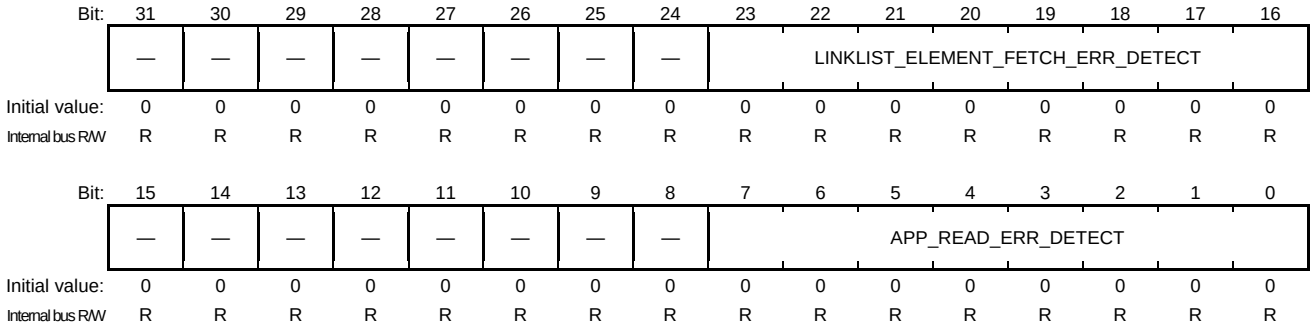
| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 23 to 16 | WR_ABORT_INT_CLEAR | All 0 | W | Abort Interrupt Clear. You must write a 1'b1 to clear the corresponding bit in the Abort interrupt status field of the DMA write interrupt status register. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>Note: Reading from this self-clearing register field always returns a "0". |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | WR_DONE_INT_CLEAR | All 0 | W | Done Interrupt Clear. You must write a 1'b1 to clear the corresponding bit in the Done interrupt status field of the DMA write interrupt status register. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>Note: Reading from this self-clearing register field always returns a "0". |

### 104.2.290 DMA Capabilities Registers 13 (DMACAP13)

### Common to Header TYPE00/01

- DMA Write Error Status Register




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | LINKLIST_ELEMENT_FETCH_ERR_DETECT | All 0 | R | Linked List Element Fetch Error Detected. The DMA write channel has received an error response from the AXI bus while reading a linked list element from local memory. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA write interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the Abort interrupt field of the "DMA Capabilities Registers 12" (DMACAP12) to clear this error bit. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | APP_READ_ERR_DETECT | All 0 | R | Application Read Error Detected. The DMA write channel has received an error response from the AXI bus while reading data from it. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA write interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the Abort interrupt field of the "DMA Capabilities Registers 12" (DMACAP12) to clear this error bit. |

### 104.2.291 DMA Capabilities Registers 14 (DMACAP14)

**Common to Header TYPE00/01**

- DMA Write Done IMWr Address Low Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_WRITE_DONE_LOW_REG | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_WRITE_DONE_LOW_REG | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_WRITE_DONE_LOW_REG | All 0 | R/W | The DMA uses this field to generate bits [31:0] of the address field for the Done IMWr TLP. Bits [1:0] must be "00" as this address must be DWORD aligned. |

### 104.2.292 DMA Capabilities Registers 15 (DMACAP15)

**Common to Header TYPE00/01**

- DMA Write Done IMWr Interrupt Address High Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_WRITE_DONE_HIGH_REG | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_WRITE_DONE_HIGH_REG | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_WRITE_DONE_HIGH_REG | All 0 | R/W | The DMA uses this field to generate bits [63:32] of the address field for the Done IMWr TLP. |

### 104.2.293 DMA Capabilities Registers 16 (DMACAP16)

**Common to Header TYPE00/01**

- DMA Write Abort IMWr Address Low Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_WRITE_ABORT_LOW_REG | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_WRITE_ABORT_LOW_REG | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_WRITE_ABORT_LOW_REG | All 0 | R/W | The DMA uses this field to generate bits [31:0] of the address field for the Abort IMWr TLP it generates. Bit [1:0] must be "00" as this address must be DWORD aligned. |

### 104.2.294 DMA Capabilities Registers 17 (DMACAP17)

**Common to Header TYPE00/01**

- DMA Write Abort IMWr Address High Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_WRITE_ABORT_HIGH_REG | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

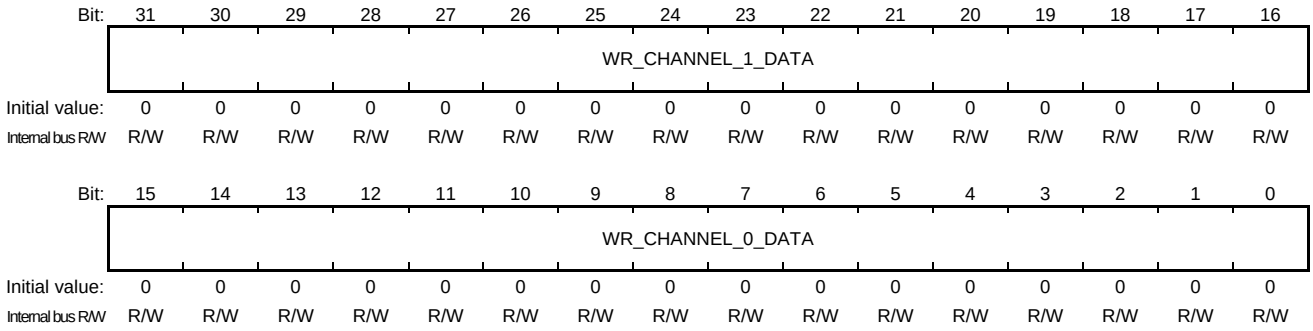
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_WRITE_ABORT_HIGH_REG | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_WRITE_ABORT_HIGH_REG | All 0 | R/W | The DMA uses this field to generate bits [63:32] of the address field for the Abort IMWr TLP. |

### 104.2.295 DMA Capabilities Registers 18 (DMACAP18)

**Common to Header TYPE00/01**

- DMA Write Channel 1 and 0 IMWr Data Register

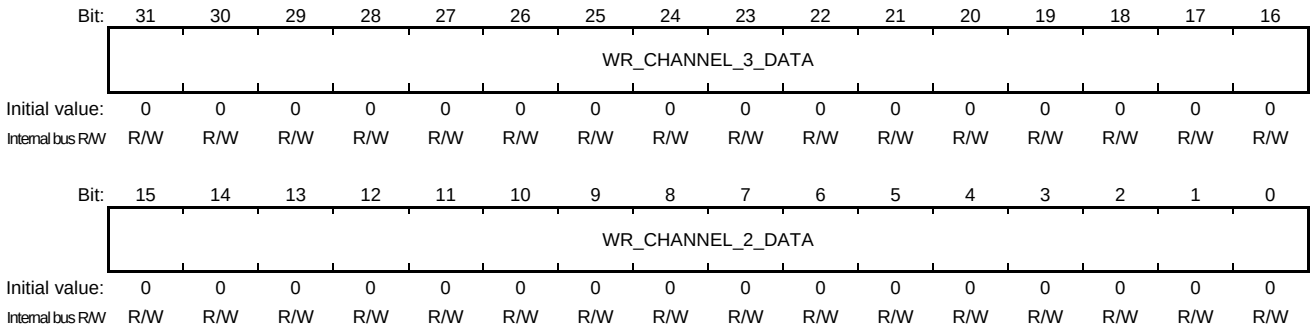



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | WR_CHANNEL_1_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for write channel 1. |
| 15 to 0 | WR_CHANNEL_0_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for write channel 0. |

### 104.2.296 DMA Capabilities Registers 19 (DMACAP19)

**Common to Header TYPE00/01**

- DMA Write Channel 3 and 2 IMWr Data Register



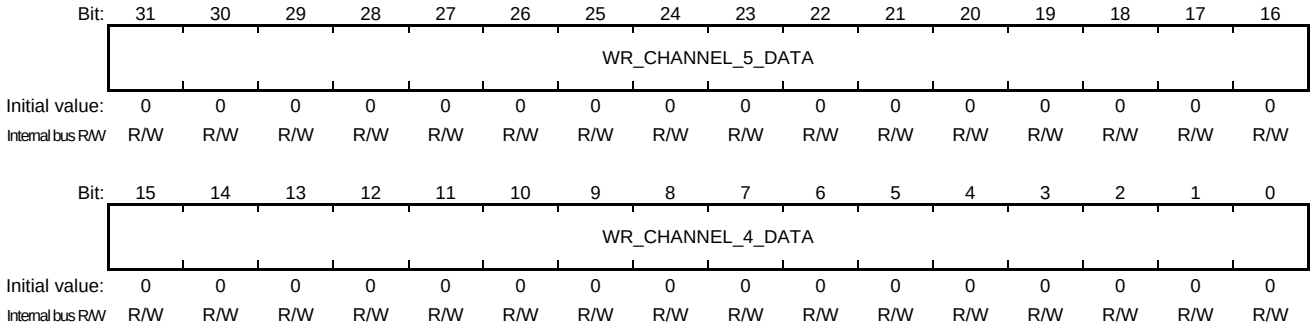

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | WR_CHANNEL_3_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for write channel 3. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 15 to 0 | WR_CHANNEL_2_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for write channel 2. |

### 104.2.297 DMA Capabilities Registers 20 (DMACAP20)

**Common to Header TYPE00/01**

- DMA Write Channel 5 and 4 IMWr Data Register

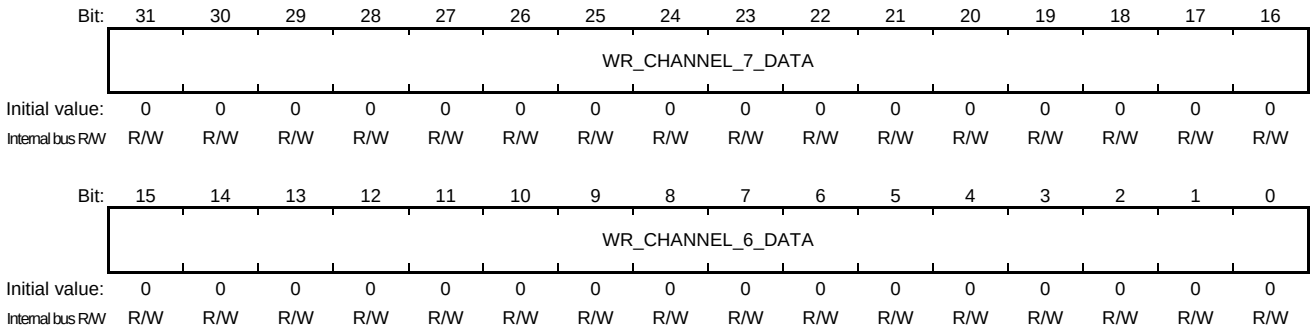



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | WR_CHANNEL_5_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for write channel 5. |
| 15 to 0 | WR_CHANNEL_4_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for write channel 4. |

### 104.2.298 DMA Capabilities Registers 21 (DMACAP21)

**Common to Header TYPE00/01**

- DMA Write Channel 7 and 6 IMWr Data Register



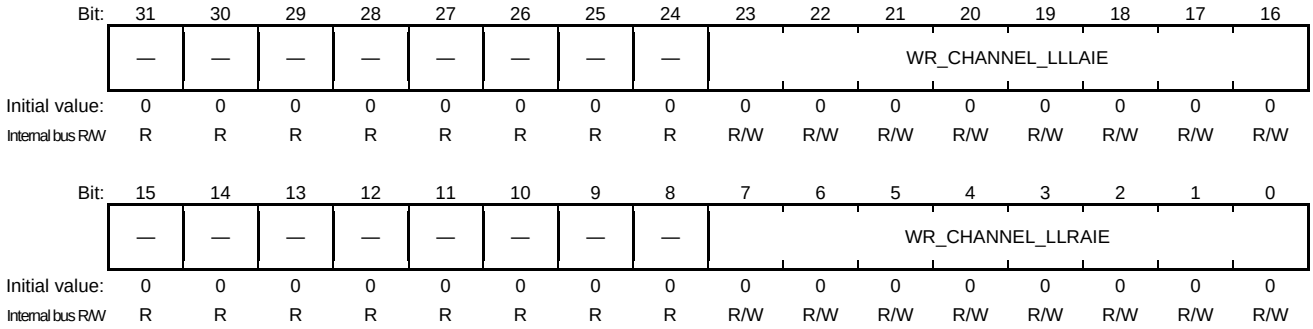

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | WR_CHANNEL_7_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for write channel 7. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 15 to 0 | WR_CHANNEL_6_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for write channel 6. |

### 104.2.299 DMA Capabilities Registers 22 (DMACAP22)

**Common to Header TYPE00/01**

- DMA Write Linked List Error Enable Register

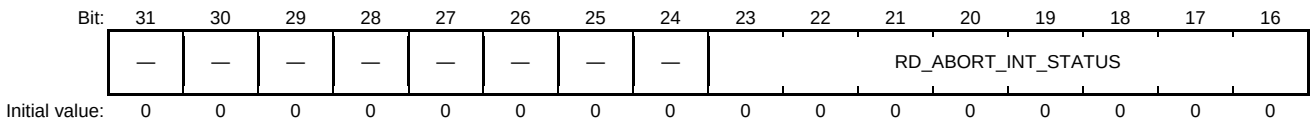



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | WR_CHANNEL_LLLAIE | All 0 | R/W | Write Channel LL Local Abort Interrupt Enable (LLLAIE).<br>You enable the write channel local abort interrupt through this bit. The LIE and RIE bits in the LL element enable the write channel done interrupts. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0. Used in linked list mode only. For more details, see "Interrupt Handling". |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | WR_CHANNEL_LLRAIE | All 0 | R/W | Write Channel LL Remote Abort Interrupt Enable (LLRAIE).<br>You enable the write channel remote abort interrupt through this bit. The LIE and RIE bits in the LL element enable the write channel done interrupts. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0. Used in linked list mode only. For more details, see "Interrupt Handling". |

### 104.2.300 DMA Capabilities Registers 23 (DMACAP23)

**Common to Header TYPE00/01**

- DMA Read Interrupt Status Register



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | — | — | — | — | — | RD_DONE_INT_STATUS | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | RD_ABORT_INT_STATUS | All 0 | R/W | Abort Interrupt Status. The DMA read channel has detected an error, or you manually stopped the transfer as described in "Stopping the DMA Transfer (Software Stop)". Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0. You can read the "DMA Capabilities Registers 26" (DMACAP26) and "DMA Capabilities Registers 27" (DMACAP27) to determine the source of the error.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA read interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the DMA read interrupt Clear register to clear this interrupt bit.<br>Note: You can write to this register to emulate interrupt generation, during software or hardware testing. A write to the address triggers an interrupt, but the DMA does not set the Done or Abort bits in this register. |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | RD_DONE_INT_STATUS | All 0 | R/W | Done Interrupt Status. The DMA read channel has successfully completed the DMA read transfer. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA read interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the DMA read interrupt Clear register to clear this interrupt bit.<br>Note: You can write to this register to emulate interrupt generation, during software or hardware testing. A write to the address triggers an interrupt, but the DMA does not set the Done or Abort bits in this register. |

### 104.2.301 DMA Capabilities Registers 24 (DMACAP24)

#### Common to Header TYPE00/01

- DMA Read Interrupt Mask Register

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | — | — | — | — | — | — | — | — | RD_ABORT_INT_MASK | | | | | | | |

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RD_ABORT_INT_CLEAR | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RD_DONE_INT_CLEAR | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | RD_ABORT_INT_CLEAR | H'FF | R/W | Abort Interrupt Clear. Clears the Abort interrupt status field in the DMA read interrupt status register. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0. (*m* = 0, 1, 2, 3)<br>0: Clear the Abort interrupt status field<br>1: Ignored |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | RD_DONE_INT_CLEAR | H'FF | R/W | Done Interrupt Clear. Clears the Done interrupt status field in the DMA read interrupt status register. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0. (*m* = 0, 1, 2, 3)<br>0: Clear the Done interrupt status field<br>1: Ignored |

### 104.2.302 DMA Capabilities Registers 25 (DMACAP25)

### Common to Header TYPE00/01

- DMA Read Interrupt Clear Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RD_ABORT_INT_CLEAR | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RD_DONE_INT_CLEAR | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 23 to 16 | RD_ABORT_INT_CLEAR | All 0 | W | Abort Interrupt Clear. You must write a 1'b1 to clear the corresponding bit in the Abort interrupt status field of the DMA read interrupt status register. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>Note: Reading from this self-clearing register field always returns a "0". |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | RD_DONE_INT_CLEAR | All 0 | W | Done Interrupt Clear. You must write a 1'b1 to clear the corresponding bit in the Done interrupt status field of the DMA read interrupt status register. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>Note: Reading from this self-clearing register field always returns a "0". |

### 104.2.303 DMA Capabilities Registers 26 (DMACAP26)

**Common to Header TYPE00/01**

- DMA Read Error Status Low Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LINK_LIST_ELEMENT_FETCH_ERR_DETECT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | APP_WR_ERR_DETECT | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

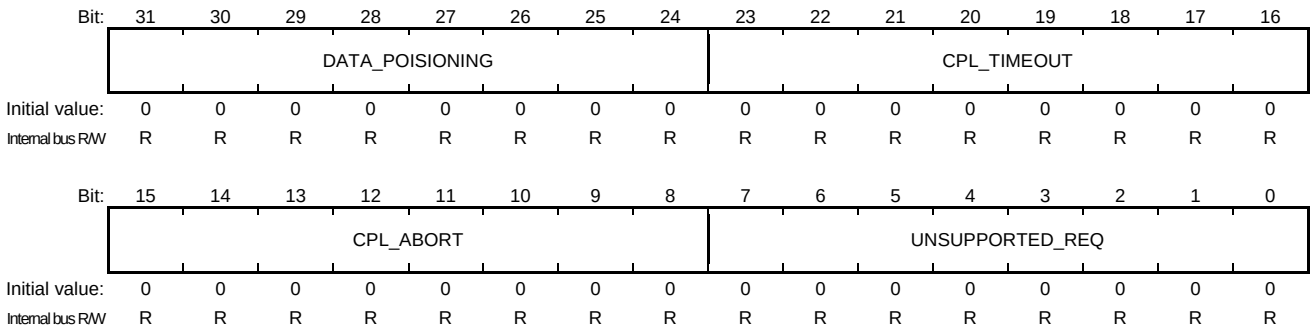
| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | LINK_LIST_ELEMENT_FETCH_ERR_DETECT | All 0 | R | Linked List Element Fetch Error Detected.<br>■ The DMA read channel has received an error response from the AXI bus while reading a linked list element from local memory. Each bit corresponds to a DMA channel.<br>Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA read interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the Abort interrupt field of the "DMA Capabilities Registers 25" (DMACAP25) to clear this error bit.<br>Note, this clears all bits in this register, and also the DMA Capabilities Registers 27 (DMACAP27). |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | APP_WR_ERR_DETECT | All 0 | R | Application Write Error Detected. The DMA read channel has received an error response from the AXI bus (or TRGT1 interface when the AXI Bridge is not used) while writing data to it. This error is fatal. You must restart the transfer from the beginning, as the channel context is corrupted, and the transfer is not rolled back. For more details, see "Linked List Mode". Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA read interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the Abort interrupt field of the "DMA Capabilities Registers 25" (DMACAP25) to clear this error bit.<br>Note, this clears all bits in this register, and also the DMA Capabilities Registers 27 (DMACAP27). |

### 104.2.304 DMA Capabilities Registers 27 (DMACAP27)

### Common to Header TYPE00/01

- DMA Read Error Status High Register



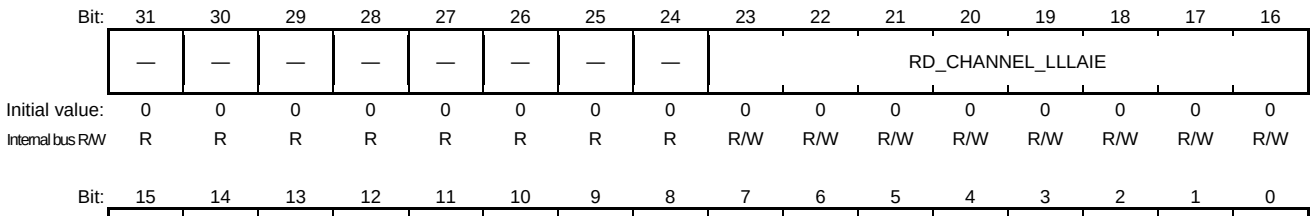

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | DATA_POISIONING | All 0 | R | Data Poisoning. The DMA read channel has detected data poisoning in the completion from the remote device (in response to the MRd request). The DMA read channel will drop the completion and then be halted. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA read interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the Abort interrupt field of the "DMA Capabilities Registers 25" (DMACAP25) to clear this error bit.<br>Note, this also clears the other error bits for the same channel in this register and in the DMA Capabilities Registers 26. |

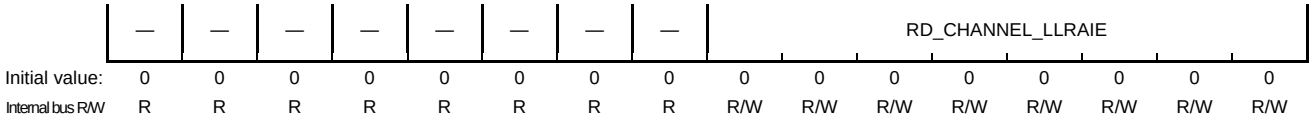
| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 23 to 16 | CPL_TIMEOUT | All 0 | R | Completion Time Out. The DMA read channel has timed-out while waiting for the remote device to respond to the MRd request, or a malformed CplD has been received. For more details, see "Linked List Mode". Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA read interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the Abort interrupt field of the "DMA Capabilities Registers 25" (DMACAP25) to clear this error bit.<br>Note, this also clears the other error bits for the same channel in this register and in the DMA Capabilities Registers 26. |
| 15 to 8 | CPL_ABORT | All 0 | R | Completer Abort. The DMA read channel has received a PCIe completer abort completion status from the remote device in response to the MRd request. For more details, see "Linked List Mode". Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA read interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the Abort interrupt field of the " DMA Capabilities Registers 25" (DMACAP25) to clear this error bit.<br>Note, this also clears the other error bits for the same channel in this register and in the DMA Capabilities Registers 26. |
| 7 to 0 | UNSUPPORTED_REQ | All 0 | R | Unsupported Request. The DMA read channel has received a PCIe unsupported request completion status from the remote device in response to the MRd request. For more details, see "Linked List Mode". Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0.<br>■ Enabling: For details, see "Interrupts and Error Handling".<br>■ Masking: The DMA read interrupt Mask register has no effect on this register.<br>■ Clearing: You must write a 1'b1 to the corresponding channel bit in the Abort interrupt field of the "DMA Capabilities Registers 25" (DMACAP25) to clear this error bit.<br>Note, this also clears the other error bits for the same channel in this register and in the DMA Capabilities Registers 26. |

### 104.2.305 DMA Capabilities Registers 28 (DMACAP28)

### Common to Header TYPE00/01

- DMA Read Linked List Error Enable Register



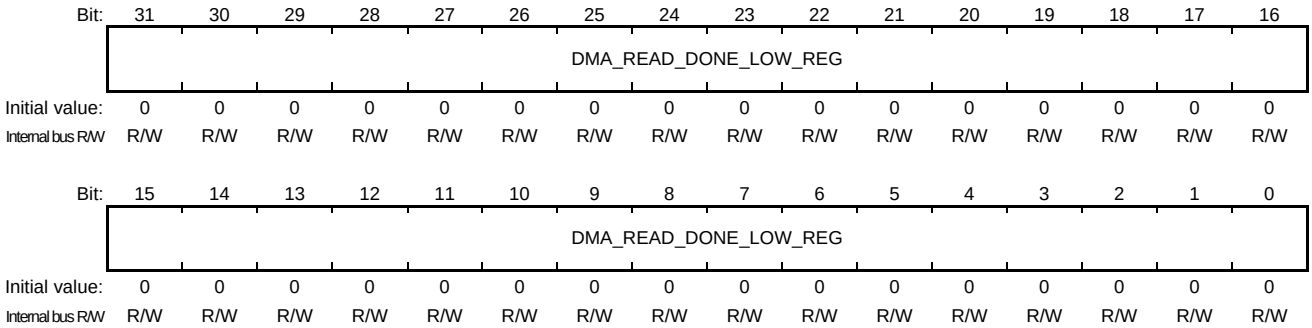



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 to 16 | RD_CHANNEL_LLLAIE | All 0 | R/W | Read Channel LL Local Abort Interrupt Enable (LLLAIE). You enable the read channel Local Abort interrupt through this bit. The LIE and RIE bits in the LL element enable the read channel done interrupts. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0. Used in linked list mode only. For more details, see "Interrupt Handling". |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 to 0 | RD_CHANNEL_LLRAIE | All 0 | * R/W | Read Channel LL Remote Abort Interrupt Enable (LLRAIE). You enable the read channel Remote Abort interrupt through this bit. The LIE and RIE bits in the LL element enable the read channel done interrupts. Each bit corresponds to a DMA channel. Bit [0] corresponds to channel 0. Used in linked list mode only. For more details, see "Interrupt Handling". |

### 104.2.306 DMA Capabilities Registers 29 (DMACAP29)

**Common to Header TYPE00/01**

- DMA Read Done IMWr Address Low Register

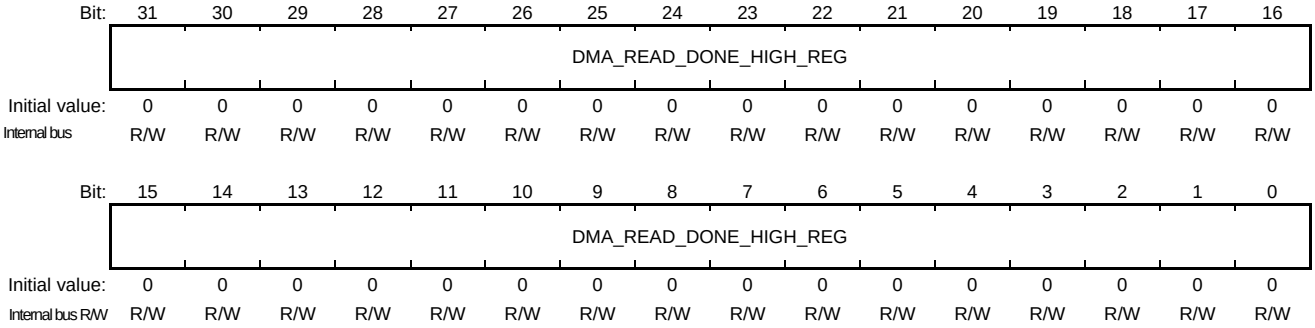



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_READ_DONE_LOW_REG | All 0 | R/W | The DMA uses this field to generate bits [31:0] of the address field for the Done IMWr TLP. Bits [1:0] must be "00" as this address must be DWORD aligned. |

### 104.2.307 DMA Capabilities Registers 30 (DMACAP30)

**Common to Header TYPE00/01**

- DMA Read Done IMWr Address High Register

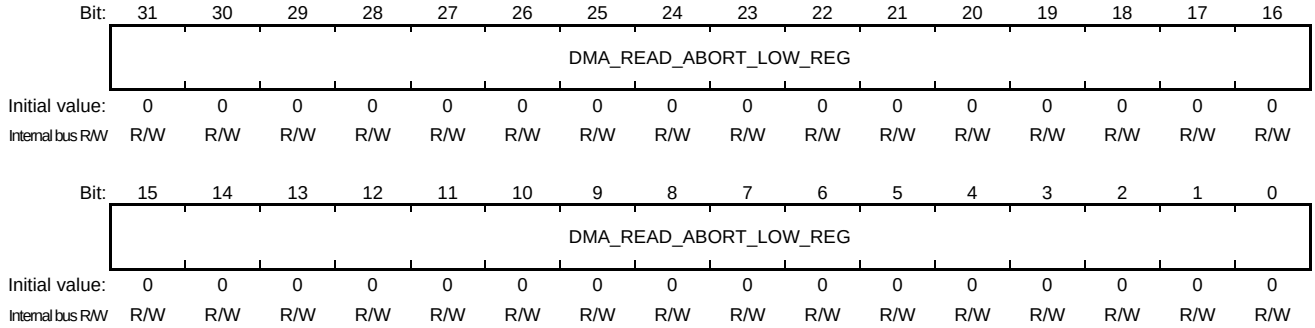



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_READ_DONE_HIGH_REG | All 0 | R/W | The DMA uses this field to generate bits [63:32] of the address field for the Done IMWr TLP. |

### 104.2.308 DMA Capabilities Registers 31 (DMACAP31)

**Common to Header TYPE00/01**

- DMA Read Abort IMWr Address Low Register

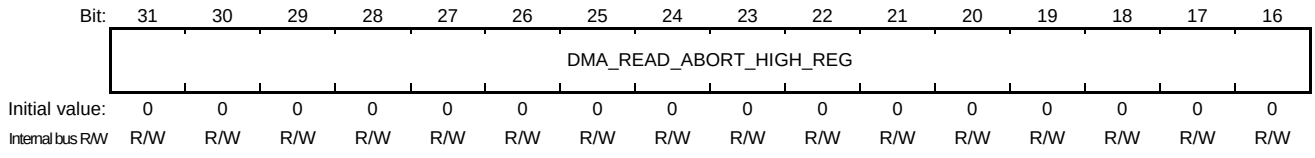


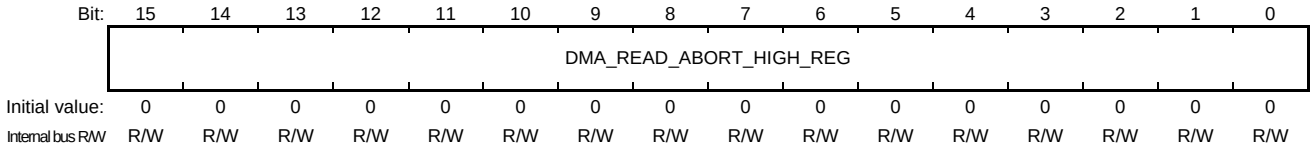

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_READ_ABORT_LOW_REG | All 0 | R/W | The DMA uses this field to generate bits [31:0] of the address field for the Abort IMWr TLP. Bits [1:0] must be "00" as this address must be DWORD aligned. |

### 104.2.309 DMA Capabilities Registers 32 (DMACAP32)

**Common to Header TYPE00/01**

- DMA Read Abort IMWr Address High Register



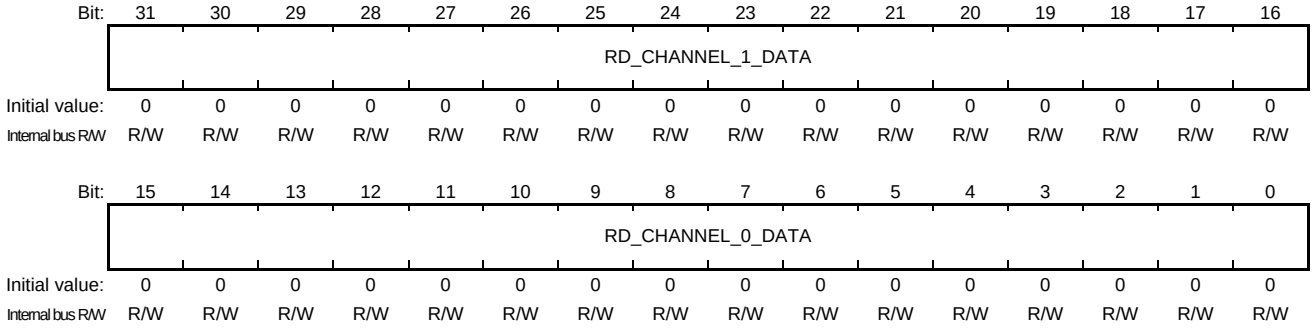



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_READ_ABORT_ HIGH_REG | All 0 | R/W | The DMA uses this field to generate bits [63:32] of the address field for the Abort IMWr TLP. |

### 104.2.310 DMA Capabilities Registers 33 (DMACAP33)

**Common to Header TYPE00/01**

- DMA Read Channel 1 and 0 IMWr Data Register

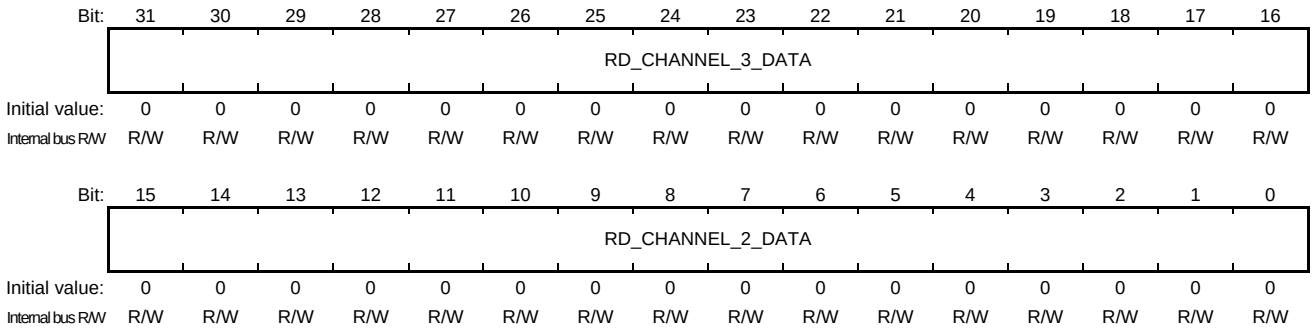



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | RD_CHANNEL_1_DAT A | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for read channel 1. |
| 15 to 0 | RD_CHANNEL_0_DAT A | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for read channel 0. |

### 104.2.311 DMA Capabilities Registers 34 (DMACAP34)

**Common to Header TYPE00/01**

- DMA Read Channel 3 and 2 IMWr Data Register

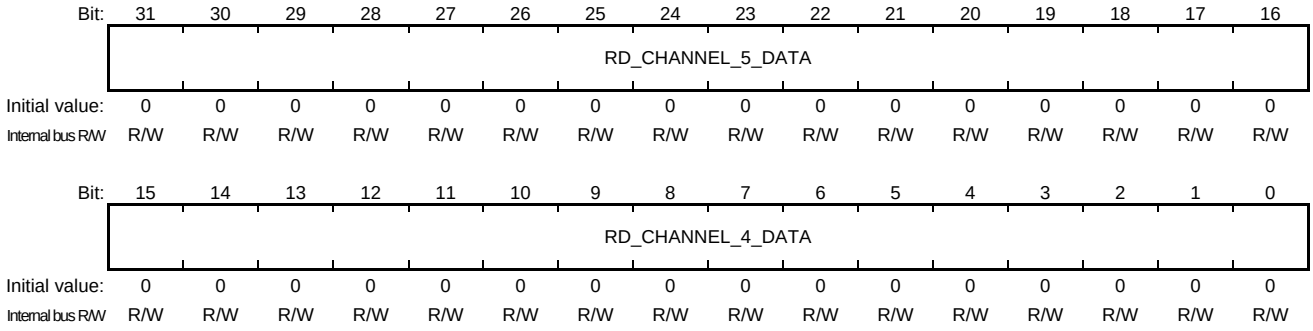


| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | RD_CHANNEL_3_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for read channel 3. |
| 15 to 0 | RD_CHANNEL_2_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for read channel 2. |

### 104.2.312 DMA Capabilities Registers 35 (DMACAP35)

**Common to Header TYPE00/01**

- DMA Read Channel 5 and 4 IMWr Data Register

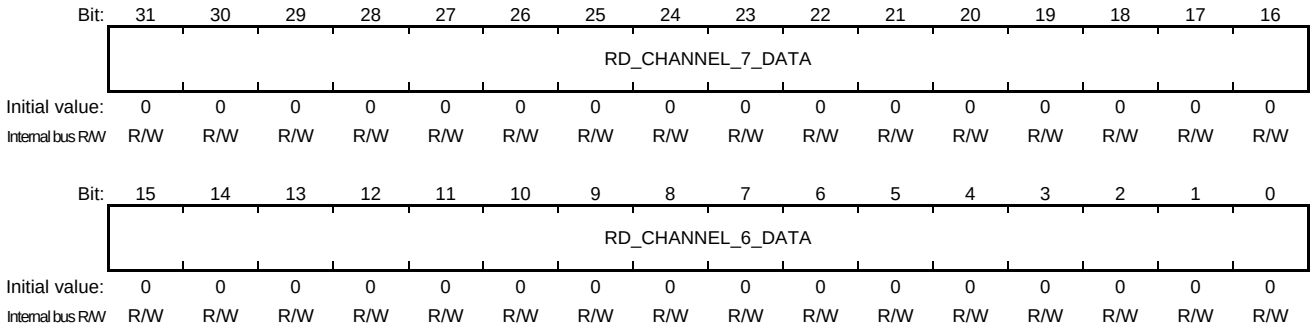



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | RD_CHANNEL_5_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for read channel 5. |
| 15 to 0 | RD_CHANNEL_4_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for read channel 4. |

### 104.2.313 DMA Capabilities Registers 36 (DMACAP36)

**Common to Header TYPE00/01**

- DMA Read Channel 7 and 6 IMWr Data Register



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | RD_CHANNEL_7_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for read channel 7. |
| 15 to 0 | RD_CHANNEL_6_DATA | All 0 | R/W | The DMA uses this field to generate the data field for the Done or Abort IMWr TLPs it generates for read channel 6. |

### 104.2.314 DMA Capabilities Registers [37+*i**16] (DMACAP[37+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Write Channel Control 1 Register

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_AT | | DMA_TC | | | — | DMA_RO | DMA_NS_SRC | DMA_NS_DST | DMA_ME | — | — | — | — | — | DMA_FUNC_NUM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMA_FUNC_NUM | | | | — | — | LLE | CCS | — | CS | | RIE | LIE | LLP | TCB | CB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 30 | DMA_AT | All 0 | R/W | Address Translation TLP Header Bit (AT)<br>The DMA uses this TLP header field when generating MRd/MWr (not IMWr) TLPs. |
| 29 to 27 | DMA_TC | All 0 | R/W | Traffic Class TLP Header Bit (TC)<br>The DMA uses this TLP header field when generating MRd/MWr (not IMWr) TLPs. |
| 26 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 | DMA_RO | 0 | R/W | Relaxed Ordering TLP Header Bit (RO)<br>The DMA uses this TLP header field when generating MRd/MWr (not IMWr) TLPs. |
| 24 | DMA_NS_SRC | 0 | R/W | Source No Snoop TLP Header Bit (DMA_NS_SRC).<br>The DMA uses this TLP header field when generating MRd (SAR addressing space) (not IMWr) TLPs. |
| 23 | DMA_NS_DST | 0 | R/W | Destination No Snoop TLP Header Bit (DMA_NS_DST).<br>The DMA uses this TLP header field when generating MWr (DAR addressing space) (not IMWr) TLPs. |
| 22 | DMA_MEM_TYPE | 0 | R/W | Master AXI ACE-Lite Cache Coherency Control. This field sets the DMA channel memory type of the address space of the data transfer as follows:<br>■ 0x0:peripheral type<br>■ 0x1:memory type |

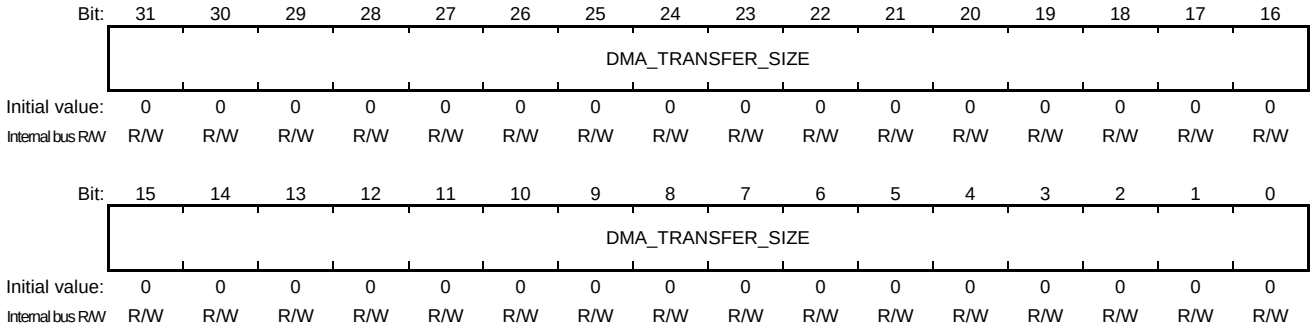
| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 21 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 to 12 | DMA_FUNC_NUM | All 0 | R/W | DMA Channel Physical Function Number. This field indicates the physical function associated with the DMA channel. The controller uses this field to form the requester ID of the generated MRd/MWr DMA TLPs. |
| 11 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | LLE | 0 | R/W | Linked List Enable (LLE).<br>■ 0: Disable linked list operation<br>■ 1: Enable linked list operation |
| 8 | CCS | 0 | R/W | Consumer Cycle State (CCS). Used in linked list mode only.<br>It is used to synchronize the producer (software) and the consumer (DMA). For more details, see "PCS-CCS-CB-TCB Producer-Consumer Synchronization". You must initialize this bit. The DMA updates this bit during linked list operation. |
| 7 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 to 5 | CS | All 0 | R | Channel Status (CS). The channel status bits identify the current operational state of the DMA channel. The operation state encoding for each DMA channel is a s follows:<br>■ 00: Reserved<br>■ 01: Running. This channel is active and transferring data.<br>■ 10: Halted. An error condition has been detected, and the DMA has stopped this channel.<br>■ 11: Stopped. The DMA has transferred all data for this channel, or you have prematurely stopped this channel by writing to the Stop field of the "DMA Capabilities Registers 3" (DMACAP3) or "DMA Capabilities Registers 7" (DMACAP7). |
| 4 | RIE | 0 | R/W | Remote Interrupt Enable (RIE). You must set this bit to enable the generation of the Done or Abort Remote interrupts. For more details, see "Interrupts and Error Handling". In LL mode, the DMA overwrites this with the RIE of the LL element. The RIE bit in a LL element only enables the Done interrupt. In non-LL mode, the RIE bit enables the Done and Abort interrupts.<br>This field is not defined in a link LL element. |
| 3 | LIE | 0 | R/W | Local Interrupt Enable (LIE). You must set this bit to enable the generation of the Done or Abort Local interrupts. For more details, see "Interrupts and Error Handling". In LL mode, the DMA overwrites this with the LIE of the LL element. The LIE bit in a LL element only enables the Done interrupt. In non-LL mode, the LIE bit enables the Done and Abort interrupts.<br>This field is not defined in a link LL element. |
| 2 | LLP | 0 | R/W | Load Link Pointer (LLP). Used in linked list mode only.<br>Indicates that this linked list element is a link element, and its LL element pointer dwords are pointing to the next (noncontiguous) element. The DMA loads this field with the LLP of the linked list element. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 1 | TCB | 0 | R/W | Toggle Cycle Bit (TCB). Indicates to the DMA to toggle its interpretation of the CB. Used in linked list mode only. It is used to synchronize the producer (software) and the consumer (DMA). For more details, see "PCS-CCS-CB-TCB Producer-Consumer Synchronization". The DMA loads this field with the TCB of the linked list element. this field is not defined in a data LL element. |
| 0 | CB | 0 | R/W | Cycle Bit (CB). Used in linked list mode only. It is used to synchronize the producer (software) and the consumer (DMA). For more details, see "PCS-CCS-CB-TCB Producer-Consumer Synchronization". The DMA loads this field with the CB of the linked list element. |

### 104.2.315 DMA Capabilities Registers [38+*i**16] (DMACAP[38+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Write Transfer Size Register

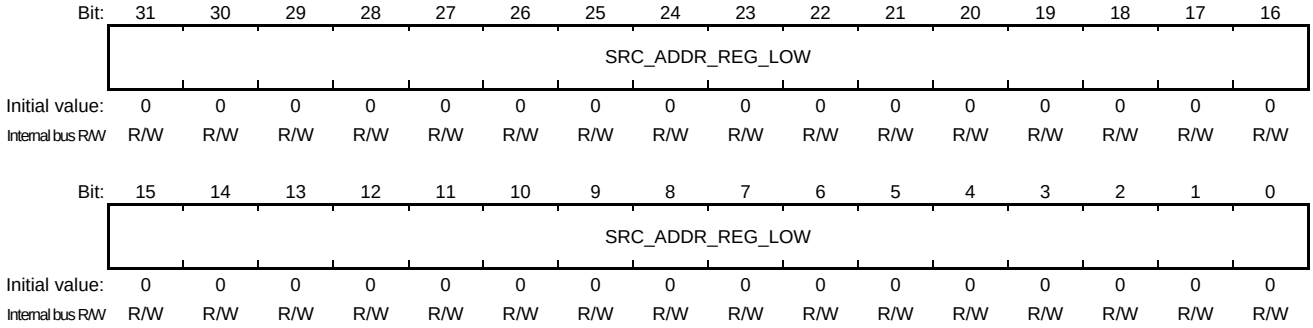



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_TRANSFER_SIZE | 0 | R/W | DMA Transfer Size. You program this register with the size of the DMA transfer. The maximum DMA transfer size is 4Gbytes. The minimum transfer size is one byte (0x1). This field is automatically decremented by the DMA as the DMA write channel transfer progresses. This field indicates the number bytes remaining to be transferred. When all bytes are successfully transferred the current transfer size is zero. In LL mode, the DMA overwrites this register with the corresponding dword of the LL element. You can read this register to monitor the transfer progress, however in some scenarios this register is updated after a delay. For example, when less than 3 channels are doorbelled, this register is updated only after a descriptor finishes (linked list mode), or the transfer ends (non-linked list mode). |

### 104.2.316 DMA Capabilities Registers [39+*i**16] (DMACAP[39+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Write SAR Low Register

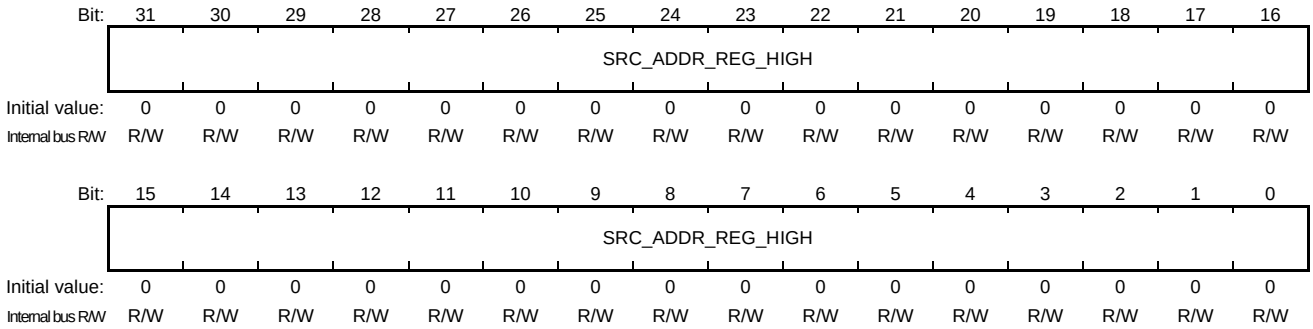



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRC_ADDR_REG_LOW | 0 | R/W | Source Address Register (Lower 32 bits). Indicates the next address to be read from. The DMA increments the SAR as the DMA transfer progresses. In LL mode, the DMA overwrites this with the corresponding DWORD of the LL element.<br>■ DMA Read: The SAR is the address of the remote memory.<br>■ DMA Write: The SAR is the address of the local memory. |

### 104.2.317 DMA Capabilities Registers [40+*i**16] (DMACAP[40+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Write SAR High Register

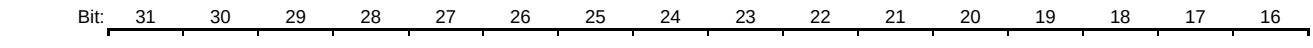


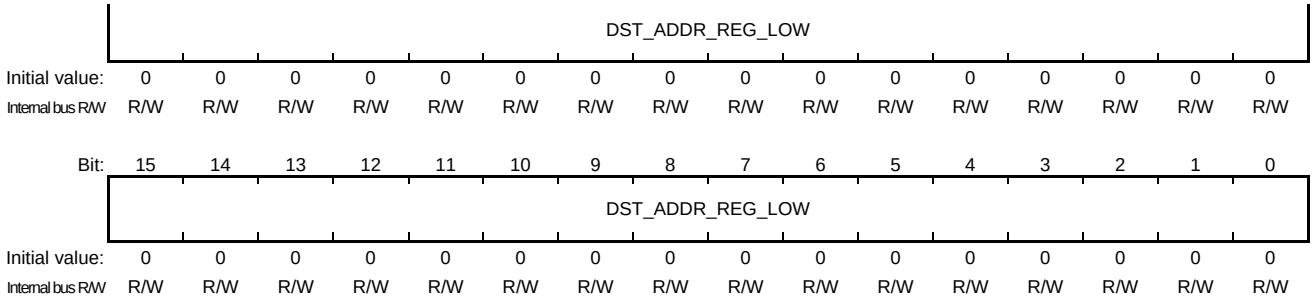

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRC_ADDR_REG_HIGH | All 0 | R/W | Source Address Register (Higher 32 bits). In LL mode, the DMA overwrites this with the corresponding DWORD of the LL element. |

### 104.2.318 DMA Capabilities Registers [41+*i**16] (DMACAP[41+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Write DAR Low Register



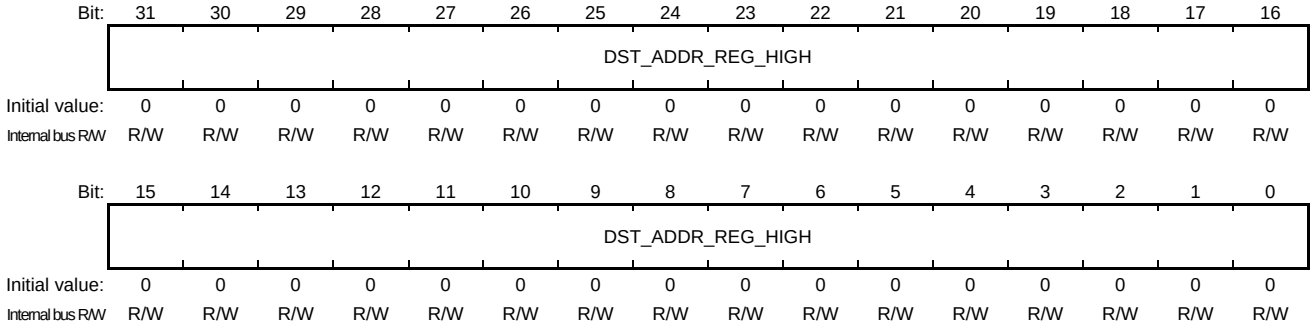



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DST_ADDR_REG_LOW | All 0 | R/W | Destination Address Register (Lower 32 bits). Indicates the next address to be written to. The DMA increments the DAR as the DMA transfer progresses. In LL mode, the DMA overwrites this with the corresponding dword of the LL element.<br>■ DMA Read: The DAR is the address of the local memory.<br>■ DMA Write: The DAR is the address of the remote memory. |

### 104.2.319 DMA Capabilities Registers [42+*i**16] (DMACAP[42+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Write DAR High Register

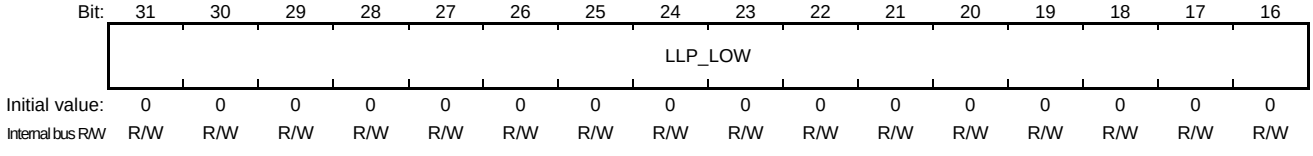


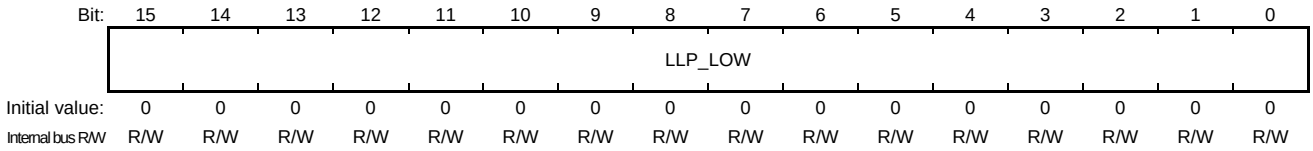

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DST_ADDR_REG_HIGH | All 0 | R/W | Destination Address Register (Higher 32 bits). In LL mode, the DMA overwrites this with the corresponding DWORD of the LL element. |

### 104.2.320 DMA Capabilities Registers [43+*i**16] (DMACAP[43+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Write Linked List Pointer Low Register



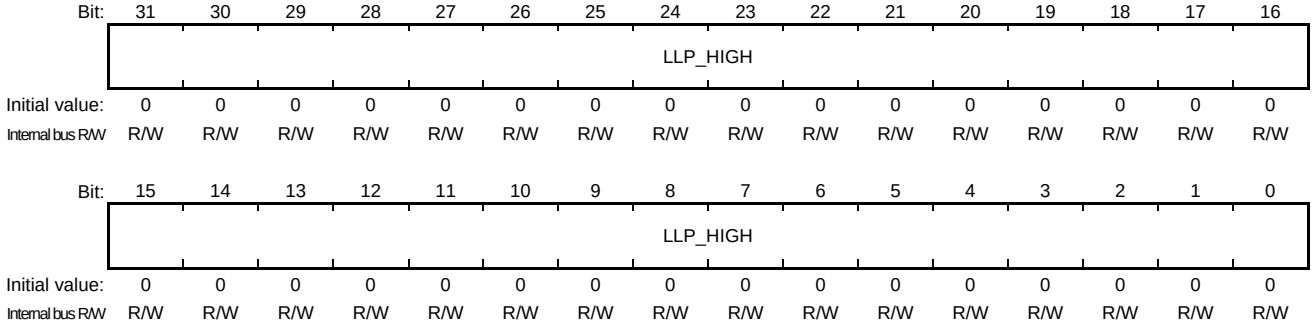



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LLP_LOW | All 0 | R/W | Lower bits of the address of the linked list transfer list in local memory. Used in linked list mode only. Updated by the DMA to point to the next element in the transfer list after the previous element is consumed.<br>■ When the current element is a data element; this field is incremented by 6 DWORDS.<br>■ When the current element is a link element; this field is overwritten by the LL Element Pointer of the element. |

### 104.2.321 DMA Capabilities Registers [44+*i**16] (DMACAP[44+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Write Linked List Pointer High Register

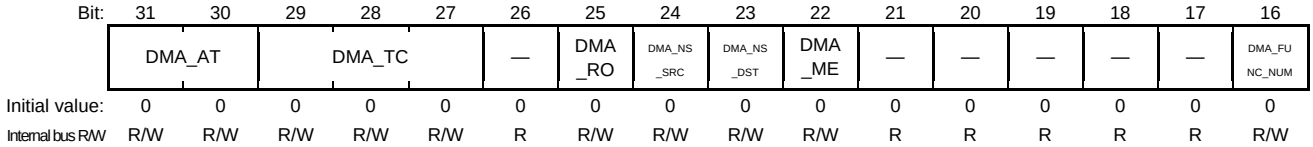


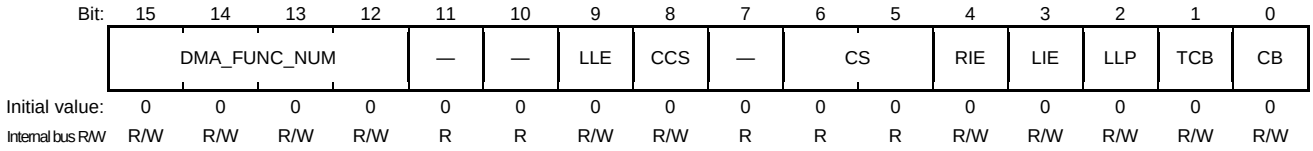

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LLP_HIGH | All 0 | R/W | Upper 32 bits of the address of the linked list transfer list in local memory. Used in linked list mode only. Updated by the DMA to point to the next element in the transfer list as elements are consumed. |

### 104.2.322 DMA Capabilities Registers [45+*i**16] (DMACAP[45+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Read Channel Control 1 Register





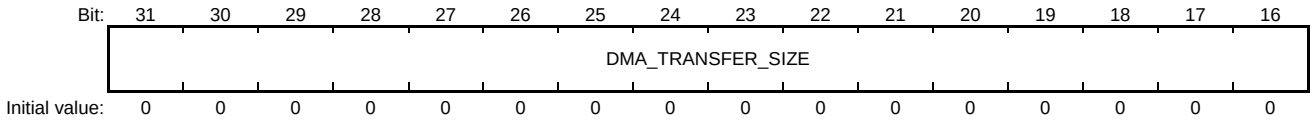

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 30 | DMA_AT | All 0 | R/W | Address Translation TLP Header Bit (AT)<br>The DMA uses this TLP header field when generating MRd/MWr (not IMWr) TLPs. |
| 29 to 27 | DMA_TC | All 0 | R/W | Traffic Class TLP Header Bit (TC)<br>The DMA uses this TLP header field when generating MRd/MWr (not IMWr) TLPs. |
| 26 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 | DMA_RO | 0 | R/W | Relaxed Ordering TLP Header Bit (RO)<br>The DMA uses this TLP header field when generating MRd/MWr (not IMWr) TLPs. |
| 24 | DMA_NS_SRC | 0 | R/W | Source No Snoop TLP Header Bit (DMA_NS_SRC).<br>The DMA uses this TLP header field when generating MRd (SAR addressing space) (not IMWr) TLPs. |
| 23 | DMA_NS_DST | 0 | R/W | Destination No Snoop TLP Header Bit (DMA_NS_DST).<br>The DMA uses this TLP header field when generating MWr (DAR addressing space) (not IMWr) TLPs. |
| 22 | DMA_MEM_TYPE | 0 | R/W | Master AXI ACE-Lite Cache Coherency Control. This field sets the DMA channel memory type of the address space of the data transfer as follows:<br>■ 0x0:peripheral type<br>■ 0x1:memory type |
| 21 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 to 12 | DMA_FUNC_NUM | All 0 | R/W | DMA Channel Physical Function Number. This field indicates the physical function associated with the DMA channel. The controller uses this field to form the requester ID of the generated MRd/MWr DMA TLPs. |
| 11 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | LLE | 0 | R/W | Linked List Enable (LLE).<br>■ 0: Disable linked list operation<br>■ 1: Enable linked list operation |
| 8 | CCS | 0 | R/W | Consumer Cycle State (CCS). Used in linked list mode only.<br>It is used to synchronize the producer (software) and the consumer (DMA). For more details, see "PCS-CCS-CB-TCB Producer-Consumer Synchronization". You must initialize this bit. The DMA updates this bit during linked list operation. |
| 7 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

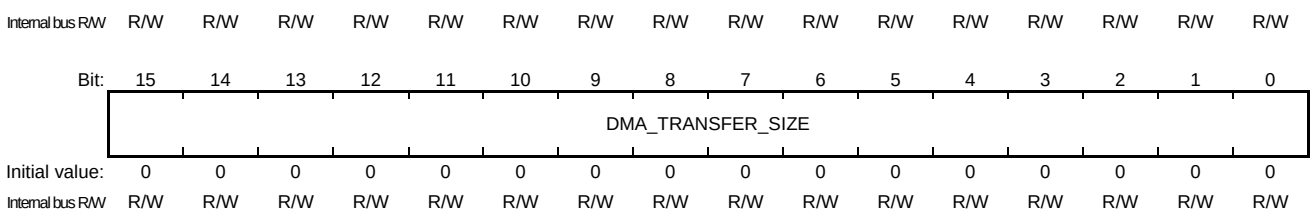
| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 6 to 5 | CS | 0 | R | Channel Status (CS). The channel status bits identify the current operational state of the DMA channel. The operation state encoding for each DMA channel is a s follows:<br>■ 00: Reserved<br>■ 01: Running. This channel is active and transferring data.<br>■ 10: Halted. An error condition has been detected, and the DMA has stopped this channel.<br>■ 11: Stopped. The DMA has transferred all data for this channel, or you have prematurely stopped this channel by writing to the Stop field of the "DMA Capabilities Registers 3" (DMACAP3) or "DMA Capabilities Registers 7" (DMACAP7). |
| 4 | RIE | 0 | R/W | Remote Interrupt Enable (RIE). You must set this bit to enable the generation of the Done or Abort Remote interrupts. For more details, see "Interrupts and Error Handling". In LL mode, the DMA overwrites this with the RIE of the LL element. The RIE bit in a LL element only enables the Done interrupt. In non-LL mode, the RIE bit enables the Done and Abort interrupts. This field is not defined in a link LL element. |
| 3 | LIE | 0 | R/W | Local Interrupt Enable (LIE). You must set this bit to enable the generation of the Done or Abort Local interrupts. For more details, see "Interrupts and Error Handling". In LL mode, the DMA overwrites this with the LIE of the LL element. The LIE bit in a LL element only enables the Done interrupt. In non-LL mode, the LIE bit enables the Done and Abort interrupts. This field is not defined in a link LL element. |
| 2 | LLP | 0 | R/W | Load Link Pointer (LLP). Used in linked list mode only. Indicates that this linked list element is a link element, and its LL element pointer DWORDs are pointing to the next (noncontiguous) element. The DMA loads this field with the LLP of the linked list element. |
| 1 | TCB | 0 | R/W | Toggle Cycle Bit (TCB). Indicates to the DMA to toggle its interpretation of the CB. Used in linked list mode only. It is used to synchronize the producer (software) and the consumer (DMA). For more details, see "PCS-CCS-CB-TCB Producer-Consumer Synchronization". The DMA loads this field with the TCB of the linked list element. this field is not defined in a data LL element. |
| 0 | CB | 0 | R/W | Cycle Bit (CB). Used in linked list mode only. It is used to synchronize the producer (software) and the consumer (DMA). For more details, see "PCS-CCS-CB-TCB Producer-Consumer Synchronization". The DMA loads this field with the CB of the linked list element. |

### 104.2.323 DMA Capabilities Registers [46+*i**16] (DMACAP[46+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Read Transfer Size Register



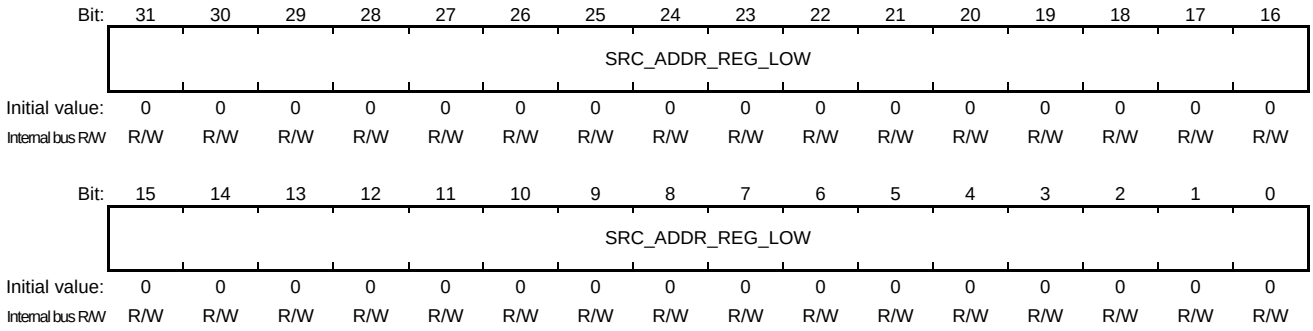



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMA_TRANSFER_SIZE | All 0 | R/W | DMA Transfer Size. You program this register with the size of the DMA transfer. The maximum DMA transfer size is 4Gbytes. The minimum transfer size is one byte (0x1). This field is automatically decremented by the DMA as the DMA read channel transfer progresses. This field indicates the number bytes remaining to be transferred. When all bytes are successfully transferred the current transfer size is zero. In LL mode, the DMA overwrites this register with the corresponding DWORD of the LL element. You can read this register to monitor the transfer progress, however in some scenarios this register is updated after a delay. For example, when less than 3 channels are doorbelled, this register is updated only after a descriptor finishes (linked list mode), or the transfer ends (non-linked list mode). |

### 104.2.324 DMA Capabilities Registers [47+*i**16] (DMACAP[47+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Read SAR Low Register

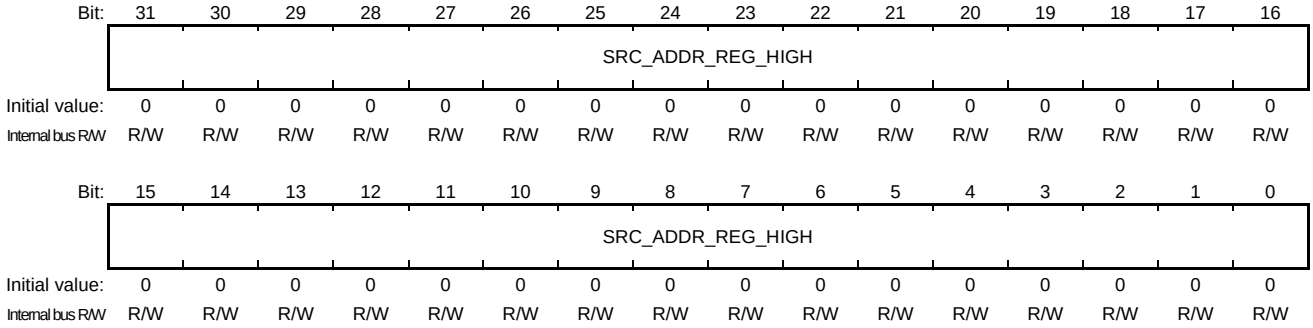



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRC_ADDR_REG_LOW | All 0 | R/W | Source Address Register (Lower 32 bits). Indicates the next address to be read from. The DMA increments the SAR as the DMA transfer progresses. In LL mode, the DMA overwrites this with the corresponding DWORD of the LL element.<br>■ DMA Read: The SAR is the address of the remote memory.<br>■ DMA Read: The SAR is the address of the local memory. |

### 104.2.325 DMA Capabilities Registers [48+*i**16] (DMACAP[48+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Read DAR Low Register

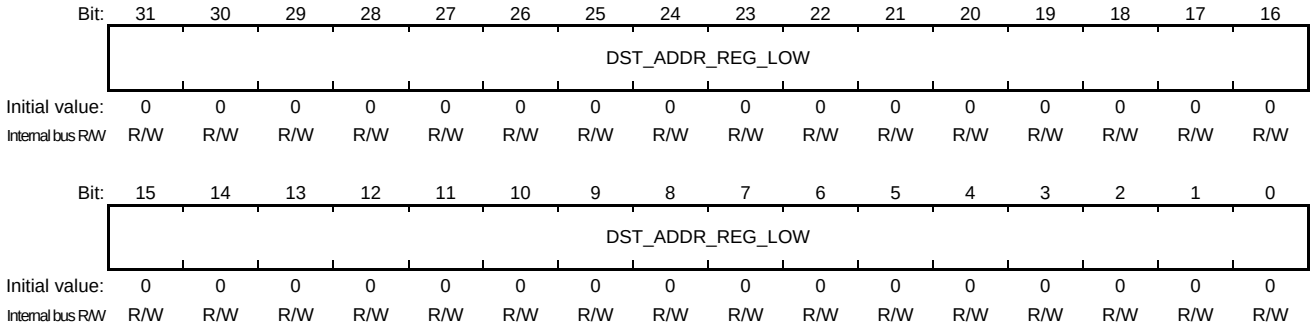



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRC_ADDR_REG_HIGH | All 0 | R/W | Source Address Register (Higher 32 bits). In LL mode, the DMA overwrites this with the corresponding dword of the LL element. |

### 104.2.326 DMA Capabilities Registers [49+*i**16] (DMACAP[49+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Read DAR Low Register

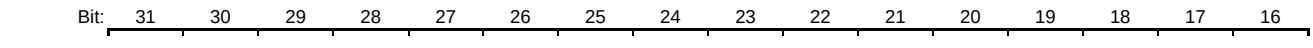


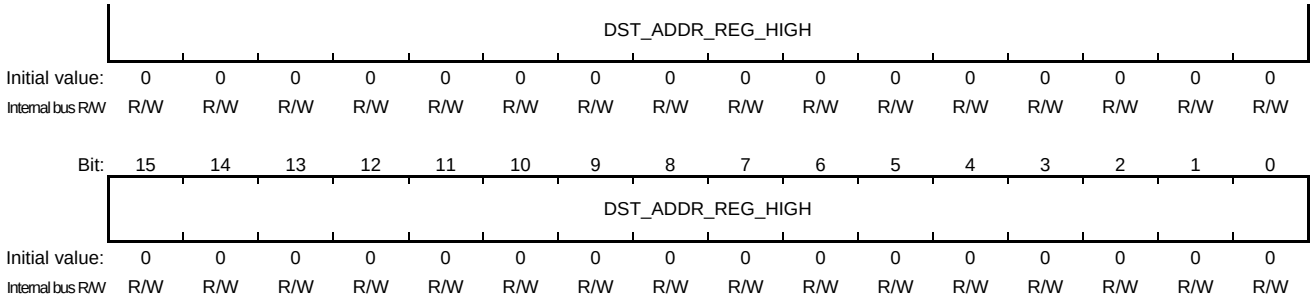

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DST_ADDR_REG_LOW | All 0 | R/W | Destination Address Register (Lower 32 bits). Indicates the next address to be written to. The DMA increments the DAR as the DMA transfer progresses. In LL mode, the DMA overwrites this with the corresponding DWORD of the LL element.<br>■ DMA Read: The DAR is the address of the local memory.<br>■ DMA Read: The DAR is the address of the remote memory. |

### 104.2.327 DMA Capabilities Registers [50+*i**16] (DMACAP[50+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Read DAR High Register



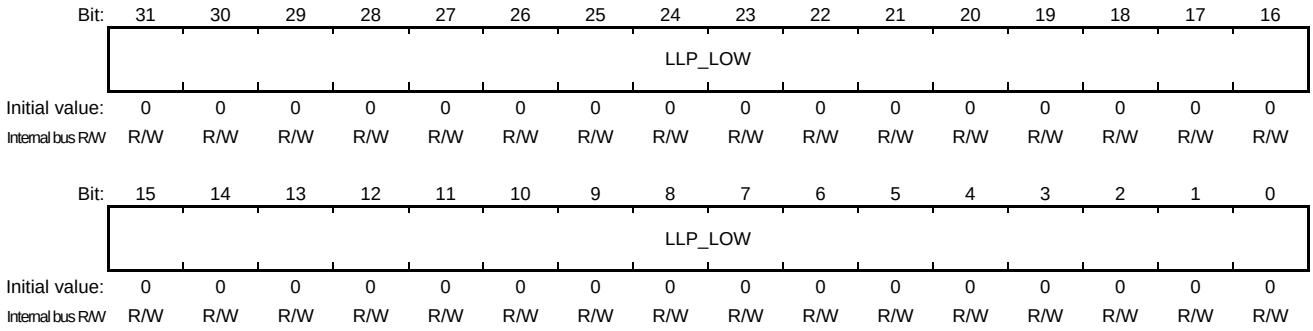



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DST_ADDR_REG_HIGH | All 0 | R/W | Destination Address Register (Higher 32 bits). In LL mode, the DMA overwrites this with the corresponding DWORD of the LL element. |

### 104.2.328 DMA Capabilities Registers [51+*i**16] (DMACAP[51+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Read Linked List Pointer Low Register

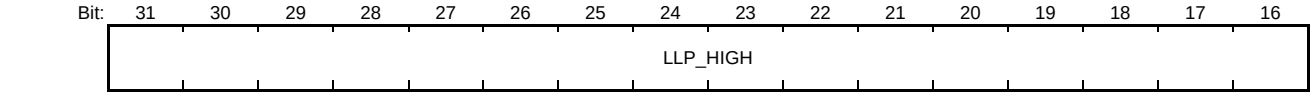



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LLP_LOW | All 0 | R/W | Lower bits of the address of the linked list transfer list in local memory. Used in linked list mode only. Updated by the DMA to point to the next element in the transfer list after the previous element is consumed.<br>■ When the current element is a data element; this field is incremented by 6 DWORDS.<br>■ When the current element is a link element; this field is overwritten by the LL Element Pointer of the element. |

### 104.2.329 DMA Capabilities Registers [52+*i**16] (DMACAP[52+*i**16]) (*i* = 0 to 7)

**Common to Header TYPE00/01**

- DMA Read Linked List Pointer High Register



| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LLP_HIGH | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LLP_HIGH | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | LLP_HIGH | All 0 | R/W | Upper 32 bits of the address of the linked list transfer list in local memory. Used in linked list mode only. Updated by the DMA to point to the next element in the transfer list as elements are consumed. |

### 104.2.330 PCIe Mode Setting Register 0 (PCIEMSR0)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | ELBI_MODE | | app_sris_ | device_type | | | | BFRCTNST | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 9 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 8 to 7 | ELBI_MODE | H'0 | R/W | Handshake between lbc_ext_cs and ext_lbc_ack<br>0: return an error response (per bit)<br>1: do not return an error response (per bit) |
| 6 | app_sris_mode | H'0 | R/W | SRIS operating mode.<br>0: non-SRIS mode<br>1: SRIS mode |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 5 to 2 | device_type | H'0 | R/W | Device/port type.<br>Indicates the specific type of this PCI Express function.<br>It is also used to set the 'Device/Port Type' field of the 'PCI Express Capabilities Register'. The controller uses this input to determine the operating mode of the controller at run time.<br>4'b0000: PCI Express endpoint<br>4'b0001: Legacy PCI Express endpoint<br>4'b0100: Root port of PCI Express root complex<br>Others: Setting prohibited<br>Note: Set device_type before deasserting software reset by CPG SRSTCLR6[24]. |
| 1 to 0 | BFRCTNST | H'0 | R/W | Bifurcation mode setting.<br>00: 1x Quad lane mode (Bifurcation off)<br>01: 2x Dual lanes mode (Bifurcation on)<br>Others: Setting prohibited |

### 104.2.331 PCIe Mode Setting Register 1 (PCIEMSR1)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | tx_lane_flip_en | rx_lane_flip_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 2 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 1 | tx_lane_flip_en | H'0 | R/W | Performs manual lane reversal for transmit lanes.<br>For use when automatic lane reversal does not occur because lane 0 is not detected. In most cases, tx_lane_flip_en should be wired to a static value at the chip level. For more details, see 'Lane Reversal and Broken Lanes'. This signal is only used in Conventional PCIe mode. When in M-PCIe mode, mpcie_tx_lane_flip_en is used. |
| 0 | rx_lane_flip_en | H'0 | R/W | Performs manual lane reversal for receive lanes.<br>For use when automatic lane reversal does not occur because lane 0 is not detected. In most cases, rx_lane_flip_en should be wired to a static value at the chip level. For more details, see 'Lane Reversal and Broken Lanes'. This signal is only used in Conventional PCIe mode. When in M-PCIe mode, mpcie_rx_lane_flip_en is used. |

### 104.2.332 PCIe Mode Setting Register 2 (PCIEMSR2)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | app_dbi_ro_wr_dis | app_req_retry_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 2 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 1 | app_dbi_ro_wr_disable | H'0 | R/W | DBI Read-only Write Disable.<br>0: PRTLGC58.DBI_RO_WR_EN register field is read-write<br>1: PRTLGC58.DBI_RO_WR_EN register field is forced to 0 and is read-only. |
| 0 | app_req_retry_en | H'0 | R/W | Provides a capability to defer incoming configuration requests until initialization is complete.<br>When app_req_retry_en is asserted, the controller completes incoming configuration requests with a configuration request retry status. Other incoming requests complete normally. When the Readiness Notification mechanism is supported, DRS messaging is blocked when app_req_retry_en=1. When app_req_retry_en=0 and app_drs_ready=1, the controller autonomously transmits a DRS message when the link transitions from DL_Down to DL_Up. |

### 104.2.333 PCIe Mode Setting Register 3 (PCIEMSR3)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | APB2DBITOSET | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | APB2DBITOSET | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | APB2DBITOSET | All 0 | R/W | APB2DBI timeout setting |

### 104.2.334 PCIe Reset Control Register 0 (PCIERSTCTRL0)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | refclk_share_prop_en | — | — | — | — | — | — | — | perst_n |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | refclk_share | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 25 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 24 | refclk_share_prop_en | H'0 | R/W | pcie_clkreq_dis_in propagation enable.<br>0: Disable<br>1: Enable |
| 23 to 17 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 16 | perst_n | H'1 | R/W | Indicates when the main power supply is within its tolerated voltage range and is stable.<br>This signal is syncronized using aux_clk and can be driven/supplied<br>asynchronously to the controller in certain low-power modes |
| 15 to 9 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 8 | refclk_share | H'0 | R/W | Reference clock share bit.<br>When shared, previous CLKREQ# is invalid<br>0: Not Shared<br>1: Shared |
| 7 to 0 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

### 104.2.335 PCIe Reset Control Register 1 (PCIERSTCTRL1)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | app_hold_phy_rst |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | app_ltssm_enable |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 17 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 16 | app_hold_phy_rst | H'1 | R/W | Set this signal to 1 during the power on reset sequence to hold the PHY in reset while the PHY is configured through the PHY viewport. |
| 15 to 1 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 0 | app_ltssm_enable | H'0 | R/W | To do otherwise (that is, de-assert it outside of the Detect LTSSM state) causes the controller to be reset and the LTSSM moves immediately back to the Detect state. This transition is outside of the PCIe Specification and it might cause a PIPE protocol violation. |

### 104.2.336 PCIe Reset Control Register 2 (PCIERSTCTRL2)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | app_init_rst |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 1 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 0 | app_init_rst | H'0 | R/W | Request from your application to send a hot reset to the upstream port.<br>The hot reset request is sent when a single cycle pulse is applied to this<br>pin. In an upstream port, you should set this input to '0'.<br>Note: This signal is not used by the controller to set the SBR field in the<br>PCICONF15F*i* register. During the transition from DL_Active to DL_inactive, assertion of app_init_rst signal indicates a Surprise Down Error, but setting of SBR field in the PCICONF15F*i* register through DBI does not trigger a Surprise Down Error. |

### 104.2.337 PCIe Reset Control Register 3 (PCIERSTCTRL3)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | app_flr_pf_done | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 2 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 1 to 0 | app_flr_pf_done | H'0 | R/W | Assert 1 when your application has stopped / reset the PFx application to initiate FLR of PFx.<br>After PCIERSTSTS.cfg_flr_pf_active is cleared, this bit is automatically cleared to 0.<br>Note : After writting 1 during the PCIERSTSTS.cfg_flr_pf_active =1, do not write "0" until it changes to "0" by the automatic clear. |

### 104.2.338 PCIe Reset Status Register (PCIERSTSTS)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | cfg_flr_pf_ | training_r | smlh_req | link_re | link_req_rs | cfg_2nd_reset |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 6 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 5 to 4 | cfg_flr_pf_active | H'0 | R | Set when the software initiates FLR a physical function by writing to the "Initiate FLR" register bit of that function. This signal is held asserted until reset of both the application and controller logic has been completed. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 3 | training_rst_n | H'0 | R | Hot reset from upstream component.<br>When the controller LTSSM receives two consecutive TS1 ordered sets with the hot_reset bit asserted, it asserts training_rst_n for one clock cycle. This signal is only kept for legacy purposes. You should use the link_req_rst_not signal to reset the controller after a 'hot reset' request. |
| 2 | smlh_req_rst_not | H'0 | R | Early version of the link_req_rst_not signal.<br>A low level indicates that the controller is requesting external logic to reset the controller because the PHY link is down. When the AXI bridge module is enabled the link reset request is delayed until all pending AXI transfers have completed. When the AXI Bridge module is not enabled the link reset request is sent immediately. |
| 1 | link_req_rst_not | H'0 | R | Reset request because the link has gone down or the controller received a hot-reset request.<br>A low level indicates that the controller is requesting external logic to reset the controller because the PHY link is down. When the AXI bridge module is enabled the link reset request is delayed until all pending AXI transfers have completed. When the AXI Bridge module is not enabled the link reset request is sent immediately. |
| 0 | cfg_2nd_reset | H'0 | R | Secondary Bus Reset.<br>Indicates that your application has requested this downstream port to start Link hot reset. This signal is asserted when your application writes to the Secondary Bus Reset field (bit 6) of the Bridge Control Register (Offset 0x3E) in the Type 1 Configuration Space header. Setting this bit triggers a hot reset the corresponding PCI Express port. |

### 104.2.339 MSI Control Register 0 (MSICTRL0)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ven_msi_req |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 1 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 0 | ven_msi_req | H'0 | R/W | Request from your application to send an MSI when MSI is enabled.<br>When MSI-X is enabled instead of MSI, assertion of ven_msi_req<br>causes the controller to generate an MSI-X message. Once asserted,<br>ven_msi_req must remain asserted until the controller asserts ven_msi_grant. |

### 104.2.340 MSI Control Register 1 (MSICTRL1)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ven_msi_vector | | | | | ven_msi_tc | | | ven_msi_func_num | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 11 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 10 to 6 | ven_msi_vector | H'0 | R/W | Used to modulate the lower five bits of the MSI Data register when multiple message mode is enabled. |
| 5 to 3 | ven_msi_tc | H'0 | R/W | Traffic Class of the MSI request, valid when ven_msi_req is asserted |
| 2 to 0 | ven_msi_func_num | H'0 | R/W | The function number of the MSI request.<br>Function numbering starts at '0'. |

### 104.2.341 MSI Control Register 2 (MSICTRL2)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | cfg_msi_pending_lo | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | cfg_msi_pending_lo | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | cfg_msi_pending_lo | All 0 | R/W | Lower bits [31:0]. Indication from application about which functions have a pending associated message.<br>Used to determine the value of the Vector Interrupt Pending register in the MSI Capability structure. |

### 104.2.342 MSI Control Register 3 (MSICTRL3)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | cfg_msi_pending_hi | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | cfg_msi_pending_hi | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | cfg_msi_pending_hi | All 0 | R/W | Upper bits [63:32]. Indication from application about which functions have a pending associated message<br>Used to determine the value of the Vector Interrupt Pending register in the MSI Capability structure. |

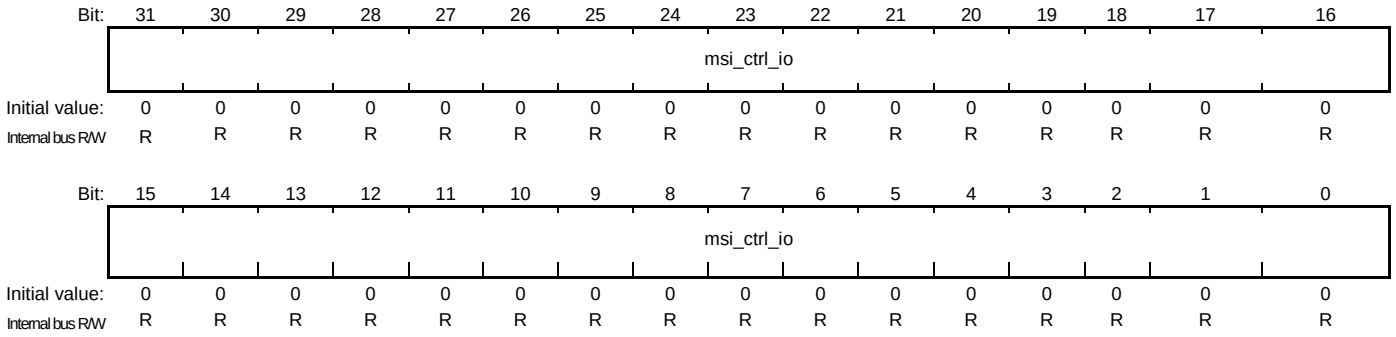
### 104.2.343 MSI Control Register 4 (MSICTRL4)

**Common to Header TYPE00/01**

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 26 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 25 to 24 | cfg_msi_en | H'0 | R | Indicates that MSI is enabled (INTx message is not sent), one bit per configured function. |
| 23 to 16 | msi_ctrl_int_vec | H'0 | R | DSP AXI MSI Interrupt Vector.<br>Asserted when an MSI interrupt is pending. De-asserted when there is no MSI interrupt pending. |
| 15 to 1 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 0 | ven_msi_grant | H'0 | R | Indicates that the controller has accepted the request to send an MSI.<br>After asserting ven_msi_grant for one cycle, the controller does not wait for ven_msi_req to be de-asserted then reasserted to generate another MSI. When ven_msi_req remains asserted after the controller asserts ven_msi_grant for one cycle, the controller generates another MSI. |

### 104.2.344 MSI Control Register 5 (MSICTRL5)

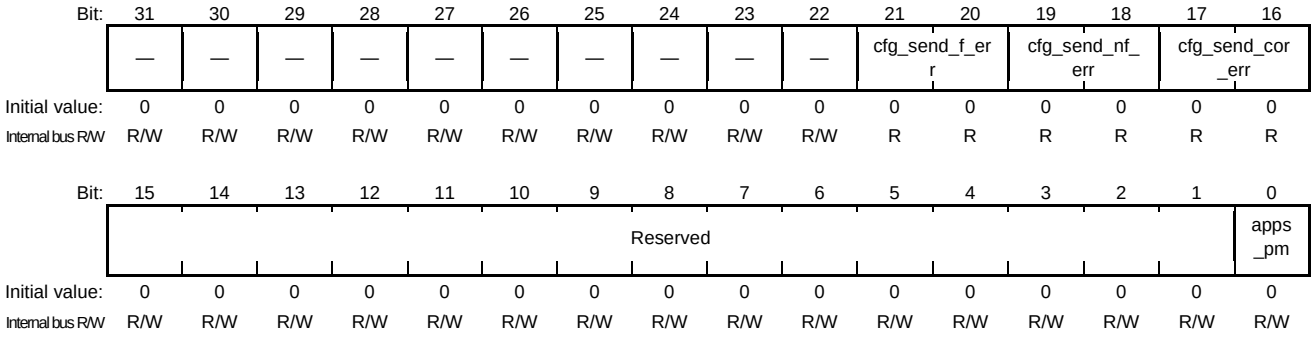
**Common to Header TYPE00/01**



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | msi_ctrl_io | All 0 | R | DSP AXI iMRM: Integrated MSI Reception Module GPIO.<br>General purpose IO driven by the MSI interrupt controller GPIO register |

### 104.2.345 PCIe Message Transmission (PCIEMSGTX)

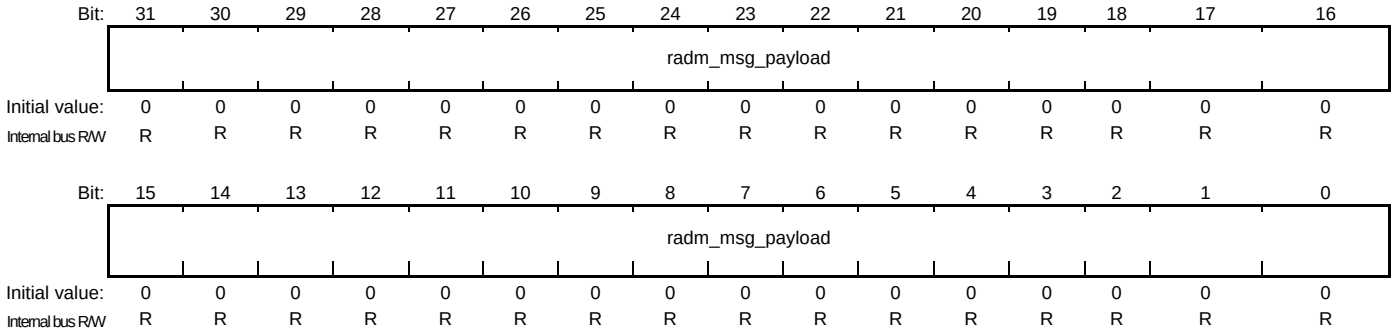
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 22 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 21 to 20 | cfg_send_f_err | H'0 | R | Sent Fatal Error.<br>Core has sent a message towards the Root Complex indicating that an Rx TLP that contained a fatal error, and that cannot be corrected, was received by the EndPoint |
| 19 to 18 | cfg_send_nf_err | H'0 | R | Sent Non-Fatal Error.<br>Core has sent a message towards the Root Complex indicating that an Rx TLP that contained an non-fatal error, and that cannot be corrected, was received by the EndPoint |
| 17 to 16 | cfg_send_cor_err | H'0 | R | Sent Correctable Error.<br>Core has sent a message towards the Root Complex indicating that an Rx TLP that contained an error, and that can be corrected, was received by the EndPoint |
| 15 to 1 | Reserved | H'0 | R/W | These bits are read as 0. The write value must be 0. |
| 0 | apps_pm_xmt_turnoff | H'0 | R/W | Request from your application to generate a PM_Turn_Off message.<br>You must assert this signal for one clock cycle. The controller does not<br>return an acknowledgment or grant signal. You must not pulse the same<br>signal again, until the previous message has been transmitted. |

### 104.2.346 PCIe Message Reception 0 (PCIEMSGRX0)

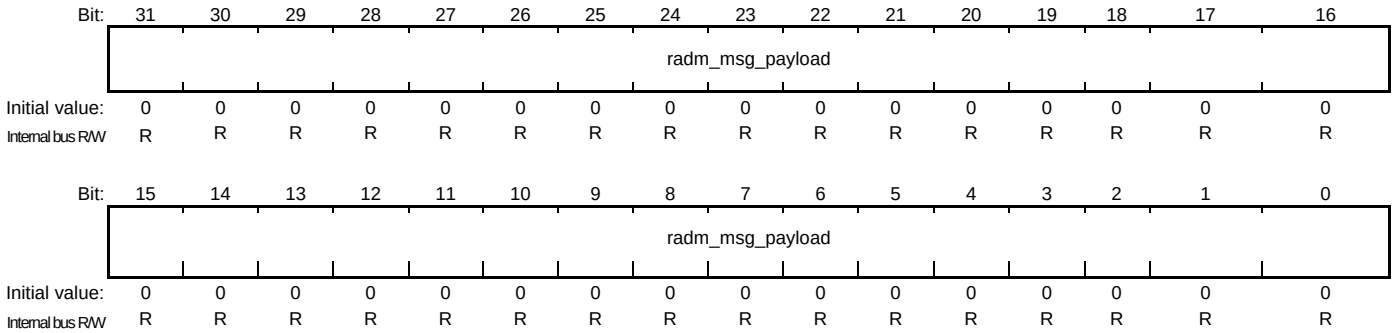
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | radm_msg_payload | All 0 | R | Lower bits [31:0].<br>Received message header information.<br>When a vendor-defined or ltr message is received (radm_vendor_msg=1 or radm_ltr_msg=1), the controller maps radm_msg_payload to the Rx TLP header dwords as follows:<br>■ [31:0] = bytes 12-15 (4th dword), where [7:0] =byte 15<br>■ [63:32] = bytes 8-11 (3rd dword)<br>When any other type of message is received, the controller maps<br>radm_msg_payload to the Rx TLP header dwords as follows:<br>■ [31:0] = DW Data of Message<br>■ [63:32] = '0' |

### 104.2.347 PCIe Message Reception 1 (PCIEMSGRX1)

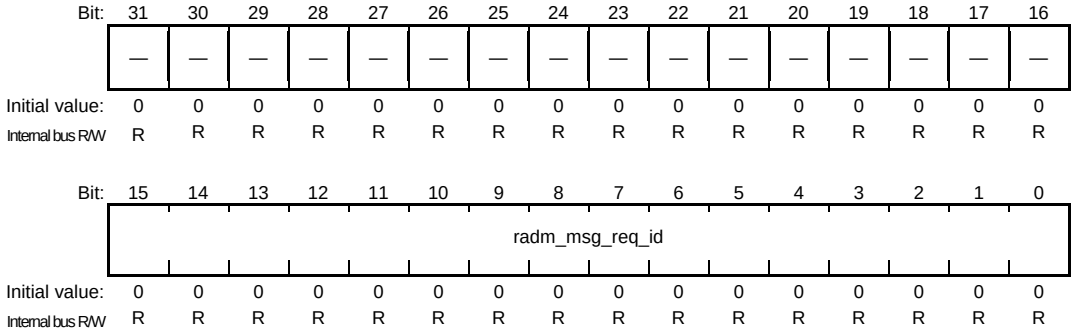
**Common to Header TYPE00/01**



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | radm_msg_payload | All 0 | R | Upper bits [63:32].<br>Received message header information.Received message header information.<br>When a vendor-defined or ltr message is received<br>(radm_vendor_msg=1 or radm_ltr_msg=1), the controller maps<br>radm_msg_payload to the Rx TLP header dwords as follows:<br>■ [31:0] = bytes 12-15 (4th dword), where [7:0] =byte 15<br>■ [63:32] = bytes 8-11 (3rd dword)<br>When any other type of message is received, the controller maps<br>radm_msg_payload to the Rx TLP header dwords as follows:<br>■ [31:0] = DW Data of Message<br>■ [63:32] = '0' |

### 104.2.348 PCIe Message Reception 2 (PCIEMSGRX2)

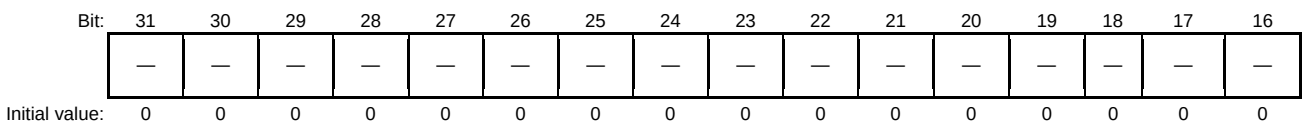
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 15 to 0 | radm_msg_req_id | H'0 | R | The requester ID of the received Message.<br>■ [15:8]: Bus number<br>■ [7:3]: Device number<br>■ [2:0]: Function number |

### 104.2.349 PCIe Message Reception Status (PCIEMSGRCPSTS)

**Common to Header TYPE00/01**



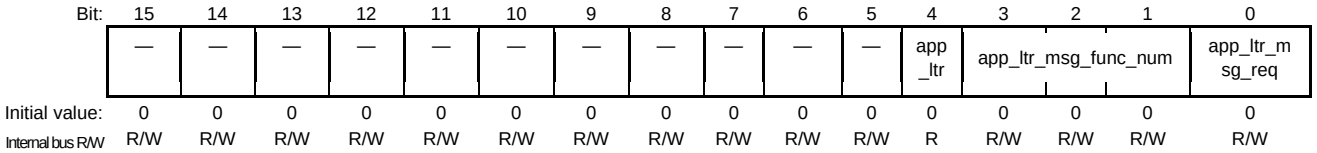
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | — | — | — | — | radm_msg | radm_ven | radm_pm | radm_pm | radm_pm | radm_fatal | radm_ | radm_correc | radm_ms |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 9 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 8 | radm_msg_unlock | H'0 | R | Indicates that the controller received an Unlock message.<br>When two messages of the same type are received in the same clock cycle (back-to-back), then no separate indication is given for the second message. |
| 7 | radm_vendor_msg | H'0 | R | Indicates the controller received a vendor-defined message.<br>The controller makes the message header available the radm_msg_payload output. When FX_TLP > 1 and when two messages<br>of the same type are received in the same clock cycle (back-to-back),<br>then both bits are asserted. |
| 6 | radm_pm_turnoff | H'0 | R | Indicates that the controller received a PME Turnoff message.<br>When two messages of the same type are received in the same clock cycle (back-to-back), then no indication is given for the second message. |
| 5 | radm_pm_to_ack | H'0 | R | Indicates that the controller received a PME_TO_Ack message. Upstream port: Reserved. |
| 4 | radm_pm_pme | H'0 | R | Indicates that the controller received a PM_PME message |
| 3 | radm_fatal_err | H'0 | R | Indicates that the controller received an ERR_FATAL message |
| 2 | radm_nonfatal_err | H'0 | R | Indicates that the controller received an ERR_NONFATAL message |
| 1 | radm_correctable_err | H'0 | R | Indicates that the controller received an ERR_COR message.<br>The controller makes the message header available the radm_msg_payload output. |
| 0 | radm_msg_ltr | H'0 | R | Indicates that the controller received an LTR message.<br>The controller makes the message header available the radm_msg_payload output. It is also available the app_ltr_latency<br>output. When two messages of the same type are received in the same clock cycle (back-to-back), then no separate indication is given for the second message. |

### 104.2.350 PCIe LTR Message Control 0 (PCIELTRMSGCTRL0)

**Common to Header TYPE00/01**

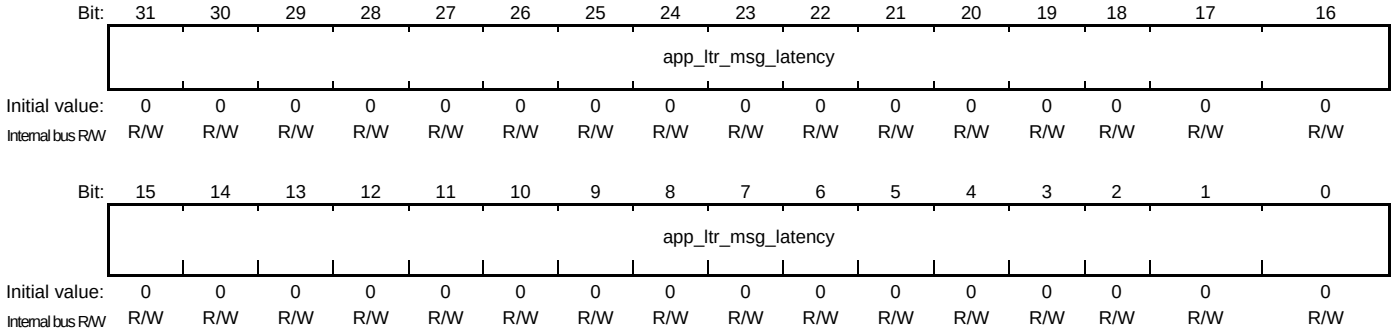
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 5 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 4 | app_ltr_msg_grant | H'0 | R | Indicates that the controller has accepted your request to send an LTR message. |
| 3 to 1 | app_ltr_msg_func_num | H'0 | R/W | Function number in your application that is requesting to send an LTR<br>message.<br>Function numbering starts at '0'. |
| 0 | app_ltr_msg_req | H'0 | R/W | Indicates that your application is requesting to send an LTR message.<br>Once asserted, app_ltr_msg_req must remain asserted until the controller asserts app_ltr_msg_grant |

### 104.2.351 PCIe LTR Message Control 1 (PCIELTRMSGCTRL1)

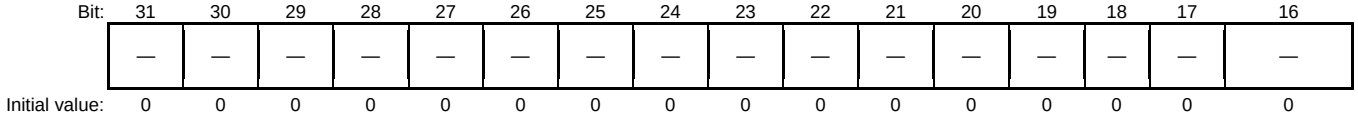
**Common to Header TYPE00/01**

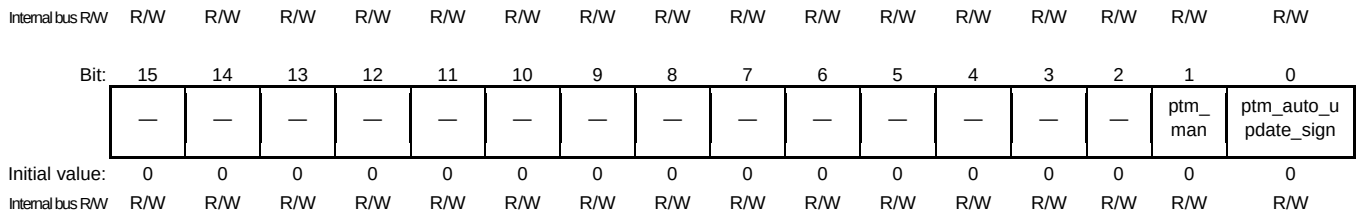



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | app_ltr_msg_latency | All 0 | R/W | LTR message that your application is requesting to send.<br>The message format is defined in the PCI Express Specification. |

### 104.2.352 PCIe PTM Control (PCIEPTMCTRL)

**Common to Header TYPE00/01**

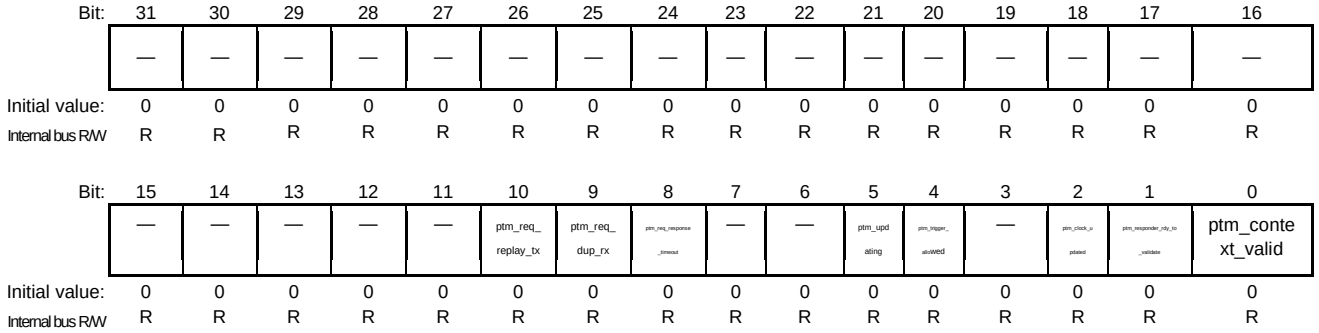





| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 2 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 1 | ptm_manual_update_pulse | H'0 | R/W | Indicates that the controller should update the PTM Requester Context and Clock now. |
| 0 | ptm_auto_update_signal | H'0 | R/W | Indicates that the controller should update the PTM Requester Context and Clock automatically every 10ms. |

### 104.2.353 PCIe PTM Status 0 (PCIEPTMSTS0)

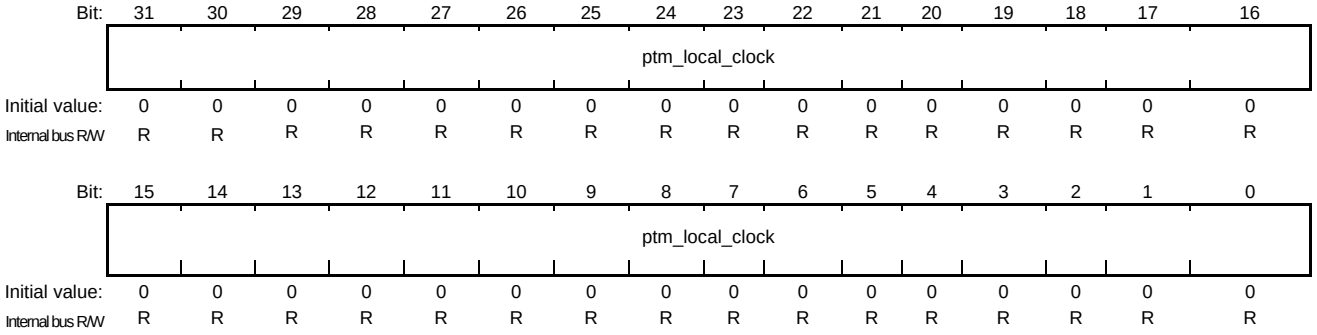
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 11 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 10 | ptm_req_replay_tx | H'0 | R | PTM Requester Replay Sent. |
| 9 | ptm_req_dup_rx | H'0 | R | PTM Requester Duplicate Received. |
| 8 | ptm_req_response_timeout | H'0 | R | PTM Requester Response Timeout. |
| 7 to 6 | Reserved | All 0 | R | These bits are read as 0. The write value is ignored. |
| 5 | ptm_updating | H'0 | R | Indicates that a PTM update is in progress |
| 4 | ptm_trigger_allowed | H'0 | R | Indicates that a PTM Requester manual update trigger is allowed |
| 3 | Reserved | H'0 | R | These bits are read as 0. The write value is ignored. |
| 2 | ptm_clock_updated | H'0 | R | Indicates that the controller has updated the Local Clock. |
| 1 | ptm_responder_rdy_to_validate | H'0 | R | PTM Responder Ready to Validate. |
| 0 | ptm_context_valid | H'0 | R | Context Valid |

### 104.2.354 PCIe PTM Status 1 (PCIEPTMSTS1)

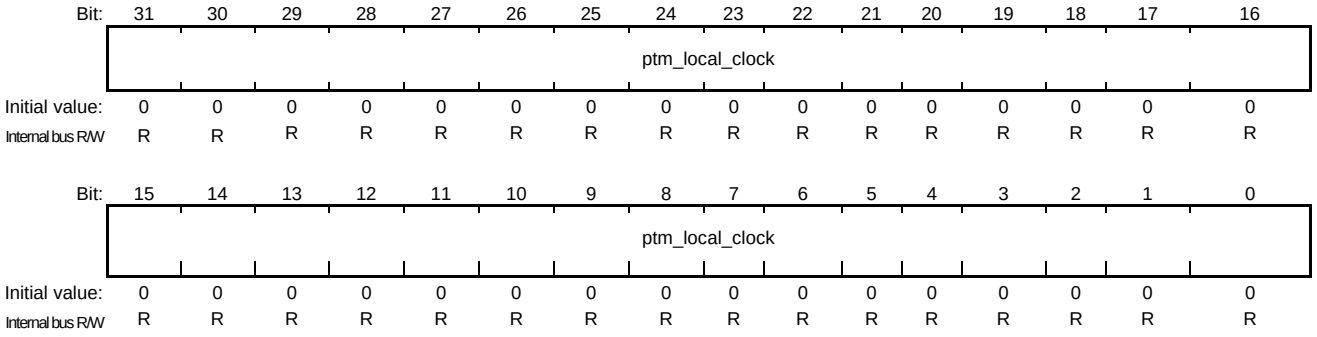
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ptm_local_clock | All 0 | R | Lower bits [31:0]<br>Local Clock value |

### 104.2.355 PCIe PTM Status 2 (PCIEPTMSTS2)

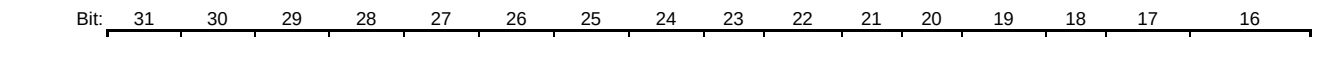
**Common to Header TYPE00/01**

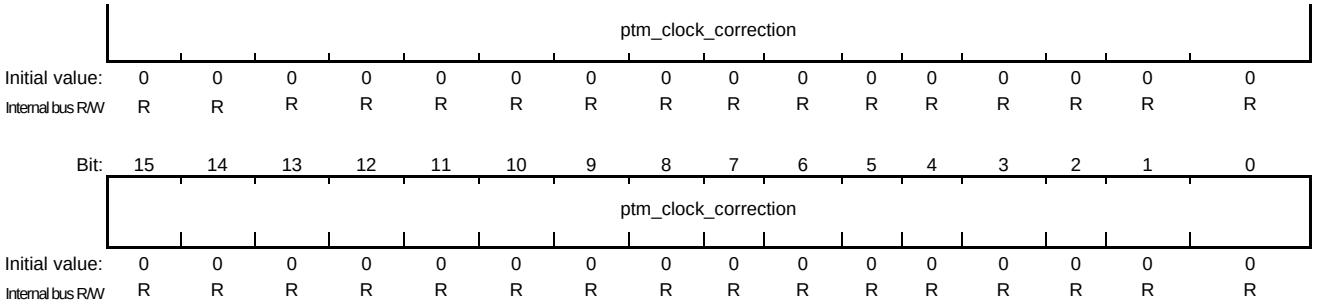



| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ptm_local_clock | All 0 | R | Upper bits [63:32]<br>Local Clock value |

### 104.2.356 PCIe PTM Status 3 (PCIEPTMSTS3)

**Common to Header TYPE00/01**

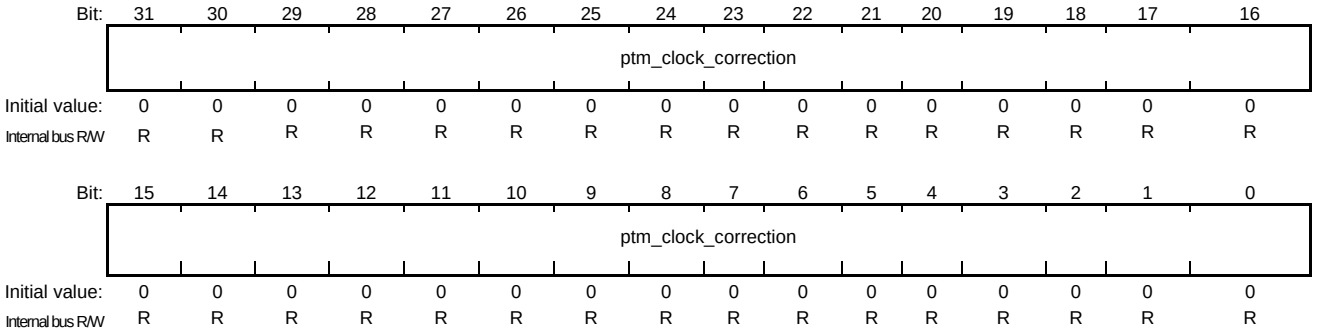





| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ptm_clock_correction | All 0 | R | Lower bits [31:0]<br>Amount by which Local Clock has been corrected. |

### 104.2.357 PCIe PTM Status 4 (PCIEPTMSTS4)

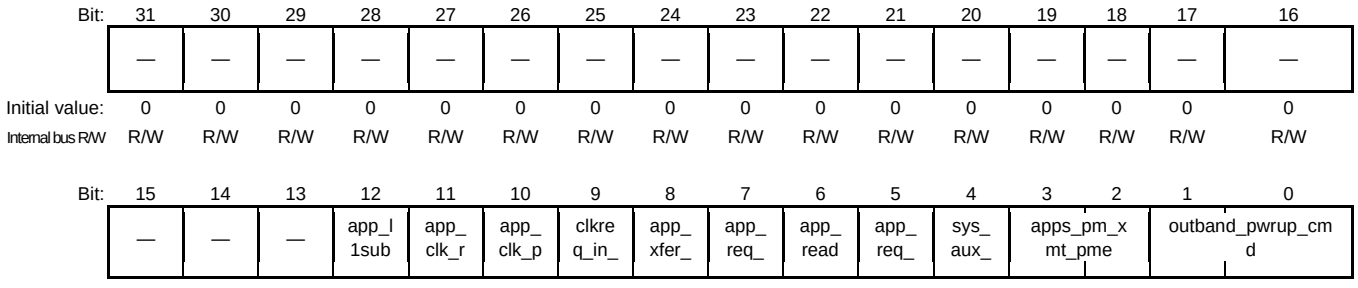
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ptm_clock_correction | All 0 | R | Upper bits [63:32]<br>Amount by which Local Clock has been corrected. |

### 104.2.358 PCIe Power Management Control (PCIEPWRMNGCTRL)

**Common to Header TYPE00/01**



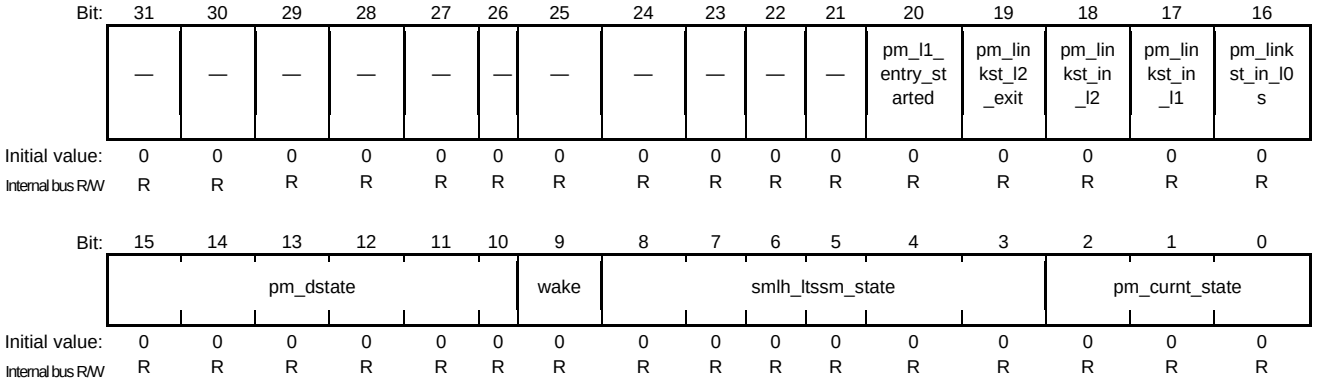
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 13 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 12 | app_l1sub_disable | H'0 | R/W | The application can set this input to 1'b1 to prevent entry to L1<br>Sub-states.<br>This pin is used to gate the L1 sub-state enable bits from the L1 PM Substates Control 1 Register |
| 11 | app_clk_req_n | H'0 | R/W | Indicates that the application logic is ready to have reference clock removed<br>In designs which support reference clock removal through either L1 PM Sub-states or L1 CPM, the application should set this signal to 1'b when it is ready to have reference clock removed. If the application does not want to remove reference clock it should set this signal to 1'b0. |
| 10 | app_clk_pm_en | H'0 | R/W | Clock PM feature enabled by application.<br>Used to inhibit the programming of the Clock PM in Link Control Register |
| 9 | clkreq_in_n | H'0 | R/W | Status of the CLKREQ# bidirectional CMOS board-level signal.<br>Used by the controller to determine when to enter and exit L1 Substates when using the CLKREQ#-based mechanism. |
| 8 | app_xfer_pending | H'0 | R/W | Indicates that your application has transfers pending and prevents the controller from entering L1.<br>This is a level signal used to inform the controller about the state of external queues and pipeline stages that contain transactions to be transmitted by the controller. The controller uses this information to determine when to enter/exit L1. When this signal is asserted, it indicates that there are transactions outside the controller that the controller needs to transmit. When de-asserted, it indicates that there are no transactions outside the controller. The controller responds to an assertion on this signal as follows:<br>■ Upstream Ports: Prevents generation of requests to enter L1. Triggers<br>exit if already in L1.<br>■ Downstream Ports: Triggers exit if already in L1.<br>You can instruct the controller to exit L1 by asserting either or both of<br>app_xfer_pending and app_req_exit_l1. The controller only samples<br>app_req_exit_l1 when the controller is already in the L1 state. |
| 7 | app_req_exit_l1 | H'0 | R/W | Application request to Exit L1.<br>Request from your application to exit L1. It is only effective when L1 is enabled |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 6 | app_ready_entr_l23 | H'0 | R/W | Application Ready to Enter L23.<br>Indication from your application that it is ready to enter the L23 state. The app_ready_entr_l23 signal is provided for applications that must control L23 entry (in case certain tasks must be performed before going into L23). The controller delays sending PM_Enter_L23 (in response to PM_Turn_Off) until this signal becomes active. When this signal has been asserted by the application, it must be kept asserted until L2 entry has completed. Hardwire to 1 for applications that do not require this feature.<br>Note: The controller ignores this input in RC mode. |
| 5 | app_req_entr_l1 | H'0 | R/W | Application request to Enter L1 ASPM state.<br>The app_req_entr_l1 signal is for use by applications that need to control L1 entry instead of using the L1 entry timer as defined in the PCI Express Specification. It is only effective when L1 is enabled. The controller latches this request when in L0 or L0s; to be acted upon later.<br>Note: The controller ignores this input in RC mode. |
| 4 | sys_aux_pwr_det | H'1 | R/W | Auxiliary Power Detected.<br>Used to report to the host software that auxiliary power (Vaux) is present.<br>For a configuration where the auxiliary power is not available (sys_aux_pwr_det =0), the controller cannot exit from L23 Ready with beacon or PERST# de-assertion when main power is not removed. |
| 3 to 2 | apps_pm_xmt_pme | H'0 | R/W | Wake Up.<br>If PME is enabled and PME support is configured for current<br>PMCSR D-state asserting this signal will cause the controller to wake<br>from either L1 or L2 state. When the controller has transitioned back to<br>the L0 state it will transmit a PME message and set the PME_Status.<br>Upon receiving the PME message the root complex should clear the<br>PME_Status and change the D-state back to D0. |
| 1 to 0 | outband_pwrup_cmd | H'0 | R/W | Wake Up.<br>If PME is enabled and PME support is configured for current PMCSR D-state asserting this signal will cause the controller to wake<br>from either L1 or L2 state. When the controller has transitioned back to<br>the L0 state it will transmit a PME message and set the PME_Status.<br>There is a separate outband_pwrup_cmd input bit for each function in<br>your controller configuration. This port is functionally identical to apps_pm_xmt_pme. Upon receiving the PME message the root complex should clear the PME_Status and change the D-state back to D0. |

### 104.2.359 PCIe Power Management Status 0 (PCIEPWRMNGSTS0)

**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 21 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 20 | pm_l1_entry_started | H'0 | R | L1 entry process is in progress |
| 19 | pm_linkst_l2_exit | H'0 | R | Power management is exiting L2 state |
| 18 | pm_linkst_in_l2 | H'0 | R | Power management is in L2 state |
| 17 | pm_linkst_in_l1 | H'0 | R | Power management is in L1 state |
| 16 | pm_linkst_in_l0s | H'0 | R | Power management is in L0s state |
| 15 to 10 | pm_dstate | H'0 | R | The current power management D-state of the function.<br>■ 000b: D0<br>■ 001b: D1<br>■ 010b: D2 (not support D2 Power Management State, Read/write value B'10 always read as 0. The write value should always be 0.)<br>■ 011b: D3<br>■ 100b: Uninitialized<br>■ Other values: Not applicable<br>There are 3 bits of pm_dstate for each configured function. |
| 9 | wake | H'0 | R | Wake Up.<br>Wake up from power management unit. The controller generates wake to request the system to restore power and clock when a wakeup event has been detected such as apps_pm_xmt_pme, apps_pm_vf_xmt_pme, or outband_pwrup_cmd. The wake signal is an active high signal and its rising edge should be detected to drive the WAKE# the connector. Assertion of wake could be for a single clock cycle or multiple clock cycles. Not used in RC mode. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 8 to 3 | smlh_ltssm_state | H'0 | R | Current state of the LTSSM.<br>Encoding is defined as follows:<br>■ 6'h00: S_DETECT_QUIET<br>■ 6'h01: S_DETECT_ACT<br>■ 6'h02: S_POLL_ACTIVE<br>■ 6'h03: S_POLL_COMPLIANCE<br>■ 6'h04: S_POLL_CONFIG<br>■ 6'h05: S_PRE_DETECT_QUIET<br>■ 6'h06: S_DETECT_WAIT<br>■ 6'h07: S_CFG_LINKWD_START<br>■ 6'h08: S_CFG_LINKWD_ACEPT<br>■ 6'h09: S_CFG_LANENUM_WAI<br>■ 6'h0A: S_CFG_LANENUM_ACEPT<br>■ 6'h0B: S_CFG_COMPLETE<br>■ 6'h0C: S_CFG_IDLE<br>■ 6'h0D: S_RCVRY_LOCK<br>■ 6'h0E: S_RCVRY_SPEED<br>■ 6'h0F: S_RCVRY_RCVRCFG<br>■ 6'h10: S_RCVRY_IDLE<br>■ 6'h11: S_L0<br>■ 6'h12: S_L0S<br>■ 6'h13: S_L123_SEND_EIDLE<br>■ 6'h14: S_L1_IDLE<br>■ 6'h15: S_L2_IDLE<br>■ 6'h16: S_L2_WAKE<br>■ 6'h17: S_DISABLED_ENTRY<br>■ 6'h18: S_DISABLED_IDLE<br>■ 6'h19: S_DISABLED<br>■ 6'h1A: S_LPBK_ENTRY<br>■ 6'h1B: S_LPBK_ACTIVE<br>■ 6'h1C: S_LPBK_EXIT<br>■ 6'h1D: S_LPBK_EXIT_TIMEOUT<br>■ 6'h1E: S_HOT_RESET_ENTRY<br>■ 6'h1F: S_HOT_RESET<br>■ 6'h20: S_RCVRY_EQ0<br>■ 6'h21: S_RCVRY_EQ1<br>■ 6'h22: S_RCVRY_EQ2<br>■ 6'h23: S_RCVRY_EQ3 |
| 2 to 0 | pm_curnt_state | H'0 | R | Indicates the current power state.<br>The pm_curnt_state output is intended for debugging purposes, not for system operation. |

### 104.2.360 PCIe Power Management Status 1 (PCIEPWRMNGSTS1)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | cfg_l1sub |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pm_linkst_in_l1 | pm_status | | pm_l1sub_state | | | pm_slave_state | | | | | pm_master_state | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 17 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 16 | cfg_l1sub_en | H'0 | R | Indicates that any of the L1 Substates are enabled in the L1 Substates Control 1 Register.<br>Could be used by your application in a downstream port to determine when not to drive CLKREQ# such as when L1 Substates are not enabled. |
| 15 | pm_linkst_in_l1sub | H'0 | R | Power management is in L1 substate.<br>Indicates when the link has entered L1 substates.For L1.2 this signal is deasserted at the end of the L1.2.Exit state, after the t_power_on constraint has been satisfied. External logic can use the transition high to low on this signal to initiate REFCLK restore. |
| 14 to 13 | pm_status | H'0 | R | PME Status bit from the PMCSR.<br>There is 1 bit of pm_status for each configured function. |
| 12 to 10 | pm_l1sub_state | H'0 | R | Power management L1 sub-states FSM state |
| 9 to 5 | pm_slave_state | H'0 | R | Power management slave FSM state |
| 4 to 0 | pm_master_state | H'0 | R | Power management master FSM state |

### 104.2.361 PCIe Error Status 0 (PCIEERRSTS0)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | if_timeout | cfg_sys_err_rc | | app_parity_errs | | | cfg_safety | cfg_safety | surprise_down_err | | cdm_ | cdm_ |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 12 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 11 | if_timeout_status | H'0 | R | Error Status if_timeout_status |
| 10 to 9 | cfg_sys_err_rc | H'0 | R | Error Status cfg_sys_err_rc |
| 8 to 6 | app_parity_errs | H'0 | R | Error Status app_parity_errs |
| 5 | cfg_safety_uncorr | H'0 | R | Error Status cfg_safety_uncorr |
| 4 | cfg_safety_corr | H'0 | R | Error Status cfg_safety_corr |
| 3 to 2 | surprise_down_err | H'0 | R | Error Status surprise_down_err |
| 1 | cdm_reg_chk_cmp_err | H'0 | R | Error Status cdm_reg_chk_cmp_err |
| 0 | cdm_reg_chk_logic_err | H'0 | R | Error Status cdm_reg_chk_logic_err |

### 104.2.362 PCIe Interrupt Status 0 (PCIEINTSTS0)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | cfg_aer_rc_err_msi | | cfg_aer_rc_err_int | | msi_ctrl_i | cfg_2nd_r | cfg_link_e | cfg_bw_m | cfg_bw_m | cfg_link_a | cfg_link_a | hp_msi | | hp_int | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

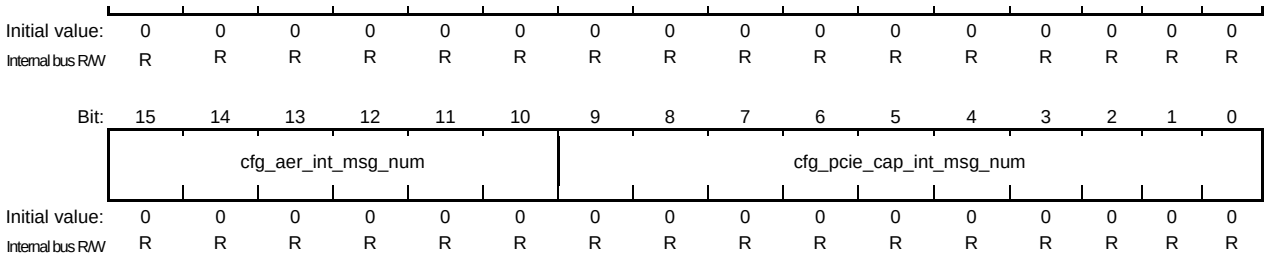
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | hp_pme | | cfg_pme_msi | | cfg_pme_int | | radm_ven | radm_qov | smlh_link | rdlh_link_u | rbar_ctrl_update | | cfg_mem_space_en | | cfg_bus_master_en | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 30 to 29 | cfg_aer_rc_err_msi | H'0 | R | Interrupt Status cfg_aer_rc_err_msi |
| 28 to 27 | cfg_aer_rc_err_int | H'0 | R | Interrupt Status cfg_aer_rc_err_int |
| 26 | msi_ctrl_int | H'0 | R | Interrupt Status msi_ctrl_int |
| 25 | cfg_2nd_reset | H'0 | R | Interrupt Status cfg_2nd_reset |
| 24 | cfg_link_eq_req_int | H'0 | R | Interrupt Status cfg_link_eq_req_int |
| 23 | cfg_bw_mgt_msi | H'0 | R | Interrupt Status cfg_bw_mgt_msi |
| 22 | cfg_bw_mgt_int | H'0 | R | Interrupt Status cfg_bw_mgt_int |
| 21 | cfg_link_auto_bw_msi | H'0 | R | Interrupt Status cfg_link_auto_bw_msi |
| 20 | cfg_link_auto_bw_int | H'0 | R | Interrupt Status cfg_link_auto_bw_int |
| 19 to 18 | hp_msi | H'0 | R | Interrupt Status hp_msi |
| 17 to 16 | hp_int | H'0 | R | Interrupt Status hp_int |
| 15 to 14 | hp_pme | H'0 | R | Interrupt Status hp_pme |
| 13 to 12 | cfg_pme_msi | H'0 | R | Interrupt Status cfg_pme_msi |
| 11 to 10 | cfg_pme_int | H'0 | R | Interrupt Status cfg_pme_int |
| 9 | radm_vendor_msg | H'0 | R | Interrupt Status radm_vendor_msg |
| 8 | radm_qoverflow | H'0 | R | Interrupt Status radm_qoverflow |
| 7 | smlh_link_up | H'0 | R | Interrupt Status smlh_link_up |
| 6 | rdlh_link_up | H'0 | R | Interrupt Status rdlh_link_up |
| 5 to 4 | rbar_ctrl_update | H'0 | R | Interrupt Status rbar_ctrl_update |
| 3 to 2 | cfg_mem_space_en | H'0 | R | Interrupt Status cfg_mem_space_en |
| 1 to 0 | cfg_bus_master_en | H'0 | R | Interrupt Status cfg_bus_master_en |

### 104.2.363 PCIe Interrupt Status 1 (PCIEINTSTS1)

**Common to Header TYPE00/01**

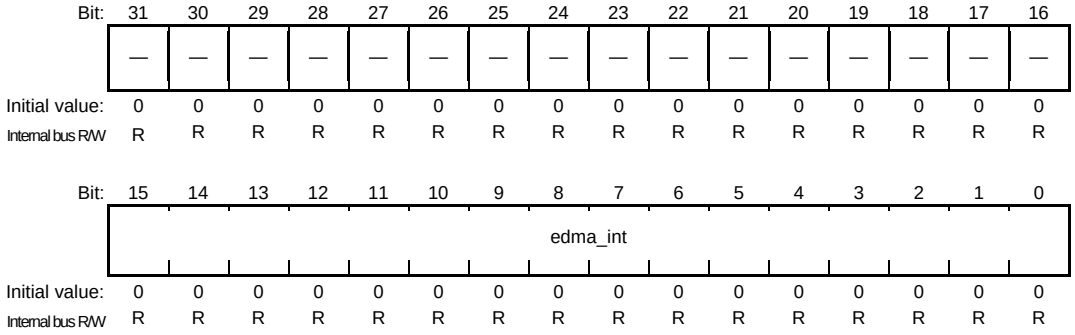
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | lbc_ext_cs | | cfg_aer_int_msg_num | | | |




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 22 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 21 to 20 | lbc_ext_cs | H'0 | R | Interrupt Status lbc_ext_cs |
| 19 to 10 | cfg_aer_int_msg_num | H'0 | R | Interrupt Status cfg_aer_int_msg_num |
| 9 to 0 | cfg_pcie_cap_int_msg_num | H'0 | R | Interrupt Status cfg_pcie_cap_int_msg_num |

### 104.2.364 PCIe DMA Interrupt Status (PCIEDMAINTSTS)

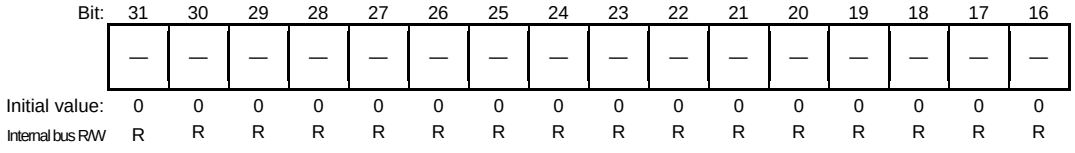
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 15 to 0 | edma_int | H'0 | R | Interrupt Status edma_int.<br>[15:8] : read channel 7-0<br>[7:0] : write channel 7-0 |

### 104.2.365 PCIe INTx Interrupt Status (PCIEINTXSTS)

**Common to Header TYPE00/01**



| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | deassert_int | deasse | deasse | deasse | assert | assert | assert | assert | radm_i | radm_i | radm_i | radm_i | radm_i | radm_i | radm_i | radm_i |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 15 | deassert_intd_grt | H'0 | R | Interrupt Status deassert_intd_grt |
| 14 | deassert_intc_grt | H'0 | R | Interrupt Status deassert_intc_grt |
| 13 | deassert_intb_grt | H'0 | R | Interrupt Status deassert_intb_grt |
| 12 | deassert_inta_grt | H'0 | R | Interrupt Status deassert_inta_grt |
| 11 | assert_intd_grt | H'0 | R | Interrupt Status assert_intd_grt |
| 10 | assert_intc_grt | H'0 | R | Interrupt Status assert_intc_grt |
| 9 | assert_intb_grt | H'0 | R | Interrupt Status assert_intb_grt |
| 8 | assert_inta_grt | H'0 | R | Interrupt Status assert_inta_grt |
| 7 | radm_intd_deasserted | H'0 | R | Interrupt Status radm_intd_deasserted |
| 6 | radm_intc_deasserted | H'0 | R | Interrupt Status radm_intc_deasserted |
| 5 | radm_intb_deasserted | H'0 | R | Interrupt Status radm_intb_deasserted |
| 4 | radm_inta_deasserted | H'0 | R | Interrupt Status radm_inta_deasserted |
| 3 | radm_intd_asserted | H'0 | R | Interrupt Status radm_intd_asserted |
| 2 | radm_intc_asserted | H'0 | R | Interrupt Status radm_intc_asserted |
| 1 | radm_intb_asserted | H'0 | R | Interrupt Status radm_intb_asserted |
| 0 | radm_inta_asserted | H'0 | R | Interrupt Status radm_inta_asserted |

### 104.2.366 PCIe RAS D.E.S Control (PCIERASDESCTRL)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

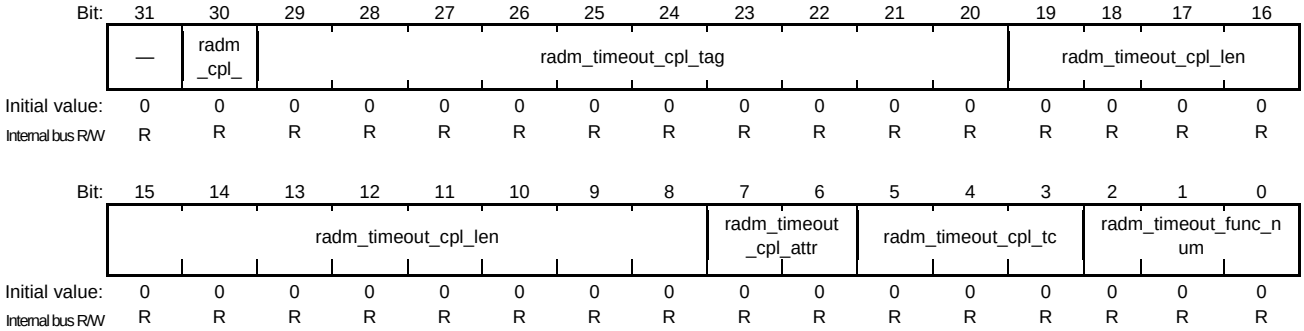
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | cdm_reg_ch | cdm_reg |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 2 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 1 | cdm_reg_chk_cmplt | H'0 | R | Signal to indicate that a Register Checking Sequence has Completed. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 0 | cdm_reg_chk_test_en | H'0 | R/W | Signal to enter test mode in the Register Checking Logic. u_cdm_b will not be written to when this signal is high. Allows a value to be written to u_cdm to compare against the previous value in u_cdm_b |

### 104.2.367 PCIe Receive CPL Timeout Status (PCIERCVCPLTOSTS)

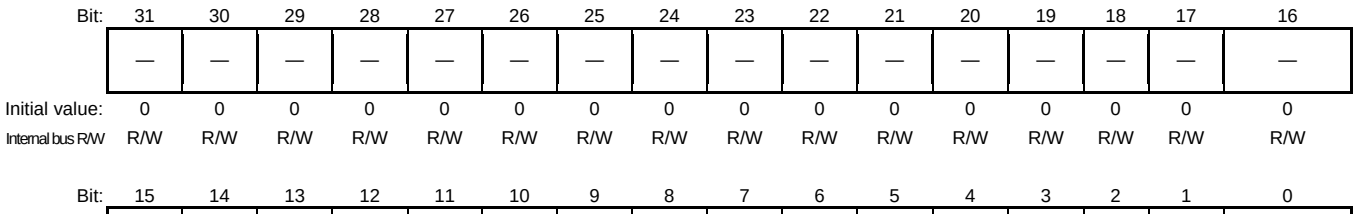
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 30 | radm_cpl_timeout | H'0 | R | Indicates that the completion TLP for a request has not been received within the expected time window. |
| 29 to 20 | radm_timeout_cpl_tag | H'0 | R | The Tag field of the timed out completion |
| 19 to 8 | radm_timeout_cpl_len | H'0 | R | Length (in bytes) of the timed out completion.<br>For a split completion, it indicates the number of bytes remaining to be delivered when the completion timed out. |
| 7 to 6 | radm_timeout_cpl_attr | H'0 | R | The Attributes field of the timed out completion |
| 5 to 3 | radm_timeout_cpl_tc | H'0 | R | The Traffic Class of the timed out completion |
| 2 to 0 | radm_timeout_func_num | H'0 | R | The function Number of the timed out completion. Function numbering starts at '0'. |

### 104.2.368 PCIe Transmit Control 0 (PCIETXCNTRL0)

**Common to Header TYPE00/01**



| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | app_hdr_valid |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 1 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 0 | app_hdr_valid | H'0 | R/W | Indicates that the data app_hdr_log, app_err_bus, app_err_func_num, and app_tlp_prfx_log is valid. |

### 104.2.369 PCIe Transmit Control 1 (PCIETXCNTRL1)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | app_err_func_num |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

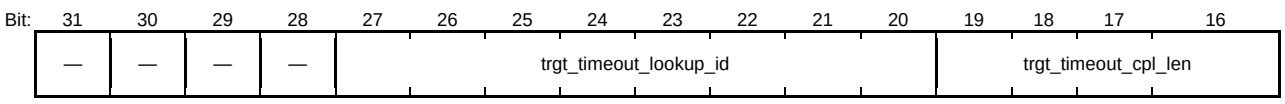
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | app_err_func_num | | app_err_a | app_err_bus | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 17 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 16 to 14 | app_err_func_num | H'0 | R/W | The number of the function that is reporting the error indicated app_err_bus, valid when app_hdr_valid is asserted.<br>Correctable and Uncorrected Internal errors (app_err_bus[10:9]) are not function specific, and are recorded for all physical functions, regardless of the value this bus. Function numbering starts at '0'. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 13 | app_err_advisory | H'0 | R/W | Indicates that your application error is an advisory error..<br>Your application should assert app_err_advisory under either of the following conditions:<br>■ The controller is configured to mask completion timeout errors, your<br>application is reporting a completion timeout error app_err_bus, and<br>your application intends to resend the request. In such cases the<br>error is an advisory error, as described in PCI Express Specification.<br>When your application does not intend to resend the request, then<br>your application must keep app_err_advisory de-asserted when<br>reporting a completion timeout error.<br>■ The controller is configured to forward poisoned TLPs to your application<br>and your application is going to treat the poisoned TLP as a normal TLP, as described in PCI Express Specification. Upon receipt of a poisoned TLP, your application must report the error app_err_bus, and either assert app_err_advisory (to indicate an advisory error) or de-assert app_err_advisory (to indicate that your application is dropping the TLP). |
| 12 to 0 | app_err_bus | H'0 | R/W | The type of error that your application detected.<br>The controller combines the values the app_err_bus bits with the internally-detected error signals to set the corresponding bit in the Uncorrectable or Correctable Error<br>Status Registers:<br>■ app_err_bus[0]: Malformed TLP<br>■ app_err_bus[1]: Receiver Overflow<br>■ app_err_bus[2]: Unexpected completion<br>■ app_err_bus[3]: Completer abort<br>■ app_err_bus[4]: Completion Timeout<br>■ app_err_bus[5]: Unsupported request<br>■ app_err_bus[6]: ECRC Check Failed<br>■ app_err_bus[7]: Poisoned TLP received<br>■ app_err_bus[8]: AtomicOp Egress Blocked: only valid when CX_ATOMIC_ROUTING_EN =1<br>■ app_err_bus[9]: Uncorrectable Internal Error<br>■ app_err_bus[10]: Corrected Internal Error<br>■ app_err_bus[11]: TLP Prefix Blocked Error Status: only valid for RC<br>and when CX_NPRFX >0<br>■ app_err_bus[12]: ACS Violation: only valid when CX_ACS_ENABLE<br>=1 |

### 104.2.370 PCIe Target Completion LUT 0 (PCIETRGTCMPLUT0)

**Common to Header TYPE00/01**



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | trgt_timeout_cpl_len | | | | | | | | trgt_timeout_cpl_attr | | trgt_timeout_cpl_tc | | | trgt_timeout_cpl_func_num | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 28 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 27 to 20 | trgt_timeout_lookup_id | H'0 | R | The target completion LUT lookup ID of the timed out completion. |
| 19 to 8 | trgt_timeout_cpl_len | H'0 | R | The Length of the timed out completion |
| 7 to 6 | trgt_timeout_cpl_attr | H'0 | R | The Attributes value of the timed out completion.<br>■ Width is three bits when CX_IDO_ENABLE =1<br>■ Width is two bits when CX_IDO_ENABLE =0 |
| 5 to 3 | trgt_timeout_cpl_tc | H'0 | R | The TC of the timed out completion. |
| 2 to 0 | trgt_timeout_cpl_func_num | H'0 | R | The function number of the timed out completion. Function numbering starts at '0'. |

### 104.2.371 PCIe Target Completion LUT 1 (PCIETRGTCMPLUT1)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | trgt_looku | trgt_cpl_ti | trgt_lookup_id | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 10 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 9 | trgt_lookup_empty | H'0 | R | When this signal is asserted with radm_trgt1_hv, it indicates that the target completion LUT is not full.<br>The application can use this signal to determine a back-pressure mechanism to the controller so that there is not an overflow condition when the number of outstanding non-posted transactions received is more than expected. Your application logic must drive trgt1_radm_halt with the logical OR of your application halt signal and the inverse of this output. |
| 8 | trgt_cpl_timeout | H'0 | R | Indicates that the application has not generated a completion for an incoming request within the required time interval.<br>Information about the timed out completion is available the trgt_timeout_* outputs listed later. When a completion timeout occurs, the controller does not clear the corresponding entry from the completion lookup table. The default completion timeout value is approximately 10 ms. |
| 7 to 0 | trgt_lookup_id | H'0 | R | The target completion LUT lookup ID for the incoming request TLP.<br>When using the optional target completion lookup table feature, the<br>application must save the lookup ID and assert the same lookup ID<br>client0/1/2_cpl_lookup_id when generating a completion for the request. |

### 104.2.372 PCIe Mode Information (PCIEMODEINFO)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

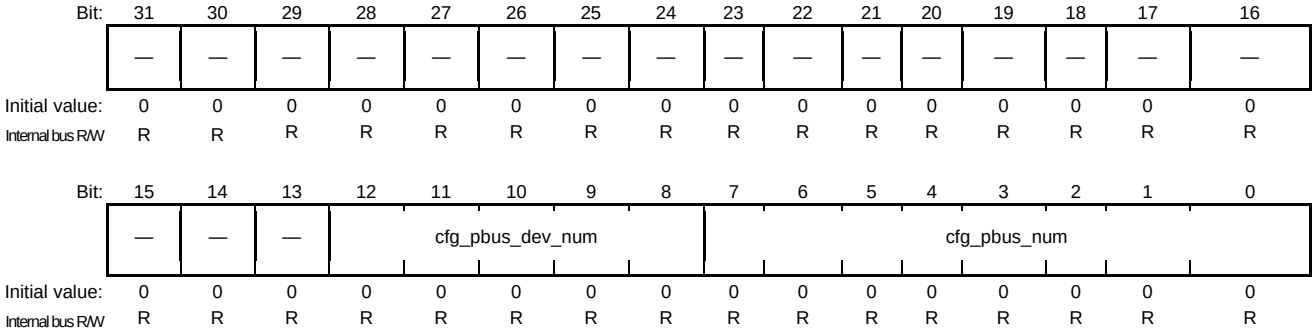
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | slv_rasdp_e | mstr_rasdp_err_mod |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 2 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 1 | slv_rasdp_err_mode | H'0 | R | Indication (slv_aclk version) from the controller that it has entered RASDP error mode.<br>The controller enters RASDP error mode (if the ERROR_MODE_EN register field =1) upon detection of the first uncorrectable error. During this mode:<br>■ Core sets mstr_rasdp_error_mode=1<br>■ Core sets slv_rasdp_error_mode=1<br>■ Rx TLPs that are forwarded to your application are not guaranteed<br>to be correct; you must discard them. |
| 0 | mstr_rasdp_err_mode | H'0 | R | Indication (mstr_aclk version) from the controller that it has entered<br>RASDP error mode.<br>The controller enters RASDP error mode (if the ERROR_MODE_EN register field =1) upon detection of the first uncorrectable error. During this mode:<br>■ Core sets slv_rasdp_err_mode=1<br>■ Core sets mstr_rasdp_err_mode=1<br>■ Rx TLPs that are forwarded to your application are not guaranteed<br>to be correct; you must discard them. |

### 104.2.373 PCIe Configuration Information 0 (PCIECNFGINFO0)

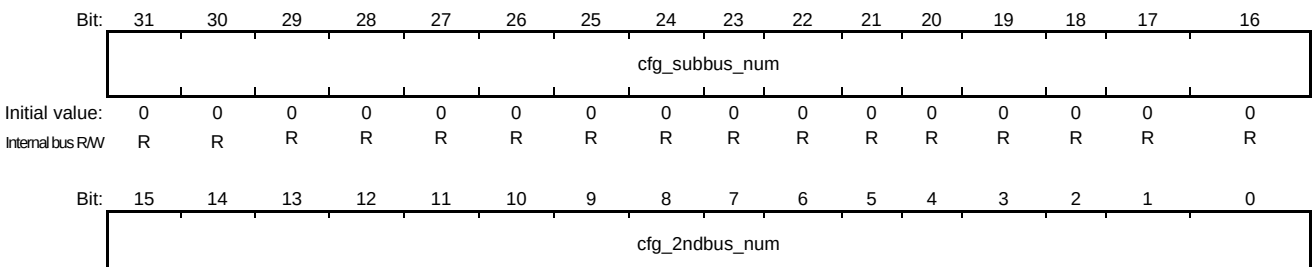
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 13 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 12 to 8 | cfg_pbus_dev_num | H'0 | R | The device number assigned to the function.<br>The number of bits depends the value of MULTI_DEVICE_AND_BUS_PER_FUNC_EN:<br>■ If MULTI_DEVICE_AND_BUS_PER_FUNC_EN =0, there are five<br>bits of cfg_pbus_dev_num ([4:0]), regardless of the number of functions.<br>■ If MULTI_DEVICE_AND_BUS_PER_FUNC_EN =1, there are five<br>bits of cfg_pbus_dev_num for each configured function. |
| 7 to 0 | cfg_pbus_num | H'0 | R | The primary bus number assigned to the function.<br>The number of bits depends the value of MULTI_DEVICE_AND_BUS_PER_FUNC_EN:<br>■ If MULTI_DEVICE_AND_BUS_PER_FUNC_EN =0, there are eight<br>bits of cfg_pbus_num ([7:0]), regardless of the number of functions.<br>■ If MULTI_DEVICE_AND_BUS_PER_FUNC_EN =1, there are eight<br>bits of cfg_pbus_num for each configured function. |

### 104.2.374 PCIe Configuration Information 1 (PCIECNFGINFO1)

**Common to Header TYPE00/01**



| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | cfg_subbus_num | H'0 | R | Configured Subordinate Bus Number.<br>When the switch controller receives a configuration request from the upstream port, it must see the secondary and subordinate bus number fields from the downstream ports to route the request. |
| 15 to 0 | cfg_2ndbus_num | H'0 | R | Configured Secondary Bus Number.<br>The secondary bus number assigned to the device. When the host software detects the upstream switch port, it assigns a secondary bus number. This bus number is then used in configuration requests to detect the downstream ports. The switch controller must see this so that it can route configuration requests. |

### 104.2.375 PCIe Margining Control (PCIEMARGINGCTRL)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | app_marg | app_margining_ready |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 2 | Reserved | All 0 | R/W | These bits are read as 0. The write value must be 0. |
| 1 | app_margining_software_ready | H'0 | R/W | Margining Software Ready.<br>Indicates that the required margining software has performed the required initialization when margining is partially implemented using Device Driver software. This signal is reflected in Margining Software Ready field of Margining Port Status register. If MARGINING_USES_DRIVER_SOFTWARE field of Margining Port Capabilities Register is '0', you can set this signal to a fixed value of '0'. This signal should be tied to '0'. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 0 | app_margining_ready | H'0 | R/W | Margining Ready.<br>Indicates when the PHY or Margining software is ready to accept margining commands. This signal is reflected in Margining Ready field of Margining Port Status register. If the PHY can accept margining commands when the current link speed is 16GT/s, you can set this signal to a fixed value of '1'. This signal should be tied to '1'. |

### 104.2.376 PCIe Reset Status Enable Register (PCIERSTSTSEN)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | | | | | | | | | | cfg_flr_pf_active | | — | smlh_req_rst | link_req_rst_not | cfg_2nd_reset |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 6 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 5 to 4 | cfg_flr_pf_active | H'0 | R/W | Status cfg_flr_pf_active enable<br>0: Disable<br>1: Enable |
| 3 | Reserved | 0 | R | These bits are read as 0. The write value must be 0. |
| 2 | smlh_req_rst_not | 0 | R/W | Status smlh_req_rst_not enable<br>0: Disable<br>1: Enable |
| 1 | link_req_rst_not | 0 | R/W | Status link_req_rst_not enable<br>0: Disable<br>1: Enable |
| 0 | cfg_2nd_reset | 0 | R/W | Status cfg_2nd_reset enable<br>0: Disable<br>1: Enable |

### 104.2.377 PCIe Message Reception Status Enable (PCIEMSGRCPSTSEN)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

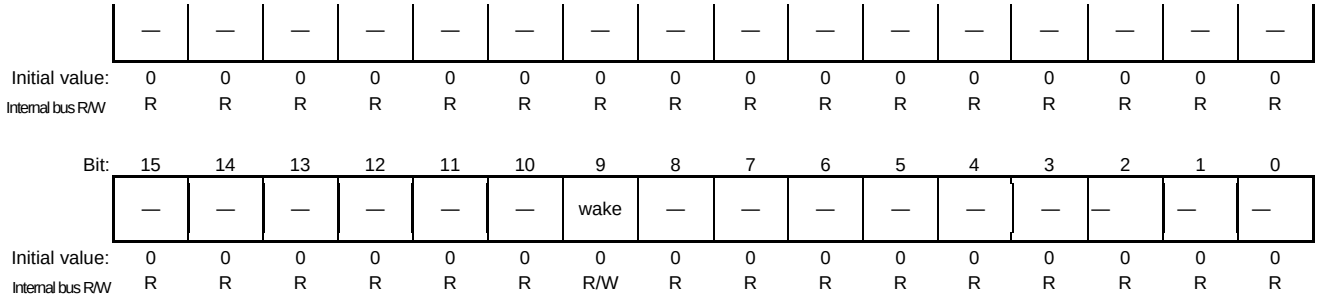
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | radm_msg_un | radm_vendor_ | radm_pm_turn | radm_pm_to_a | radm_pm_pme | radm_fatal_err | radm_nonfatal_ | radm_correcta | radm_msg_ltr |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 9 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 8 | radm_msg_unlock | 0 | R/W | Status training_rst_n enable<br>0: Disable<br>1: Enable |
| 7 | radm_vendor_msg | 0 | R/W | Status radm_vendor_msg enable<br>0: Disable<br>1: Enable |
| 6 | radm_pm_turnoff | 0 | R/W | Status radm_pm_turnoff enable<br>0: Disable<br>1: Enable |
| 5 | radm_pm_to_ack | 0 | R/W | Status radm_pm_to_ack enable<br>0: Disable<br>1: Enable |
| 4 | radm_pm_pme | 0 | R/W | Status radm_pm_pme enable<br>0: Disable<br>1: Enable |
| 3 | radm_fatal_err | 0 | R/W | Status radm_fatal_err enable<br>0: Disable<br>1: Enable |
| 2 | radm_nonfatal_err | 0 | R/W | Status radm_nonfatal_err enable<br>0: Disable<br>1: Enable |
| 1 | radm_correctable_err | 0 | R/W | Status radm_correctable_err enable<br>0: Disable<br>1: Enable |
| 0 | radm_msg_ltr | 0 | R/W | Status radm_msg_ltr enable<br>0: Disable<br>1: Enable |

### 104.2.378 PCIe Power Management Status 0 Enable (PCIEPWRMNGSTS0EN)

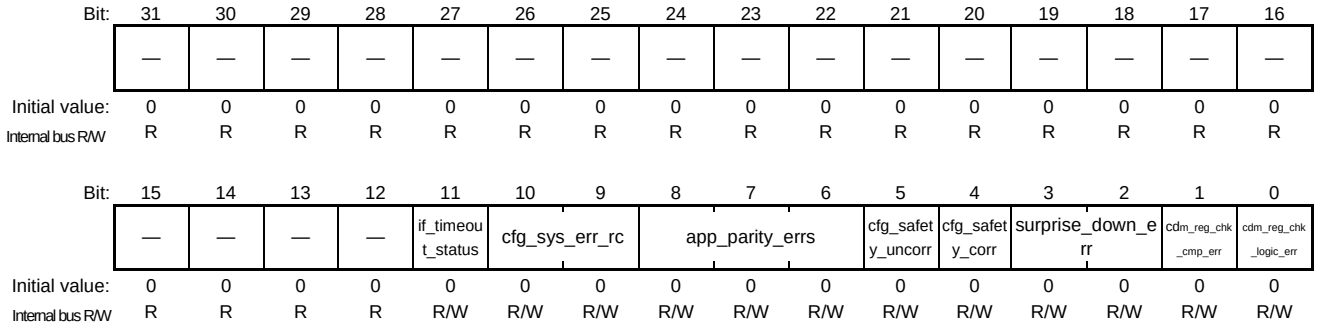
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 10 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 9 | wake | 0 | R/W | Status wake enable<br>0: Disable<br>1: Enable |
| 8 to 0 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

### 104.2.379 PCIe Error Status 0 Enable (PCIEERRSTS0EN)

**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 12 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 11 | if_timeout_status | 0 | R/W | Status if_timeout_status enable<br>0: Disable<br>1: Enable |
| 10 to 9 | cfg_sys_err_rc | 0 | R/W | Status cfg_sys_err_rc enable<br>0: Disable<br>1: Enable |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 8 to 6 | app_parity_errs | 0 | R/W | Status app_parity_errs enable<br>0: Disable<br>1: Enable |
| 5 | cfg_safety_uncorr | 0 | R/W | Status cfg_safety_uncorr enable<br>0: Disable<br>1: Enable |
| 4 | cfg_safety_corr | 0 | R/W | Status cfg_safety_corr enable<br>0: Disable<br>1: Enable |
| 3 to 2 | surprise_down_err | 0 | R/W | Status surprise_down_err enable<br>0: Disable<br>1: Enable |
| 1 | cdm_reg_chk_cmp_err | 0 | R/W | Status cdm_reg_chk_cmp_err enable<br>0: Disable<br>1: Enable |
| 0 | cdm_reg_chk_logic_err | 0 | R/W | Status cdm_reg_chk_logic_err enable<br>0: Disable<br>1: Enable |

### 104.2.380 PCIe Interrupt Status 0 Enable (PCIEINTSTS0EN)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | cfg_aer_rc_err_msi | | cfg_aer_rc_err_int | | msi_ctrl_int | — | cfg_link_eq_re | cfg_bw_mgt_ | cfg_bw_mgt_in | cfg_link_auto_ | cfg_link_auto_ | hp_msi | | hp_int | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | hp_pme | | cfg_pme_msi | | cfg_pme_int | | — | — | smlh_link | rdlh_link_u | — | — | cfg_mem_space_en | | cfg_bus_master_en | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 30 to 29 | cfg_aer_rc_err_msi | H'0 | R/W | Interrupt cfg_aer_rc_err_msi enable<br>0: Disable<br>1: Enable |
| 28 to 27 | cfg_aer_rc_err_int | H'0 | R/W | Interrupt cfg_aer_rc_err_int enable<br>0: Disable<br>1: Enable |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 26 | msi_ctrl_int | 0 | R/W | Interrupt msi_ctrl_int enable<br>0: Disable<br>1: Enable |
| 25 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 24 | cfg_link_eq_req_int | 0 | R/W | Interrupt cfg_link_eq_req_int enable<br>0: Disable<br>1: Enable |
| 23 | cfg_bw_mgt_msi | 0 | R/W | Interrupt cfg_bw_mgt_msi enable<br>0: Disable<br>1: Enable |
| 22 | cfg_bw_mgt_int | 0 | R/W | Interrupt cfg_bw_mgt_int enable<br>0: Disable<br>1: Enable |
| 21 | cfg_link_auto_bw_msi | 0 | R/W | Interrupt cfg_link_auto_bw_msi enable<br>0: Disable<br>1: Enable |
| 20 | cfg_link_auto_bw_int | 0 | R/W | Interrupt cfg_link_auto_bw_int enable |
| 19 to 18 | hp_msi | H'0 | R/W | Interrupt hp_msi enable<br>0: Disable<br>1: Enable |
| 17 to 16 | hp_int | H'0 | R/W | Interrupt hp_int enable<br>0: Disable<br>1: Enable |
| 15 to 14 | hp_pme | H'0 | R/W | Interrupt hp_pme enable<br>0: Disable<br>1: Enable |
| 13 to 12 | cfg_pme_msi | H'0 | R/W | Interrupt cfg_pme_msi enable<br>0: Disable<br>1: Enable |
| 11 to 10 | cfg_pme_int | H'0 | R/W | Interrupt cfg_pme_int enable<br>0: Disable<br>1: Enable |
| 9 to 8 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 7 | smlh_link_up | 0 | R/W | Interrupt smlh_link_up enable<br>0: Disable<br>1: Enable |
| 6 | rdlh_link_up | 0 | R/W | Interrupt rdlh_link_up enable<br>0: Disable<br>1: Enable |
| 5 to 4 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 3 to 2 | cfg_mem_space_en | H'0 | R/W | Interrupt cfg_mem_space_en enable<br>0: Disable<br>1: Enable |
| 1 to 0 | cfg_bus_master_en | H'0 | R/W | Interrupt cfg_bus_master_en enable<br>0: Disable<br>1: Enable |

### 104.2.381 PCIe DMA Interrupt Status Enable (PCIEDMAINTSTSEN)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | edma_int | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 15 to 0 | edma_int | 0 | R/W | Interrupt edma_int enable<br>0: Disable<br>1: Enable<br>[15:8] : read channel 7-0<br>[7:0] : write channel 7-0 |

### 104.2.382 PCIe INTx Interrupt Status Enable (PCIEINTXSTSEN)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | radm_intd_de | radm_intc_deas | radm_intb_de | radm_inta_de | radm_intd_asse | radm_intc_asse | radm_intb_ass | radm_inta_asse |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 8 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 7 | radm_intd_deasserted | 0 | R/W | Interrupt radm_intd_deasserted enable<br>0: Disable<br>1: Enable |
| 6 | radm_intc_deasserted | 0 | R/W | Interrupt radm_intc_deasserted enable<br>0: Disable<br>1: Enable |
| 5 | radm_intb_deasserted | 0 | R/W | Interrupt radm_intb_deasserted enable<br>0: Disable<br>1: Enable |
| 4 | radm_inta_deasserted | 0 | R/W | Interrupt radm_inta_deasserted enable<br>0: Disable<br>1: Enable |
| 3 | radm_intd_asserted | 0 | R/W | Interrupt radm_intd_asserted enable<br>0: Disable<br>1: Enable |
| 2 | radm_intc_asserted | 0 | R/W | Interrupt radm_intc_asserted enable<br>0: Disable<br>1: Enable |
| 1 | radm_intb_asserted | 0 | R/W | Interrupt radm_intb_asserted enable<br>0: Disable<br>1: Enable |
| 0 | radm_inta_asserted | 0 | R/W | Interrupt radm_inta_asserted enable<br>0: Disable<br>1: Enable |

### 104.2.383 PCIe Interrupt Status 1 Enable (PCIEINTSTS1EN)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | lbc_ext_cs | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 22 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 21 to 20 | lbc_ext_cs | H'0 | R/W | Interrupt lbc_ext_cs enable<br>0: Disable<br>1: Enable |
| 19 to 0 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

### 104.2.384 PCIe Target Completion LUT 1 Enable (PCIETRGTCMPLUT1EN)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | trgt_cpl_timeout | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 9 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 8 | trgt_cpl_timeout | 0 | R/W | Status trgt_cpl_timeout enable<br>0: Disable<br>1: Enable |
| 7 to 0 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

### 104.2.385 PCIe Reset Status Clear Register (PCIERSTSTSCLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | — | — | — | — | — | — | — | — | — | — | — | — | training_rst_n | smlh_req_rst_not | link_req_rst_not | — |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 4 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 3 | training_rst_n | 0 | RW1C | Status training_rst_n clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 2 | smlh_req_rst_not | 0 | RW1C | Status smlh_req_rst_not clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 1 | link_req_rst_not | 0 | RW1C | Status link_req_rst_not clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 0 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

### 104.2.386 PCIe Message Reception Status Clear (PCIEMSGRCPSTSCLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | radm_msg_un | radm_vendor_ | radm_pm_turn | radm_pm_to_a | radm_pm | radm_fatal_err | radm_nonfatal_ | radm_correcta | radm_msg_ltr |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 9 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 8 | radm_msg_unlock | 0 | RW1C | Status radm_msg_unlock clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 7 | radm_vendor_msg | 0 | RW1C | Status radm_vendor_msg clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 6 | radm_pm_turnoff | 0 | RW1C | Status radm_pm_turnoff clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 5 | radm_pm_to_ack | 0 | RW1C | Status radm_pm_to_ack clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 4 | radm_pm_pme | 0 | RW1C | Status radm_pm_pme clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 3 | radm_fatal_err | 0 | RW1C | Status radm_fatal_err clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 2 | radm_nonfatal_err | 0 | RW1C | Status radm_nonfatal_err clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 1 | radm_correctable_err | 0 | RW1C | Status radm_correctable_err clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 0 | radm_msg_ltr | 0 | RW1C | Status radm_msg_ltr clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |

### 104.2.387 PCIe Power Management Status 0 Clear (PCIEPWRMNGSTS0CLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | wake | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | RW1C | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 10 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 9 | wake | 0 | RW1C | Status wake clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 8 to 0 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

### 104.2.388 PCIe Error Status 0 Clear (PCIEERRSTS0CLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | cfg_sys_err_rc | | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | R | R | R | R | RW1C | RW1C | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 to 11 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 10 to 9 | cfg_sys_err_rc | H'0 | RW1C | Status cfg_sys_err_rc clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 8 to 0 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

### 104.2.389 PCIe Interrupt Status 0 Clear (PCIEINTSTS0CLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | cfg_aer_rc_err_msi | | — | — | — | — | — | cfg_bw_mgt_m | — | cfg_link_auto_ | — | hp_msi | | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | R | RW1C | RW1C | R | R | R | R | R | RW1C | R | RW1C | R | RW1C | RW1C | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | hp_pme | | cfg_pme_msi | | — | — | radm_v | radm_q overflo | smlh_link | rdlh_link_u | — | — | cfg_mem_space_en | | cfg_bus_master_en | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus R/W | RW1C | RW1C | RW1C | RW1C | R | R | RW1C | RW1C | RW1C | RW1C | R | R | RW1C | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 31 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 30 to 29 | cfg_aer_rc_err_msi | H'0 | RW1C | Interrupt cfg_aer_rc_err_msi clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 28 to 24 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 23 | cfg_bw_mgt_msi | 0 | RW1C | Interrupt cfg_bw_mgt_msi clear<br>[Writing]<br>0: Ignored<br>1: Interrupt is cleared<br>This bit is read as 0. |
| 22 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 21 | cfg_link_auto_bw_msi | 0 | RW1C | Interrupt cfg_link_auto_bw_msi clear<br>[Writing]<br>0: Ignored<br>1: Interrupt is cleared<br>This bit is read as 0. |
| 20 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 19 to 18 | hp_msi | H'0 | RW1C | Interrupt hp_msi clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 17 to 16 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 15 to 14 | hp_pme | H'0 | RW1C | Interrupt hp_pme clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 13 to 12 | cfg_pme_msi | H'0 | RW1C | Interrupt cfg_pme_msi clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 11 to 10 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 9 | radm_vendor_msg | 0 | RW1C | Interrupt radm_vendor_msg clear<br>[Writing]<br>0: Ignored<br>1: Interrupt is cleared<br>This bit is read as 0. |
| 8 | radm_qoverflow | 0 | RW1C | Interrupt radm_qoverflow clear<br>[Writing]<br>0: Ignored<br>1: Interrupt is cleared<br>This bit is read as 0. |
| 7 | smlh_link_up | 0 | RW1C | Interrupt smlh_link_up clear<br>[Writing]<br>0: Ignored<br>1: Interrupt is cleared<br>This bit is read as 0. |
| 6 | rdlh_link_up | 0 | RW1C | Interrupt rdlh_link_up clear<br>[Writing]<br>0: Ignored<br>1: Interrupt is cleared<br>This bit is read as 0. |
| 5 to 4 | Reserved | All 0 | R | These bits are read as 0. The write value must be 0. |
| 3 to 2 | cfg_mem_space_en | H'0 | RW1C | Interrupt cfg_mem_space_en clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |

| Bit | Bit Name | Initial Value | Internal Bus R/W | Description |
|---|---|---|---|---|
| 1 to 0 | cfg_bus_master_en | H'0 | RW1C | Interrupt cfg_bus_master_en clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |

### 104.2.390 PCIe INTx Interrupt Status Clear (PCIEINTXSTSCLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

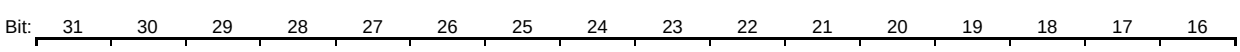
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | deassert_intd_g | deas | deas | deas | asser | asser | asser | asser | radm | radm | radm | radm | radm | radm | radm | radm |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | deassert_intd_grt | H'0 | RW1C | Interrupt deassert_intd_grt clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 14 | deassert_intc_grt | H'0 | RW1C | Interrupt deassert_intc_grt clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 13 | deassert_intb_grt | H'0 | RW1C | Interrupt deassert_intb_grt clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 12 | deassert_inta_grt | H'0 | RW1C | Interrupt deassert_inta_grt clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 11 | assert_intd_grt | H'0 | RW1C | Interrupt assert_intd_grt clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 10 | assert_intc_grt | H'0 | RW1C | Interrupt assert_intc_grt clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 9 | assert_intb_grt | H'0 | RW1C | Interrupt assert_intb_grt clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 8 | assert_inta_grt | H'0 | RW1C | Interrupt assert_inta_grt clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 7 | radm_intd_deasserted | H'0 | RW1C | Interrupt radm_intd_deasserted clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 6 | radm_intc_deasserted | H'0 | RW1C | Interrupt radm_intc_deasserted clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 5 | radm_intb_deasserted | H'0 | RW1C | Interrupt radm_intb_deasserted clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 4 | radm_inta_deasserted | H'0 | RW1C | Interrupt radm_inta_deasserted clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 3 | radm_intd_asserted | H'0 | RW1C | Interrupt radm_intd_asserted clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 2 | radm_intc_asserted | H'0 | RW1C | Interrupt radm_intc_asserted clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 1 | radm_intb_asserted | H'0 | RW1C | Interrupt radm_intb_asserted clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 0 | radm_inta_asserted | H'0 | RW1C | Interrupt radm_inta_asserted clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |

### 104.2.391 PCIe Interrupt Status 1 Clear (PCIEINTSTS1CLR)

**Common to Header TYPE00/01**



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | lbc_ext_cs | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 22 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 21 to 20 | lbc_ext_cs | H'0 | RW1C | Interrupt lbc_ext_cs clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 19 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.392 PCIe Target Completion LUT 1 Clear (PCIETRGTCMPLUT1CLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | trgt_cpl_timeout | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | RW1C | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 8 | trgt_cpl_timeout | H'0 | RW1C | Status trgt_cpl_timeout clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.393 PCIe PTM Status 0 Clear (PCIEPTMSTS0CLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | ptm_req_ replay_tx | ptm_req_ dup_rx | ptm_req_response _timeout | — | — | — | — | — | ptm_clock_ updated | ptm_responder_ rdy_to_validate | ptm_cont ext_valid |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | RW1C | RW1C | RW1C | R | R | R | R | R | RW1C | RW1C | RW1C |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 10 | ptm_req_replay_tx | H'0 | RW1C | PTM Requester Replay Sent.<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 9 | ptm_req_dup_rx | H'0 | RW1C | PTM Requester Duplicate Received.<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 8 | ptm_req_response_timeout | H'0 | RW1C | PTM Requester Response Timeout.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 7 to 3 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value is ignored. |
| 2 | ptm_clock_updated | H'0 | RW1C | Status ptm_clock_updated clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 1 | ptm_responder_rdy_to_validate | H'0 | RW1C | Status ptm_responder_rdy_to_validate clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |
| 0 | ptm_context_valid | H'0 | RW1C | Status ptm_context_valid clear<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>This bit is read as 0. |

### 104.2.394 PCIe Message Transmission Clear (PCIEMSGTXCLR)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | cfg_send_f_err | | cfg_send_nf_ | | cfg_send_cor | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 22 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 21 to 20 | cfg_send_f_err | H'0 | RW1C | Status cfg_send_f_err clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 19 to 18 | cfg_send_nf_err | H'0 | RW1C | Status cfg_send_nf_err clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |
| 17 to 16 | cfg_send_cor_err | H'0 | RW1C | Status cfg_send_cor_err clear.<br>[Writing]<br>0: Ignored<br>1: Status is cleared<br>These bits are read as 0. |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 15 to 0 | Reserved | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.395 PIPE current link operating rate (PMRATE)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | pm_current_data_rate | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | pm_current_data_rate | All 0 | R | The current link operating rate. |

### 104.2.396 SII: BDF Signals (SIIBDF)

**Common to Header TYPE00/01**

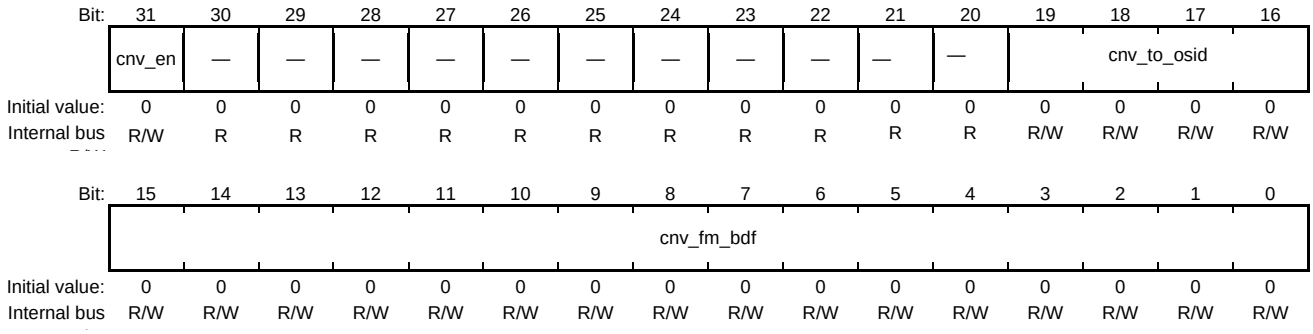
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | app_bus_num | | | | | | | | — | — | — | app_dev_num | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | app_bus_num | H'0 | R/W | Bus number |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 to 0 | app_dev_num | H'0 | R/W | Device number. |

### 104.2.397 Convert Requester ID to OSID Register *i* (CNVID*i* )(*i* = 0 to 15)

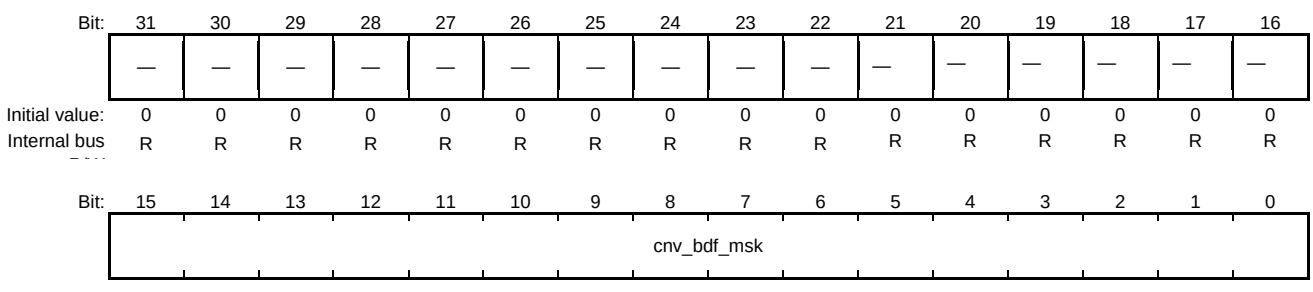
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 | cnv_en | H'0 | R/W | Conversion enabled |
| 30 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 16 | cnv_to_osid | H'0 | R/W | OSID value when matched |
| 15 to 0 | cnv_fm_bdf | H'0 | R/W | Value to compare with BDF |

### 104.2.398 Convert Requester ID to OSID Mask Register *i* (CNVIDMSK*i* )(*i* = 0 to 15)

**Common to Header TYPE00/01**



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 0 | cnv_bdf_msk | H'0 | R/W | Mask used for matching |

### 104.2.399 Convert Requester ID to OSID Control Register (CNVOSIDCTRL)

**Common to Header TYPE00/01**

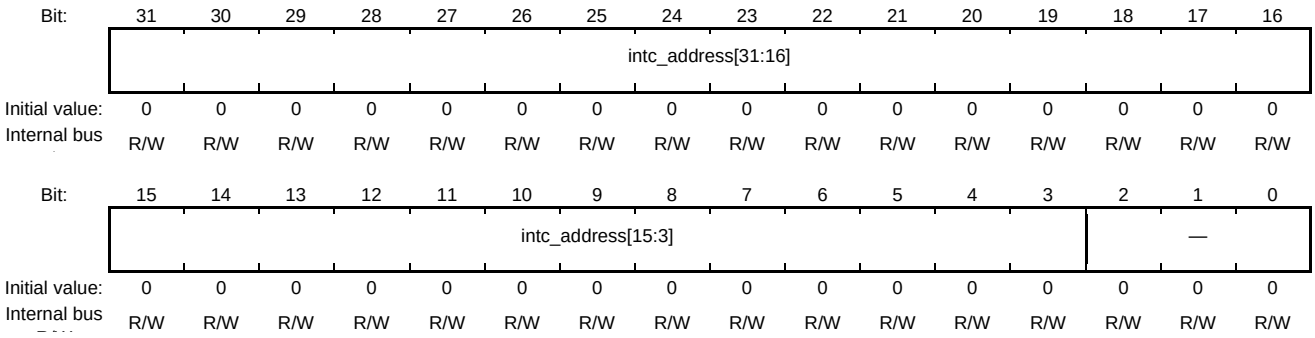
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | elsosid | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 16 | elsosid | H'0 | R/W | OSID value to use when none hit |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.400 INTC address(AXIINTCADDR)

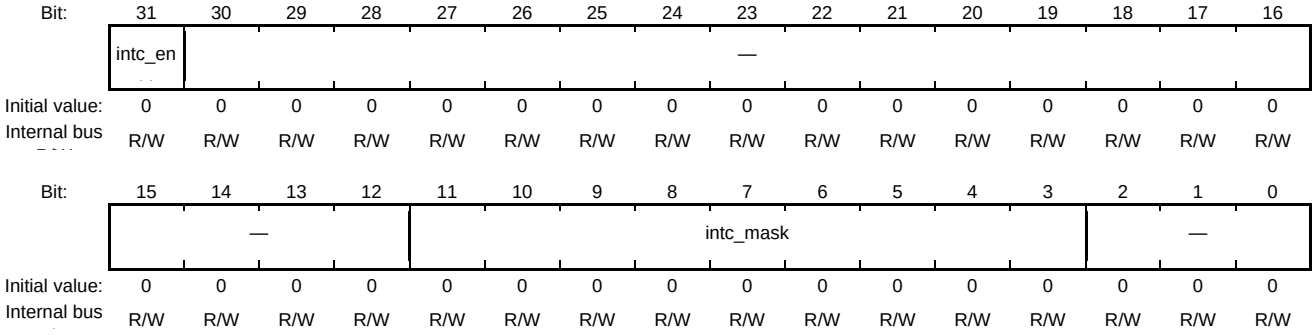
**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 3 | intc_address | H'0 | R/W | Address judged as INTC area |
| 2 to 0 | — | All 0 | R/W | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.401 INTC control & mask(AXIINTCCONT)

**Common to Header TYPE00/01**




| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 | intc_enable | H'0 | R/W | INTC judgment enable |
| 30 to 12 | — | All 0 | R/W | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 3 | intc_mask | H'0 | R/W | Mask used for INTC judgment |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 2 to 0 | — | All 0 | R/W | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.402 Reference Clock Control Signals PHY0 Register (REFCLKCTRLP0)

**Common to Header TYPE00/01**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | phy0 | — | — | — | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | phy0_ref_repeat_clk_en | H'0 | R/W | Repeat reference clock enable |
| 8 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | — | H'1 | R | Reserved<br>These bits are always read as 1. The write value should always be 1. |
| 1 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 104.2.403 Reference Clock Control Signals PHY1 Register (REFCLKCTRLP1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | | — | | — | phy1_ref_ | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal bus | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R | R |

| Bit | Bit Name | Initial Value | Internal bus R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | H'0 | R | Reserved<br><br>These bits are read as 0. The write value must be 0. |
| 2 | phy1_ref_use_pad | H'0 | R/W | Select reference clock connected to ref_pad_clk_p/ref_pad_clk_m |
| 1 to 0 | — | H'0 | R | Reserved<br><br>These bits are read as 0. The write value must be 0. |

## 104.3 Operation

The following describes the operation of this module.

Note: The following operation flowcharts are each divided in two parts and described. One part covers the H/W, hardware operation of this module and the other covers the S/W, software operation to control this module. The arrows originating from the outside of the figure frame indicate that the event arises from the user program.

In the explanation of the operation flows, interrupt sources are described in the H/W part. The S/W should first verify a given interrupt flag and then clear it.

If generation of interrupt sources is expected, enable each of the interrupt sources referring to Figure 104.41, Figure 104.42, Figure 104.43 and Figure 104.44.

### 104.3.1 Clocks

The following describes the clocks specification for PCIEC.

#### 104.3.1.1 Overview

PCIEC uses the clocks outlined in Table 104.7.

**Table 104.7 Clocks list in PCIEC**

| Clock Name | Frequency[MHz] | Descrition |
|---|---|---|
| S0D2_HSCφ | Refer to section 8(CPG) | This clock is used for AXI bus I/F. |
| S0D4_HSCφ | Refer to section 8(CPG) | This clock is used for APB bus I/F. |
| core_clk@ Gen{1,2,3,4} | {62.5, 125, 250, 500} | This clock is the primary clock for the PCIe Controller module |
| PCIE*m*_CLK_P/M<br>(*m* = 0) | 100 | Reference clock for the PLL incorporated in the PCIe Controller module (differential inputs). |

#### 104.3.1.2 Reference Clock

PCIEC physical unit is able to repeat the input reference clocks. A reference clock repeater is provided to propagate the reference clock through PCIEC physical unit pins. This is called reference clock repeat mode.

Figure 104.2 shows connection of reference repeat mode.

Reference clock repeat mode is enabled by setting PCIEC0 Local register REFCLKCTRLP0. phy0_ref_use_pad = 1 and REFCLKCTRLP0. phy0_ref_repeat_clk_en = 1, PCIEC1 Local register REFCLKCTRLP*i*. phy*i*_ref_use_pad = 0 and REFCLKCTRLP0. phy*i*_ref_repeat_clk_en= 1 (*i* = 0,1). Starting point of reference clock is only PCIE0_CLK_P/M supported.

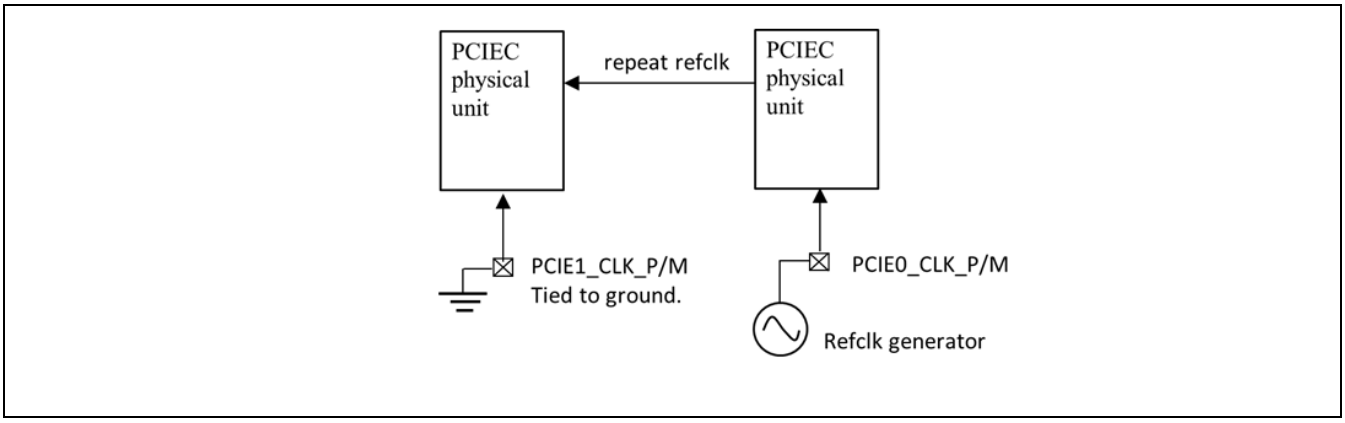



**Figure 104.2 Connection of reference repeat mode**

PCIEC supports the Separate Reference clock with Independent Spread Spectrum Clocking (SRIS) feature as defined by the PCI-SIG. SRIS is enabled by setting PCIEMSR0. app_sris_mode = 1. When SRIS is enabled, then L0s is not supported and must not be advertised in the ASPM Support Capability Registers.

### 104.3.2 Resets

The following describes the resets specification for PCIEC.

PCIEC uses the resets outlined in Table 104.8. These resets are implemented per link.

Cold-reset (power_up_rst_n) is controlled by CPG registers, Warm-reset (perst_n) and Hot-reset (app_init_rst) are controlled by PCIEC control register (PCIERSTCTRL0 / PCIERSTCTRL2) in PCIEC module.

**Table 104.8 Resets list in PCIEC**

| Reset level | Description |
|---|---|
| Cold-reset | power_up_rst_n. This reset is used as PCIEC fundamental reset.<br><br>Assertion of this reset resets all registers in PCIEC. |
| Warm-reset | PCIERSTCTRL0.perst_n. This reset is used as same as PERST#.<br><br>Assertion of this reset resets registers except PCIEC control registers partial. |
| Hot-reset | PCIERSTCTRL2.app_init_rst. This reset is used as Hot-reset.<br><br>Assertion of this reset resets non-sticky registers. |

### 104.3.3 Endian

The ordering of AXI/APB address and data is a little endian.

### 104.3.4 Initialization

#### (1) Standby State Cancellation

- After canceling a reset, cancel the module standby state. For the module standby state, refer to section “8A. Module Standby and Software Reset”. This module is in the module standby state after a power-on reset is canceled.

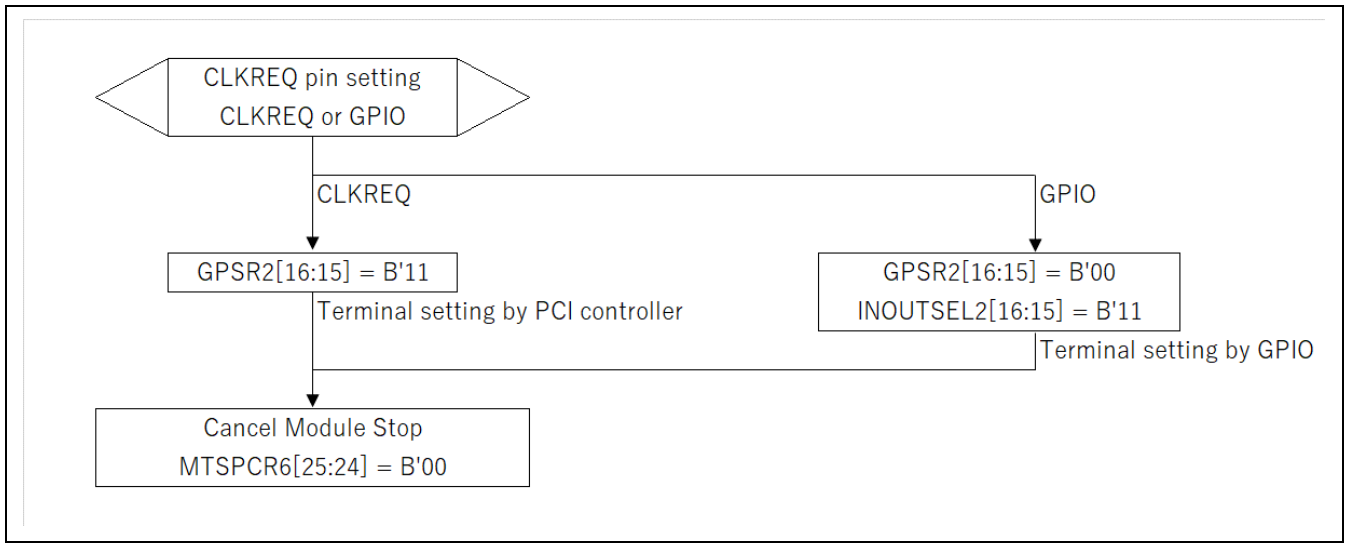



**Figure 104.3a Initial Setting of PCIEC**

### (2) Initial Setting of PCI Express

- Figure 104.3 shows the PCIEC operation in initial setting example.
- After start-up processing for the PCIe control register access is completed, the initial setting is executed to start operation as a PCI Express. Until the initial setting is completed, this module cannot work correctly as a PCI Express device. Therefore, a link is not established immediately after a reset is canceled. When the initial setting is completed, set the PCIERSTCTRL1.app_ltssm_enable=1 and PCIERSTCTRL1. app_hold_phy_rst =0 to start operation.
- Set the bits in the configuration registers as directed in the related register description, if there is a direction on initialization.
- When a register is used with the default setting, setting the register is not required.
- Set PRTLGC5.LANE_CONFIG =H'00_0040.

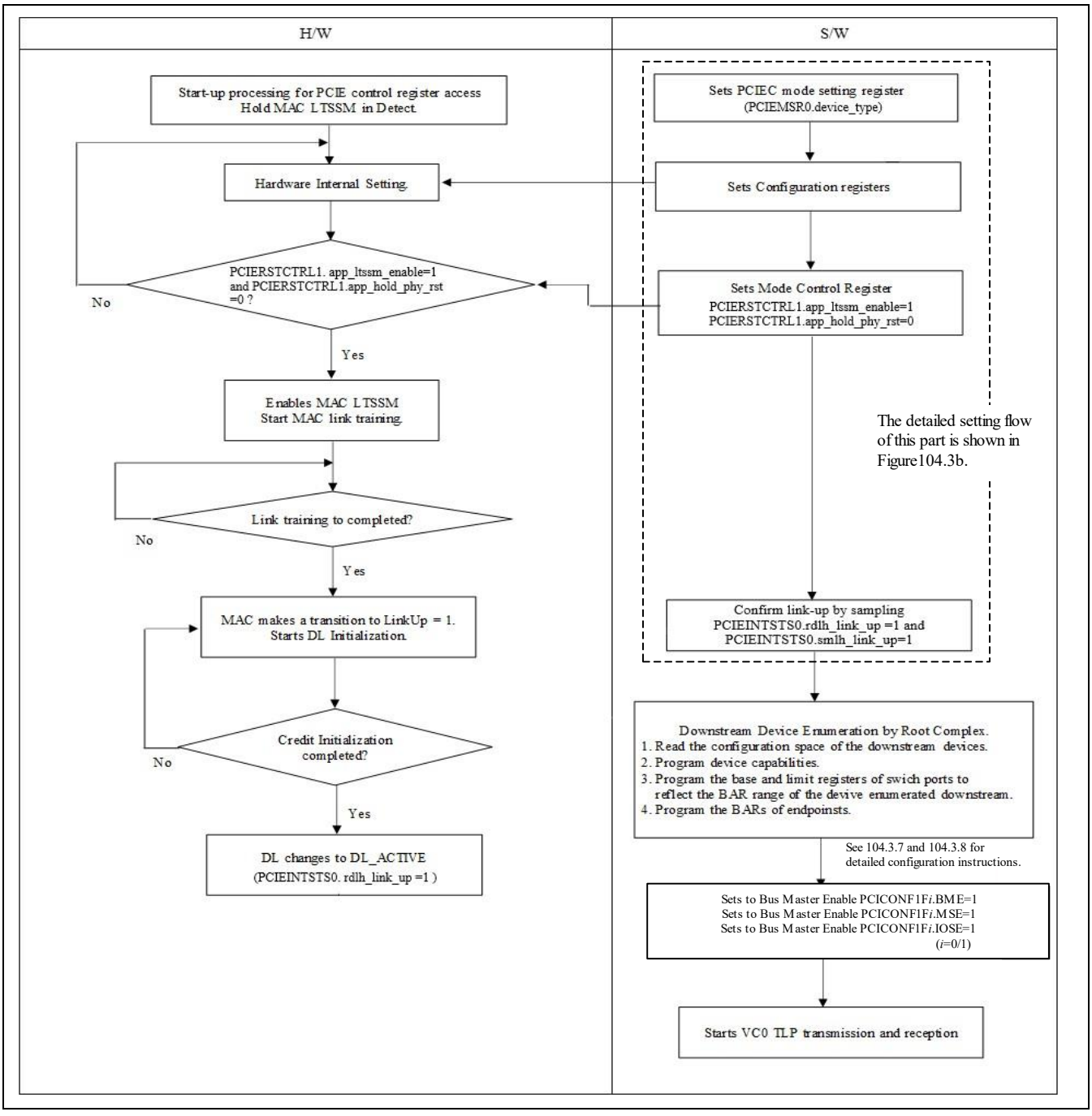
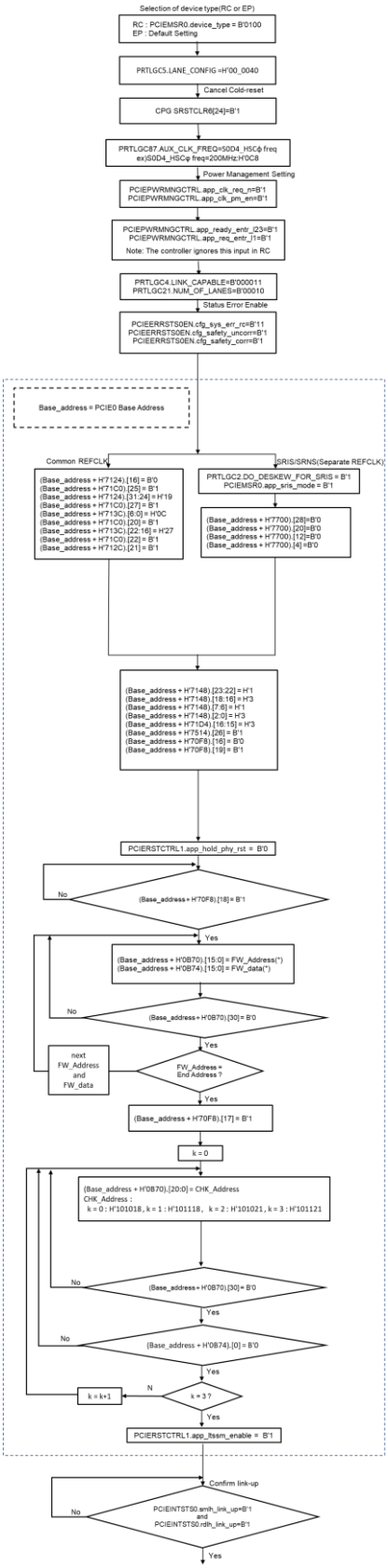



**Figure 104.3 Initial Setting of PCIEC(example)**




(*): FW_Address(ADDR) and FW_data(DATA) refer to attachment File "104_PCIe_fw_addr_data_ver1.05.txt"

**Figure 104.3b Initial Setting of PCIEC(example)**

### (3) DL_Down Operation

Figure 104.4 shows the operation when a link goes down (LinkUP = 0) in the state where a link is established, thus causing DL_Down. When DL_Down is caused, PCIERSTCTRL1.app_ltssm_enable is set to 0. Since the current data is invalidated, all the resources of the data transmitter/receiver of this module are initialized.

- Upon DL_Down, a write to the transmission buffer is disabled thus stopping transmission.
- Upon DL_Down, the receive buffer is forcibly cleared.
- If a link goes down, transmission and reception are suspended; therefore, data before and after DL_Down becomes illegal. Since the specific process here differs depending on the application, the operation is simply indicated as “Processes errors.” in the flowchart shown in Figure 104.4.

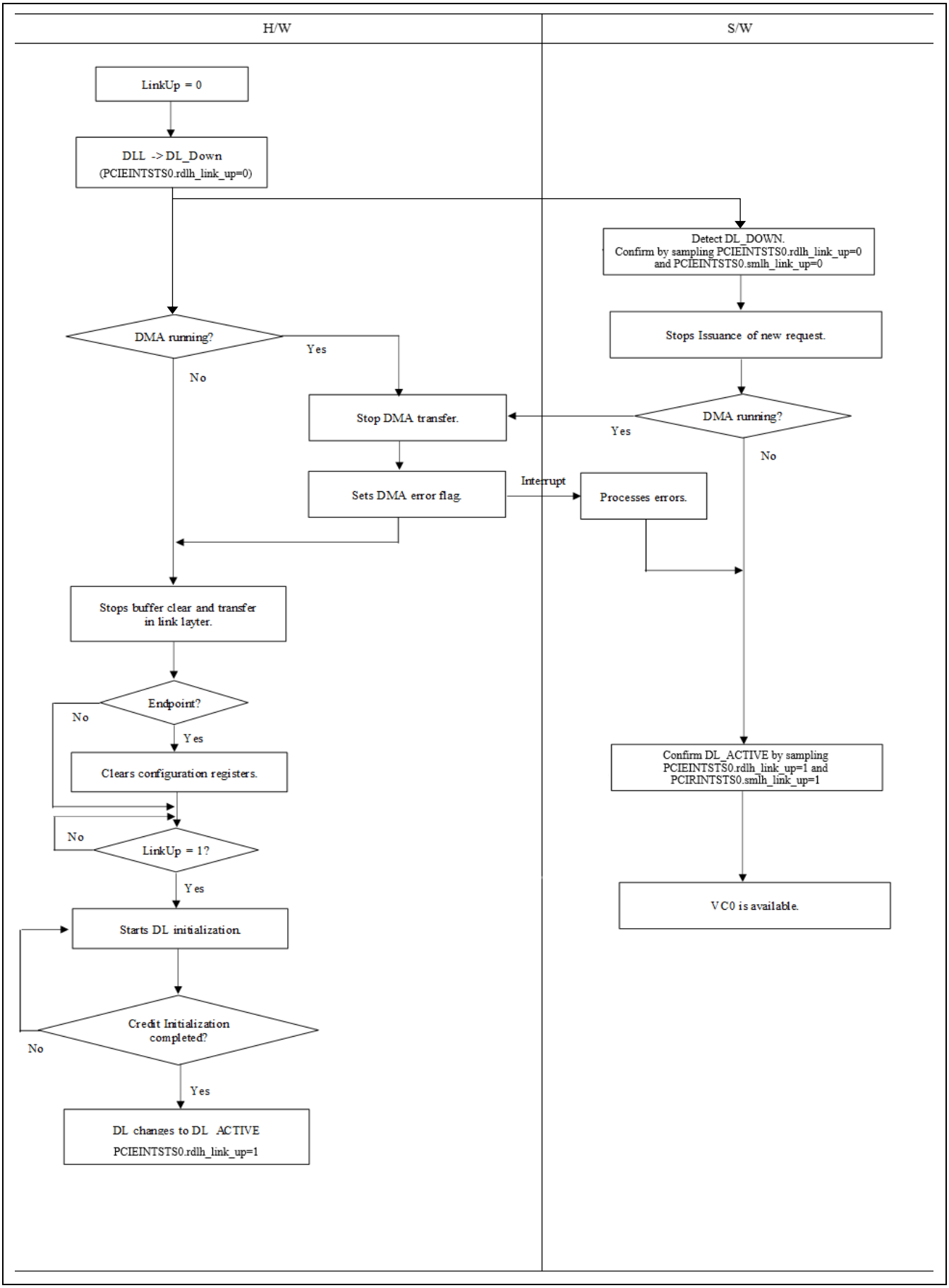



**Figure 104.4 DL_Down Operation**

#### (4) Hot Reset Operation

##### (a) Hot Reset Transmission

In Root Port mode, to send a hot reset to a downstream port, write 1 to the secondary bus reset bit in the configuration register. Before issuing a hot reset, confirm that this module has completed transfer.

##### (b) Hot Reset Reception

In Endpoint mode, when a hot reset is received from a downstream port, the LTSSM on the link layer makes a transition to the hot reset state. This causes LinkUp = 0 and then DL_Down = 1. After this, the same operation is performed as the operation upon the DL_Down.

### 104.3.5 Configuration Cycle

When using this module as a Root Port, a configuration cycle must be generated to configure the connection-target device. A configuration cycle refers to the process of checking the status of the configuration register of the connection-target Endpoint, switch, or bridge through the use of configuration access, and assigning values to the configuration registers of the Root Port itself and the Endpoint according to the result of the checking. When the Root Port accesses its own configuration register, such access is normally made through the internal bus.

This section describes the transmission (write) and reception (read) operations of the configuration access, and the items to be specified during a configuration cycle.

When this module is used as an Endpoint, it receives a configuration access from a Root Port, and accepts the initialization processing.

#### (1) Initialization of PCIEC in Root Port Mode

In Root Port mode, set all the Endpoints (indicated as system in the flow chart) connected to the PCIEC after link establishment.Figure 104.5 shows a sample setting flowchart.

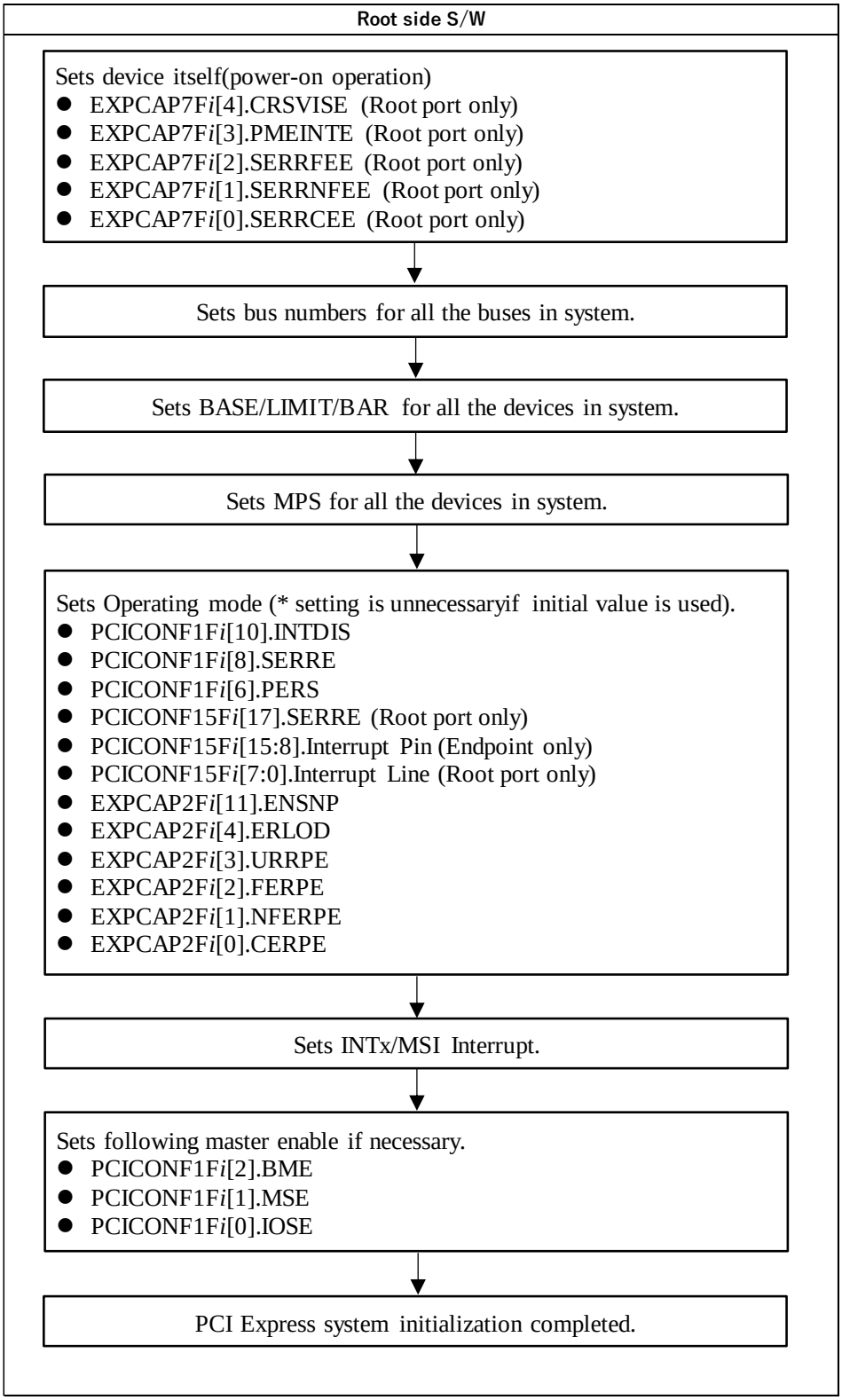



**Figure 104.5 PCIEC Initialization in Root Port Mode**

### 104.3.6 Overview of Transmission and Reception

PCI Express transfers are classified into two types of requests and one type of completion.

The following three methods are provided for PCI Express transactions. For details on each transfer, refer to the following sections.

1. Outbound: A request is issued from the internal bus side to a device on the PCI Express.
2. Inbound: A request is received by the internal bus side from the PCI Express.
3. DMA transfer: A great quantity of data is automatically written and read between the PCI Express and the internal bus without intervention of the CPU.

Table 104.9 shows the relationship between two types of requests and PCI Express transactions and shows the transaction issuance methods with this module.

**Table 104.9 Relationship between Posted/Non-Posted Requests and PCI Express Transactions**

| Transaction Type | Transfer Type | Completion | Device Type that can Issue Transaction | Transaction Issuance Method with This Module | Operation |
|---|---|---|---|---|---|
| Memory Write | Posted | Not provided | Root Port &Endpoint | PIO, DMA, (MSI Endpoint only) | Data is written to the memory of the communication peer. 1 byte to 4 Kbytes of data not exceeding MPS*1 can be written. |
| Memory Read | Non-Posted | Provided | Root Port &Endpoint | PIO, DMA | A quantity of data not exceeding MRRS*2 is read from the memory of the communication peer. The read data is returned with a completion separately from this request; no data is included in this request. |
| I/O Write | Non-Posted | Provided | Root Port only | PIO | Data is written to I/O. Data size is 1 DW. A completion is necessary that indicates whether this request has succeeded. |
| I/O Read | Non-Posted | Provided | Root Port only | PIO | Data is read from I/O. No data is included. The read data and a completion that indicates whether this request has succeeded are necessary. |
| Configuration Write | Non-Posted | Provided | Root Port only | PCI Express transfer control register | Configuration write is performed. Data size is 1 DW. A completion is necessary that indicates whether this request has succeeded. |
| Configuration Read | Non-Posted | Provided | Root Port only | PCI Express transfer control register | Configuration read is performed. No data is included. The read data and a completion that indicates whether this request has succeeded are necessary. |
| Message | Posted | Not provided | Root Port & Endpoint | Issued by this IP, message transmit register, message transmit module | A message request is transmitted. This is used for power management and interrupt control. |

Notes: 1. MPS: Max Payload Size
2. MRRS: Max Read Request Size

In Table 104.9, the transactions for which “Provided” is indicated in the “Completion” column, make a response with a completion. The data returned with a completion differs depending on the transaction. Table 104.10 shows the particulars of the completion contents returned. Specifically, the table shows whether

completion data for each type of transactions is included or not, the devices that can issue the transaction, and some other related information.

**Table 104.10 Relationship between Completion and PCI Express Transactions**

| Transaction Type | Completion Data | Device Type that can Issue Transaction | Transaction Issuance Method with This Module | Operation |
|---|---|---|---|---|
| Memory Read | Included | Root Port & Endpoint | Automatically issued by this module. | Data requested by memory read request and status are returned. |
| I/O Write | Not included | Root Port only | Automatically issued by this module. | No data is included. Status in response to I/O write is returned. |
| I/O Read | Included | Root Port only | Automatically issued by this module. | Data size is 1 DW. The read data and a completion that indicates whether this request has succeeded are returned. |
| Configuration Write | Not included | Root Port only | Automatically issued by this module. | No data is included. Status in response to configuration write is returned. |
| Configuration Read | Included | Root Port only | Automatically issued by this module. | Data size is 1 DW. The read data and a completion that indicates whether this request has succeeded are returned. |

### 104.3.7 Outbound (Data Transfer from Internal Bus → PCIEC → External Device)

This section describes Outbound. Outbound here refers to the transfer method to create a PCI Express packet by accessing the memory space assigned to the PCI Express via the internal bus.

#### (1) Outbound Features

Number of Outbound Address Translation Regions is 16.

Address translation is used for mapping different address ranges to different memory spaces supported by your application. A typical example maps your application memory space to PCI memory space. The iATU also supports type translation. Without address translation, your application address is passed unmodified to the TLPs directly through the Tx application interface. You can program the iATU to implement your own outbound address translation scheme without external logic.

**Outbound Features:**

- Address Match mode operation for MEM and I/O, CFG, and MSG TLPs. No translation for completions.
- Supports type translation through TLP type header field replacement for MEM or I/O types to MSG/CFG types.
  - Includes posted to non-posted translation (for example, MWr to CfgWr0)
  - No translation from completions.
- Programmable TLP header field replacement.
  - TYPE/TC/AT/ATTR/MSG-Code.
  - Function Number
- Multiple (up to 256) address regions programmable for location and size.
- Programmable enable/disable per region.
- Automatic FMT field translation between three DWORDs and four DWORDs for 64-bit addresses.

■ Invert Address Matching mode to translate accesses outside of a successful address match.

■ Configuration Shift mode. Optimizes the memory footprint of CFG accesses destined for the Rx application interface in a multifunction device.

■ Response code which defines the completion status to return for accesses matching a region.

■ Supports regions from 4 KB to 16 ZB in size.The minimum and maximum region sizes are determined by CX_ATU_MIN_REGION_SIZE and CX_ATU_MAX_REGION_SIZE. Smaller region sizes consume extra decode logic.

For AXI, this feature is applicable only when SLAVE_BUS_ADDR_WIDTH = 64; If SLAVE_BUS_ADDR_WIDTH < 64, CX_ATU_MAX_REGION_SIZE is restricted to 4GB. For AHB, this features is not applicable.

■ Payload Inhibit marks all TLPs as having no payload data.

■ Header Substitution replaces bytes 8 to 11 (for 3 DWORD header) or bytes 12 to 15 (for 4 DWORD header), inclusive, of the outbound TLP header.

■ Tag Substitution of the outbound TLP tag field.

■ Function number bypass mode to allow function number information to be supplied from your application transmit interface while translating the address and other attributes of the TLP.

■ DMA bypass mode to allow TLPs which are initiated by the embedded DMA engine, to pass through the iATU untranslated.

■ TLP Header fields bypass mode to allow header information to be supplied from your application transmit interface or, if AMBA is configured, from the AMBA sideband bus while translating the address and Type of the TLP.

***Note:** The default behavior of the ATU when there is no address match in the outbound direction or no TLP attribute match in the inbound direction, is to pass the transaction through.

### (2) Outbound Limitations

■ Generation of ATS requests and messages not supported (no facility to insert NW and iTAG fields).

### (3) Outbound Basic Operation (Address Match Mode)

The address field of each request MEM and I/O TLP is checked to see if it falls into any of the enabled address regions defined by the Start and End addresses as defined in Figure 104.6. When an address match is found, then the TLP address field is modified as follows:

*Translated Address = Original Address - Base Address + Target Address*

and the TYPE, TC, AT, TH, PH, ST, Function Number, and ATTR header fields are replaced with the corresponding fields in the ATUCAP[0+i*18] register. When your application (XALI0/1/2) address field matches more than one of the CX_ATU_NUM_OUTBOUND_REGIONS address regions, then the first (lowest of the numbers from 0 to CX_ATU_NUM_OUTBOUND_REGIONS-1) enabled region to be matched is used. For details on what happens when there is no address match, see “No Address Match Result”. This operational mode (called Address Match Mode) is always used for outbound translation. Figure 104.6 to Figure 104.7 provide more details on this translation process.

***Note:** The iATU does not translate DBI requests.

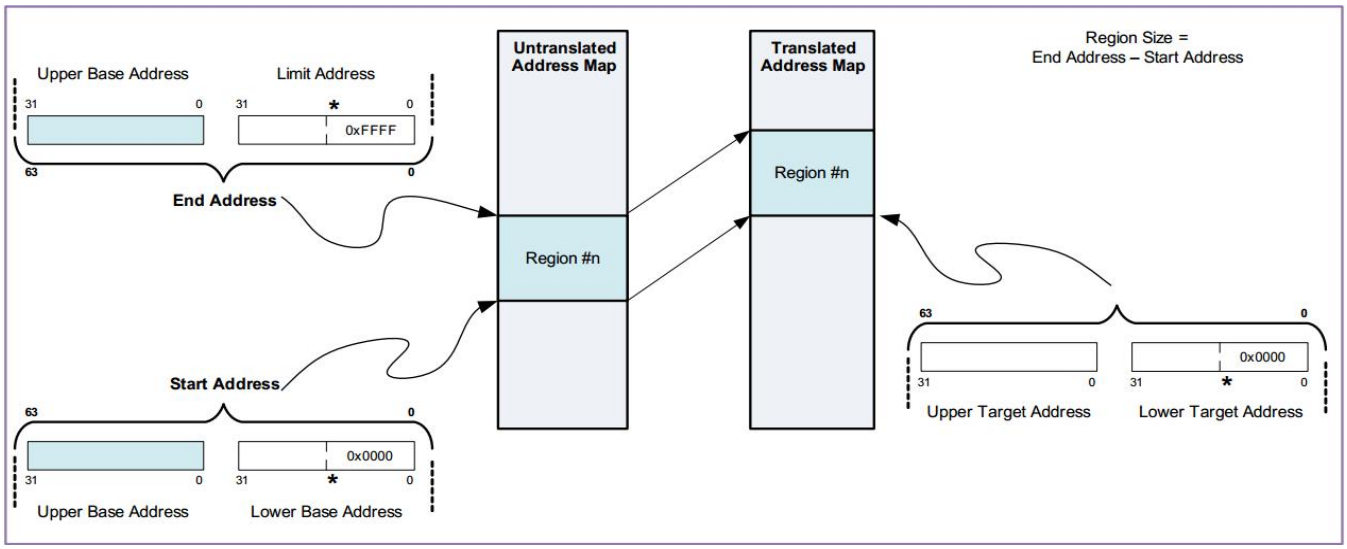



**Figure 104.6 64-bit Region Mapping: Outbound and Inbound (Address Match Mode); INCREASE_REGION_SIZE=0**

**Note:** * is log2(CX_ATU_MIN_REGION_SIZE)

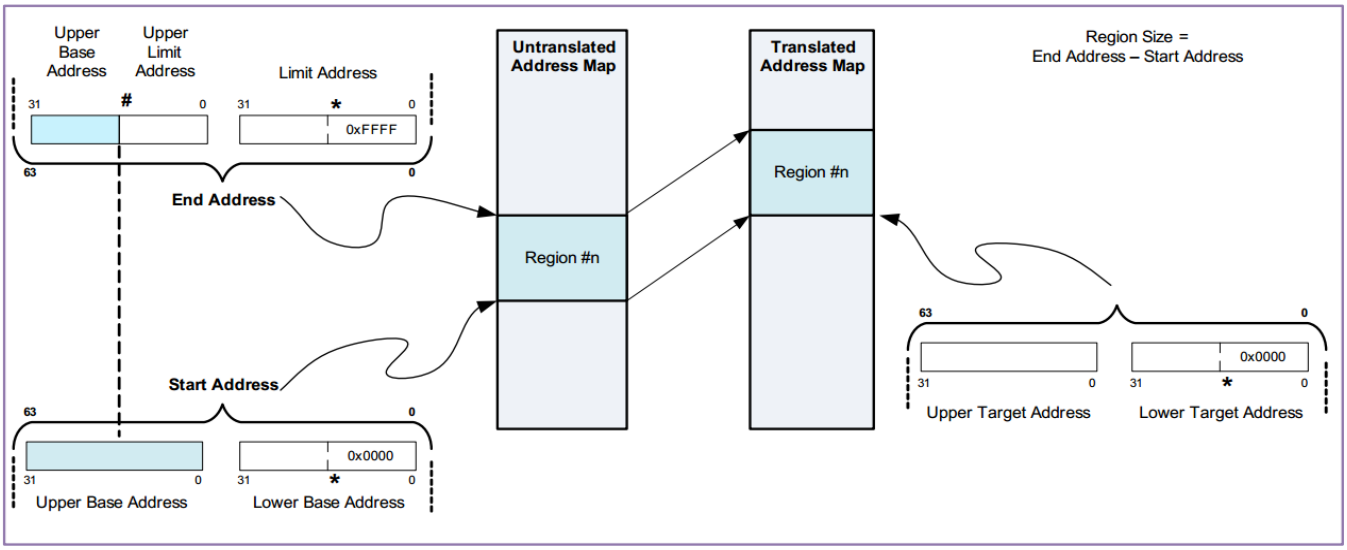



**Figure 104.7 64-bit Region Mapping: Outbound and Inbound (Address Match Mode); INCREASE_REGION_SIZE=1**

**Note:** * is log2(CX_ATU_MIN_REGION_SIZE)

The CX_ATU_MIN_REGION_SIZE configuration parameter (4 kB - 64 kB) specifies the minimum size of an address translation region. For example, when set to 64 kB, the lower 16 bits of the Base and Target registers are zero, the lower 16 bits of the Limit register is 0xFFFF, and all address regions are aligned on 64 kB boundaries. More precisely, the lower log2(CX_ATU_MIN_REGION_SIZE) bits are zero. In Figure 104.8, # is CX_ATU_MAX_REGION_SIZE; if =0, Upper Limit Address register does not exist.
The INCREASE_REGION_SIZE field in ATUCAP[0+i*18] determines if the maximum region size is 4 GB or is set by the CX_ATU_MAX_REGION_SIZE configuration parameter. This only applies to 64-bit addressing.

***Caution**: *The combination of region size and number of regions must not consume the maximum space that is addressable with your 32-bit or 64-bit system address*.*

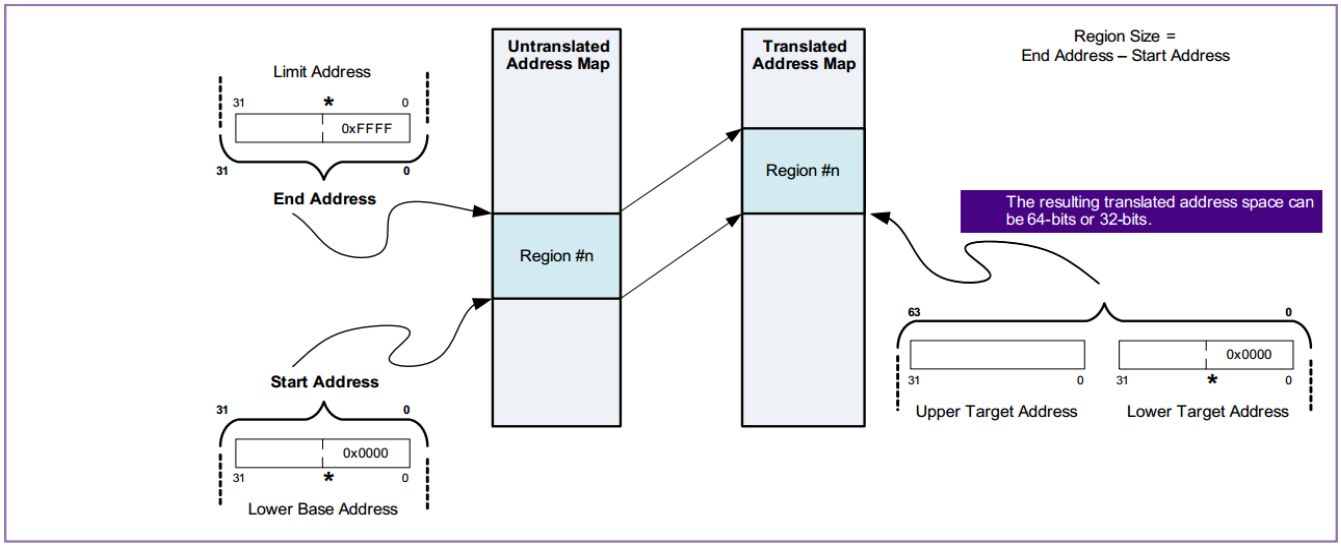



**Figure 104.8 32-bit Region Mapping: Outbound and Inbound (Address Match Mode)**

**Note:** * is log2(CX_ATU_MIN_REGION_SIZE)

### (4) Outbound Detailed Operation

The following advanced topics are discussed:

**Default Operation:**

- RID BDF Number Replacement.
- IO/MEM to Msg/MsgD Type Translation.
- IO/MEM to CFG Type Translation.
- FMT Translation.
- Writing to a MRdLk Region.
- No Address Match Result

**Optional Features:**

- CFG Shift.
- Address Match Invert.
- General Bypass.
- DMA Bypass.
- Function Number Translation Bypass.
- TLP Header Field Translation Bypass.
- Payload Inhibit.

- Header Substitution.
- Tag Substitution.

**(a) RID BDF Number Replacement:**

**Overview:** When there is a successful address match on an outbound TLP, then the function number used in generating the function part of the requester ID field of the TLP is taken from the 3-bit Function Number field of the ATUCAP[0+i*18] register. The value in this field must be 0x0 unless multifunction operation in the controller is enabled. To override this behavior, use the “Function Number Translation Bypass”.

**(b) IO/MEM to Msg/MsgD Type Translation:**

**Overview:** The iATU supports TYPE translation/conversion of MEM and I/O TLPs to Msg/MsgD TLPs.
When there is a successful address match on an outbound MEM TLP, and the translated TLP type field is MSG (that is, the type field of the ATUCAP[0+i*18] register is 10xxx), then the message code field of the TLP is set to the value in the Message Code field of the
ATUCAP[1+i*18] register. A MWr with an effective length of ‘0’ is converted to Msg and all other MWr TLPs are converted to MsgD. For more information on generating messages, see “Message Generation”.

***Caution:** *The iATU does not translate outbound messages.*

**Usage Scenario:** This is useful for applications that are unable to generate Msg/MsgD TLPs.

**(c) IO/MEM to CFG Type Translation:**

**Overview:** The iATU supports translation of I/O and MEM TLPs to CFG TLPs. The 16-bit BDF is located at bits [31:16] of the translated address where:

*Translated Address = Original Address - Base Address + Target Address*

As an example

*Original Address[31:16] = {13h0,function_no[2:0]}*

*Base Address[31:16] = 16h0*

*Target Address[31:16] = {bus_no[7:0],device_no[4:0],3h0}*

Then

*Translated Address[31:16] = BDF ={bus_no[7:0],device_no[4:0],function_no[2:0]}*

To handle eight functions (as the previous example indicates), you should use a 19-bit wide region size.
For CFG transactions created directly by your application (as opposed to the iATU), you must ensure that the BDF field does not match any programmed iATU address region or else unintentional type translation could occur.
**Usage Scenario:** This is useful for applications that are unable to generate CFG TLPs.

**(d) FMT Translation:**

**Overview:** The iATU automatically sets the TLP format field for three DWORDs when it detects all zeros in the upper 32 bits of the translated address. Otherwise, it sets it to four DWORDs when it detects a 64-bit address (that is, when there is a ‘1’ in the upper 32 bits of the translated address). When the original address and the translated address are of different format, the iATU ensures that the TLP header size matches the translated address format.

**(e) Writing to a MRdLk Region:**

**Overview:** When there is a successful address match for an outbound write and the type header field in the ATUCAP[0+i*18]_i register is “00001” indicating a locked MEM transfer, then the controller sets the type field to “00000” (MEM).

**(f) No Address Match Result:**

**Overview:** When there is no address match then the address is untranslated but the TLP header information (for fields that are programmable) comes from the relevant fields on the application transmit interface XALI* or AXI slave.

**(g) CFG Shift:**

**Overview:** The iATU uses bits [27:12] of the original address to form bits [31:16] (BDF location) of the outgoing CFG TLP. This supports the Enhanced Configuration Address Mapping (ECAM) mechanism (Section 7.2.2 of the *PCI Express Base Specification, Revision 4.0, Version 1.0*) by allowing all outgoing I/O and MEM TLPs (that have been translated to CFG) to be mapped from memory space into any 256 MB region of the PCIe configuration space.
**Usage Scenario:** This is useful for applications that implement Enhanced Configuration Address Mapping (ECAM) mechanism (Section 7.2.2 of the *PCI Express Base Specification, Revision 4.0, Version 1.0*).
**Feature Configuration:** To enable CFG shift feature, set the CFG_SHIFT_MODE field of the ATUCAP[1+i*18] register to ‘1’.

**(h) Address Match Invert:**

**Overview:** In normal operation an address match on an outbound TLP occurs when the untranslated address is in the region bounded by the base address and limit address. When the invert feature is enabled, an address match occurs when the untranslated address is not in the region bounded by the base address and limit address.
**Feature Configuration:** To enable Invert feature, set the INVERT_MODE field of the ATUCAP[1+i*18] register to ‘1’.

**(i) General Bypass:**

**Not supported.**

**(j) DMA Bypass:**

**Overview:** When you do not want the iATU to translate outbound requests that are generated by the DMA, you must implement one of the following approaches:

■ Ensure that the combination of DMA channel address programming and iATU control register programming causes no translation of DMA traffic to be done in the iATU, or (see below)

■ Enable the DMA bypass mode to allow request TLPs which are initiated by the embedded DMA controller to pass through the iATU untranslated.
**Feature Configuration:** To enable DMA bypass, set the DMA_BYPASS field of the ATUCAP[1+*i**18] register to '1'.

**(k) Function Number Translation Bypass**:

**Overview:** When function number translation bypass is enabled and region address is matched, the function number of the translated TLP is taken from your application transmit interface and not from the CTRL_1_FUNC_NUM field of the ATUCAP[0+*i**18] register.

**Feature Configuration:** To enable function number translation bypass, set the FUNC_BYPASS field of the ATUCAP[1+*i**18] register to '1'.

**(l) TLP Header Field Translation Bypass:**

**Overview:** When TLP header field bypass is enabled and region address is matched, header fields of the translated TLP are taken from your application transmit interface or, if AMBA is configured, from the AMBA sideband bus and not from the corresponding fields of the ATUCAP[0+*i**18] or ATUCAP[1+*i**18] registers. The header fields are TC, ATTR *(RO and NS)* fields.
**Feature Configuration:** To enable TLP header field bypass, set the TLP_HEADER_FIELDS_BYPASS field of the ATUCAP[1+*i**18] register to '1'.

**(m) Payload Inhibit:**

**Overview:** When payload inhibit is enabled and region address is matched, the iATU converts transactions with data payload to TLPs without payload data by forcing the TLP header Fmt[1] bit =0, regardless of the application inputs.
***Note**:* This feature is not supported for MemWr/IOWr with data payload greater than 1DW.

***Caution:** This feature must not be used for outbound transactions with data poisoning, because the EP bit will be set for TLPs without data payload, this is a violation of data poisoning rules defined in section "2.7.2.2. Rules For Use of Data Poisoning" of the PCI Express Base Specification.

**Usage Scenario:** This is useful for Vendor Defined Msg (without data); your application sends a MWr over the AXI interface but does not set the byte write enables to 0.
**Feature Configuration:** To enable payload inhibit, set the the INHIBIT_PAYLOAD field of ATUCAP[1+*i**18] _i to '1'.

**(n) Header Substitution:**

**Overview:** When header substitution is enabled and region address is matched, the iATU fully substitutes bytes 8-11 (for 3 DWORD header) or bytes 12-15 (for 4 DWORD header) of the outbound TLP header with the contents of the of LWR_TARGET_RW_OUTBOUND field in ATUCAP[5+*i**18].
***Note:** For a Vendor Defined Msg/MsgD having 4 DWORD header, when HEADER_SUBSTITUTE_EN =1, the 4th DWORD (bytes 12-15) is replaced with the contents of LWR_TARGET_RW_OUTBOUND field in the ATUCAP[5+*i**18] register. But regardless of the value of HEADER_SUBSTITUTE_EN, the 3rd DWORD is replaced with the contents of UPPER_TARGET_RW field in the ATUCAP[6+*i**18] register. So for a Vendor Defined Msg, BDF (or reserved) and Vendor ID must be programmed in UPPER_TARGET_RW.
**Usage Scenario:** This is useful for Vendor Defined Msg/MsgD and ATS transactions over the AXI bridge which is normally inefficient requiring a very large iATU region.

**Feature Configuration:** To enable header substitution, set the HEADER_SUBSTITUTE_EN field in ATUCAP[1+*i**18] register to '1'.

**(o) Tag Substitution:**

**Overview:** When tag substitution is enabled and region address is matched, the iATU substitutes the TAG field of the outbound TLP header with the contents of the TAG field in ATUCAP[1+*i**18].
ST substitution can still take place using the MSG_CODE field in ATUCAP[1+*i**18]. Your application must not attempt to perform TAG substitution for outgoing non-posted TLPs.
**Usage Scenario:** This is useful for translation from AXI MWr to Vendor Defined Msg/MsgD.
**Feature Configuration:** To enable tag substitution, set the TAG_SUBSTITUTE_EN field of the ATUCAP[1+*i**18] register to '1'.

### (5) Outbound Programming

This section discusses the register programming interface and provides details of programming examples.

**(a) iATU Outbound Programming Overview**

You can access the iATU registers through the DBI interface or through BAR Matched Mem/IO requests.
The following registers are used for programming the iATU.

Table 104.11 iATU Register Map

| Address | Label (i = 0 to CX_ATU_NUM_OUTBOUND_REGIONS-1) | Description |
|---|---|---|
| **0x000** | ATUCAP[0+*i**18] | Region Control 1 |
| **0x004** | ATUCAP[1+*i**18] | Region Control 2 |
| **0x008** | ATUCAP[2+*i**18] | Lower Base Address |
| **0x00C** | ATUCAP[3+*i**18] | Upper Base Address |
| **0x010** | ATUCAP[4+*i**18] | Limit Address |
| **0x014** | ATUCAP[5+*i**18] | Lower Target Address |
| **0x018** | ATUCAP[6+*i**18] | Upper Target Address |
| **0x01C** | ATUCAP[7+*i**18] | Region Control 3 |
| **0x020** | ATUCAP[8+*i**18] | Upper Limit Address |

**Table 104.12 iATU Register Address Bus Layout**

| Register Bit | 31-20 | 19 | 18-17 | 16-9 | 8 | 7-2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Value | Not used | 0 | Reserved | Region | Inbound/Outbound | Register Address | 0 | 1 |

**Table 104.13 Example: Address Table for Accessing iATU Outbound Region 1 Configuration Register**

| Address | Register | Register Bit Value | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **31-20** | **19** | **18-17** | **16-9** | **8** | **7-2** | **1** | **0** |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0x200 | Region Control 1 | 0 | 0 | 0 | 1 | 001 |
| 0x204 | Region Control 2 | 0 | 0 | 0 | 1 | 005 |
| 0x208 | Lower Base Address | 0 | 0 | 0 | 1 | 009 |
| 0x20C | Upper Base Address | 0 | 0 | 0 | 1 | 00D |
| 0x210 | Limit Address | 0 | 0 | 0 | 1 | 011 |
| 0x214 | Lower Target Address | 0 | 0 | 0 | 1 | 015 |
| 0x218 | Upper Target Address | 0 | 0 | 0 | 1 | 019 |

**(b) Outbound iATU Programming Example**

Define Outbound Region 1 as:
64 kB I/O region from 0x80000000_d0000000 to 0x80000000_d000ffff, to be mapped to 0x00010000 in the PCIe I/O space.

1. Setup the Region Base and Limit Address Registers.
   Write 0xd0000000 to Address {0x208} to set the Lower Base Address.
   Write 0x80000000 to Address {0x20C} to set the Upper Base Address.
   Write 0xd000ffff to Address {0x210} to set the Limit Address.
2. Setup the Target Address Registers.
   Write 0x00010000 to Address {0x214} to set the Lower Target Address.
   Write 0x00000000 to Address {0x218} to set the Upper Target Address.
3. Configure the region through the Region Control 1 Register.
   Write 0x00000002 to Address {0x200} to define the type of the region to be I/O.
4. Enable the region.
   Write 0x80000000 to Address {0x204} to enable the region.

### 104.3.8 Inbound (Data Transfer from External Device → PCIEC → Internal Bus)

This section describes Inbound. With Inbound, the device on the PCI Express accesses the device on the internal bus of this module through a PCI Express request packet.

#### (1) Inbound Features

Number of Inbound Address Translation Regions is 16.

Address translation is used for mapping different address ranges to different memory spaces supported by your application. A typical example maps your application memory space to PCI memory space. The iATU supports type translation. Without address translation, your application address is passed from the TLPs directly through the TRGT1 application interface. You can program the iATU to implement your own inbound address translation scheme without external logic.

- ■ Programmable Match mode operation for MEM, I/O, CFG, and MSG TLPs. No translation for completions.
- ■ Selectable BAR Match mode operation for I/O and MEM TLPs.
  - ❑ TLPs destined for the internal CDM (or ELBI) in an upstream port are not translated.
  - ❑ TLPs that are not error-free (ECRC, malformed and so on) are not translated.
- ■ Selectable VF BAR Match mode operation for I/O and MEM TLPs which target VF BARs.
  - ❑ The VF BAR match mode enables all VFs in a PF which match a BAR to be matched with a single ATU region instead of an ATU region per VF.
- ■ Programmable TLP header field matching.
  - ❑ TYPE/TD/TC/AT/ATTR/MSG-Code.
  - ❑ Function Number.
- ■ Multiple (up to 256) address regions programmable for location and size.
- ■ Programmable enable/disable per region.
- ■ Automatic FMT field translation between three DWORDs and four DWORDs for 64-bit addresses.
- ■ Invert Address Matching mode to translate accesses outside of a successful address match.
- ■ ECAM Configuration Shift mode to allow a 256 MB CFG1 space to be located anywhere in the 64-bit address space.
- ■ Supports regions from 4 KB to 16 ZB in size. The minimum and maximum region sizes are determined by CX_ATU_MIN_REGION_SIZE and CX_ATU_MAX_REGION_SIZE. Smaller region sizes consume extra decode logic.
  For AXI, this feature is applicable only when SLAVE_BUS_ADDR_WIDTH =64; If SLAVE_BUS_ADDR_WIDTH <64, CX_ATU_MAX_REGION_SIZE is restricted to 4GB.
- ■ Single Address Location to allow all TLPs to be translated to a single address location.
- ■ Msg Type Match Mode to allow matching of any TLP of type Message.

***Note:** The default behavior of the ATU when there is no address match in the outbound direction or no TLP attribute match in the inbound direction, is to pass the transaction through.

#### (2) Inbound Limitations

■ Generation of ATS requests and messages not supported (no facility to insert NW and iTAG fields).

#### (3) Inbound Basic Operation

This section discusses how the iATU processes inbound requests. The topics are:

■ “Overview”

■ “I/O and MEM Match Modes”.

**(a) Overview**

***Note:** ■ The main difference between inbound and outbound iATU operation is that the TLP type is never changed in the inbound direction. Instead, the type field is used for more precise matching. Other fields can also be optionally used to further refine the matching process.

■ Another difference is that for MEM and I/O TLPs, you can select between address matching (as used in outbound operation) or BAR matching. Normally, an endpoint uses BAR match mode, and a root complex uses address mode as a root complex normally does not implement BARs.

The following translation rules and limitations apply:

■ When there is no match, then the address is untranslated. In addition

■ TLPs destined for the internal CDM or ELBI in an upstream port are not translated.

■ TLPs that are not error-free (ECRC, malformed and so on) are not translated.

■ Address translation of all TLP types (MEM, I/O, CFG, and MSG) except completion is supported in Address Match mode. In BAR Match mode, only translation of I/O and MEM is supported.

The setting of the MATCH_MODE field in ATUCAP[10+I*18] determines how iATU inbound matching is done for each TLP type.

**Table 104.14 Determination of Match Mode**

| TLP Type | MATCH_MODE=0 | MATCH_MODE=1 |
|---|---|---|
| MEM or I/O | Address Match Mode | BAR Match Mode |
| CFG0 | Routing ID Match Mode | Accept Mode |
| MSG/MSGD | Address Match Mode | Vendor ID Match Mode |

**(b) I/O and MEM Match Modes**

Inbound address translation for I/O and MEM TLPs operates in one of two matching modes as determined by the “Inbound Match Mode” field in the ATUCAP[10+I*18] register.

**Address Match Mode**: The operation is similar to “Outbound Basic Operation (Address Match Mode)”. The address field of each request TLP is checked to see if it falls into any of the enabled address regions as shown in Figure 104.9. When an address match is found then the TLP address field is modified as follows:

Address = Address - Base Address + Target Address

When the TLP address field matches more than one of the CX_ATU_NUM_INBOUND_REGIONS address regions, then the first (lowest of the numbers from 0 to CX_ATU_NUM_INBOUND_REGIONS -1)enabled region to be matched is used.

**BAR Match Mode**

Looking for an address match is a two-step process.

1. The standard internal PCI Express BAR Matching Mechanism checks if the address field of any MEM and I/O request TLP falls into any address region defined by the enabled BAR addresses and masks.

2. When a matched BAR is found, then the iATU compares the BAR ID to the BAR Number field in the ATUCAP[10+I*18] register for all enabled regions. Figure 104.9 and Figure 104.10 below provide more details on inbound translation in BAR Match Mode.

****Attention:** BAR Match Mode can only be used for MEM and I/O transactions.*

When the PCIe controller is operating with 64-bit BARs, the operation is defined as in Figure 104.9 on where * is log2(BAR_MASK+1).

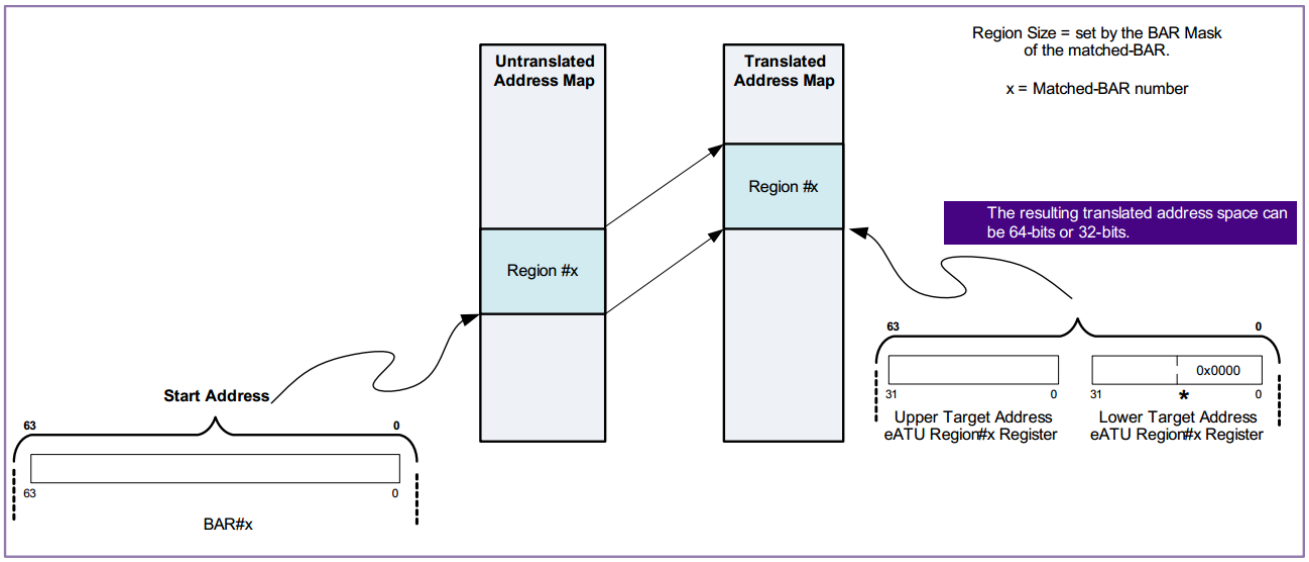
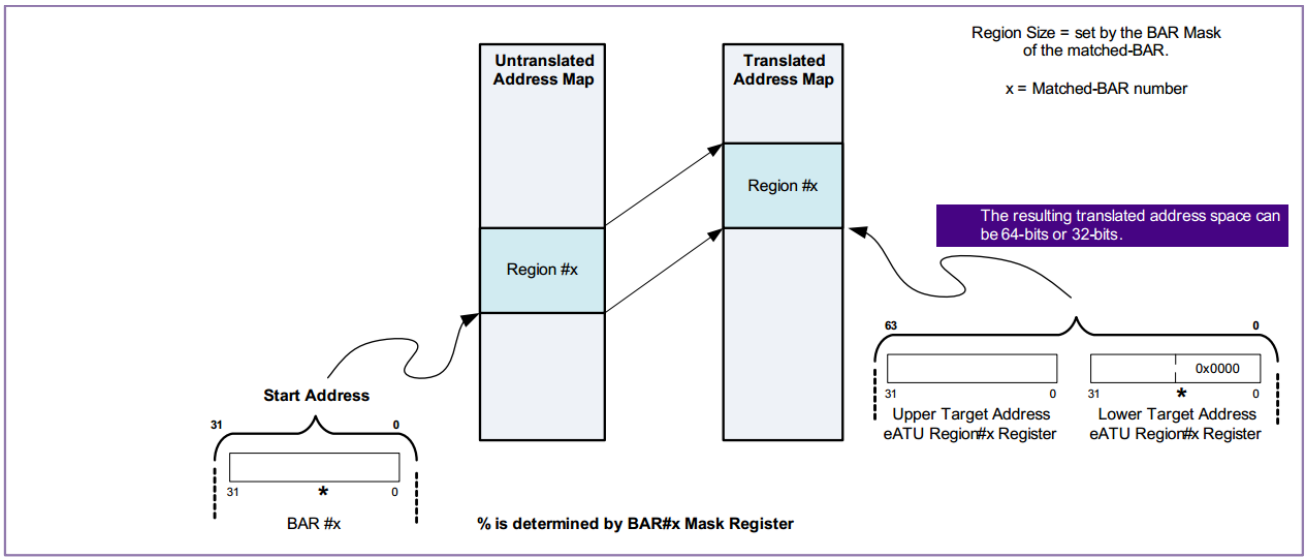



**Figure 104.9 iATU Address Region Mapping: Inbound (BAR Match Mode), 64-bit BAR**

In address match mode, when the address range does not match one of its BAR ranges in an upstream port, then the device rejects the request with unsupported request (UR) completion status and no translation occurs. When the PCIe controller is operating with 32-bit BARs, the operation is defined as in Figure 104.10

**Figure 104.10 iATU Address Region Mapping: Inbound (BAR Match Mode), 32-bit BAR**




### (4) Inbound Detailed Operation

The following advanced topics are discussed:

- “Single Address Location”.
- “CFG Handling (Upstream Port)”.
- “Optional Matching Fields”.
- “Response Code Feature”.
- “iATU Inbound MSG Handling”.

- “Msg Type Match Mode”.
- “Fuzzy Type Match Mode”.
- “FMT Translation”.
- “Invert Feature”.

**(a) Single Address Location**

When enabled and region address is matched, the TLPs can be translated to a single address location as determined by the target address register of the iATU region. This feature can be enabled using the SINGLE_ADDR_LOC_TRANS_EN field in ATUCAP[10+i*18]. This feature is useful for translating messages when AXI bridge is enabled.

**Received Message TLP Header Bytes 8 - 15 at TRGT1 Interface**

Bytes 8 to 15 of the received TLPs header destined for TRGT1 are made available on radm_trgt1_hdr_uppr_bytes irrespective of TLP translation. When radm_trgt1_hdr_uppr_bytes_valid is asserted your application should use radm_trgt1_hdr_uppr_bytes along with radm_trgt1_data.The radm_trgt1_hdr_uppr_bytes_valid is set only when an ATU match occurs for Single Address Location enabled region.
For example, if a received MsgD matches a Single Address Location enabled ATU region, bytes 8 to 15 of the MsgD header are available on radm_trgt1_hdr_uppr_bytes when radm_trgt1_hdr_uppr_bytes_valid is asserted, and radm_trgt1_hv is true. The available radm_trgt1_hdr_uppr_bytes can be concatenated with radm_trgt1_data and used by your application.

*Note: The width of radm_trgt1_ hdr_uppr_bytes is FLT_Q_ADDR_WIDTH bits. For radm_trgt1_hdr_uppr_bytes[63:32] to exist FLT_Q_ADDR_WIDTH must be 64. If 32 < FLT_Q_ADDR_WIDTH < 64, radm_trgt1_hdr_uppr_bytes[FLT_Q_ADDR_WIDTH-1:32] = 0.

**(b) CFG Handling (Upstream Port)**

The controller normally routes CFG TLPs (to the internal CDM or ELBI) without translation. The iATU only translates CFG0 TLPs that the controller has routed to the TRGT1. Inbound address translation for CFG0 TLPs operates in one of two matching modes as determined by the Inbound CFG0 MATCH_MODE field in the ATUCAP[10+i*18] register.

**Accept Mode**

The controller always accepts CFG0 TLPs even when the CFG bus number does not match the current bus number of the device. This mode follows that behavior. The routing ID of received CFG0 TLPs are ignored when determining a match.

**Routing ID Match Mode**

The operation is similar to “Outbound Basic Operation (Address Match Mode)”. The routing ID of the inbound CFG0 TLP must fall within the Base and Limit of the defined iATU region for matching to proceed. The iATU interprets the routing ID (Bytes1 8-11 of TLP header) as an address. This corresponds to the upper 16 bits of the address in MEM and I/O transactions.

**CFG Shift Feature**

Inbound CFG transactions routed to the Rx application interface can exist anywhere in address space, because the PCIe controller filter processes the routing ID (BDF) as bits [31:16] of an address. This BDF changes according on the PCIe bus topology. A compressor feature (CFG Shift Feature) can be enabled by setting the CFG Shift bit in the ATUCAP[10+i*18] register. Bits [15:12] of the third DWORD1 of CFG TLPs are reserved. The compressor feature uses this fact to reduce the memory requirement. This shifts/maps the BDF (bits [31:16] of the third header DWORD, which would be matched against the Base and Limit addresses) of the incoming CfgRd0/CfgWr0 down

to bits[27:12]of the translated address.

**CFG1 Transactions**
For CfgRd1/CfgWr1 transactions, the base and limit addresses could enclose the entire 32-bit 4GB memory space with the routing ID forming the upper 16 bits. The target address maps these CFG transactions to anywhere in application address space.

**(c) Optional Matching Fields**
In address and BAR match modes, a successful address/BAR match can be optionally gated by successful matching of the following programmable TLP header fields (per region):

- TD/TC/AT/ATTR/TH/PH/ST.
- MSG Code (MSG TLPs only).
- Function number (MEM, I/O, or CFG TLPs only).
- Virtual function number (MEM or I/O TLPs only).

For each of the previous fields in the ATUCAP[9+i*18] register, there is an associated Match Enable bit in the ATUCAP[10+i*18] register. Address translation only proceeds when compares on all enabled field are successful.

**(d) Response Code Feature**
When the Response Code field of the ATUCAP[10+i*18] register is set to a value other than 00, the controller uses it to determine the completion status field of completion TLPs sent in response to successfully matched non-posted TLPs. This can be set to unsupported request (UR) or completer abort (CA). When the error response field is set to 2b00, then the normal receive filter response for this TLP is used.

***Note:** Response Code feature is not available in regions where Single Address Location Enable is set.

**(e) iATU Inbound MSG Handling**
Inbound message (Msg/MsgD) transactions can use one of two matching modes:

- **Address Match Mode**
  The third and fourth header DWORDs are treated as an address and are compared against the iATU Region Base and Limit Address registers. For vendor defined messages this allows specific messages to be filtered into memory at the target address. The Upper Base address should be set to Bus Device Function (BDF) and Vendor ID. The Lower Base address can be used as a filter for specific messages.
- **Vendor ID Match Mode**
  This mode is relevant for ID-routed vendor defined messages. The iATU ignores the routing ID (BDF) in bits [31:16] of the third DWORD of the TLP header1, but compares it against the vendor ID in bits [15:0] of the third DWORD of the TLP header (bytes1 10 and 11). This allows vendor defined messages to be filtered against specific vendor IDs without needing to know the BDF number which might vary depending on the PCI topology. Bits [15:0] of the Region Upper Base register should be programmed with the required vendor ID as follows:
  Region Upper Base[15:8] =byte 10
  Region Upper Base[7:0] =byte 11
  The lower base and limit register should be programmed to translate TLPs based on vendor-specific information in the fourth DWORD of the TLP header.

■ For a proper understanding of messages, you should be familiar with Section 2.2.8, Message Request Rules of the PCI Express Base Specification.

**(f) Msg Type Match Mode**

Inbound message (Msg/MsgD) transactions can also use Msg Type matching mode. When this mode is enabled and “Single Address Location” is enabled, the iATU matches Msg TLP Type field with TYPE field of ATUCAP[9+i*18] register.
If “Fuzzy Type Match Mode” is also enabled, then any Msg received is matched (that is, Msg Type's sub-field r[2:0], which specifies the Message routing mechanism, is ignored).

****Caution:** The Message should be consumed by your application before the next message arrives as all messages go to the same address.

***Note:** If SINGLE_ADDR_LOC_TRANS_EN is set for any region, then you must ensure that the same TLP cannot be matched in any other region where SINGLE_ADDR_LOC_TRANS_EN is not set. If this *happens radm_trgt1_hdr_uppr_bytes* could have incorrect data.

**(g) Fuzzy Type Match Mode**

When enabled, the iATU relaxes the matching of the TLP type field against the expected type field so that:

- CfgRd0 and CfgRd1 TLPs are seen as identical. Similarly with CfgWr0 and CfgWr1.
- MWr, MRd and MRdLk TLPs are seen as identical.
- The routing field of MsgD TLPs is ignored.
- Atomic Ops TLPs - FetchAdd, Swap, and CAS are seen as identical.

For example, CFG0 in the type field in ATUCAP[9+*i**18] register matches against an inbound CfgRd0, CfgRd1, CfgWr0, or CfgWr1 TLP. To enable this feature, set the Fuzzy Type Match Mode bit of the ATUCAP[10+i*18] register.

**(h) FMT Translation**

The iATU automatically sets the TLP format field for three DWORDs when it detects all zeroed in the upper 32 bits of the translated address. Otherwise it sets it to four DWORDs when it detects a 64-bit address (when there is a 1 in the upper 32 bits of the translated address). When the original address and the translated address are of a different format then the iATU ensures that the TLP header size matches the translated address format.

**(i) Invert Feature**

Normally an address match on an inbound TLP occurs when the untranslated address is in the region bounded by the Base address and Limit address. When the Invert feature is activated, an address match occurs when the untranslated address is not in the region bounded by the Base address and Limit address. This feature is activated by setting the INVERT_MODE field of the ATUCAP[10+i*18] register.

## (5) Inbound Programming

This section discusses the register programming interface and provides details of programming examples.

**(a) iATU Inbound Programming Overview**

You can access the iATU registers through the DBI interface or through BAR Matched Mem/IO requests.
The following registers are used for programming the iATU.

**Table 104.15 iATU Register Map**

| Address | Label (i=0 to CX_ATU_NUM_INBOUND_REGIONS-1) | Description |
|---|---|---|
| 0x100 | ATUCAP[9+*i**18] | Region Control 1 |
| 0x104 | ATUCAP[10+*i**18] | Region Control 2 |
| 0x108 | ATUCAP[11+*i**18] | Lower Base Address |
| 0x10c | ATUCAP[12+*i**18] | Upper Base Address |
| 0x110 | ATUCAP[13+*i**18] | Limit Address |
| 0x114 | ATUCAP[14+*i**18] | Lower Target Address |
| 0x118 | ATUCAP[15+*i**18] | Upper Target Address |
| 0x11C | ATUCAP[16+*i**18] | Region Control 3 |
| 0x120 | ATUCAP[17+*i**18] | Upper Limit Address |

**Table 104.16 iATU Register Address Bus Layout**

| **Register** | **31-20** | **19** | **18-17** | **16-9** | **8** | **7-2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| Value | Not used | 0 | Reserved | Region | Inbound/Outbound | Register Address | 0 | 1 |

**Table 104.17 Example: Address Table for Accessing iATU Inbound Region 1 Configuration Registers**

| **Address** | **Register** | **Register Bit Value** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **31-20** | **19** | **18-17** | **16-9** | **8** | **7-2** | **1** | **0** |
| 0x300 | Region Control 1 | 0 | 0 | 0 | 1 | 101 | | | |
| 0x304 | Region Control 2 | 0 | 0 | 0 | 1 | 105 | | | |
| 0x308 | Lower Base Address | 0 | 0 | 0 | 1 | 109 | | | |
| 0x30C | Upper Base Address | 0 | 0 | 0 | 1 | 10D | | | |
| 0x310 | Limit Address | 0 | 0 | 0 | 1 | 111 | | | |
| 0x314 | Lower Target Address | 0 | 0 | 0 | 1 | 115 | | | |
| 0x318 | Upper Target Address | 0 | 0 | 0 | 1 | 119 | | | |
| 0x31C | Region Control 3 | 0 | 0 | 0 | 1 | 11D | | | |
| 0x320 | Upper Limit Address | 0 | 0 | 0 | 1 | 121 | | | |

**(b) Inbound Programming Examples**

*Address Match Mode Example*

Define Inbound Region 0 as, MEM region matching TLPs with addresses in the range 0x00010000 to 0x0005ffff mapped to 0x1000_0000_2000_0000 - 0x1000_0000_2004_ffff in your application memory space.

1. Setup the Region Base and Limit Address Registers.
   Write 0x00010000 to Address {0x108} to set the Lower Base Address.
   Write 0x00000000 to Address {0x10C} to set the Upper Base Address.
   Write 0x0005ffff to Address {0x110} to set the Limit Address
2. Setup the Target Address Registers.
   Write 0x20000000 to Address {0x114} to set the Lower Target Address.
   Write 0x10000000 to Address {0x118} to set the Upper Target Address.
3. Configure the region through the Region Control 1 Register.
   Write 0x00000000 to Address {0x100} to define the type of the region to be MEM.
4. Enable the region.
   Write 0x80000000 to Address {0x104} to enable the region in address match mode.

*BAR Match Mode Example*

Define Inbound Region 2 as MEM region matching BAR4 (BAR match mode) mapping to 0x8000_0000_2000_0000 in your application memory space.

1. Setup the Target Address Registers.
   Write 0x20000000 to Address {0x514} to set the Lower Target Address.
   Write 0x80000000 to Address {0x518} to set the Upper Target Address.

2. Configure the region through the Region Control 1 Register.
   Write 0x00000000 to Address {0x500} to define the type of the region to be MEM.

3. Enable the region for BAR Match Mode.
   Write 0xC0000400 to Address {0x504} to enable the region for BAR match mode for BAR#4.

**Note**: Default number of inbound regions is 2 (CX_ATU_NUM_INBOUND_REGIONS), so region 2 is not accessible by default.

****Attention:* EP port**: Defined MEM or I/O regions must be inside an enabled BAR range.
**RC port**: Defined MEM or I/O regions must either match a BAR or be outside of the base and limit ranges defined for the port in the Type 1 configuration header.

### 104.3.9 DMA Transfer

This section describes the DMA transfer using the DMAC (PCIEC-DMAC) incorporated in this module.

#### (1) Overview of DMA Transfer

The RC system CPU, or the EP application CPU, can offload the transferring of large blocks of data to the embedded DMA controller, leaving the CPU free to perform other tasks. You can configure the DMA to have one to eight read channels and one to eight write channels. It can simultaneously perform the following types of memory transactions (as shown in Figure 104.12):

- **DMA write**
  Transfer (copy) of a block of data from local (application) memory to remote (link partner) memory.

- **DMA read**
  Transfer (copy) of a block of data from remote (link partner) memory to local (application) memory.

Therefore the DMA supports full duplex operation, processing read and write transfers at the same time, and in parallel with normal (non-DMA) traffic. Upon completion of a DMA transfer or an error, the DMA optionally interrupts the local CPU or sends an interrupt MWr (IMWr) to the remote CPU. The DMA is highly configurable and you can program it using the local DBI or over the PCIe wire.

In linked list mode, the DMA fetches the transfer control information (called channel context) for each transfer (block), from a list of DMA elements that you have constructed in local memory.

Table 104.18 identifies the limitations when using the DMA module.

PCIEC-DMAC has features shown in Table 104.18.

**Table 104.18 DMA Module Features**

| No | DMA Module Feature | Support | Note |
|---|---|---|---|
| 1 | Number of Read Channels | 8 ch | |
| 2 | Number of Write Channels | 8 ch | |
| 3 | AXI Support | Y | |
| 4 | Native Controller Support (no AXI) | N | Not support |
| 5 | Linked List Mode | Y | Independent on implementation setting. |
| 6 | Linked List Descriptor Alignment | DWORD | Independent on implementation setting. |
| 7 | Scatter Gather Support (by using Linked Lists) | Y | Independent on implementation setting.<br>Perform Scatter Gather using Linked lists function. |
| 8 | Full Duplex Read/Write Channel Operation | Y | Independent on implementation setting.<br>Read channel and Write channel are independent |
| 9 | Separate Tag Pool For DMA Traffic | Y | Independent on implementation setting.<br>In the packet of DMA, use the managed tag separately from PCIe tag.<br>(There is no need to be aware of tag duplication) |
| 10 | SAR/DAR Address Alignment | BYTE | Independent on implementation setting. |
| 11 | Transfer Size | 1byte - 4 GB | Independent on implementation setting. |
| 12 | AXI Address Bus Width | 32 | |
| 13 | AXI Databus Width | 256 | |

| | | | |
|---|---|---|---|
| 14 | Independent Internal, Master and Slave Databus Widths | Y | Internal bus data width and Master/Slave data width are independent.<br><br>Data width of Master and Slave is the same. |
| 15 | Independent Controller and AXI Clocks | Y | Clocks of AXI and DMA are independent.<br><br>(AXI S0D2_HSCφ, DMA module 500MHz(pclk)) |
| 16 | Multifunction Support | Y | Independent on implementation setting.<br><br>Support 2 function. |
| 17 | DMA | Y | |
| 18 | ARI | N | Not support |
| 19 | SR-IOV | N | Not support |
| 20 | Latency Tolerance Reporting (LTR) Support | Y | |
| 21 | OBFF Support | N | Not support |
| 22 | Address Translation Services (ATS) Support | N | Not support |
| 23 | Atomic Ops Support | - | Not support |
| 24 | TLP Processing Hints (TPH) Support | N | Not support |
| 25 | TLP Prefix Support (Including PASID) | - | Not support |
| 26 | Programmable Interrupt Generation | Y | Independent on implementation setting.<br><br>Function to generate interrupt with MWr after DMA transfer completed. |
| 27 | Reordering Of DMA Read Transfer Completions | Y | Independent on implementation setting.<br><br>Function to re-order and return it on the AXI bus even if Completion for Read Request is returned out of bound. |
| 28 | ID Based Ordering (IDO) Signaling Support | N | Not support |
| 29 | Datapath and RAM Protection | N | Not support |
| 30 | RAM Read Enable for Low-power Optimization | Y | Independent on implementation setting. |
| 31 | Automatic Flushing/Reset of Bridge (AXI Configurations) When Link Goes Down | Y | Independent on implementation setting. |
| 32 | Automatic Flushing of Slave (AXI Configurations) When Link is Not Up | Y | Independent on implementation setting. |

**Table 104.19 DMA Module Limitations**

| Limitation | Note |
|---|---|
| Memory DMA Traffic Only | I/O and Type 0 or Type 1 configuration DMA transfers are not supported. |

### (2) Architecture Overview

A DMA write and read channel operate independently to maximize the performance of the DMA read and write data transfers over the PCIe link. When you configure the DMA with multiple read channels, then it uses a weighted round robin (WRR) arbitration scheme to select the next read channel to be serviced. The same applies when you have multiple write channels.

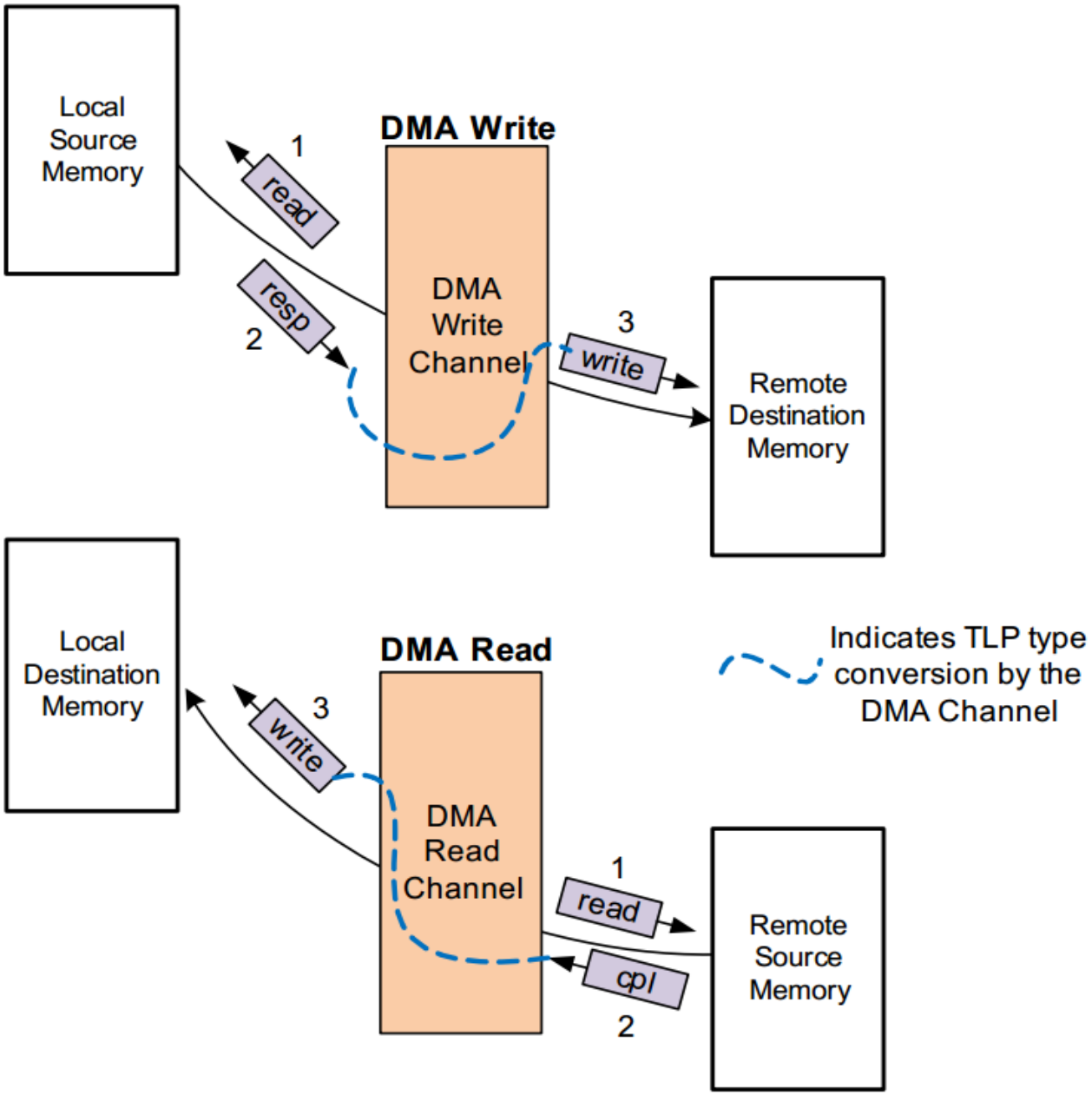



**Figure 104.12 System Level View of DMA**

**DMA Write Transfer**

The DMA injects multiple MRd requests of size less than or equal to Max_Payload_Size into the inbound request path, directed toward the local application. The DMA converts the read responses into MWr TLPs that it then transmits to the remote link partner. When the DMA data transfer is complete the CPU is notified. For more details, see “Interrupts and Error Handling”. For a write transfer, the SAR is the address of the local memory, and the DAR is the address of the remote memory.

**DMA Read Transfer**

The DMA injects multiple MRd requests of size less than or equal to the minimum of {Max_Read_Request_Size, Max_Payload_Size} into the outbound request path, directed toward the remote link partner. The DMA converts the read responses into write requests that it then transmits to the local application. When the DMA data transfer is complete, the CPU is notified. For more details, see “Interrupts and Error Handling”. For a read transfer, the SAR is the address of the remote memory, and the DAR is the address of the local memory.

**RAM Requirements**

The DMA requires two RAMs.

**Registers and Context Memory**

You program each DMA transfer (channel) using the registers described in “DMA Software Register Map”. The registers that are specific to each channel are called the channel context registers. As the transfer progresses, the DMA updates these registers. You program the channel context information and other DMA control registers directly with the local CPU through the DBI or remotely over the PCIe link.

Some of the registers are implemented in a high-speed wide interface context RAM. If you implement your RAMs external to the PCIe controller, you must connect this RAM through the external RAM interfaces.

### (3) Interrupts and Error Handling

The registers referenced in Figure 104.13 and Figure 104.14 that are related to error and interrupt handling are described in “DMA Software Register Map”.
The DMA generates two interrupts per channel:

- **Done**: The DMA successfully completes the transfer.
- **Abort**: The DMA fails to complete the transfer, or an error occurs during the transfer.

The interrupts are signaled to the software on your CPU, using one or both of the following mechanisms:

- Locally through the edma_int[CC_NUM_DMA_RD_CHAN+CC_NUM_DMA_WR_CHAN-1:0] bus.
- Remotely using a posted memory write (IMWr), which can be interpreted as an MSI when directed toward the RC.
  - There are two programmable IMWr addresses per channel, one each for the done and abort interrupts. For MSI, you must program all IMWRr address registers with the same MSI address, as PCIe only supports a single MSI address per function.
  - A single IMWr data register is used for both types of interrupts, so you must read DMACAP23 (or DMACAP10) to identify the interrupt type.
- The relaxed ordering (RO) bit is automatically set to zero in the TLP header of all IMWr’s so that subsequent MWr TLPs do not arrive at the remote device before the IMWr.

The interrupt handling mechanism is different for linked list (LL) mode (than non LL mode), and there are also some differences between the read and write channels.

**Non Linked List Mode Interrupt Handling**

You enable the local and remote interrupts through the local and remote interrupt enable (LIE and RIE) bits: DMACAP[37+i*16].LIE and DMACAP[37+i*16].RIE.

In the write channel, there is only one error condition that results in an abort interrupt. For more details, see “Linked List Mode”. You mask, clear, and read the status of each of the two interrupts (done and abort) through the DMA interrupt registers as indicated in Figure 104.13.
In the read channel, there are five error conditions that results in an abort interrupt. For more details, see “Linked List Mode”. You mask and clear each of the two interrupts (done and abort) through the DMA interrupt registers as indicated in Figure 104.14. However, you can read the status of each of the five abort errors (that contribute to the abort interrupt) through DMACAP26 and DMACAP27.

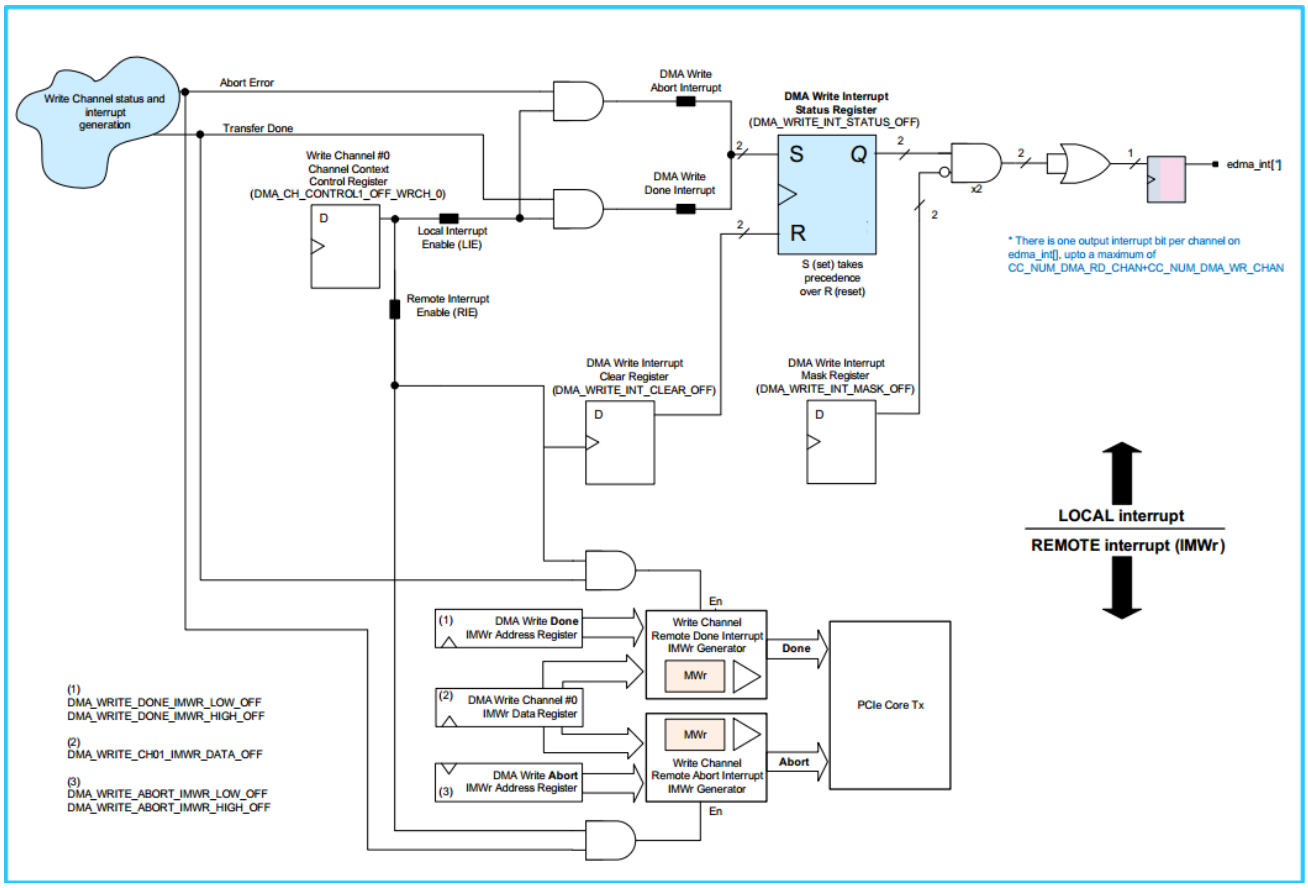



**Figure 104.13 Write Interrupt Generation - Non Linked List Mode - Shown For Write Channel #0**

****Hint:** If you want a remote interrupt and not a local interrupt then:

- Set LIE and RIE
- Mask the local interrupt using the mask register

This allows you to poll the status register to distinguish a DONE from an ABORT.

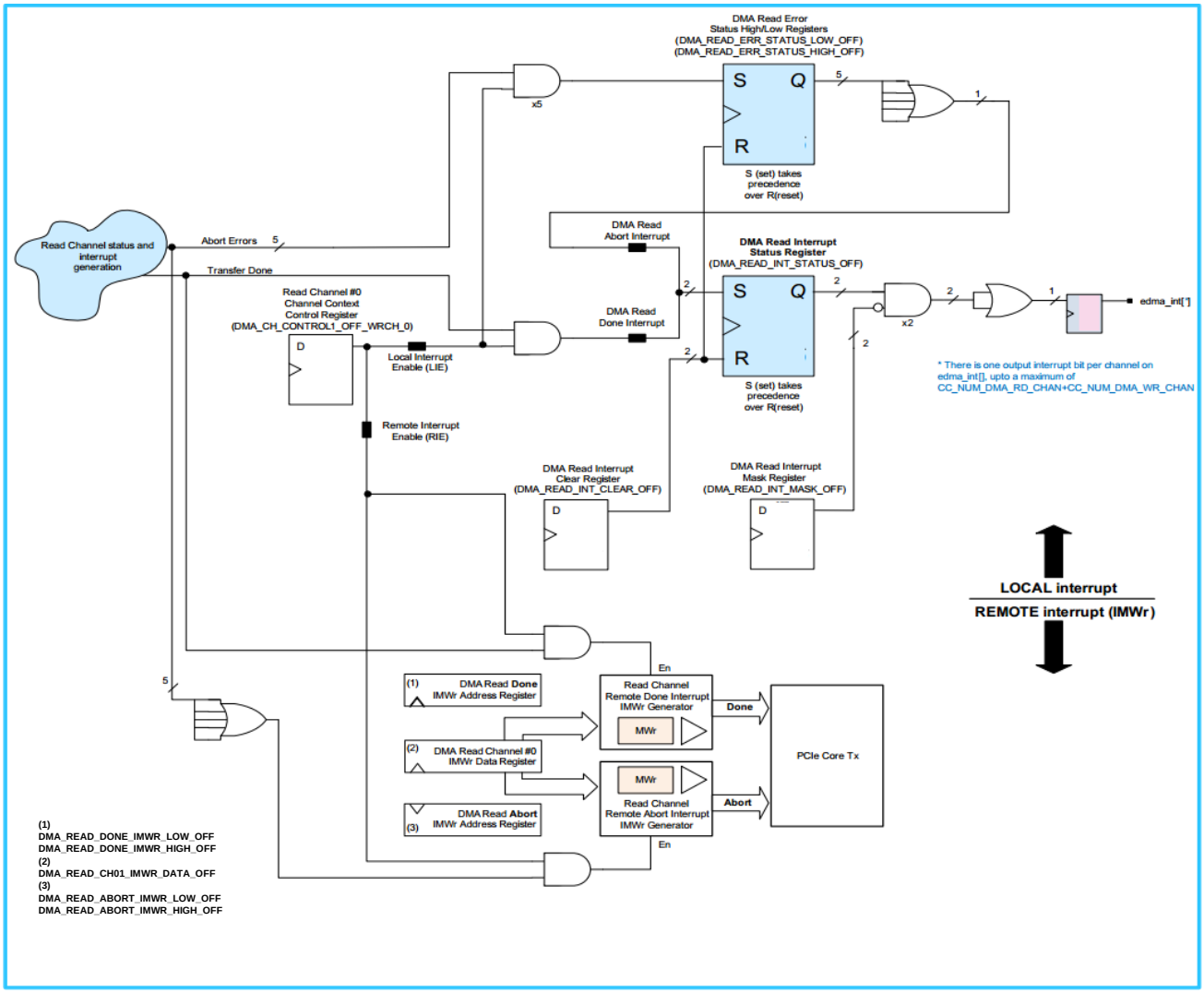



**Figure 104.14 Read Interrupt Generation - Non Linked List Mode - Shown For Read Channel #0**

### Linked List Mode Interrupt Handling

The LIE and RIE bits in the LL element enable the channel done interrupts (local and remote). The LLLAIE and LLRAIE bits of the DMACAP22/ DMACAP28 registers enable the channel abort interrupts (local and remote). In the write channel, there are two error conditions that results in an abort interrupt. For more details, see “Linked List Mode”. You mask and clear each of the two interrupts (done and abort) through the DMA interrupt registers as indicated in Figure 104.15. You can read the status of each of the two abort errors (that contribute to the abort interrupt) through the DMACAP13 register.

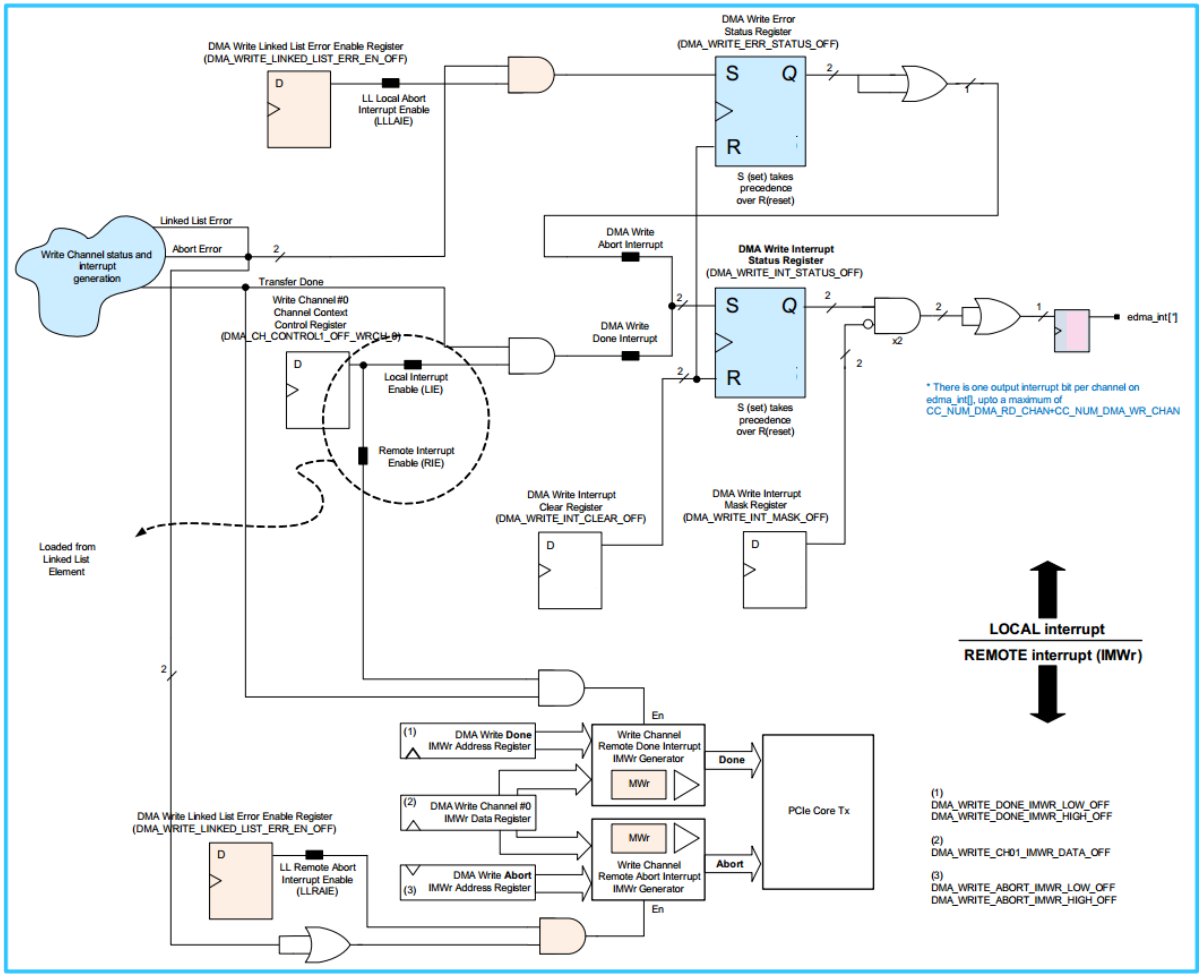



**Figure 104.15 Write Interrupt Generation - Linked List Mode - Shown For Write Channel #0**

In the read channel, there are six error conditions that results in an abort interrupt. For more details, see “Linked List Mode. You mask and clear each of the two interrupts (done and abort) through the DMA interrupt registers as indicated in Figure 104.16. You can read the status of each of the six abort errors (that contribute to the abort interrupt) through the DMACAP26 and DMACAP27 registers.

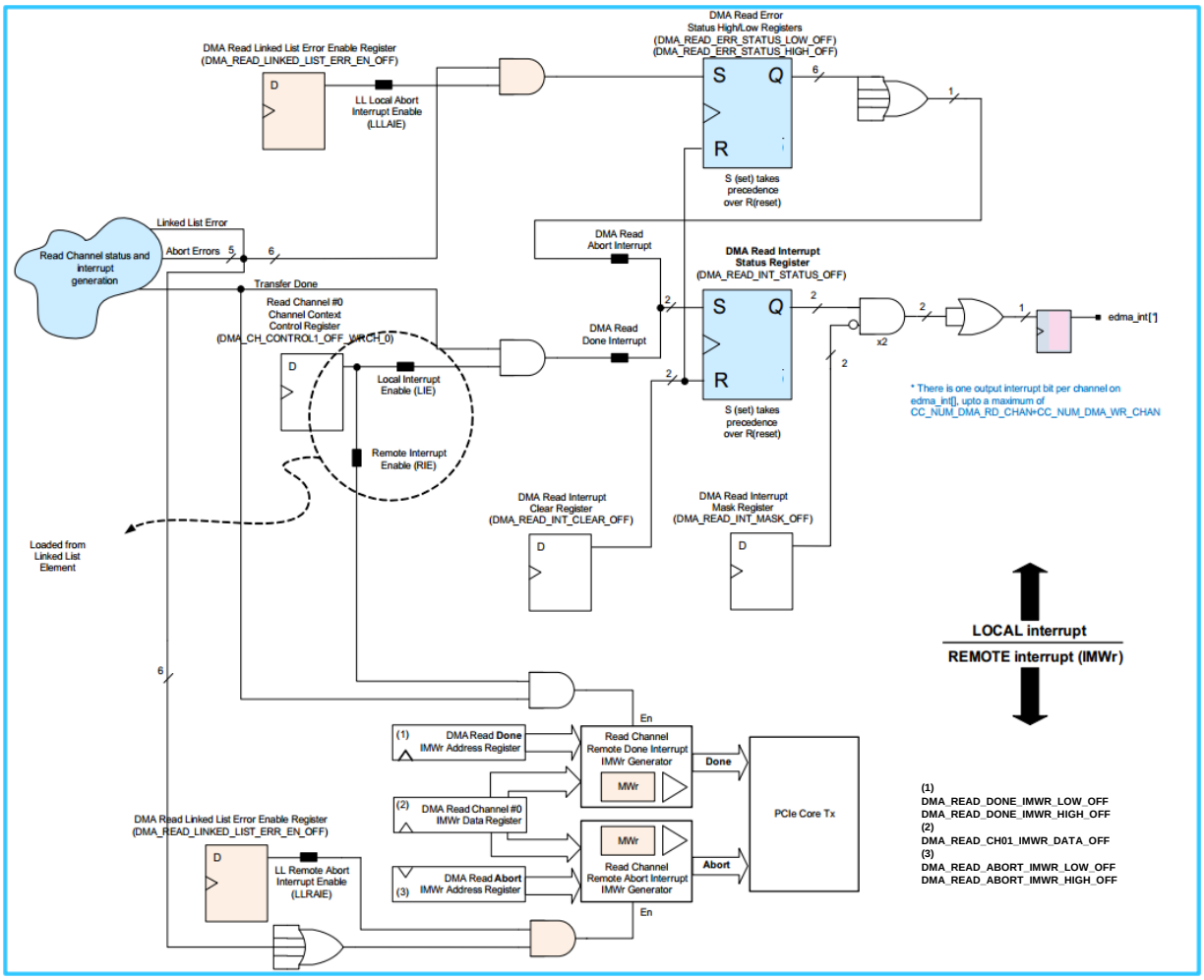



**Figure 104.16 Read Interrupt Generation - Linked List Mode - Shown For Read Channel #0**

***Note:** In non-linked list mode, LIE acts as a global switch. However when in linked list mode, LIE is just local to the current linked list element and the global switch is LLLAIE.

****Attention: Software Programming of an Endpoints LIE and RIE Bits for Linked List Transfers.** If the DMA driver is running on the host and the interrupt service routine is reading local interrupts to determine if the transfer is successful, then you must set LIE and RIE in the same element and you should mask or ignore the local interrupt pin. Setting RIE and LIE in element A followed by RIE (only) in element B is not a verified usage scenario.

### (4) Linked List Mode

The DMA provides a linked list (LL) mode to efficiently move data from source to destination with minimal intervention from the local CPU. This mode provides an alternative to programming the DMA multiple times to transfer multiple blocks of data. The programming information (address, size, and so on) for each block of memory is pre-programmed by your software into a LL element (also known as a descriptor) in local memory. Each element (called a data element) in the LL structure (called a transfer list) can transfer up to 4GB of data.

You enable LL operation for a channel, by setting the LLE field of the DMACAP[37+*i**16] register to 1. You can enable LL mode independently for each channel. When you enable LL for more than one channel, then you must have a separate LL structure in local memory for each channel. Your application must produce the LL element structure in local memory as shown in Figure 104.17. Normally, all of the elements are contiguous (one after the other) in memory, and each element has six DWORDs containing the information about the block of data to be transferred. You program the channel context

registers (DMACAP[43+i*16] and DMACAP[44+i*16]) with the location of where you have placed the LL element structure in local memory.

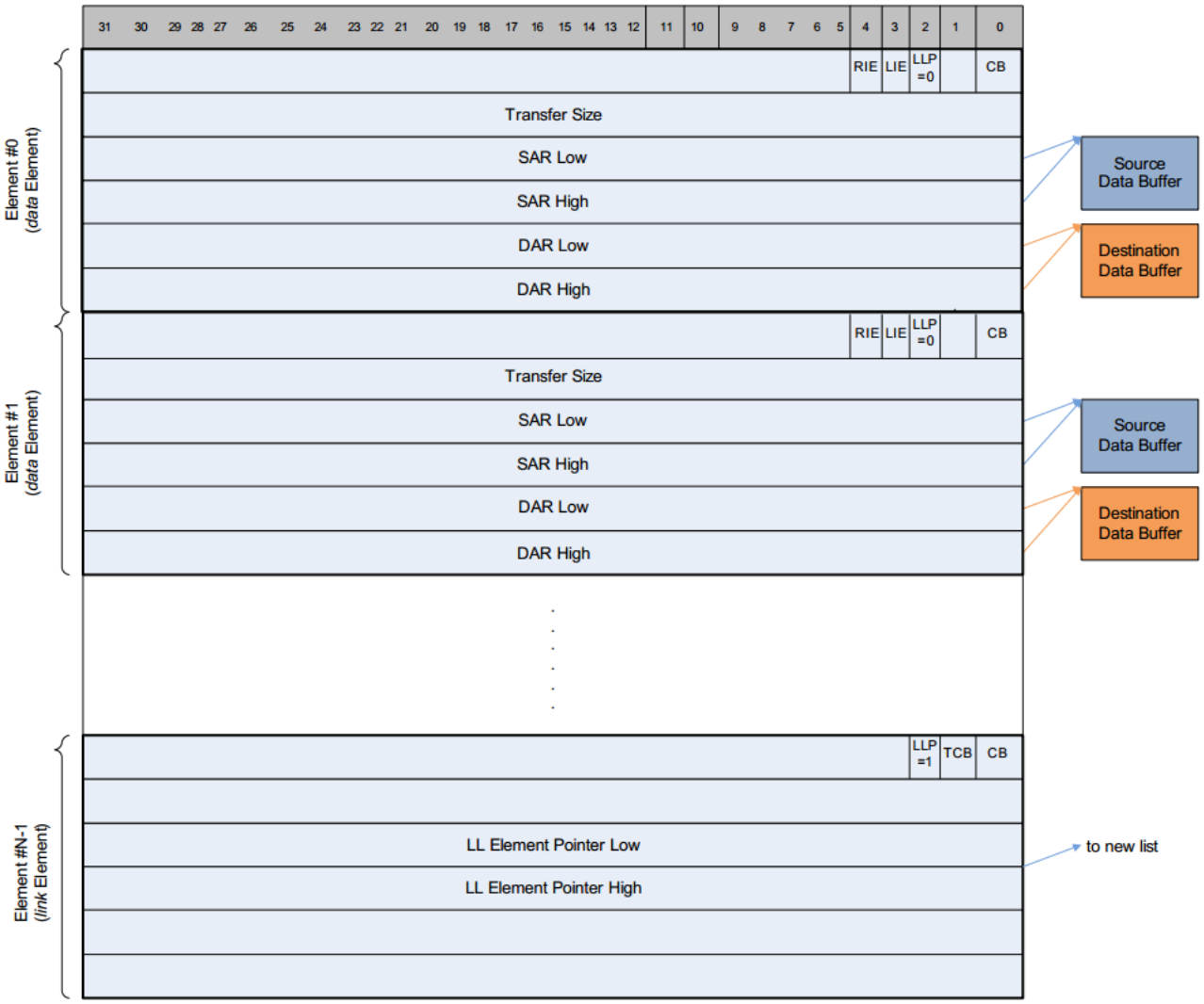



**Figure 104.17 Linked List Element/Descriptor Structure in Local Memory with N Elements.**

When you start the DMA transfer (by writing to the DMA Write Doorbell Register DMACAP3 or DMA Read Doorbell Register DMACAP7), the DMA read (consumes) each element from local memory, and loads the information (SAR, DAR, size, and so on) from that element into the channel context registers in the DMA. These channel context registers determine the operation of the channel that the DMA controller is currently servicing. The DMA then proceeds to transfer the block of data (as defined by the element), and when it is finished, reads the next element from local memory. Normally, all of the elements are contiguous (one after the other) in memory, with the starting address defined in the channel context DMA Linked List Pointer Low Register DMACAP[43+i*16].

When you want to jump in local memory to another element list (or recycle the consumed elements), then you set the LLP bit in the element (for example, link element #N-1 in Figure 104.17), specify the location of the next element structure using the LL Element Pointer DWORDs (as indicated in Figure 104.17), and, set TCB to 1 (for recycling) or to 0 (to jump to another list).

**Relationship Between Element DWORDs and Channel Context Registers**

Notice the similarity between a data element and the DMA Channel Context registers for each channel. Each element has six DWORDs as in Figure 104.17. There are eight channel context registers (DWORDs) for a channel. The DMA loads the six element DWORDs into the following channel context:

- CB, LLP, LIE, and RIE fields of the DMA Channel Control 1 register.

- DMA Transfer Size.
- DMA SAR Low and DMA SAR High.
- DMA DAR Low and DMA DAR High

The definitions of the element DWORD bit fields are the same as the DMA Channel Context registers described in the “DMA Software Register Map”, with the exception of the LIE and RIE bits. The LIE and RIE bits in a LL element, only enable the done interrupt. In non-LL mode, the RIE and LIE bits (in the channel context registers) enable the done and abort interrupts.

When the LLP field of the first DWORD in the element is set to 1, the element is a link element. The DMA only loads the following information into the channel context registers:

- CB, TCB, and LLP bits of the first DWORD. The LIE and RIE bits are not defined in a link element.
- LL Element Pointer (3rd and 4th DWORDs) into the DMA Linked List Pointer registers (DMACAP[43+i*16] and DMACAP[44+i*16]).

***Caution:* Linked List Operation**

When you enable linked list operation (DMACAP[37+i*16].LLE=1’b1), the DMA overwrites the following DMA channel context registers with information from linked list data elements:

- ☐ The CB, LLP, LIE, and RIE fields of the DMA Channel Control 1 register.
- ☐ DMA Transfer Size.
- ☐ DMA SAR Low & High.
- ☐ DMA DAR Low & High

The structure of a link element is different to that of the data element. A data element has no TCB field. A link element has no LIE or RIE fields. It has no SAR, DAR, or Transfer Size DWORDs, but has LL Element Pointer DWORDs instead of the SAR DWORDs.
The LIE and RIE bits in a LL element, only enable the done interrupt. In non-LL mode, the RIE and LIE bits enable the done and abort interrupts. For more information, see “Interrupts and Error Handling”.

**Descriptor Early Fetch (AXI Bridge Configuration)**

In AXI bridge configuration, the DMA serializes the next descriptor fetch with the previous descriptor’s last data fetch. This allows the next descriptor to be loaded into DMA’s context memory just in time. Early descriptor fetch reduces the number of idle cycles between subsequent descriptor fetches introduced because of high round-trip read-to-completion latencies in scenarios where the descriptor’s transfer size is small.

The descriptor is fetched and loaded into AXI Bridge’s master completion buffer. While the previous descriptor’s CPL and MWr transactions are in progress, DMA back-pressures the descriptor loaded in the master completion buffer. It is recommended to have the configuration respecting the following relationship to avoid any potential performance issues:

CC_MAX_MSTR_TAGS_AXI > (CC_NUM_DMA_RD_CHAN + CC_NUM_DMA_WR_CHAN)

The early fetched descriptor completes concurrently with previous descriptor’s data completion. If previous descriptor transfer aborts due to an error in any data completion, the early-fetched descriptor is flushed. Figure 104.18 and Figure 104.19 describe Read/Write channel operation example for a scenario where each descriptor transfer size is 1 TLP.

***Note:** You can revert to legacy descriptor fetch by setting the hidden parameter CC_DMA_EFETCH_EN to 0.

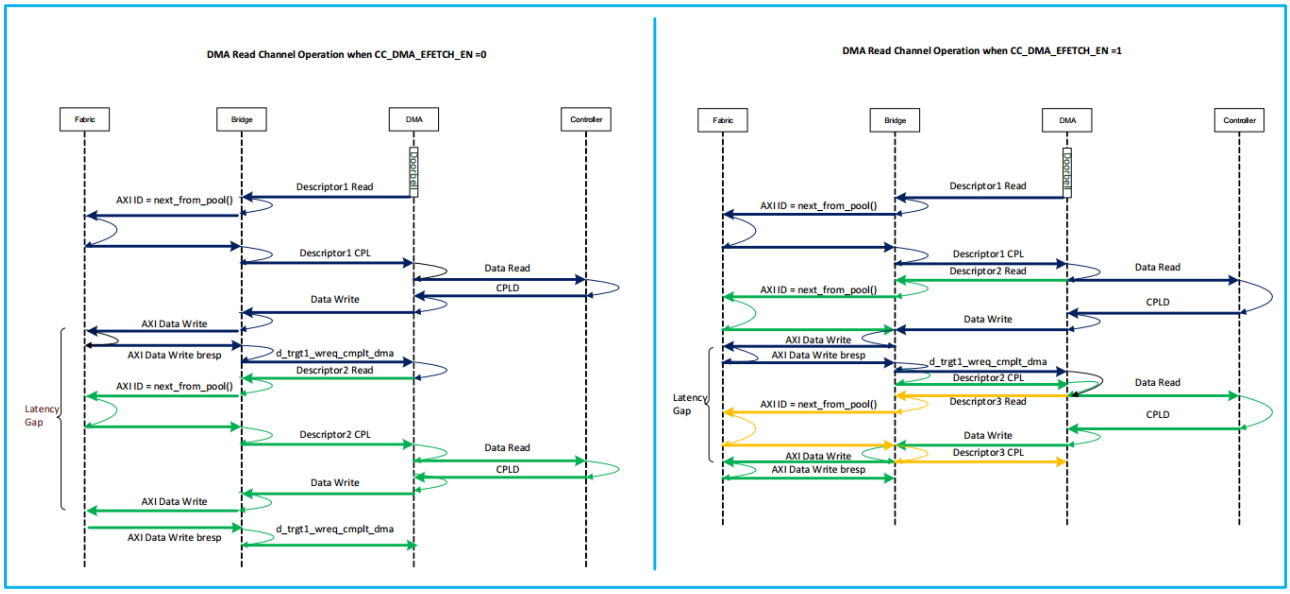



**Figure 104.18 Read Channel Operation Example: Before and After Early Fetch Optimization**

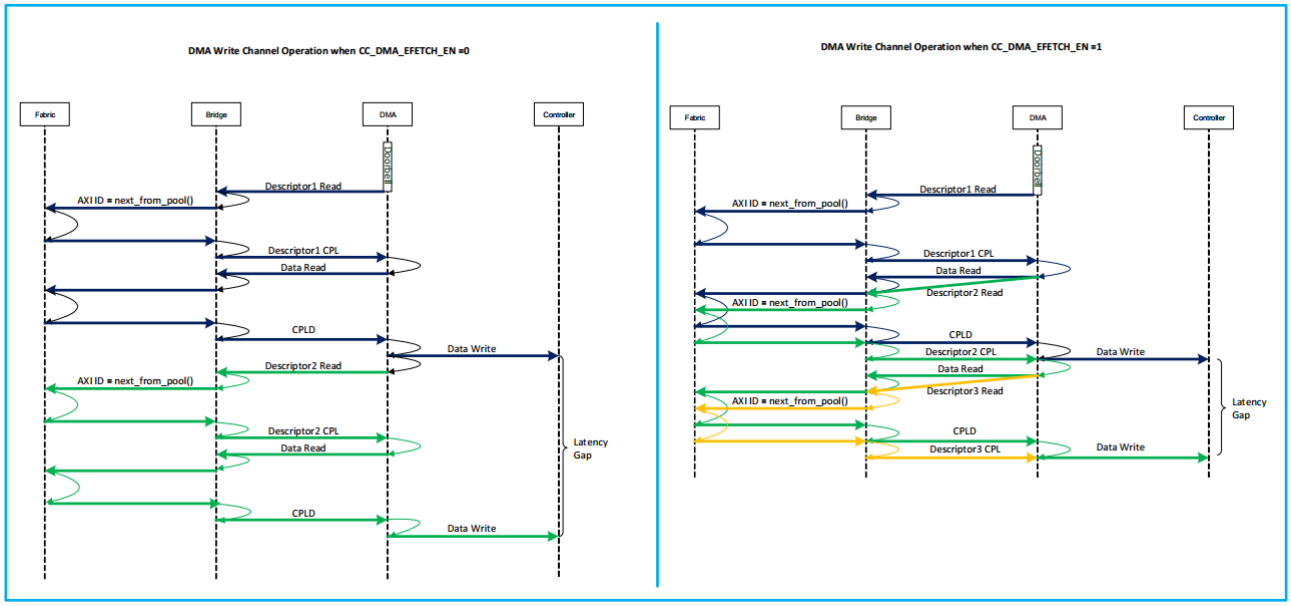



**Figure 104.19 Write Channel Operation Example: Before and After Early Fetch Optimization**

### (5) Using the DMA

This section describes how to use the DMA. The topics for this section are:

- "Source and Destination Address Registers (SAR, DAR)".
- "DMA Transfer Size Register".
- "Starting The DMA Transfer".
- "Detecting the End of The DMA Transfer".
- "Stopping the DMA Transfer (Software Stop)".

- "TLP Generator Header Control".
- "DMA Software Register Map".
- "Programming Examples".

You can configure the DMA to have from one to eight read channels and one to eight write channels. It can simultaneously perform the following types of memory transactions:

- DMA write
  Transfer (copy) of a block of data from local memory to remote memory.
- DMA read
  Transfer (copy) of a block of data from remote memory to local memory.

The DMA supports full duplex operation, processing read and write transfers at the same time, and in parallel with normal (non-DMA) traffic. Upon completion of a DMA transfer or an error, the DMA optionally interrupts the local CPU or sends an interrupt MWr (IMWr) to the remote CPU. The DMA is highly configurable and you can program it using the local DBI, or over the PCIe wire.
The channels are named read channel 7…0 and write channel 7...0. Each channel is programmed using a number of registers.

**(a) Source and Destination Address Registers (SAR, DAR)**

The DMA channel context SAR and DAR registers (DMACAP[39+*i**16]/ DMACAP[40+*i**16]/ DMACAP[41+*i**16]/ DMACAP[42+*i**16]) provide support for remote-to-local, and local-to-remote PCIe address mapping. You program the start of the local and remote data buffers using these registers, and the DMA increments the SAR and DAR as the DMA transfer progresses. For a write transfer, the SAR is the address of the local memory, and the DAR is the address of the remote memory, as shown in Figure 104.20.

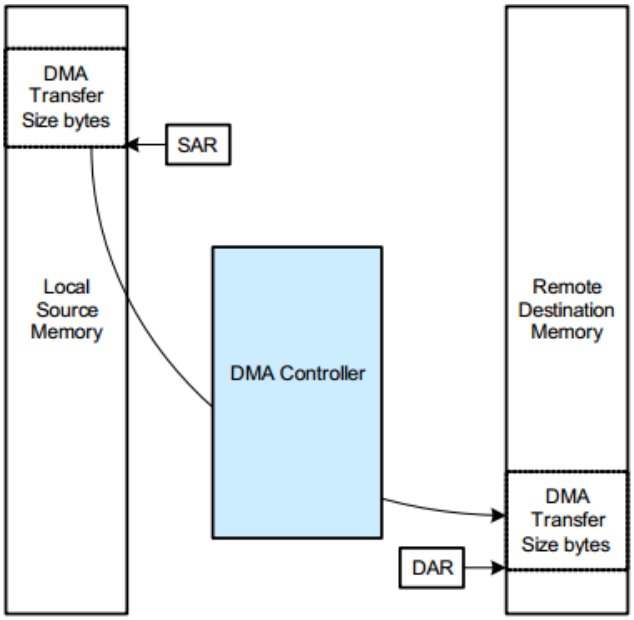



**Figure 104.20 Write Transfer: SAR and DAR for Write Channel 0**

For a read transfer, the SAR is the address of the remote memory, and the DAR is the address of the local memory, as shown in Figure 104.21.

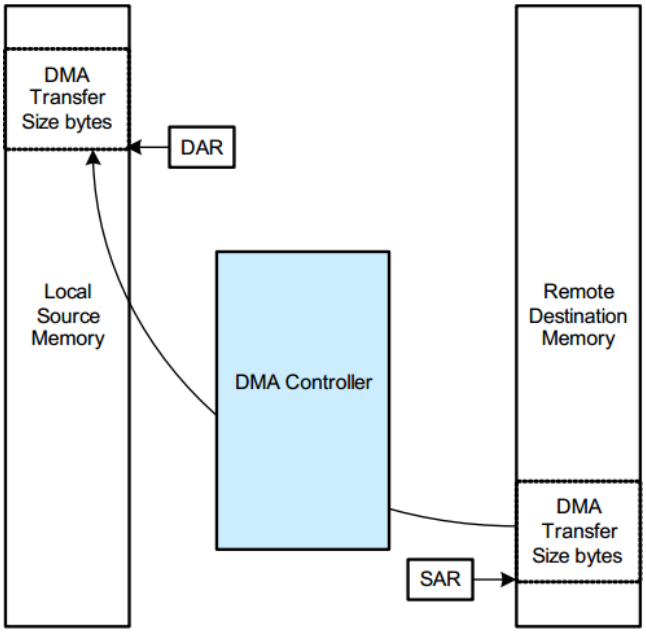



**Figure 104.21 Read Transfer: SAR and DAR for Read Channel 0**

The presence of the DMA controller does not affect:

- Normal filtering rules for inbound TLPs (for example, BAR checking in an Endpoint, as described in “Internal Address Translation Unit (iATU)”).
- The operation of any internal or external address translation, as described in “Receive Filtering”.

****Caution:** When you do not want the iATU to translate outbound requests that are generated by the internal DMA module, then you must implement one of the following approaches:

- Ensure that the combination of DMA channel programming and iATU control register programming, causes no translation of DMA traffic to be done in the iATU.
- Activate the DMA bypass mode to allow request TLPs which are initiated by the DMA controller to pass through the iATU untranslated. You can activate DMA bypass mode by setting the DMA Bypass field of the iATU Control 2 Register (ATUCAP[1+*i**18]).

### (b) DMA Transfer Size Register

You program the DMA transfer size using the DMA Transfer Size Register (DMACAP[38+*i**16] or DMACAP[46+*i**16]). The maximum DMA transfer size is 4GB, and the minimum transfer size is one byte (0x1). The DMA decrements the value in this register as the DMA transfer progresses. When all bytes are successfully transferred, the value in this register is zero. In LL mode, the DMA overwrites this register with the corresponding dword of the LL element.

You can read this register to monitor the transfer progress. However, in some scenarios there is a delay before the controller updates this register. For example, when less than three channels are doorbelled, this register is only updated after a descriptor finishes (LL mode), or the transfer ends (non-LL mode).

### (c) Starting The DMA Transfer

After you program the DMA controller registers (including writing to the DMA Read Engine Enable or DMA Write Engine Enable register), you start a DMA transfer by writing the channel number to the Doorbell Number field of the DMA Write Doorbell Register (DMACAP3) or DMA Read Doorbell Register (DMACAP7). You can program and start both a read and a write transfer at the same time. The DMA supports full duplex operation, processing read and write transfers at the same time and in parallel with normal (non-DMA) traffic. You can program and start any number of channels in the DMA controller sequentially or simultaneously.

- **Sequentially**: configure and start a channel, and then configure and start another channel
- **Simultaneously**: configure multiple channels, and then start multiple channels

****Caution:** You must not write to any of the context registers for a particular channel after you start the channel by writing the channel number to the Doorbell Number field of the DMA Write Doorbell Register (DMACAP3) or DMA Read Doorbell Register (DMACAP7)**.**

**(d) Detecting the End of The DMA Transfer**

If a DMA transfer proceeds without any errors, it stops automatically when finished.

**Detecting End of Transfer without Errors**
The normal end of a DMA transfer is detected by any of the following methods:

- Done local interrupt (pin) asserted.
- Done remote interrupt (IMWr) received.
- Channel status field of the Channel Control 1 register is stopped, and the DMA Transfer Size register is 0x0.
- Polling of the DMA Write Interrupt Status Register (DMACAP10) or DMA Read Interrupt Status Register (DMACAP23).

**Detecting End of Transfer with Errors**

The abnormal end of a DMA transfer is detected by any of the following methods:

- Abort local interrupt (pin) asserted.
- Abort remote interrupt (IMWr) received.
- Polling of the DMA Write Interrupt Status Register (DMACAP10) or DMA Read Interrupt Status Register (DMACAP23).
- Channel Status field of the Channel Control 1 register is Halted. When the DMA controller detects an error, it forces the DMA to stop issuing requests for the channel. It also sets the channel status field of the Channel Control 1 register to Halted, generates an abort interrupt (if enabled), and sends an abort IMWr (if enabled). The DMA Transfer Size register indicates the remaining number bytes to be transferred, except when there is an AXI write error during a DMA read transfer.
- Channel Status field of the Channel Control 1 register is stopped, and the DMA Transfer Size register is not 0x0. You have prematurely stopped this channel as described in “Stopping the DMA Transfer (Software Stop)”.

**(e) Stopping the DMA Transfer (Software Stop)**

You can manually abort (stop) the DMA transfer by writing the channel number to the Doorbell Number field and writing 1 to the Stop field in DMACAP3. This causes the DMA to:

- Place the channel in a Stopped state.
  The channel Status field in DMACAP[37+*i**16] is stopped and the value in DMACAP[38+*i**16]0 is not 0x0.
- Wait for all outstanding pending transactions.
- Assert the abort interrupt (if it is enabled) in DMACAP10.

You might do this as part of error handling which is described in “Error Handling Assistance by Remote Software”, which is only necessary during software development if you incorrectly program the DMA write channel DAR.

You might also do this as part of a function level reset (FLR). FLR does not directly affect the DMA transfer so you must manually stop the DMA transfer before initiating an FLR.

***Caution:* Use This Feature With Caution**
You should not use the STOP feature in operational mode. It is only a debug feature and has associated hazards.

- Before setting the Stop bit.
  - You must read the channel status field (CS) of the DMA Channel Control 1 register to ensure that the corresponding channel is running (transferring data).
  - To eliminate the possibility of a race condition between these two actions (read and write), you should confirm the presence of the abort bit in the DMA Write Interrupt Status Register (DMACAP10) or DMA Read Interrupt Status Register (DMACAP23) and check the status of any fatal error if some other event has occurred.
- After a Stop event, you cannot seamlessly resume the transfer again because the DMA does not resume operation exactly from the point it stopped. Therefore, you must setup and start the complete channel context again, including the
  - Linked list pointer.
  - Linked list structure in memory.
  - Related PCS and CCS values.
  - When operating in non-linked list mode, setup channel context registers

### (f) TLP Generator Header Control

You must program bits [31:12] of the DMA Channel Control 1 Register (DMACAP[37+*i**16]/ DMACAP[45+*i**16), so that the DMA TLP generator correctly sets the TC, RO, NS, and FN TLP header fields for any TLPs that it generates.

### (g) DMA Software Register Map

This section discusses the grouping and access methods of the DMA registers. The following registers are present when the DMA is enabled.

- DMA Global Registers: these registers control global settings

**Table 104.20 DMA Global Register**

| | Description |
|---|---|
| base address + 0x000 | DMA Arbitration Scheme for TRGT1 Interface |
| +0x008 | Number of DMA Channels |
| +0x00C | DMA Write Engine Enable |
| +0x010 | DMA Write Doorbell |
| +0x018 | DMA Write Engine Channel Arbitration Weight Low |
| +0x01C | DMA write Engine Channel Arbitration Weight High |
| +0x02C | DMA Read Engine Enable |

| | |
|---|---|
| +0x030 | DMA Read Doorbell |
| +0x038 | DMA Read Engine Channel Arbitration Weight Low |
| +0x03C | DMA Read Engine Channel Arbitration Weight High |

- DMA Interrupt Registers: these registers control the generation of local interrupts and remote IMWr. The IMWr addresses and data are stored in RAM.

**Table 104.21 DMA Interrupt Register**

| Byte Offset | Description |
|---|---|
| Base address + 0x4C | DMA Write Interrupt Status |
| +0x50 | Not Used (RsvdP) |
| +0x54 | DMA Write Interrupt Mask |
| +0x58 | DMA Write Interrupt Clear |
| +0x5C | DMA Write Error Status |
| +0x60 | DMA Write Done IMWr Address Low |
| +0x64 | DMA Write Done IMWr Address High |
| +0x68 | DMA write Abort IMWr Address Low |
| +0x6C | DMA Write Abort IMWr Address High |
| +0x70 | DMA Write Channels 1 and 0 IMWr Data |
| +0x74 | DMA Write Channels 3 and 2 IMWr Data |
| +0x78 | DMA Write Channels 5 and 4 IMWr Data |
| +0x7c | DMA Write Channels 7 and 6 IMWr Data |
| +0x90 | DMA Write Linked List Error Enable |
| Up to + 0x9C | Not Used (RsvdP) |
| +0xA0 | DMA Read Interrupt Status |
| +0xA4 | Not Used (RsvdP) |
| +0xA8 | DMA Read Interrupt Mask |
| +0xAC | DMA Read Interrupt Clear |
| +0xB0 | Not Used (RsvdP) |
| +0xB4 | DMA Read Error Status Low |
| +0xB8 | DMA Read Error Status High |

| | |
|---|---|
| Up to + 0xC0 | Not Used (RsvdP) |
| +0xC4 | DMA Read Linked List Error Enable |
| +0xc8 | Not Used (RsvdP) |
| +0xCC | DMA Read Done IMWr Address Low |
| +0xD0 | DMA Read Done IMWr Address High |
| +0xD4 | DMA Read Abort IMWr Address Low |
| +0xD8 | DMA Read Abort IMWr Address High |
| +0xDC | DMA Read Channels 1 and 0 IMWr data |
| +0xE0 | DMA Read Channels 3 and 2 IMWr data |
| +0xE4 | DMA Read Channels 5 and 4 IMWr data |
| +0xE8 | DMA Read Channels 7 and 6 IMWr data |

- DMA Channel Context Registers: these registers (which are stored in RAM) are for the control and status information that is specific to each channel. There is a set of registers for each channel. The number of channels in the DMA is configurable in each direction (write and read) from 1 to 8 using the CC_NUM_DMA_WR_CHAN and CC_NUM_DMA_RD_CHAN configuration parameters.

| **Byte Offset** | **Description** |
|---|---|
| DMA Channel Context | |
| Base Address + 0x200*(i+1) + 0x100*(WR\|RD) | DMA Channel Control 1 Register |
| +0x004*i + 0x200*(i+1) + 0x100*(WR\|RD) | DMA Channel Control 2 Register |
| +0x008*i + 0x200*(i+1) + 0x100*(WR\|RD) | DMA Transfer Size |
| +0x00C*i + 0x200*(i+1) + 0x100*(WR\|RD) | DMA SAR Low |
| +0x010*i + 0x200*(i+1) + 0x100*(WR\|RD) | DMA SAR High |
| +0x014*i + 0x200*(i+1) + 0x100*(WR\|RD) | DMA DAR Low |
| +0x018*i + 0x200*(i+1) + 0x100*(WR\|RD) | DMA DAR High |
| +0x01C*i + 0x200*(i+1) + 0x100*(WR\|RD) | DMA Linked List Pointer Low |
| +0x020*i + 0x200*(i+1) + 0x100*(WR\|RD) | DMA Linked List Pointer High |

***Caution:* Implementation of Channel Context Registers**

- The channel context registers are stored in RAM whose contents are automatically initialized after power-on.

- You must not write to any of the context registers for a particular channel after you start the channel by writing the channel number to the Doorbell Number field of the DMA Write Doorbell Register (DMACAP3) or DMA Read Doorbell Register (DMACAP7).

**Linked List Operation**
When you enable linked list operation (Linked List Enable field of DMA Channel Control 1 Register DMACAP[37+*i**16] to 1b1, the DMA overwrites the following DMA channel context registers with the following information from linked list data elements:

- The CB, LLP, LIE, and RIE fields of the DMA Channel Control 1 register.
- DMA Transfer Size.
- DMA SAR Low.
- DMA SAR High.
- DMA DAR Low.
- DMA DAR High

**Note**:

- The structure of a link element is different to that of the data element. A data element has no TCB field. A link element has no LIE or RIE fields. It has no SAR, DAR, or Transfer Size DWORDs, but has LL Element Pointer DWORDs instead of the SARDWORDs. For more details, see Relationship Between Element DWORDs and Channel Context Registers.
- The LIE and RIE bits in a LL element, only enable the Done interrupt. In non-LL mode, the RIE and LIE bits enable the Done and Abort interrupts.

### (h) Programing Examples

****Attention:** You must enable the DMA before you start a DMA channel. Therefore you must write ‘1’ to DMA Read Engine Enable or DMA Write Engine Enable register before you write ‘1’ to the DMA Read Doorbell or DMA Write Doorbell register.

**1 MB Write Transfer On Write Channel #0 Initiated by Local CPU (Non LL mode)**

In this example, the IMWr generation is disabled, as the local CPU initiates the DMA transfer. The local CPU is interrupted using the edma_int bus. The SAR is the address of the local memory, and the DAR is the address of the remote memory, as shown in Figure 104.22. Table 104.22 provides the programming details for this example transfer.

**Figure 104.22 1 MB Write DMA Transfer from 0xBEEF_BEE0 to 0xCAFE_CAF0**

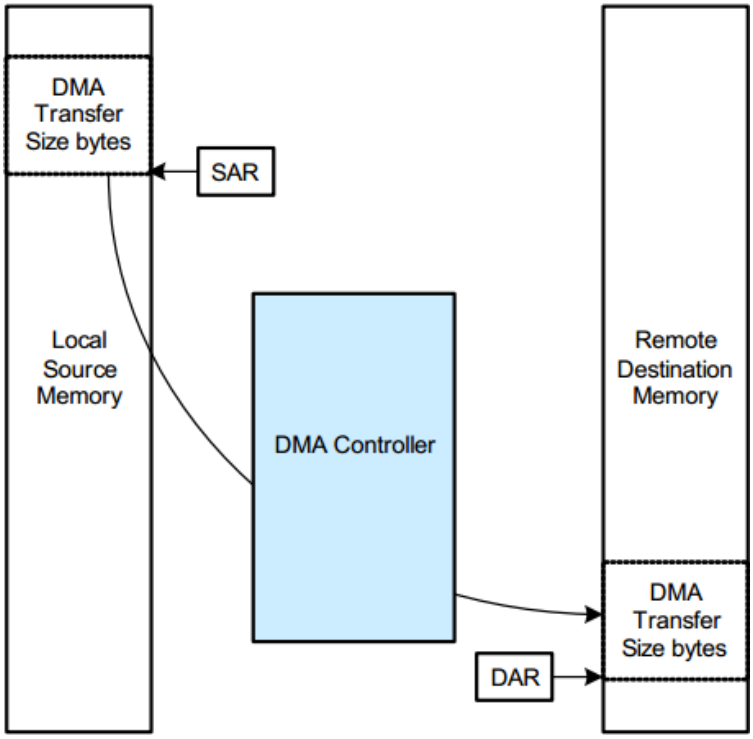



**Table 104.22 1 MB Write DMA Transfer from 0xBEEF_BEE0 to 0xCAFE_CAF0**

| Address (0x5000+) | Name | Value |
|---|---|---|
| 0x00C | DMA Write Engine Enable<br><br>You must not write 0 to this register. Even temporarily writing 0 to this register resets the DMA logic. For more detail, see DMA Write Engine Enable Register (DMACAP2). | 0x1 |
| 0x054 | DMA Write Interrupt Mask | 0x0 |
| 0x200 | DMA Channel Control 1 register<br><br>■ Local Interrupt Enable (LIE) =1<br><br>■ Remote Interrupt Enable (RIE) =0<br><br>■ RO, NS, TC, Function Number =0 | 0x04000008 |
| 0x208 | DMA Transfer Size | 0x00100000 |
| 0x20C | DMA SAR Low | 0xBEEF_BEE0 |
| 0x210 | DMA SAR High | 0x0000_0000 |
| 0x214 | DMA DAR Low | 0xCAFE_CAF0 |
| 0x218 | DMA DAR High | 0x0000_0000 |
| 0x010 | DMA Write Doorbell | 0x0 |

**1 MB Read Transfer On Read Channel #0 Initiated by Local CPU (Non LL mode)**

In this example, the IMWr generation is disabled, as the local CPU initiates the DMA transfer. The local CPU is interrupted using the edma_int bus. The SAR is the address of the remote memory, and the DAR is the address of the local memory, as shown in Figure 104.23. Table 104.23 provides the programming details for this example transfer.

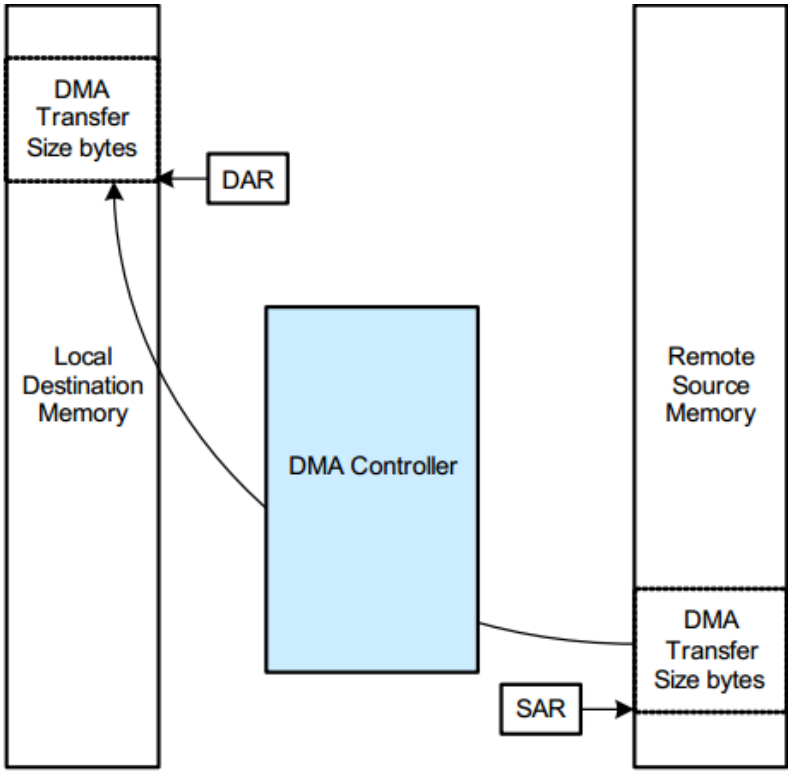



**Figure 104.23 1 MB Read DMA Transfer from 0xBEEF_BEE0 to 0xCAFE_CAF0**

**Table 104.23 DMA Register Setup**

| Address (0x5000+) | Name | Value |
|---|---|---|
| 0x02C | DMA Read Engine Enable<br><br>You must not write 0 to this register. Even temporarily writing 0 to this register resets the DMA logic. For more detail, see DMA Read Engine Enable Register (DMACAP6). | 0x1 |
| 0x0A8 | DMA Read Interrupt Mask | 0x0 |
| 0x300 | DMA Channel Control 1 register<br><br>■ Local Interrupt Enable (LIE) =1<br><br>■ Remote Interrupt Enable (RIE) =0<br><br>■ RO, NS, TC, Function Number =0 | 0x04000008 |
| 0x308 | DMA Transfer Size | 0x00100000 |
| 0x30C | DMA SAR Low | 0xBEEF_BEE0 |
| 0x310 | DMA SAR High | 0x0000_0000 |
| 0x314 | DMA DAR Low | 0xCAFE_CAF0 |

| | | |
|---|---|---|
| 0x318 | DMA DAR High | 0x0000_0000 |
| 0x030 | DMA Read Doorbell | 0x0 |

**1 MB Write Transfer On Write Channel #5 Initiated by Remote CPU (Non LL mode)**

In this example, the local interrupt generation is disabled, as the remote CPU initiates the DMA transfer. The remote CPU is interrupted using an IMWr. The SAR is the address of the local memory, and the DAR is the address of the remote memory, as shown in Figure 104.24. Table 104.24 provides the programming details for this example transfer.

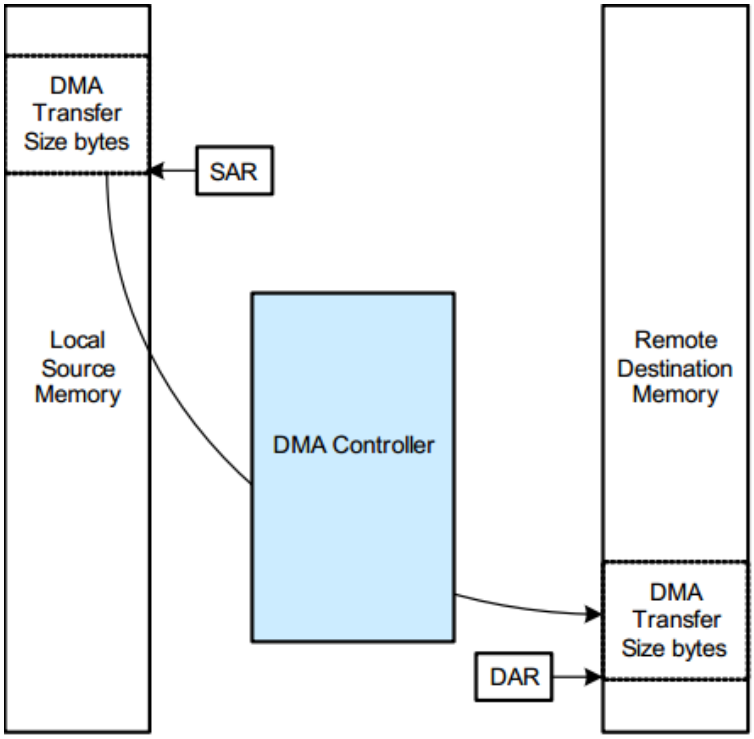



**Figure 104.24 1 MB Write DMA Transfer from 0xBEEF_BEE0 to 0xCAFE_CAF0**

**Table 104.24 DMA Register Setup**

| Address (0x5000+) | Name | Value |
|---|---|---|
| 0x00C | DMA Write Engine Enable<br><br>You must not write 0 to this register. Even temporarily writing 0 to this register resets the DMA logic. For more detail, see DMA Write Engine Enable Register (DMACAP2) | 0x1 |
| 0x060 / 0x064 | DMA Write Done IMWr Address Low and High | Your IMWr Address #1 |
| 0x068 / 0x06C | DMA Write Abort IMWr Address Low and High | Your IMWr Address #2 |
| 0x088 | DMA Write Channel 5 IMWr Data | Your IMWr Data |
| 0xC00 | DMA Channel Control 1 register<br><br>■ Local Interrupt Enable (LIE) =0 | 0x04000010 |

| | | |
|---|---|---|
| | ■ Remote Interrupt Enable (RIE) =1<br>■ RO, NS, TC, Function Number =0 | |
| 0xC08 | DMA Transfer Size | 0x00100000 |
| 0xC0C | DMA SAR Low | 0xBEEF_BEE0 |
| 0xC10 | DMA SAR High | 0x0000_0000 |
| 0xC14 | DMA DAR Low | 0xCAFE_CAF0 |
| 0xC18 | DMA DAR High | 0x0000_0000 |
| 0x010 | DMA Read Doorbell | 0x5 |

**1 MB Read Transfer On Read Channel #5 Initiated by Remote CPU (Non LL mode)**

In this example, the local interrupt generation is disabled, as the remote CPU initiates the DMA transfer. The remote CPU is interrupted using an IMWr. The SAR is the address of the remote memory, and the DAR is the address of the local memory, as shown in Figure 104.25. Table 104.25 provides the programming details for this example transfer

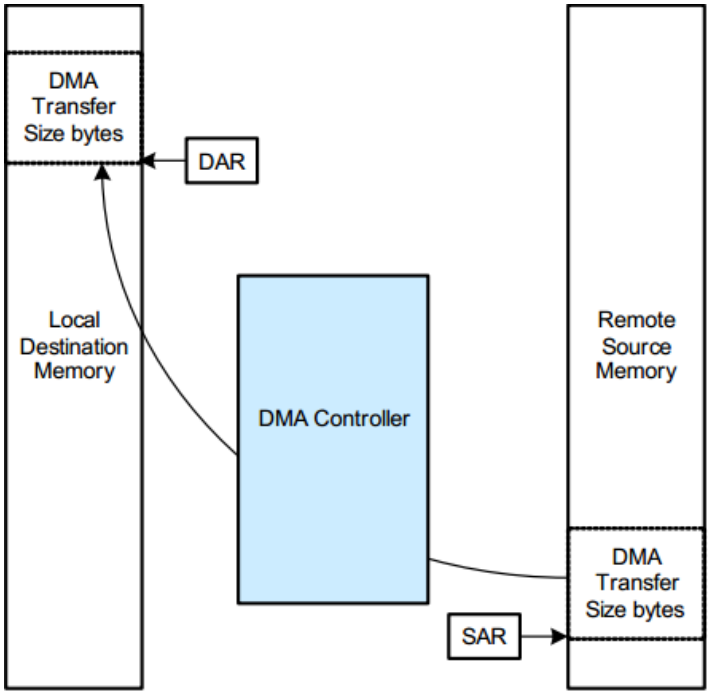



**Figure 104.25 1 MB Read DMA Transfer from 0xBEEF_BEE0 to 0xCAFE_CAF0**

**Table 104.25 DMA Register Setup**

| Address (0x5000+) | Name | Value |
|---|---|---|
| 0x02C | DMA Read Engine Enable<br><br>You must not write 0 to this register. Even temporarily writing 0 to this register resets the DMA logic. For more detail, see DMA Read Engine Enable Register (DMACAP6). | 0x1 |
| 0x0CC / 0x0D0 | DMA Read Done IMWr Address Low and High | Your IMWr Address #1 |
| 0x0D4 / 0x0D8 | DMA Read Abort IMWr Address Low and High | Your IMWr Address #2 |
| 0x0E4 | DMA Write Channel 5 IMWr Data | Your IMWr Data |
| 0xD00 | DMA Channel Control 1 register<br>■ Local Interrupt Enable (LIE) =0<br>■ Remote Interrupt Enable (RIE) =1<br>■ RO, NS, TC, Function Number =0 | 0x04000010 |
| 0xB08 | DMA Transfer Size | 0x00100000 |
| 0xD0C | DMA SAR Low | 0xBEEF_BEE0 |
| 0xD10 | DMA SAR High | 0x0000_0000 |
| 0xD14 | DMA DAR Low | 0xCAFE_CAF0 |
| 0xD18 | DMA DAR High | 0x0000_0000 |
| 0x030 | DMA Read Doorbell | 0x5 |

**Linked List Mode Programming Example (Write Channel #0)**

This write channel example with the following features:

- Recycled linked list (LL) of nine data elements and one link element.
- Transfer is initiated by local CPU.
- Each data element corresponds to a 1 MB (0x0010_0000) write transfer.
- The SARs for each element are separated by 2 MB (0x0020_0000). The DAR for each element are separated by 16 MB (0x0100_0000). The SAR is the address of the local memory, and the DAR is the address of the remote memory.
- The LL transfer list is located at address 0x0000_0200 in local memory.
- No remote interrupt is generated so the IMWr generation is disabled, as the local CPU initiates the DMA transfer. The local CPU is interrupted using the edma_int bus.
- There is a Watermark interrupt at element #5, and an Empty interrupt at element #8. Both of these interrupts are done interrupts (as opposed to abort interrupts).

You should familiarize yourself with the Linked List Flow for Producer and Consumer. These are the main steps for this process in this example:

Step 1: Create TL in local memory

- Create the 10 elements:
  - For elements 0, 1, 2, 3, 4, 6, 7: RIE, LIE, LLP, CB =0, 0, 0, 1.
  - For element 5: RIE, LIE, LLP, CB =0, 1, 0, 1.
  - For element 8: RIE, LIE, LLP, CB =0, 1, 0, 1.
  - For element 9: LLP, TCB, CB =1, 1, 0.

Step 2: Program and Start DMA

- Configure and start the DMA registers.

Step 3: Wait for done interrupt and recycle TL

****Attention: Software Programming of an Endpoints LIE and RIE Bits for Linked List Transfers.**
If the DMA driver is running on the host and the interrupt service routine is reading local interrupts to determine if the transfer is successful, then you must set LIE and RIE in the same element and you should mask or ignore the local interrupt pin. Setting RIE and LIE in element A followed by RIE (only) in element B is not a verified usage scenario.

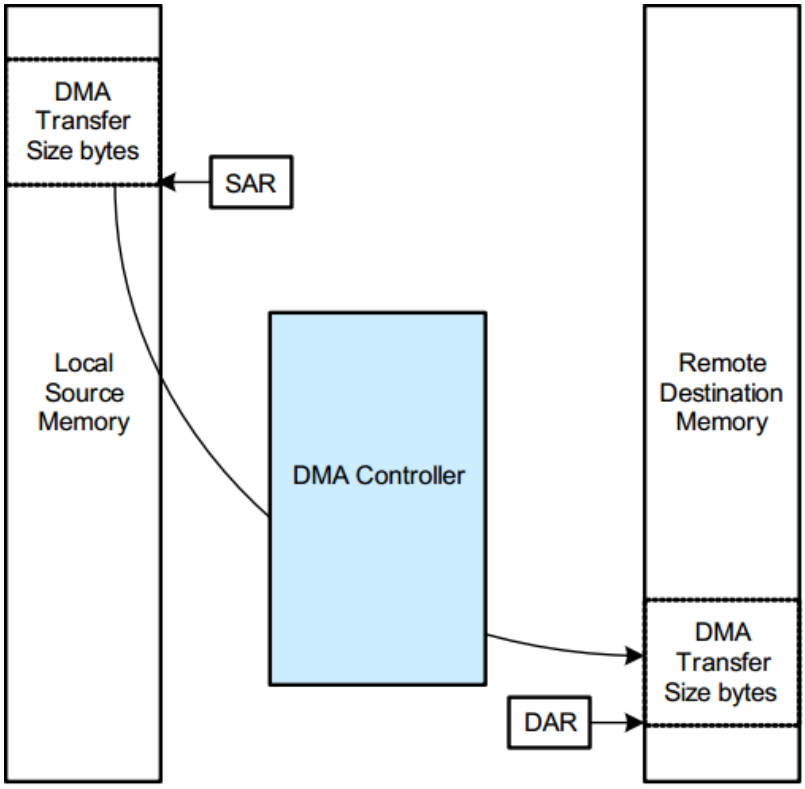



**Figure 104.26 1 MB Write DMA Transfer from SAR to DAR**

**Table 104.26 Linked List Element Initial Setup**

| Element# | Address | Data | Label |
|---|---|---|---|
| 0 | 0x0000_0200 | 0x0000_0001 | Channel Control |
| 0 | 0x0000_0204 | 0x0010_0000 | Transfer Size |
| 0 | 0x0000_0208 | 0xBEEF_BEE0 | SAR Low |
| 0 | 0x0000_020C | 0x0 | SAR High |
| 0 | 0x0000_0210 | 0xCAFE_CAF0 | DAR Low |
| 0 | 0x0000_0214 | 0x0 | DAR High |
| 1 | 0x0000_0218 | 0x0000_0001 | Channel Control |
| 1 | 0x0000_021C | 0x0010_0000 | Transfer Size |
| 1 | 0x0000_0220 | 0xBF0F_BEE0 | SAR Low |
| 1 | 0x0000_0224 | 0x0 | SAR High |
| 1 | 0x0000_0228 | 0xCBFE_CAF0 | DAR Low |
| 1 | 0x0000_022C | 0x0 | DAR High |
| 2 | 0x0000_0230 | 0x0000_0001 | Channel Control |
| 2 | 0x0000_0234 | 0x0010_0000 | Transfer Size |
| 2 | 0x0000_0238 | 0xBF2F_BEE0 | SAR Low |
| 2 | 0x0000_023C | 0x0 | SAR High |
| 2 | 0x0000_0240 | 0xCCFE_CAF0 | DAR Low |
| 2 | 0x0000_0244 | 0x0 | DAR High |
| 3 | 0x0000_0248 | 0x0000_0001 | Channel Control |
| 3 | 0x0000_024C | 0x0010_0000 | Transfer Size |
| 3 | 0x0000_0250 | 0xBF4F_BEE0 | SAR Low |
| 3 | 0x0000_0254 | 0x0 | SAR High |
| 3 | 0x0000_0258 | 0xCDFE_CAF0 | DAR Low |
| 3 | 0x0000_025C | 0x0 | DAR High |
| 4 | 0x0000_0260 | 0x0000_0001 | Channel Control |
| 4 | 0x0000_0264 | 0x0010_0000 | Transfer Size |
| 4 | 0x0000_0268 | 0xBF6F_BEE0 | SAR Low |

| | | | |
|---|---|---|---|
| 4 | 0x0000_026C | 0x0 | SAR High |
| 4 | 0x0000_0270 | 0xCEFE_CAF0 | DAR Low |
| 4 | 0x0000_0274 | 0x0 | DAR High |
| 5 | 0x0000_0278 | 0x0000_0009 | Channel Control |
| 5 | 0x0000_027C | 0x0010_0000 | Transfer Size |
| 5 | 0x0000_0280 | 0xBF8F_BEE0 | SAR Low |
| 5 | 0x0000_0284 | 0x0 | SAR High |
| 5 | 0x0000_0288 | 0xCFFE_CAF0 | DAR Low |
| 5 | 0x0000_028C | 0x0 | DAR High |
| 6 | 0x0000_0290 | 0x0000_0001 | Channel Control |
| 6 | 0x0000_0294 | 0x0010_0000 | Transfer Size |
| 6 | 0x0000_0298 | 0xBFAF_BEE0 | SAR Low |
| 6 | 0x0000_029C | 0x0 | SAR High |
| 6 | 0x0000_0290 | 0xD0FE_CAF0 | DAR Low |
| 6 | 0x0000_0294 | 0x0 | DAR High |
| 7 | 0x0000_0298 | 0x0000_0001 | Channel Control |
| 7 | 0x0000_029C | 0x0010_0000 | Transfer Size |
| 7 | 0x0000_02A0 | 0xBFCF_BEE0 | SAR Low |
| 7 | 0x0000_02A4 | 0x0 | SAR High |
| 7 | 0x0000_02A8 | 0xD1FE_CAF0 | DAR Low |
| 7 | 0x0000_02AC | 0x0 | DAR High |
| 8 | 0x0000_02B0 | 0x0000_0009 | Channel Control |
| 8 | 0x0000_02B4 | 0x0010_0000 | Transfer Size |
| 8 | 0x0000_02B8 | 0xBFEF_BEE0 | SAR Low |
| 8 | 0x0000_02BC | 0x0 | SAR High |
| 8 | 0x0000_02C0 | 0xD2FE_CAF0 | DAR Low |
| 8 | 0x0000_02C4 | 0x0 | DAR High |
| 9 | 0x0000_02C8 | 0x0000_0006 | Channel Control |
| 9 | 0x0000_02CC | 0x0 | Reserved |

| | | | |
|---|---|---|---|
| 9 | 0x0000_02D0 | 0x0000_0200 | Linked List Element Pointer Low |
| 9 | 0x0000_02D4 | 0x0 | Linked List Element Pointer High |

**Table 104.27 DMA Register Setup**

| Address (0x5000+) | Name | Value |
|---|---|---|
| 0x030 | DMA Write Engine Enable<br><br>You must not write 0 to this register. Even temporarily writing 0 to this register resets the DMA logic. For more details, see DMA Write Engine Enable Register (DMACAP2) | 0x1 |
| 0x54 | DMA Write Interrupt Mask | 0x0 |
| 0x090 | DMA Write Linked List Error Enable<br><br>■ LLLAIE = 1 | 0x1 |
| 0x200 | DMA Channel Control 1 register<br><br>■ Linked List Enable (LLE) =1<br><br>■ Consumer Cycle Status (CCS) = PCS =1<br><br>■ RO, NS, TC, Function Number =0 | 0x0400_0300 |
| 0x21C | DMA Linked List Pointer Low | 0x0000_0200 |
| 0x220 | DMA Linked List Pointer High | 0x0000_0000 |
| 0x010 | DMA Write Doorbell | 0x0 |

### 104.3.10 Peer to Peer Transfer

The PCIEC does not support the Peer-to-Peer Transfer features defined in the PCI Express Base Specification, Revision 4.0, Version 1.0. (The Peer-to-Peer in this section means that TLP is routed from one channel to another channel like PCIe switch.)

### 104.3.11 Transfer Priority

This section describes the transfer priority.

### (1) Arbitration (Transmission to PCI Express)

Accesses to transmit packets to the PCI Express include: DMA transfer, response processing to Inbound transfer, Outbound transfer, and Message transfer. The DMAC and the modules controlling Outbound transfer output posted and non-posted requests, respectively.

Arbitration between these accesses is performed according to the priority shown below Table 104.28.

**Table 104.28 TLP Arbitration Priority (1= Highest, 6= Lowest)**

| Priority | TLP Type |
|---|---|
| 1 | Completion of any TLP currently in progress |
| 2 | Retry buffer retransmissions |
| 3 | Messages generated by PCIEC (including MSI generated by register write) |
| 4 | TLPs from the transaction layer:<br>(When Root port mode) Completions generated by PCIEC for UR or CA<br>(When End port mode) Completions generated by PCIEC for Type0 configuration read and write requests (including UR responses) |
| 5 | TLPs on the internal bus |
| 6 | DMA transfer |

### (2) Arbitration (Reception from PCI Express)

The PCIEC receiver receive packets from the PCI Express and filter into three separate FIFO queues: posted, non-posted, and completion. When more than one queue has a TLP that is allowed to proceed, it uses a priority arbitration algorithm to select a transaction. This prevents that delivery rate for a particular TLP type to internal bus is unfairly reduced.

### (3) PCI Express Transmit/Receive Ordering

The following describes the transaction ordering rules of PCIEC.

Table 104.29 shows the transmit transaction ordering rules of this module.

**Table 104.29 Transmit Transaction Ordering Rules of This Module**

| Row Pass Column? | 1st Posted | 1st Non-Posted | 1st Completion |
|---|---|---|---|
| 2nd posted | No | Yes | Yes |
| 2nd non-posted | No | No | Yes |
| 2nd completion | No | Yes | No |

Note: Yes: The second transaction can pass the first transaction.
No: The second transaction cannot pass the first transaction.

Table 104.30 shows the receive transaction ordering rules of this module.

**Table 104.30 Receive Transaction Ordering Rules of This Module**

| Row Pass Column? | 1st Posted | 1st Non-Posted | 1st Completion |
|---|---|---|---|
| 2nd posted | No[a] | Yes | Yes |
| 2nd non-posted | No | No | Yes |
| 2nd completion | No | Yes | No |

Note: Yes: The second transaction can pass the first transaction.
No: The second transaction cannot pass the first transaction.
a: A non MSI P request can pass previously-received MSI P request.

PCIEC applies the ordering rules conforming to the standard to the transmission and reception sides. According to the rules, some packets cannot be passed by (to prevent an undesired influence on the order of sequential processing and such). Therefore, depending on the packet transfer order or combination, processing of a packet may be delayed until the previous packet processing is completed.

### (4) Arbitration (Bus Master Request)

Accesses to output a master request to the internal bus include: DMA transfer, Inbound transfer, and message transfer.

Arbitration between these accesses is performed according to the priority shown below.

(High priority)

- Inbound transfer (reception of memory request and IO request)
- DMA Write Channel (transmission of memory write and command read)
- DMA Read Channel (command read)
- DMA Read Channel (transmission of memory write)

(Low priority)

### 104.3.12 Link Function

The following sections describe Changing Speed, Lane Resizing and Lane Reversal and Flipping.

#### (1) Changing Speed

PCIEC supports all of the non-optional Gen1 2.5 GT/s features, Gen2 5.0 GT/s features, Gen3 8.0 GT/s features and Gen4 16.0 GT/s features defined in the PCI Express Base Specification, Revision 4.0, Version 1.0.

PCIEC initiates the speed change automatically after link up according to EXPCAP12F*i*[3:0]. cfg_target_link_speed and clears PRTLGC21. cfg_directed_speed_change bit. The default of EXPCAP12[3:0]. int_target_link_speed value is ‘B0100 (16.0 GT/s). Therefor the first link up speed is 16.0 GT/s. If you want to link up in another speed, then write a target speed value to EXPCAP12F0[3:0]. int_target_link_speed before link up.

To manually initiate the speed change then:

- Write to EXPCAP12F0[3:0]. int_target_link_speed in the local device.
- Deassert PRTLGC21. cfg_directed_speed_change in the local device.
- Assert PRTLGC21. cfg_directed_speed_change in the local device.

If you want to change speed from Gen4 rate to Gen2 or Gen3 rate,
Request speed change from Gen4 rate to Gen1 rate. Then change speed from Gen1 rate to any higher rate.

#### (2) Gen3 8.0 GT/s , Gen4 16.0 GT/s Transmitter Equalization

PCIEC performs link equalization during link training to improve signal quality by adjusting the transmitter and receiver equalization parameters for each lane on each side of the link. There are two feedback modes for determining the optional equalization settings, programmable using PRTLGC54.

- Figure of merit (FOM)
- Direction change with optional convergence support

#### (3) Gen4 Lane Margining

The controller supports all mandatory PIPE message bus interface features specified in PIPE Specification for PCI Express, Version 4.4.1, except for elastic buffer depth control and generation of read command form the controller to PHY.

#### (4) Lane Resizing

Initially, the controller tries to linkup at maximum width. You can restrict the maximum width as follows:

- Program PRTLGC4. cfg_link_capable
- Program PRTLGC21. cfg_lane_en[4:0]

The link partners can change the link width after the link is up. Resizing option are show in Table 104.31.

**Table 104.31 Link Resizing Options**

| | **Remote Partner Initiates (L0 -> Recovery -> Configuration)** | **Software Initiates Localy (L0 -> Recovery -> Configuration)** |
|---|---|---|
| Upsizing (Upconfugure) | Yes | Yes |
| Downsizing | Yes | Yes |

Resizing only occurs if both link partners advertised the Upconfigure Capability bit (except link reliability case). PCIEC has the Upconfigure Capability bit in PRTLGC59. cfg_pl_multilane_control[7] and default setting is 1.

The following steps show how you can initiate link resizing:

- Ensure the link is in the L0 LTSSM state.
- Program PRTLGC59.cfg_pl_multilane_control[5:0]
- Program PRTLGC59.cfg_pl_multilane_control[6]

Limitation:

If your system has reversed lanes in Gen3 or Gen4 speed mode, then you should perform upsizing at Gen1 or Gen2 speed only to avoid link-down caused by LFSR mismatching.

If the lanes are reversed, software must manually flip the lanes to achieve downsizing for some configuration. See 104.3.12(5) for more details.

**(5) Lane Reversal and Flipping**

PCIEC supports lane reversal, lane flipping and polarity inversion Depending on the configuration of the link partners and the topology of link between them, the controller will attempt to reverse and / or flip lanes to form a link in any of the following scenarios:

- The lanes are reversed by the physical layout of the PCIe link
- When connecting a narrower link partner into upper lanes of a wide controller
- A lane is non-operational (broken)
  - Non-operational means that a lane cannot be detected in Detect LTSSM state or it cannot receive in Polling LTSSM state.

Lane auto flip occurs in Detect when physical Lane0 cannot be detected.

Lane auto reverse occurs in Configuration and is useful to from a link when logical Lane0 receives TS Ordered Sets from the link partner with the Lane Number field different from 0.

To enable lane auto flip and lane auto reverse, set PRTLGC21[16]. cfg_lane_en[8]= 1 and PRTLGC21[15:13]. cfg_lane_en[7:5] = 0.

You can also perform lane flip manually. To enable manual lane flip, set PRTLGC21[16]. cfg_lane_en[8] =1, PCIEMSR1[1]. tx_lane_flip_en =1 for transmission lanes and set PCIEMSR1[0]. rx_lane_flip_en =1 for reception lanes.

Figure 104.27, Figure 104.28, Figure 104.29, Figure 104.30 and Figure 104.31 show Reversing and Flipping Lanes structures.

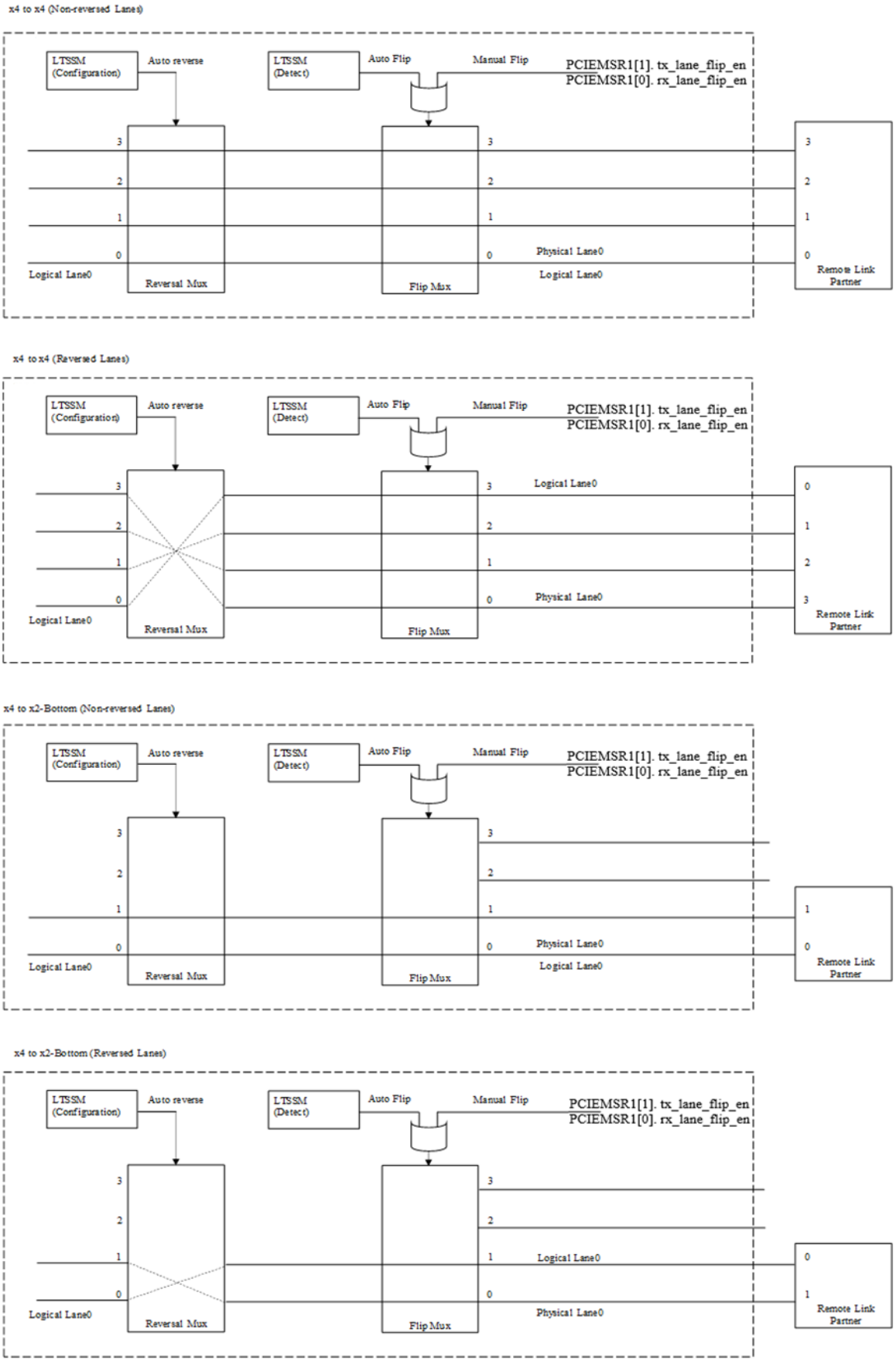



**Figure 104.27 Reversing and Flipping Lanes (1)**

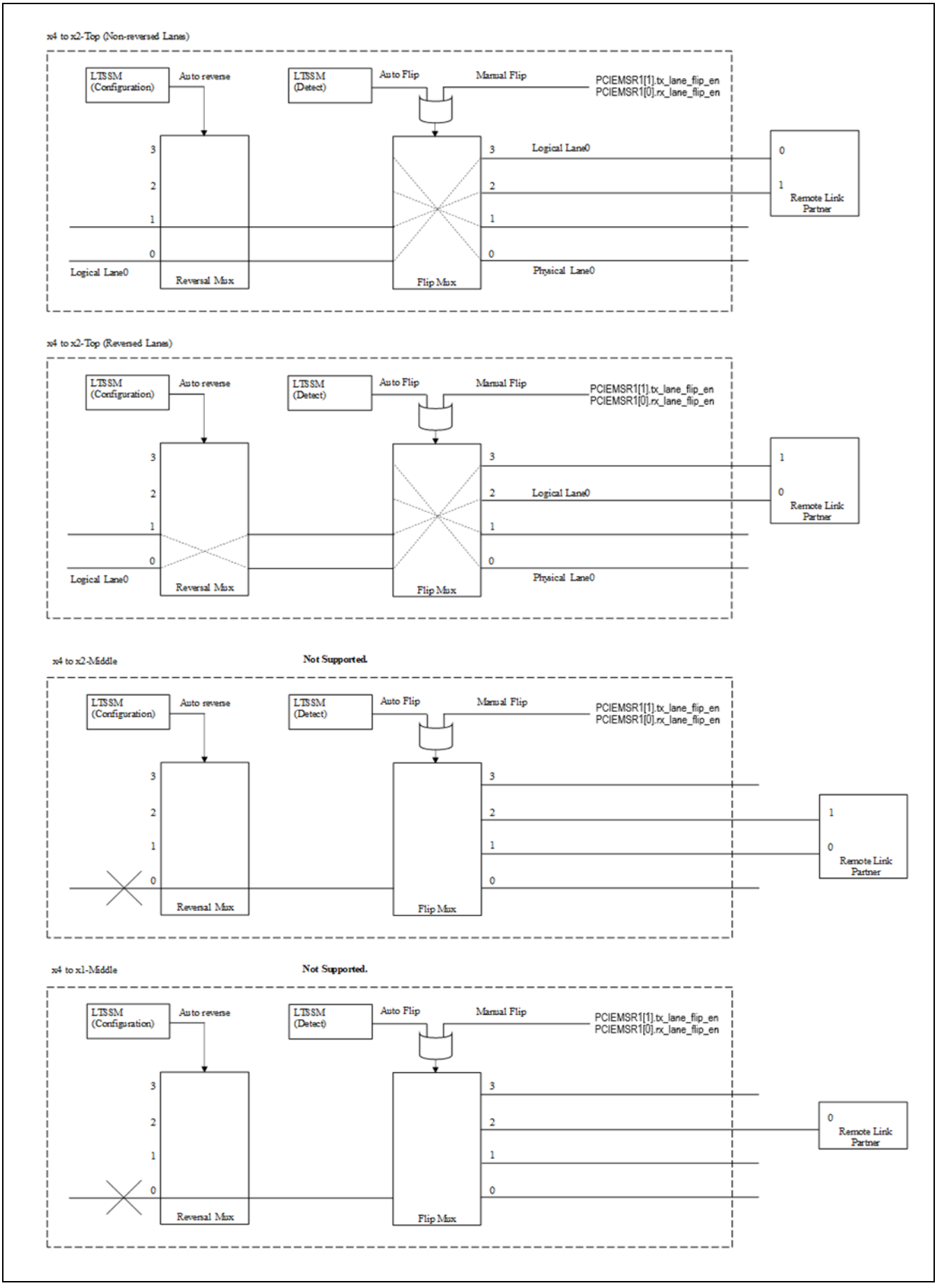



**Figure 104.28 Reversing and Flipping Lanes (2)**

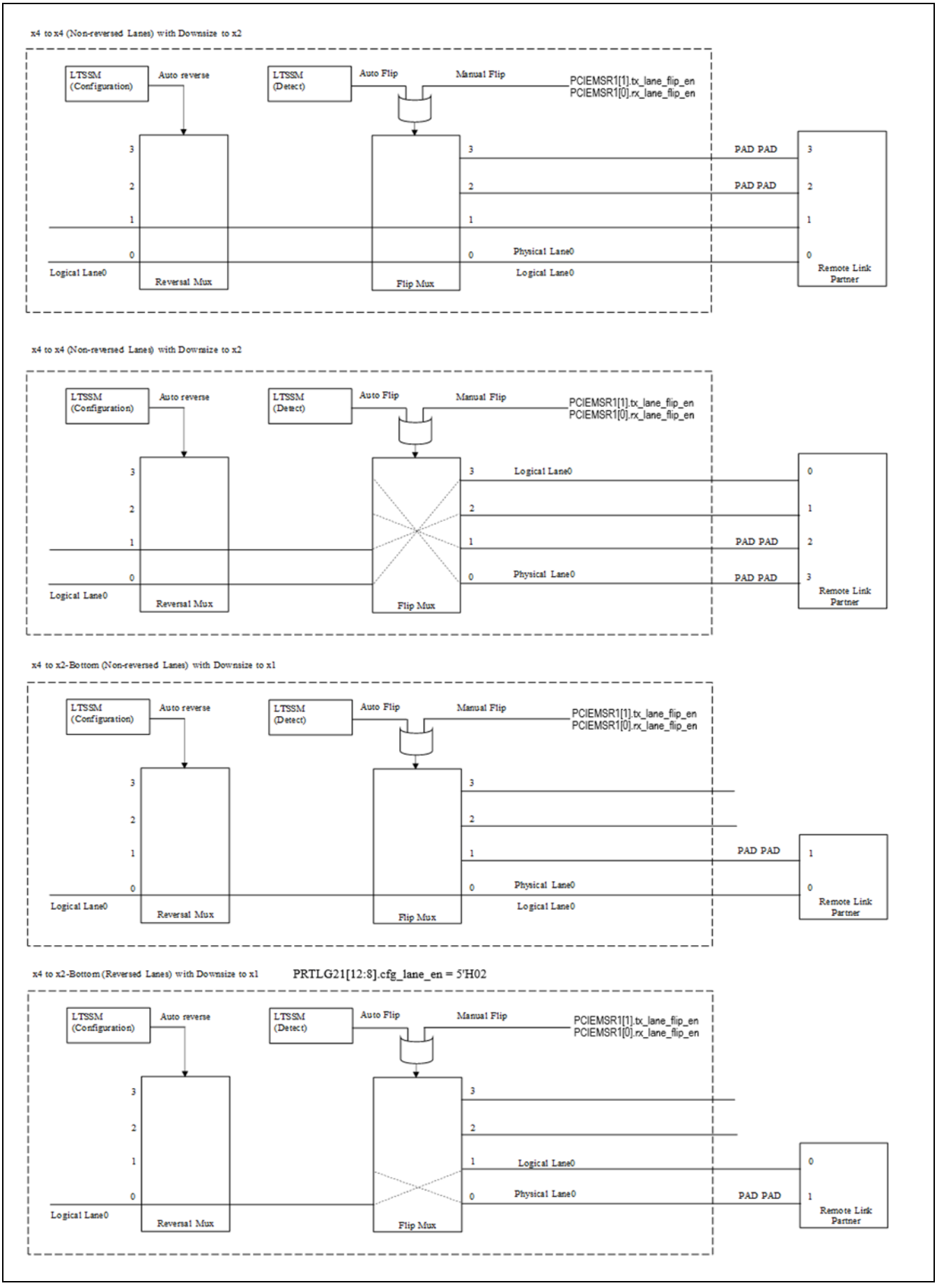



**Figure 104.29 Reversing and Flipping Lanes (3)**

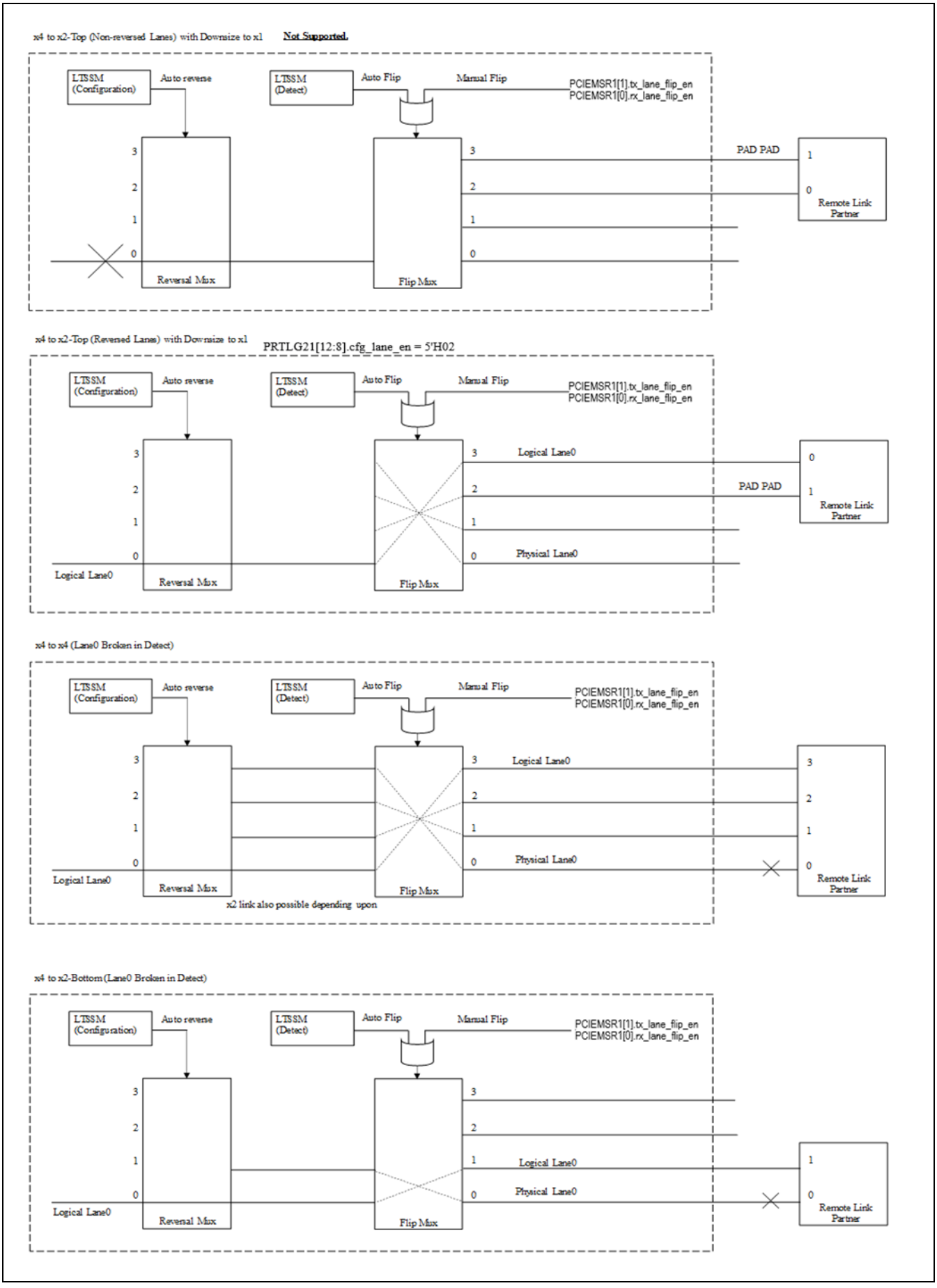



**Figure 104.30 Reversing and Flipping Lanes (4)**

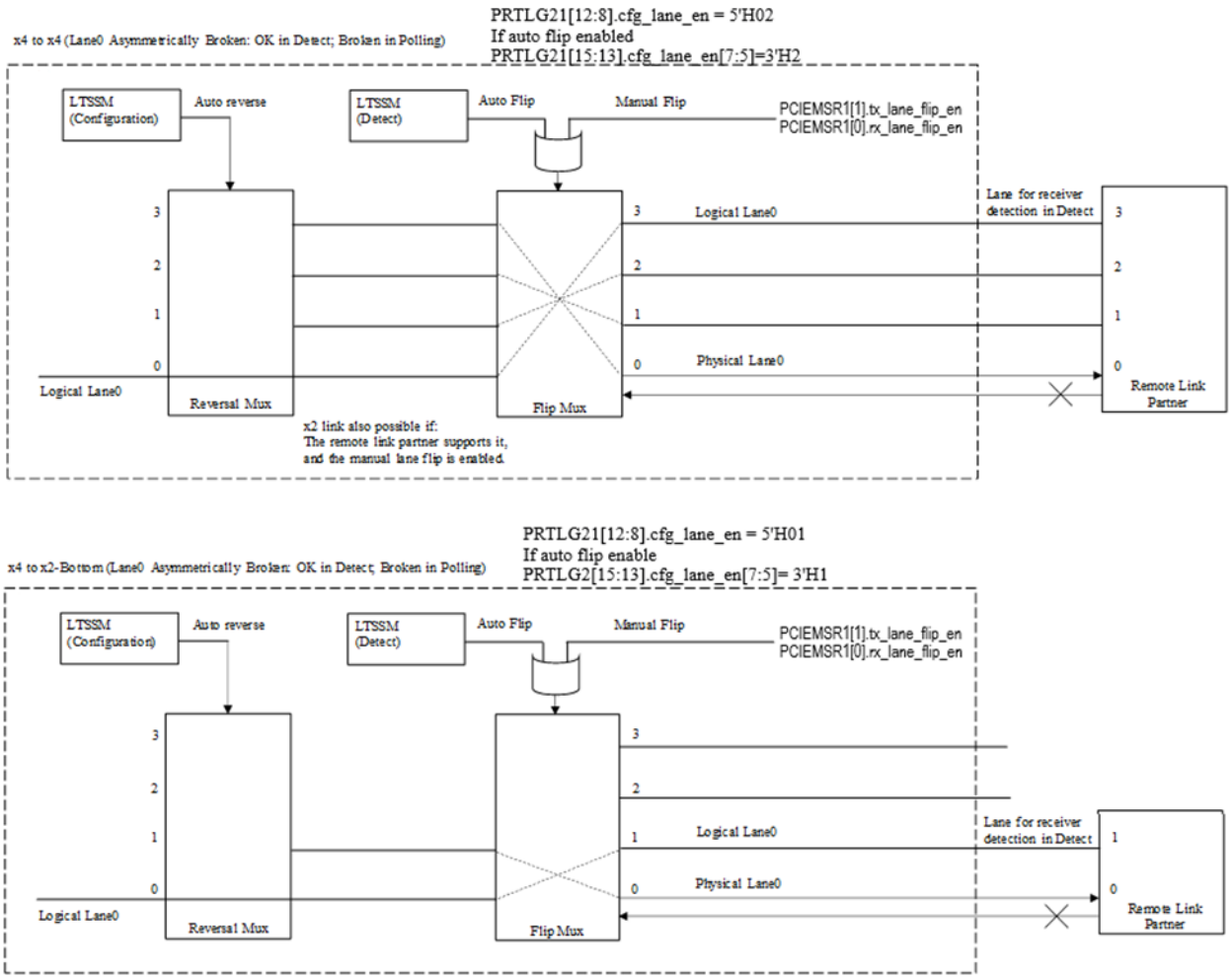



**Figure 104.31 Reversing and Flipping Lanes (5)**

### 104.3.13 Lane Bifurcation

PCIEC's channel 0 which has the quad lanes supports lane bifurcation. The channel 0 can be bifurcated to two dual lanes which are called as channel 0 and channel 1 respectively. The channel 0 and channel 1 which are bifurcated lanes have the independent link function, power management and reset architecture. Therefore, you can control each channel without effect to the other working channel.

To enable the lane bifurcation, set PCIEMSR0.BFRCTNST = H'1 before PCIEC reset de-assert. It is forbidden to change the lane bifurcation mode during PCIEC operation.

### 104.3.14 Power Management

This section discusses the power management (PM) features of the controller. For a proper understanding of PCIe power management, you should read Chapter 5, Power Management of the PCI Express Base Specification, Revision 4.0, Version 1.0. The following topics are discussed:

■ "Overview"

■ "L1 Substates"

■ "L0s Entry and Exit Conditions"

■ "L1 Operation (Non-substates)"

■ "L2 and L3 Power Down Entry and Exit Conditions (USP)"

■ "Advanced Power Management and Power Domain Gating"

#### (1) Overview

The controller supports two categories of PM operations to control the device state (D-state) and link state.

- **Software PCI Compatible PM (PCI-PM)**
  - **D-state PM of Function.** The host software can direct the function to enter any of the D1 or D3 low-power states. It does this by writing to the Power Management Control and Status Register (PMCAP1Fi) in the PCI-PM capability structure.
  - **D-state PM of Link.** Link states are not visible to PCI-PM legacy compatible software, and are derived from the power management D-states of the components connected to that link. The action of changing the D-state in the PMCAP1Fi indirectly causes a change in the link power state. The L1 state is entered whenever all functions of a USP on a link are programmed to a non-D0 state. The entry into L2 and L3 states is initiated by the DSP.
  - **Clock PM (L1 with REFCLK removal/PLL Off).** This is an optional feature that enables components on a link to further reduce idle power consumption while the link is in L1, by turning off the PLL.
- **Native PCIe PM Mechanisms**
  - **Active State PM (ASPM).** When the USP is in L0 and detects idleness on the link for a specific amount of time, it automatically transitions the link to the L0s or L1 (optional) power state.

- **L1 Substates.** This is an optional PCIe feature that enables components on a link to further reduce idle power consumption while the link is in L1, including almost complete removal of power for the high speed PHY circuits.

**Relationship Between D-state, Link State, and PHY State**

The link state and D-state determine what power and clock supplies are on or off. For implementation details of low-power clocking logic.

**Table 104.32 Power/Clock Supply Status in Low-Power States of Interest (DSP or Single Function non-ARI USP)**

| D-State | Permissible Quiescent Link States | Link State Change Trigger | Active Power Supplies | Active Clock Supplies | pipe_clk<br>core_clk | PHY State |
|---|---|---|---|---|---|---|
| $D0_{unitialize}$ | L0 | n/a | $V_{main}$<br>$V_{aux}$ | REFCLK<br>AUXCLK | ON | P0, P0s |
| $D0_{active}$ | L0 | ASPM | | | | |
| | L0s | | | | | |
| | L1 | | | | ON | P1 |
| D1, $D3_{hot}$ | L1 | PCI-PM | | | | |
| $D3_{cold}$ | L2 | PCI-PM | $V_{aux}$ | AUXCLK | OFF | P2/OFF |
| | L3 | | None | None | | |

****Note: $D0_{active}$ and $D0_{uninitialized}$**
D0 is divided into two distinct sub-states, the uninitialized sub-state and the active sub-state. When a component comes out of conventional reset or FLR, it defaults to the D0 uninitialized state. A function enters the $D0_{active}$ state whenever any of the functions Memory Space Enable, I/O Space Enable, or Bus Master Enable bits have been enabled by system software.

**Table 104.33 Link-State-Change Trigger Capabilities for Each Port Type**

| Port Type | Can Trigger L0/L0s Entry | Can Trigger L1 Entry | Can Trigger L2/L3 Entry |
|---|---|---|---|
| USP | Yes | Yes, on any of these trigger. Software program non-D0 state (PCI-PM)<br>USPs idle timer expires (ASPM)<br>Application requests the USP to Enter L1 | No |
| DSP | Yes | No | Yes |

Table below indicates the different levels of power savings (obtained by switching off circuits outside the controller) that can be expected from executing L1 in different ways. Higher exit latencies are associated with higher power savings. Power saving and latency are PHY implementation specific, and may be determined by consulting your PHY documentation.

**Table 104.34 – L1 Execution Modes and Permissible Power States of PHY Circuits**

| L1 Execution Mode | Permissible Power States for PHY Circuits | | |
|---|---|---|---|
| | **PLL/REFCLK** | **Rx Electrical Idle Detector** | **TX common mode** |
| L1 without clock PM | ON | ON | ON |
| L1 with clock PM | OFF | ON | ON |
| L1.1 | OFF | OFF | ON |
| L1.2 | OFF | OFF | OFF |

**Device State (D-state) Power Management of a Function (USP)**
The host software can direct the downstream device function to enter any of the low-power states:

- D1 (light sleep).
- D2 (deep sleep). (not support D2 Power Management State)
- $D3_{hot}$ (full of; reference clock optionally removed; $V_{main}$ normally available).
- $D3_{cold}$ (full of; reference clock removed; $V_{main}$ removed; $V_{aux}$ optionally available).

It does this by writing to the PMCAP1F*i* in the PCI-PM capability structure. This action also indirectly causes the downstream component (USP) to change the link power state. The controller negotiates with the link partner using PM TLPs and DLLPs before entering the corresponding low-power link state. The controller exits the device low-power state (and returns to D0) when it detects a CFG access to a device function, or when it detects a PME such as an event on the outband_pwrup_cmd input signal. For a function in device power states D1 and $D3_{hot}$, the controller only accepts CFG and MSG requests TLPs for that function. All other incoming request types for that function are treated as unsupported requests (UR). For more details see Section 5.3.3, “Power Management Event Mechanisms” of the *PCI Express Base Specification, Revision 4.0, Version 1.0.*

**Active State Power Management (ASPM)**

You must enable ASPM through the ASPM Control field in the Link Control register. When the USP is in L0 and detects idleness on the link for a specific amount of time (determined by the L0s Entrance Latency field in the “Ack Frequency and L0-L1 ASPM Control Register” PRTLGC3), it automatically transitions the link to the L0s power state. Each link direction (TX and RX) is processed separately. The controller exits L0s when there is traffic waiting to be sent. When the USP is in L0 or L0s and it detects idleness on the link (both directions) for a specific amount of time (determined by the L1 Entrance Latency field in PRTLGC3), it automatically transitions the link to the L1 power state. The link partners must negotiate entry into the L1 link power state using PM TLPs and DLLPs. The L1 state achieves greater power savings but the exit latency from this state is higher than from L0s. You can also instruct an USP controller (only) to enter L1 by asserting the app_req_entr_l1 input. The USP exits L1 when there is traffic waiting to be sent or when a PME (such as an event on the app_req_exit_l1 input) is received. If the ASPM exit latencies from L0s or L1 are too high for the overall PCIe system data latency requirements, then the host software can disable ASPM using the ASPM Control Field of the Link Control register. The controller reports the exit latencies using the Link Capability registers. You set these by writing to these registers or by setting the DEFAULT_L0S_EXIT_LATENCY and DEFAULT_L1_EXIT_LATENCY parameters.

**Registers Shadowed in Power Management Controller**

Table below lists the registers shadowed in the power management controller (PMC) and the reason for shadowing them.

**Table 104.35 Registers Shadowed in PMC**

| Register | Field | Description | Reason for Shadowing |
|---|---|---|---|
| Device Control Register<br>(EXPCAP2Fi) | Aux Power PM Enable<br>(aux_pm_en) | Enable aux_power support for a function | The PCI Express Base Specification, Revison4.0, Version 1.0 specifies that the value of this bit must be preserved in the $D3_{cold}$ state. In L2 when power is removed the CDM loses its context. This bits is preserved in the PM under AUX power and fed back to the CDM. |
| PMCSR<br>(PMCAP1F*i*) | PME_EN<br>(pm_pme_en) | Indicates if a function is permitted to generate a PME | PME message request, if enable, can trigger a wakeup from L2 if main power has been removed. |
| PMCSR_BSE<br>(PMCAP1F*i*) | PowerState<br>(pmstate) | Indicates the D-state of a function | Use for entry to low power states. |
| Link Capabilities Register<br>(EXPCAP3F*i*) | Clock power Management<br>(cfg_clk_pm_cap) | If set to 1 this indicates that an upstream port tolerates the removal of reference clock when the link is in L1 or L2/L3 ready | Used for the removal of reference clock in L1 or L2/L3 ready. |
| AUX_CLK_FREQ_OFF<br>(PRTLGC87) | AUX_CLK_FREQ | The frequency of the aux_clk in MHz | This field is use to scale certain timers in the power management block. It needs to be shadowed because the CDM is reset when power is removed in L2. |
| GEN3_RELATED_OFF<br>(PRTLGC53) | GEN3_ZRXDC_NONCOMPL<br>(cfg_pl_gen3_zrxdc_noncompl) | GEN3 receiver impedance zrx-dc not compliant | This bit enable a 100ms timer which can trigger exit from L1 |

Note:
Temporarily interrupt the stream of back-to-back TLPs for one or more clock cycles on the client interfaces when low power mode entry is requested. This causes performance degradation.

Implement the note 1 (if possible), or
Implement the note 2
1. The application can monitor PCIEPWRMNGSTS1.pm_master_state and PCIEPWRMNGSTS1.pm_slave_state to know when the controller is in L1 entering negotiation.
2. Disable ASPM and do not force the controller into a low power mode through PCI-PM.

### (2) L1 Substates

This section discusses the L1 link substates. The following topics are discussed:

■ “L1.1 and L1.2 Entry and Exit Conditions”

■ “L1 Exit Latency”

The L1 substates are applicable in both the ASPM and PCI-PM L1 link states. L1 substates management utilizes a per-link sideband signal called CLKREQ#. The controller supports this feature (with the additional I/O as shown in “Removing the Reference Clock” on page 69) when you set CX_L1_SUBSTATES_ENABLE =1.

**Assumptions and Limitations**

- The CLKREQ# signal is not directly controlled by the controller. You must implement external logic to do so.
- During L1 substates it is assumed that core_clk is turned off and that aux_clk is active. It is required that your application switches aux_clk to a low frequency free running clock on entry into L1.
- The controller uses aux_clk for counting time during L1 substates. You must program the frequency of this clock into the PRTLGC87 register with a value in the range 1…1000MHz to count real time. Frequencies lower than 1 MHz are possible, but with a loss of accuracy in the time counted.
- When the electrical idle detection circuitry is disabled it is assumed that the PHY holds the signal phy_mac_rxelecidle to 1.
- Access the L1 PM Substates Extended Capability register.
  System Usage Note:
  Software should not attempt to write to the mentioned fields in the L1 PM Substates Extended Capability register.

#### (a) L1.1 and L1.2 Entry and Exit Conditions

**Overview**

After the link has entered L1 through the normal L1 negotiation, the USP can initiate the sequence for entering the target L1 substate (L1.1 or L1.2) by tri-stating its CLKREQ# output buffer. The entry sequence can only proceed if the DSP is also tri-stating its CLKREQ# output buffer, resulting in the bidirectional CLKREQ# signal being pulled up to 1. Otherwise CLKREQ# remains asserted at 0 and the link state stays in L1. The exit sequence can be initiated by either ports by asserting CLKREQ# to 0. For each port there are two cases to consider, the first where the exit is initiated locally, the second where the exit is initiated remotely. The following sections provide details on the signaling that occurs between the PCIe controller as the MAC, the PHY, and the CLKREQ# output buffer, for entering and exiting L1 substates. L1 substates management utilizes a per-link sideband signal called CLKREQ#. After all of the entry conditions for a L1 substate are met, the controller asserts the mac_phy_rxelecidle_disable output. When the target L1 substate is L1.2, it also asserts the mac_phy_txcommonmode_disable output.

**L1 Substates Entry**

1. MAC transitions the PHY to P1.
2. PHY asserts phy_mac_phystatus for P1 entry.
3. Controller transitions the PHY to P1.CPM and asserts pm_sel_aux_clk to turn off core_clk.
4. PHY asserts phy_mac_phystatus for P1.CPM entry and CLK_RST asserts aux_clk_active.

5. Controller de-asserts mac_phy_pclkreq_n[1].
6. CLKREQ# output buffer tri-stated.
7. Controller waits for clkreq_in_n to de-assert.
8. Controller transitions the PHY to P1.1 (or P1.2).
9. PHY asserts phy_mac_phystatus for P1.1 (or P1.2).
10. Controller asserts mac_phy_asyncpowerchangeack.
11. PHY de-asserts phy_mac_phystatus.
12. Controller de-asserts mac_phy_asyncpowerchangeack.

**L1 Substates Exit (Locally Initiated)**

1. Any of the normal L1 exit conditions in “L1-PM/L1-ASPM Exit” on page 278 are met
2. Controller asserts mac_phy_pclkreq_n[1].
3. CLKREQ# output buffer is asserted.
4. Controller transitions the PHY to P1.CPM.
5. PHY asserts phy_mac_phystatus for P1.1 (or P1.2).
6. Controller asserts mac_phy_asyncpowerchangeack.
7. PHY de-asserts phy_mac_phystatus.
8. Controller de-asserts mac_phy_asyncpowerchangeack.
9. Controller transitions the PHY to P1 and de-asserts pm_sel_aux_clk to turn on core_clk.
10. PHY asserts phy_mac_phystatus for P1 entry and CLK_RST de-asserts aux_clk_active.
11. Controller transitions the PHY to P0 in the usual way.

**L1 Substates Exit (Remotely Initiated)**

1. The remote link partner asserts CLKREQ#.
2. Controller input clkreq_in_n asserts.
3. Controller asserts mac_phy_pclkreq_n[1].
4. Controller transitions the PHY to P1.CPM.
5. PHY asserts phy_mac_phystatus for P1.1 or (P1.2).
6. Controller asserts mac_phy_asyncpowerchangeack.
7. PHY de-asserts phy_mac_phystatus.
8. Controller de-asserts mac_phy_asyncpowerchangeack.

9. Controller transitions the PHY to P1 and de-asserts pm_sel_aux_clk to turn on core_clk.
10. PHY asserts phy_mac_phystatus for P1 entry and CLK_RST de-asserts aux_clk_active.
11. Controller transitions the PHY to P0 in the usual way.

**L1 Substates Software Control**

When the controller enters L1, the target L1 substate depends on several programming bits:

- PM Control/Status Register
  - USP: The current D-state
  - DSP: The DLLP type that is used by USP to request L1
- L1 Substates Control 1 Register
  - ASPM PM L1.1 Enabled.
  - ASPM PM L1.2 Enabled.
  - PCI PM L1.1 Enabled.
  - PCI PM L1.2 Enabled.

After the USP controller enters L1, it uses the D-state of the device to determine if L1 is entered in ASPM mode or PCI-PM mode. After the DSP controller enters L1, it uses the USPs DLLP L1 request type to determine if L1 is entered in ASPM mode or PCI-PM mode. Table 104.36 and Table 104.37 (below) show the target L1 substate as a function of the relevant programming bits.

**Table 104.36 Target L1 Substate as a Function of Software Controls (USP)**

| D-State | ASPM PM L1.1 Enabled | ASPM PM L1.2 Enabled | Reported LTR >= L1.2 Threshold | PCI PM L1.1 Enabled | PCI PM L1.2 Enabled | Target L1 Substate |
|---|---|---|---|---|---|---|
| !D0 | - | - | - | 0 | 0 | L1 |
| !D0 | - | - | - | 1 | - | L1.1 |
| !D0 | - | - | - | - | 1 | L1.2 |
| D0 | 0 | 0 | - | - | - | L1 |
| D0 | 1 | 0 | - | - | - | L1.1 |
| D0 | 0 | 1 | 0 | - | - | L1 |
| D0 | 1 | 1 | 0 | - | - | L1.1 |
| D0 | - | 1 | 1 | - | - | L1.2 |

**Table 104.37 Target L1 Substate as a Function of Software Controls (DSP)**

| DLLP Received | ASPM PM L1.1 Enabled | ASPM PM L1.2 Enabled | Reported LTR >= L1.2 Threshold | PCI PM L1.1 Enabled | PCI PM L1.2 Enabled | Target L1 Substate |
|---|---|---|---|---|---|---|
| PM_Enter_L1 | - | - | - | 0 | 0 | L1 |
| | - | - | - | 1 | 0 | L1.1 |
| | - | - | - | - | 1 | L1.2 |
| PM_Active_State_Request_L1 | 0 | 0 | - | - | - | L1 |
| | 1 | 0 | - | - | - | L1.1 |
| | 0 | 1 | 0 | - | - | L1 |
| | 1 | 1 | 0 | - | - | L1.1 |
| | - | 1 | 1 | - | - | L1.2 |

The Reported LTR is the maximum of the snoop/no-snoop latency values embedded in LTR messages transmitted by the upstream ports or received by the downstream port. The controller stores these values in the port logic LTR Latency Register (PRTLGC86). When the requirement bit in the message is 0, the latency value is considered infinite (that is, the check with the threshold always pass).

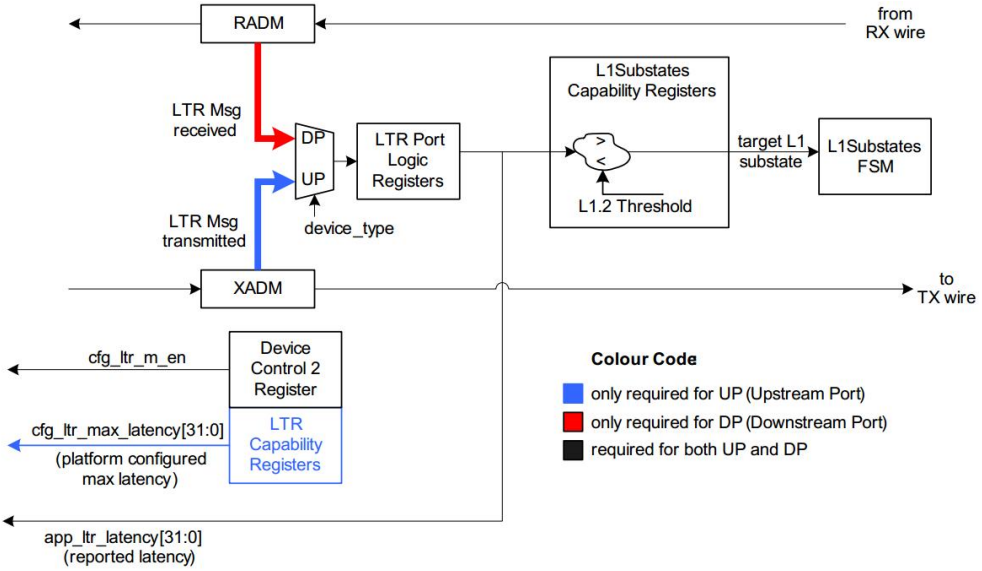



**Figure 104.36 Role of LTR in L1 Substates.**

### (b) L1 Exit Latency

This section provides a breakdown of the L1 exit latency when CX_L1_SUBSTATES_ENABLE is enabled for L1 substates or L1 Clock PM.

**Table 104.38 L1 Substates and L1 Clock PM Exit Latency Component Values**

| | A | B | C | D | E |
|---|---|---|---|---|---|
| Event in Figure 104.38 | CLKREQ# -> mac_phy_p | mac_phy_powerdown -> phy_mac_phystatus (pipe_clk_ stable) | pipe_clk stable -> mac_phy_powe | mac_phy_powerdown -> phy_mac_phystatus & LTSSM change | Recovery -> L0 |

| | owerdown | | rdown | | |
|---|---|---|---|---|---|
| Description | 0 | Time requires to produce a stable pipe_clk | Includes time to switch aux_clk back to pipe_clk and ungating of core_clk. | Time required after pipe_clk is stable to power up circuit for P0 | - |
| Values when exiting L1.1 | 0 | 400ns to reactivate REFCLK + 15us | Estimate: 3aux_clk cycles (slow freq) + 3 core_clk cycles | Time to P0, ~ 2us | Time to execute Recovery |
| Values when exiting L1.2 | 0 | T_power_on to reactivate REFCLK + 15us | | | T_common mode (~55us) + time to execute Recovery |

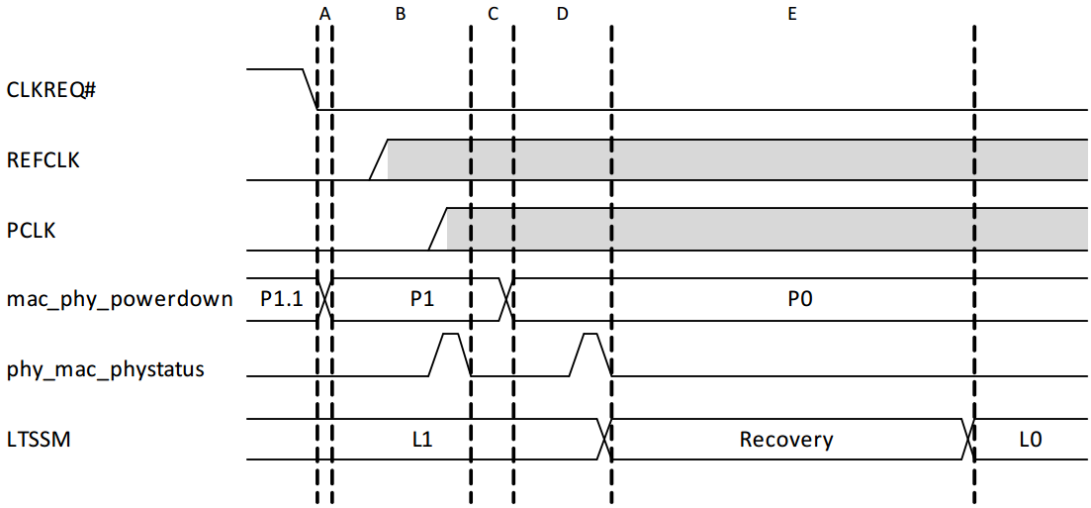



**Figure 104.38 L1 Exit Latency Components**

### (3) L0s Entry and Exit Conditions

This section discusses the L0s link state. The following topics are discussed:

■ “Overview”

■ “L0s Entry”

■ “L0s Exit”**Overview**

L0s is a low-power state enabled by ASPM. ASPM controls entry into L0s for the transmitter. The remote device controls entry into L0s for the receiver.

**L0s Entry**

All of these condition must be met:

- ASPM L0s is enabled through the ASPM Control field in the Link Control register.
- L0s entry conditions as defined in Section 5.4.1.1.1, “Entry into the L0s State” of the *PCI Express Base Specification, Revision 4.0, Version 1.0*, exist for a duration of time (determined by the cfg_l0s_entr_latency_timer field in PRTLGC3).
- No higher stage of power-down requested.

**L0s Exit**

Any of these condition can be met:

- A DLLP or TLP is pending to be sent.
- L1 entry conditions as defined in Section 5.4.1.2.1, “Entry into the L1 State” of the *PCI Express Base Specification, Revision 4.0, Version 1.0* are satisfied.
- PCIe link partner requests to enter into link recovery.

### (4) L1 Operation (Non-substates)

The following topics are discussed:

■ “L1 (ASPM/PM) Entry and Exit Conditions”

■ “L1 Clock PM (L1 with REFCLK Removal/PLL Off) Overview”

■ “L1 Clock PM (L1 with REFCLK Removal/PLL Off); Entry and Exit Conditions”

#### (a) L1 (ASPM/PM) Entry and Exit Conditions

**L1-ASPM Overview**

The L1-ASPM entry negotiation handshake uses PM_Active_State_Request_L1 DLLPs, PM_Request_Ack DLLPs, and PM_Active_State_Nak MSG TLPs. There are three scenarios that cause the controller to enter L1 under ASPM conditions.

**L1-ASPM Entry (Scenario 1): L1 Idle Timeout in L0s**

The controller enters L1-ASPM when all of these conditions are met in the USP:

- ASPM L1 and L0s are enabled through the ASPM Control field in the Link Control register.
- The cfg_l1_entr_wo_rl0s field of PRTLGC3 is set to ‘0’ and the link state is L0s for both link partners, or the cfg_l1_entr_wo_rl0s field of PRTLGC3 is set to ‘1’.
- L1 entry conditions as defined in Section 5.4.1.2.1, “Entry into the L1 State” of the *PCI Express Base Specification, Revision 4.0, Version 1.0*, exist for a duration of time (determined by the cfg_l1_entr_latency_timer field in PRTLGC3).
- No higher stage of power-down requested.
- Your USP application is not asserting the app_xfer_pending input.
- There are no in-progress transactions1 in the controller such as:
  - Rx queue P/NP Request TLPs.

- Rx queue CPL TLPs.
- Outstanding/expected wire completions.
- Outstanding/expected application (at the AXI master interface when present) completions.

- There are no pending requests at the AXI slave interface, that is, slv_a*valid must be 0.
- There are no pending DMA transfers.

**L1-ASPM Entry (Scenario 2): L1 Idle Timeout In L0**

The controller enters L1-ASPM when all of these conditions are met in the USP:

- ASPM L1 is enabled and L0s is not enabled through the ASPM Control field in the Link Control register.
- Link state is L0.
- L1 entry conditions as defined in Section 5.4.1.2.1, “Entry into the L1 State” of the *PCI Express Base Specification, Revision 4.0, Version 1.0*, exist for a duration of time (determined by the cfg_l1_entr_latency_timer field in PRTLGC3).
- No higher stage of power down-requested.
- Your USP application is not asserting the app_xfer_pending input.
- There are no in-progress transactions in the controller such as:
  - Rx queue P/NP Request TLPs.
  - Rx queue CPL TLPs.
  - Outstanding/expected wire completions.
  - Outstanding/expected application (at the AXI master interface when present) completions.
- There are no pending requests at the AXI slave interface, that is, slv_a*valid must be 0.
- There are no pending DMA transfers.

**L1-ASPM Entry (Scenario 3): Application Controlled (USP only)**

The controller enters L1-ASPM when all of these conditions are met in the USP:

- ASPM L1 is enabled through the ASPM Control field in the Link Control register.
- Your application logic pulses the app_req_entr_l1 input.
- Your USP application is not asserting the app_xfer_pending input.
- There are no in-progress transactions1 in the controller such as:
  - Rx queue P/NP Request TLPs.
  - Rx queue CPL TLPs.
  - Outstanding/expected wire completions.
  - Outstanding/expected application (at the AXI master interface when present) completions.

- There are no pending requests at the AXI slave interface, that is, slv_a*valid must be 0.
- There are no pending DMA transfers.

**L1-PM Entry**

The power management state of a link is determined by the D-state of the USP. When you change the device state of the USP to D1 or $D3_{hot}$ by writing to the PMCAP1F*i*, the controller must initiate a link state transition to L1. The USP of a non-ARI multi-function device must not initiate a link state transition to L1 until all of its functions have been programmed to a non-D0 D-state. L1 state is a bi-directional link low-power state. Both link partners must negotiate to go this state. The L1-PM negotiation handshake uses the following DLLPs:

- PM_Enter_L1.
- PM_Request_Ack.

All of these condition must be met:

- L1 entry conditions are satisfied as defined in Section 5.3.2.1, “Entry into the L1 State” of the *PCI Express Base Specification, Revision 4.0, Version 1.0*.

**Note**: An USP application asserting the app_xfer_pending input does not prevent L1-PM entry, but causes immediate exit from L1-PM.

**L1-PM/L1-ASPM Exit**

The controller exits L1-PM/L1-ASPM if any of these conditions is met:

- A DLLP or TLP is pending to be sent.
- Your application asserts the app_req_exit_l1 input.
- Link partner is requesting exit from L1.
- Your application asserts the app_xfer_pending input.
- Your application asserts the apps_pm_xmt_pme input. When the USP is programmed with capability to support PME; it sends a PME message to the RC which calls the PM software to transition the USP to the D0 state. Therefore, you should only use apps_pm_xmt_pme for L1-PM exit.
- PM software (RC) requests a higher stage of power-down by writing to the PMCAP1F*i* in the USP.
- Your application (USP) requests transmission of VDM, MSI, or LTR message.
- Your application is requesting to send traffic by asserting client0_tlp_hv (non-AXI)
- Your application enables a DMA read or write channel, or DMA controller is requesting data.
- Your application (DSP) initiates link disable, or link retrain (by setting cfg_link_dis or cfg_link_retrain field in EXPCAP4Fi to 1).
- Your application (DSP) initiates hot reset by either:
  - setting cfg_plreg_reset field in PRTLGC4 to 1, or
  - setting cfg_2nd_reset field in PCICONF15F*i* to 1, or
  - toggling app_init_rst
- Your application (DSP) requests transmission of Unlock message.

- Your application requests a speed change (by setting cfg_directed_speed_change field in PRTLGC21 to 1).
- Your application requests link width change (by setting cfg_pl_multilane_control[6] field in PRTLGC59 is set to 1).
- Any activity in the RADM, as indicated by radm_idle being set to 1'b0.
- Clearing r_radm_clk_gating_en field in PRTLGC52 register.

***Note:** The CDM space is accessible in L1, but some CDM register bits do not produce the desired results. Writing to CDM register bits which trigger LTSSM state transitions in L1 does not have the desired effect because the LTSSM clock is not running in L1. radm_idle is set to 1'b0 if AERCAP6F*i*.ECE is set to 1'b1 during L1.

****Caution:**
**DMA Doorbell L1**
- Any DMA register can be accessed in L1 because the DMA registers are clocked on aux_clk_g.
- When core_clk is removed in L1 substates or L1 clock power management, aux_clk_g is used.

**DMA Doorbell L1 Exit**
- Doorbelling any DMA read/write channel triggers L1 exit.
- Writing to other DMA registers does not trigger L1 exit.

### (b) L1 Clock PM (L1 with REFCLK Removal/PLL Off) Overview

This feature allows you to shut down the PLL and the reference clock during standard (non-substates) L1. Table 104.39 lists the inputs and outputs that are considered for L1 Clock PM.

**Table 104.39 Inputs and Outputs to Consider for L1 Clock PM**

| CX_L1_SUBSTATES_ENABLE | CX_PIPE_VER | I/O SIGNALS |
|---|---|---|
| 0 | 0 | clk_req_n<br>phy_clk_req_n |
| | 1 | clk_req_n<br>mac_phy_powerdown[3:0] = CX_PIPE43_P1CPM_ENCODING<br>phy_mac_phystatus |
| 1 | 0 | mac_phy_pclkreq_n[0]<br>phy_mac_pclkack_n |
| | 1 | mac_phy_pclkreq_n[0]<br>mac_phy_powerdown[3:0]= CX_PIPE43_P1CPM_ENCODING<br>phy_mac_phystatus |

For PIPE 4.3 (CX_PIPE_VER =1),

- The controller and PHY use mac_phy_powerdown[3:0] to indicate the PHY power state.
- You can specify the power-down encodings using the CX_PIPE43_P1CPM_ENCODING parameter.

- You can specify P1.CPM entry sequence using the CX_PIPE43_P0_P1CPM parameter.

For an USP, host software uses the Enable Clock Power Management bit in the Link Control register to enable this feature. For a DSP, this register bit is hard coded to '0' and cannot be used to control this behavior. Your application can use the app_clk_pm_en input to dynamically control whether to execute L1 with or without Clock PM. The CLKREQ# signal is not directly controlled by the controller.

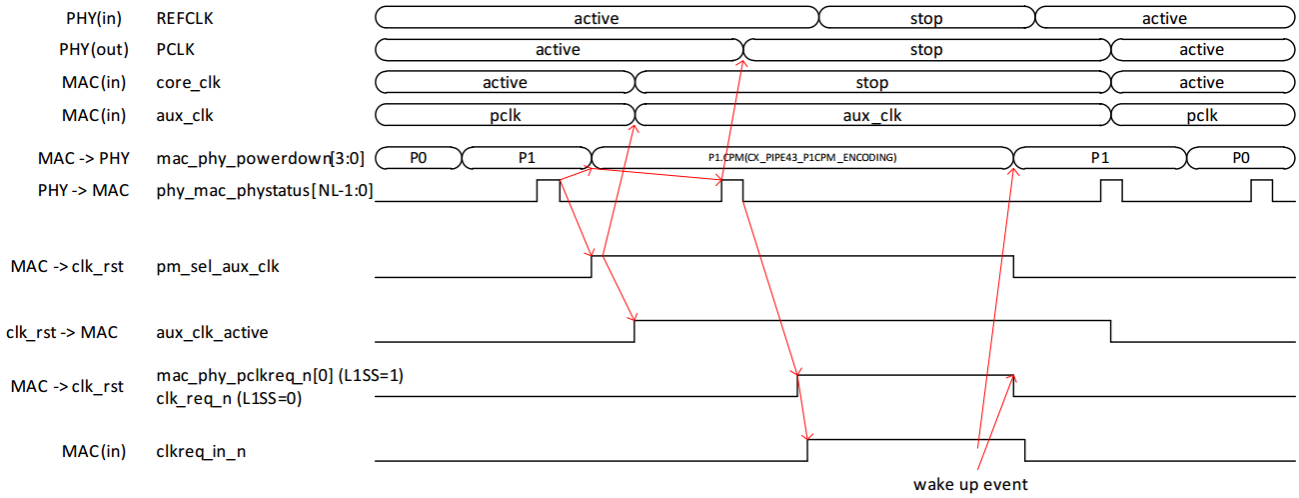



**Figure 104.39 PIPE 4.3 L1 CPM Operation (CX_PIPE43_P0_P1CPM=0)**

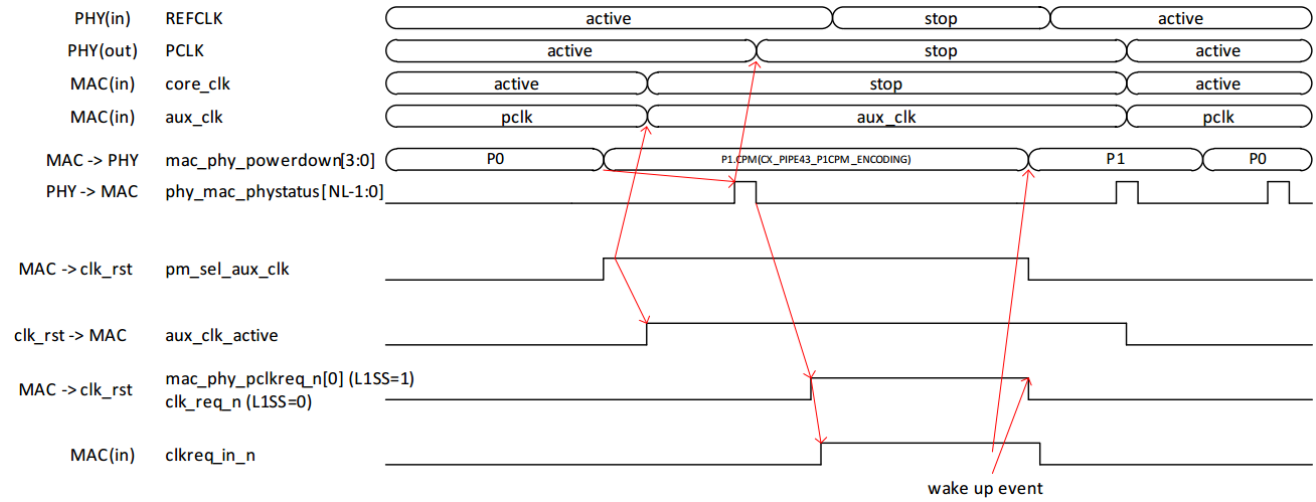



**Figure 104.40 PIPE 4.3 L1 CPM Operation (CX_PIPE43_P0_P1CPM=1)**

### (c) L1 Clock PM (L1 with REFCLK Removal/PLL Off); Entry and Exit Conditions

To enable the controller (upstream port only) to execute L1 with Clock PM:

- The Support Clock Power Management bit in the Link Capabilities register must be set. The default value of this register is controlled by the DEFAULT_CLK_PM_CAP parameter. For downstream ports it is hardcoded to 0, for upstream ports it can be accessed through the DBI.
- The Enable Clock Power Management bit in the Link Control register must be set.
- You must set the app_clk_pm_en input to 1. The controller only samples app_clk_pm_en when L1 is entered.

L1 with Clock PM and L1 substates work orthogonal to each other. L1 with Clock PM uses the mac_phy_pclkreq_n[0] signaling, and L1 substates uses the mac_phy_pclkreq_n[1]signaling. However, L1 substates takes precedence over Clock PM within the cores PM state machine. This means that when the entry conditions for any L1 substate are satisfied then the controller's LTSSM executes the corresponding L1 substate protocol.

For details on L1 exit latency, see “L1-PM/L1-ASPM Exit”.

### (5) L2 and L3 Power Down Entry and Exit Conditions (USP)

This section discusses the L2 and L3 link states. The following topics are discussed:

- “Overview”
- “L2/L3 Entry”
- “L2/L3 Exit”

**Overview**

L2 entry is initiated(1) after the RC calls power management software to initiate the removal of power and clocks. USPs of devices in D0, D1 and $D3_{hot}$ must respond to the receipt of a PME_Turn_Off MSG TLP by transmitting a PME_TO_Ack MSG TLP. The device must then request a link transition to L2_Ready. L2_Ready is a bi-directional link power down state. If your application is not ready to be shut-down, it must keep the app_ready_entr_l23 input de-asserted. This causes the controller to delay sending the PM_Enter_L23 DLLP and thereby stalling the negotiation handshake that uses the following DLLPs:

- PM_Enter_L23.
- PM_Request_Ack

When your application is eventually ready for transitioning to $D3_{cold}$, that is, loss of main power and reference clock, L2_Ready is entered and the downstream device begins preparation for the power and clock removal. After main power has been removed, the link transitions to L2 if Vaux is provided, or it transitions to L3 if no Vaux is provided.

***Note: PCIe does not support L3.**

**L2 Entry**

All of these condition must be met:

- PME_Turn_Off/PME_TO_Ack handshake has been completed.
- Your USP application is ready to be turned off; app_ready_entr_l23 =1.
- After sending the PME_TO_Ack, the USP initiates the L2 Ready transition protocol by sending the PM_Enter_L23 DLLP. The RC responds with the PM_Request_Ack.

Note:
When PCIEPWRMNGSTS0.smlh_ltssm_state is set to '6'h0D' (RCVRY_LOCK) and PCIEPWRMNGSTS1.pm_master_state is set to '5'h18' (Wait for PM DLLP flush), the application should trigger the perst_n (warm reset)

**L2 Exit**

Any of these condition can be met:

- When the USP is programmed(2) with capability to support PME, your application can assert the apps_pm_xmt_pme input to request the controller to wakeup. The USP then sends a PM_PME MSG TLP to the RC which calls the PM software to transition the USP out of the D3 state.
- Device is programmed with capability to support PME and your application requests the controller to wakeup by triggering a native hot-plug event.
- Link partner is requesting exit from L2/L3.

The controller directs the PHY to generate beacon signaling when a PCIe device initiates a wake-up event.

Note :

1. In preparation for removing the main power source, your RC application asserts the apps_pm_xmt_turnoff input which causes the RC to broadcast the PME_Turn_Off MSG TLP to all USPs.

2. pm_pme_en bit is set in PMCAP1Fi, and the cfg_pme_support RT bit is set in PMCAP0Fi for the corresponding D-state for which the function is currently in.

PCIE controller is stuck in L2.IDLE during L2 exit by Electrical Idle.
Note #1:
Set the P2NOBEACON_ENABLE field of the PRTLGC60 register to '1'.
Note #2:
Always use warm reset (perst_n) to exit L2 state.
Note #3:
Upstream port can use WAKE# to exit L2 state.
System Usage note: Beacon is not available as a method for L2 exit.

### 104.3.15 Data Protection

This section discusses how the controller maintains the integrity of the link using different methods at different points.

**(1) Overview**

PCIEC RAS consists of three fundamental components:

- Reliable protocol architecture involving LCRC, ACK/NAK handshake, and Replay mechanisms.
- Internal device-level Datapath and RAM protection (RAS DP) of data, header, and RAM control signals from application I/F to LCRC extraction/generation points.
- Debug, Error Injection, and Statistic gathering (RAS DES).

**(2) RAS Data Protection (DP)**

- For the parts of the PCIEC outside of LCRC protection, PCIEC has the RAS DP feature consisting of:
- RAM Protection (ECC on data and Parity on address)
- An ECC checksum is calculated and added to data that is written to the RAM. When the data is read from the RAM, then the PCIEC recalculates the ECC and compares it against the value read from RAM. RAM ECC protection is provided for all of the RAMs in the PCIEC, except for the RAS-DES RAM. ECC corrects single-bit errors and reports (without correction) multi-bit errors.
- Datapath Protection (Parity)
- For Tx traffic, PCIEC adds parity protection codes to AXI bus address and data.
- For Rx traffic, PCIEC checks the protection codes on AXI bus address and data.

**(3) Uncorrectable Error Handling**

Upon detection of first Uncorrectable error (RAM and Datapath/Interface Protection), depending on value of cfg_rasdp_linkdown_en field of the VSE2CAP10, the PCIEC takes one of the two actions:

- Forces the PCIe link to go down (cfg_rasdp_linkdown_en =1), or
- Enters RASDP Error Mode (cfg_rasdp_linkdown_en =0)

**(a) PCIe Link Down**

- If the PCIEC is configured to cause the link to go down upon detection of an Uncorrectable error, the controller resets and resumes normal operation.

**(b) RASDP Error Mode**

On entering RASDP error mode, the PCIEC sets

slv_rasdp_error_mode/mstr_rasdp_error_mode.

Wire access to CDM registers is suspended to prevent corruption of the register configuration space.

TLP handling in RAS DP Error mode:

- Rx TLP Handling
  - Rx Errored TLPs are Forwarded to AXI bus.
  - Subsequent Rx TLPs not Guaranteed to be Correct; you Must Discard all TLPs.
- Tx TLP Handling for Dual lane link
  - Bad TLP is Nullified (EDB Inserted and LCRC Inverted) and Transmitted.
  - Subsequent TLPs are Nullified and Transmitted.
  - In-progress Replay is Nullified and Transmitted.
  - Subsequent Replays are Silently Dropped.
- Tx TLP Handling for Quad Lane link
  - Bad TLP is Silently Dropped.
  - Subsequent TLPs are Silently Dropped.
  - In-progress Replay is Nullified and Transmitted.
  - Subsequent Replays are Silently Dropped.

Credits accuracy is not retained under uncorrectable error conditions.

Error Logging:

- Uncorrectable Error (2-bit ECC and all parity errors.) is Logged
- Uncorrectable Internal Error Reported (AER)

**(4) Correctable Error Handling**

The PCIEC takes the following action to handle correctable (RAM and Datapath/Interface Protection) errors:

- Correctable Error(1-bit ECC errors) is Logged
- Correctable Internal Error Reported (AER)
- TLP is Modified/Corrected
- Errors Reported When Detected (Immediately)
- Error Correction Enable/Disable Control (Per TX/RX Direction and Per-Layer)
- If the PCIEC is capable of single bit error correction, disabling ECC error correction does not change the nature of the errors and their reporting. A single-bit error is reported as a correctable error although correction might be disabled.

**(5) RAS Debug, Error Injection, and Statistics (DES)**

**(a) Debug**

To facilitate system level debugging the PCIEC provides the following features:

- General Diagnostic Support (Legacy Internal Probe Points).
- Silicon Debug Support (Control and Status of PIPE, Deskew, Ack/Nak, Reversal, FC, EQ, LTSSM, Replay)

**(b) Error Injection**

The PCIEC provides support to inject the following types of errors:

- CRC Error
- Sequence Number Error

- DLLP Error
- Symbol DataK Mask Error or Sync Header Error
- FC Credit Update Error
- TLP Duplicate/Nullify Error
- Specific TLP Error

**(c) Statistics**

To facilitate collection of statistics the PCIEC provides the following features:

- Time Based Analysis (RX/TX data throughput and time spent in each low-power LTSSM state)
- Event counters (Error and Non-Error)

### 104.3.16 Error Handling

This section describes handling of the errors detected by this module.

#### (1) Error Types

This module classifies detected errors into four types: three types defined in the PCI Express standard (correctable errors, non-fatal errors, and fatal errors), and DMA controller error. It reports each type of error through the corresponding interrupt. About Error Types classification, refer to 104.3.17.

#### (a) Correctable Error

Correctable errors are defined as the type of errors from which recovery is possible through the PCI Express protocol. Recovery from the errors is performed by hardware.

When the conditions described later are satisfied, occurrence of a correctable error is signaled by the intreq_pci*m*_corerr interrupt. (m = 0)

#### (b) Non-Fatal Error

Non-fatal errors are defined as the type of errors from which recovery is impossible through the PCI Express protocol but which is not fatal. After this error, a single packet is discarded but the subsequent data communication can be continued. In most cases, error processing such as retransmission of the discarded packet is needed.

When the conditions described later are satisfied, occurrence of a non-fatal error is signaled by the intreq_pci*m*_nonfaterr interrupt. (m = 0)

#### (c) Fatal Error

Fatal errors are defined as the type of errors from which recovery is impossible through the PCI Express protocol and which is fatal. Specifically, the errors that require resetting the communication path or other appropriate processing fall into this type. In most cases, PCIEC must be initialized.

When the conditions described later are satisfied, occurrence of a fatal error is signaled by the intreq_pci*m*_faterr interrupt. (m = 0)

#### (d) DMA Controller Error

DMA controller Errors are defined that the DMA fails to complete the transfer, or an error occurs during the transfer. After this error, issuing new requests are stopped and the DMA channel is placed in a halted state.

When the conditions described later are satisfied, occurrence of a fatal error is signaled by the intreq_pci*m*_dma interrupt. (m = 0)

#### (2) Priority of Errors (in A Single Packet)

Upon detecting multiple errors during reception of a single packet, PCIEC reports only one error that it determines as the most important according to the following priority among errors. About Errors, refer to Table 104.40.

(High Priority)

Uncorrectable Internal Error

Receiver Overflow

Flow Control Protocol Error

Malformed TLP

ECRC Check Failed

Unsupported request (UR), Completer Abort (CA) or unexpected completion

Poisoned TLP received

(Low Priority)

### (3) Error Factor

#### (a) Correctable Error

Correctable errors are defined as the type of errors from which recovery is possible without loss of transfer data through hardware processing such as automatic retransmission request.

If a correctable error occurs, the EXPCAP2F*i*[16].CEDTCD and any bit of register AERCAP4F*i* is set to 1. If a correctable error occurs when the EXPCAP2F*i*[0].CERPE bit is 1 and the corresponding bit of register AERCAP5F*i* is 0 in Endpoint mode, the ERR_COR message is transmitted to the Root Port to notify occurrence of an error. Both Root Port mode and End Port mode, the intreq_pci*m*_corerr interrupt is generated if PRTLGC80[4]. masked_safety_int[PCIE_CORR_INDEX] is 1.

In Root Port mode, if the ERR_COR message is received from an Endpoint when all of the PCICONF15F*i*[17].SERRE, EXPCAP2F*i*[0].CERPE, and EXPCAP7F*i*[0].SERRCEE bits are 1, the intreq_pci*m*_sub interrupt is generated. (*m* = *0*, *i* = 0, 1)

PCIE controller begins to exit the LTSSM state L0 to Recovery.Lock due to receiving TS1 Ordered Sets from the remote partner, while operating in the 4s mode at Gen1 or Gen2 data rate. If a Skip Ordered Set, with a Skip symbol removed is received, it may disrupt the deskew logic to cause an unexpected deskew alignment error (RAS D.E.S register), bad TLP or bad DLLP(correctable errors).
Not any needed, as the controller recovers itself.

The following errors are classified as correctable errors.

**(a-1) Receiver Error**

8b/10b or 128b/130b decode error

Disparity error

Elastic buffer overflow

Elastic buffer underflow

**(a-2) BAD TLP**

LCRC error

Sequence number error

**(a-3) BAD DLLP**

16-bit CRC error

**(a-4) Replay Timeout**

Ack/Nak timeout

**(a-5) Replay Num Rollover**

Replay Num rollover

**(a-6) RAM 1-bit ECC Errors**

RAM Protection ECC 1-bit errors.

**(a-7) Header Log Overflow**

Multiple Error Header Log overflow.

**(b) Non-Fatal Error**

A non-fatal error cannot be corrected through hardware control and one packet of data will be lost. Therefore, recovery processing by software, such as retransmission processing, is needed.

If a non-fatal error occurs, the EXPCAP2F*i*[17].NFEDTCD bit and the corresponding bit inside of register AERCAP1F*i* is set to 1. If a non-fatal error occurs when PCICONF1F*i* [8].SERRE is 1, the PCICONF1F*i* [30].SSE bit is set to 1. If a non-fatal error occurs when the corresponding bit in register AERCAP2F*i* is 0, the EXPCAP2F*i*[1].NFERPE bit or PCICONF1F*i* [8].SERRE bit is 1 in Endpoint mode, the ERR_NONFATAL message is transmitted to the Root Port to notify occurrence of an error. Both Root Port mode and End Port mode, the intreq_pci*m*_nonfaterr interrupt is generated if PRTLGC80 [3].masked_safety_int[PCIE_UNCORR_INDEX] is 1. However, if the error source is an unsupported request (described later), this module does not send the message or generate an interrupt when the conditions described later are satisfied.

In Root Port mode, if the ERR_NONFATAL message is received, PCICONF7F*i* [30].RSE is set to 1. If the ERR_NONFATAL message is received when the PCICONF15F*i*[17].SERRE and PCICONF1F*i*[8].SERRE bits are 1, the PCICONF1F*i*[30].SSE bit is set to 1. If the ERR_NONFATAL message is received from an Endpoint when the PCICONF15F*i*[17].SERRE bit is 1, the EXPCAP2F*i*[1].NFERPE or PCICONF1F*i*[8].SERRE bit is 1, and the EXPCAP7F*i*[1].SERRNFEE bit is 1, the intreq_pci*m*_sub interrupt is generated. However, if the error source is an unsupported request (described later), this module does not generate an interrupt when the conditions described later are satisfied. In addition, if the non-fatal error comes under the advisory non-fatal error cases described later, separately defined processing is performed. (*m* = *0*, *i* = 0, 1)

The following errors are classified as non-fatal errors.

**(b-1) Poisoned TLP Reception**

If a packet with the Poisoned attribute is written, it is handled as a non-fatal error.

This module does not transfer a write request of a packet from the PCI Express side to the internal bus side if the packet has the Poisoned attribute. If a request with the Poisoned attribute is received, the packet is discarded.

If the completion responding to the request issued by this module has the Poisoned attribute, the data in the completion is not transferred to the internal bus side.

If ConfigrationWrite Type0, I/O Write, or MSI has the Poisoned attribute, it is handled also as an unsupported request.

**(b-2) Unsupported Request (UR) Reception**

The following requests are handled as unsupported requests.

- Access to the area other than the PCI address area specified with BAR
- Reception of memory/IO request in Root Port mode when PCICONF1F*i*[2].BME = 0
- Access to memory space when PCICONF1F*i*[1].MSE = 0
- Access to IO space when PCICONF1F*i*[0].IOSE = 0
- Reception of a message with an undefined message code
- Reception of MsgD with message code Msg
- Reception of a message to be received only by an Endpoint in Root Port mode
- Reception of a message to be received only by a Root/Switch in Endpoint mode
- Reception of TLP other than configuration access and message in the Non-D0 state
- Access to a configuration register not implemented
- Reception of a Type1 Configuration Request

- Reception of a Configuration Request in Root Port mode
- Reception of a MRdLk request

If an unsupported request is received, the EXPCAP2F*i*[19].URDTCD bit is set to 1.

If an unsupported request is received when both the EXPCAP2F*i*[3].URRPE and PCICONF1F*i*[8].SERRE bits are 0, neither transmission of a message nor generation of an interrupt is performed.

If an unsupported request is received, the packet is discarded.

**(b-3) Completer Abort (CA)**

If an error occurs on the internal bus when the packet is transferred from the PCI Express to the internal bus, this module considers it as a Completer Abort and performs error handling. When the packet received from the PCI Express is a Non-Posted request, PCIEC returns a completion with the Completer Abort (CA)– Completion Status thus setting the PCICONF1F*i*[27].STA bit to 1.

If the packet received by PCIEC from the PCI Express side violates the PCI Express standard or is considered to be damaged, it is handled as an unsupported request or malformed TLP, not as a Completer Abort.

**(b-4) Unexpected Completion Reception**

If the completion not corresponding to the request issued by PCIEC is received, it is considered as an unexpected completion and error handling is performed.

Reception of an unexpected completion falls into the advisory non-fatal error cases described later.

**(b-5) ECRC Check Failed**

If the packet received by PCIEC from the PCI Express side violates ECRC check, it is considered as ECRC Check Failed. When the packet received from the PCI Express is a Non-Posted request, PCIEC returns a completion with the Completer Abort (CA)– Completion Status thus setting the PCICONF1F*i*[27].STA bit to 1.

**(c) Fatal Error**

Fatal errors are defined as the type of errors from which recovery is impossible through hardware and which usually require system-level error recovery processing such as initialization of the PCI Express.

If a fatal error occurs, the EXPCAP2F*i*[18].FEDTCD and the corresponding bit in register AERCAP1F*i* is set to 1. If a fatal error occurs when the PCICONF1F*i*[8].SERRE bit is 1, the PCICONF1F*i*[30].SSE bit is set to 1. If a fatal error occurs when the corresponding bit in register AERCAP2F*i* is 0, the EXPCAP2F*i*[2].FERPE or PCICONF1F*i*[8].SERRE bit is 1, in Endpoint mode the ERR_FATAL message is transmitted to the Root Port to notify occurrence of an error. Both Root Port mode and End Port mode, the intreq_pci*m*_faterr interrupt is generated if PRTLGC80[3].masked_safety_int[PCIE_UNCORR_INDEX] is 1.

In Root Port mode, if the ERR_FATAL message is received, PCICONF7F*i*[30].RSE is set to 1. If the ERR_FATAL message is received when both of the PCICONF15F*i*[17].SERRE and PCICONF1F*i*[8].SERRE bits are 1, the PCICONF1F*i*[30].SSE bit is set to 1. If the ERR_FATAL message is received from an Endpoint when the PCICONF15F*i*[17].SERRE bit is 1, the EXPCAP2F*i*[2].FERPE or PCICONF1F*i*[8].SERRE bit is 1, and the EXPCAP7Fi[2].SERRFEE bit is 1, the intreq_pci*m*_sub interrupt is generated. (*m* = *0*, *i* = 0, 1)

The following errors are classified as fatal errors.

(c-1) Data Link Layer Protocol Error
(c-2) Reception of malformed TLP
(c-3) Receiver Overflow
(c-4) Flow Control Protocol Error
(c-5) RAM 2-bit ECC errors

(c-6) CDM Register Check Error
(c-7) Tx Interface Progression Detection Error

The following packets are considered as malformed TLPs, which cause a fatal error when received. These packets are discarded when received.

- Reception of a packet with data size exceeding the Max Payload Size
- Reception of a packet with data of different length from the length specified in the size field
- Reception of a TLP with TD field in header = 0 and with the digest field
- Reception of a TLP with TD field in header = 1 and without the digest field
- Reception of a power management message of non TC0
- Reception of an error signal message of non TC0
- Reception of an unlock message of non TC0
- Reception of a Set_Slot_Power_Limit message of non TC0
- Reception of a Set_Slot_Power_Limit message without data
- Reception of a completion with data of different size from the size specified in the Length field
- Reception of a completion with data size exceeding the Max Payload Size
- Reception of a TLP of undefined type or fmt
- Reception of a TLP of the unassigned TC
- Reception of an IO request of TC!=0, Attr[1:0]!=0, Length[9:0]!=1, or Last DWBE[3:0]!=0
- Reception of a configuration request of TC!=0, Attr[1:0]!=0, Length[9:0]!=1, or Last DWBE[3:0]!=0
- Reception of a memory request with illegal DWBE
- Reception of a memory request crossing a 4-Kbyte boundary

**(d) DMA Controller Error**

DMA controller errors are defined that the DMA fails to complete the transfer, or an error occurs during the transfer.

PCIEC can signal the following events as the DMA controller errors by setting the corresponding registers. See 54.17 for more information.

For DMA Read transfer:

- Application Write Error Detected
- Unsupported Request (UR)
- Completer Abort (CA)
- Completion Timeout
- Data Poisoning
- Linked List Element Fetch Error Detected

For DMA Write transfer:

- Application Read Error Detected
- Linked List Element Fetch Error Detected

If any DMA controller error occurs, the intreq_pci*m*_dma interrupt is generated.

**(e) Advisory Non-Fatal Error Case**

The PCI Express standard prescribes that some of non-fatal errors should not be processed when the applicable packet is received because the errors should be judged in terms of cause and significance by the packet transmission source, software, or system. These non-fatal errors are called advisory non-fatal error

cases, which include the cases described in the following sections. If such a case is detected, general non-fatal error processing is not performed but processing prescribed individually is performed instead.

**(e-1) Return of Completion with Unsupported Request/Completer Abort Status**

If a completion with unsupported request/Completer Abort status is returned, PCIEC does not perform non-fatal error processing defined in the PCI Express standard. PCIEC records occurrence of an error in EXPCAP2F*i*[16].CEDTCD, the corresponding AERCAP1F*i* bit, and generates the ERR_COR message and the intreq_pci*m*_corerr interrupt if the corresponding bit set to 1.

**(e-2) Completion Timeout**

If a completion timeout occurs, PCIEC performs non-fatal error processing (recovery from software) defined in the PCI Express standard, does not handle this case as an Advisory Non-Fatal Error.

**(e-3) Reception of Unexpected Completion**

If this module receives an unexpected completion, it does not perform non-fatal error processing defined in the PCI Express standard. PCIEC records occurrence of an error in EXPCAP2F*i*[16].CEDTCD, the corresponding AERCAP1F*i* bits/AERCAP4F*i* bits, and generates the ERR_COR message and the intreq_pci*m*_corerr interrupt if the corresponding bit set to 1.

**(e-4) Reception of Unsupported Request/Completer Abort Completion**

If this module receives a completion of unsupported request/Completer Abort, it does not perform non-fatal error processing defined in the PCI Express standard. PCIEC records occurrence of an error in EXPCAP2F*i*[16].CEDTCD, the corresponding AERCAP1F*i* bits/ERCAP4F*i* bits, and generates the intreq_pci*m*_corerr interrupt if the corresponding bit set to 1.

**(e-5) Reception of Poisoned TLP**

According to the PCI Express standard, the case in which a Poisoned TLP is received and reception is continued is also defined as an advisory non-fatal error case; however, an advisory non-fatal error of such a case does not occur with this module since PCIEC prohibits continuation of Poisoned data processing. Although the cases related to the intermediate receiver are also defined by the standard, the corresponding processing is not performed since PCIEC not operate as an intermediate receiver.

### 104.3.17 Interrupts

Table 104.40 shows the interrupt signals of this module, definition, and the conditions under which interrupts are generated. The interrupt signals are synchronized with the internal bus clock(S0D4_HSCφ).

**Table 104.40 List of Interrupt Signals**

| Name | Definition | Conditions of Generation | Related Registers |
|---|---|---|---|
| intreq_pci*m*_corerr | PCI Express Correctable Error interrupt | Occurrence of a correctable error<br>• Receiver Error<br>• Bad DLLP<br>• Bad TLP<br>• Replay Timer Timeout<br>• RAM 1-bit ECC errors<br>• Header Log Overflow | • PCIEERRSTS0[4] |
| intreq_pci*m*_nonfaterr | PCI Express Uncorrectable Error interrupt | Occurrence of an uncorrectable (fatal & non-fatal) error<br>• Reception of Malformed TLP<br>• Data Link Layer Protocol Error<br>• Receiver Overflow<br>• Flow Control Protocol Error<br>• RAM 2-bit ECC errors<br>• CDM Register Check Error<br>• Tx Interface Progression Detection Error<br>• Unsupported request<br>• Poisoned TLP<br>• ECRC Check Failed<br>• Completion Timeout<br>• Completer Abort<br>• Unexpected Completion | • PCIEERRSTS0[5] |
| intreq_pci*m*_faterr | PCI Express Uncorrectable Error interrupt | Same as intreq_pci*m*_nonfaterr | • PCIEERRSTS0[5] |
| intreq_pci*m*_lperr | PCI Express error message reception interrupt | Reception of an ERR message from link partner.<br>• RC Error Interrupt is received.<br>• RC Error MSI is received.<br>• System Error (RC only) is received. | • PCIEINTSTS0[28:27]<br>• PCIEINTSTS0[30:29]<br>• PCIEERRSTS0[10:9] |

| Name | Definition | Conditions of Generation | Related Registers |
|---|---|---|---|
| intreq_pci*m*_sub | PCI Express message reception interrupt | Reception of a message<br>• PM_PME is received.<br>• Unlock message is received.<br>• LTR is received.<br>• PME Turnoff is received.<br>• PME_TO_Ack is received.<br>• ERR_FATAL message is received.<br>• ERR_NONFATAL message is received.<br>• ERR_COR message is received.<br>Indicate LTSSM status:<br>• Data link layer is up.<br>• PHY link is up<br>Caused by Power Management Event (PME):<br>• MSI caused by PME<br>• Interrupt caused by PME | • PCIEMSGRCPSTS[4]<br>• PCIEMSGRCPSTS[8]<br>• PCIEMSGRCPSTS[0]<br>• PCIEMSGRCPSTS[6]<br>• PCIEMSGRCPSTS[5]<br>• PCIEMSGRCPSTS[3]<br>• PCIEMSGRCPSTS[2]<br>• PCIEMSGRCPSTS[1]<br>• PCIEINTSTS0[7]<br>• PCIEINTSTS0[6]<br>• PCIEINTSTS0[13:12]<br>• PCIEINTSTS0[11:10] |
| intreq_pci*m*_sub | PCI Express link bandwidth change interrupt | Following conditions are true<br>• MSICAP0Fi[16]=1 and EXPCAP4Fi[14]=1 and EXPCAP4Fi[10]=1<br>• PCICONF1Fi[10]=0 and EXPCAP4Fi[10]=1 and EXPCAP4Fi[30]=1<br>• MSICAP0Fi[16]=1 and EXPCAP4Fi[15]=1 and EXPCAP4Fi[11]=1<br>• PCICONF1Fi[10]=0 and EXPCAP4Fi[11]=1 and EXPCAP4Fi[31]=1 | • PCIEINTSTS0[23]<br>• PCIEINTSTS0[22]<br>• PCIEINTSTS0[21]<br>• PCIEINTSTS0[20] |
| intreq_pci*m*_sub | PCI Express reset status interrupt | Change of the following status<br>• Link has gone down or Controller received a hot-reset request.<br>• Same condition as above and after all pending AXI transfers have completed.<br>• Application request downstream port to start Link hot reset.<br>• FLR | • PCIERSTSTS[2]<br>• PCIERSTSTS[1]<br>• PCIERSTSTS[0]<br>• PCIERSTSTS[5:4] |
| intreq_pci*m*_sub | PCI Express-IP power management interrupt | Detection of PM status<br>• Wake Up from power management unit. | • PCIEPWRMNGSTS0[9] |

| Name | Definition | Conditions of Generation | Related Registers |
|---|---|---|---|
| intreq_pci*m*_sub | PCI Express IP Hot-plug logic interrupt | Detection of hot-plug events<br>• INTx interrupt mode.<br>• PME mode.<br>• MSI hot-plug events. | • PCIEINTSTS0[19:18]<br>• PCIEINTSTS0[17:16]<br>• PCIEINTSTS0[15:14] |
| intreq_pci*m*_sub | MSI | MSI Detected | • PCIEINTSTS0[26] |
| intreq_pci*m*_sub | Legacy interrupt | PCI Legacy Interrupts<br>• Assert_INTx is received (radm_int*i*_asserted) .<br>• Deassert_INTx is received radm_int*i*_deasserted ( *i* = a,b,c,d). | • PCIEINTXSTS[0:3]<br>• PCIEINTXSTS[4:7] |
| intreq_pci*m*_sub | Local bus controller external access interrupt | Detection of local bus controller to external register access. | • PCIEINTSTS1[21:20] |
| intreq_pci*m*_sub | Configuration information interrupt | Change of the following<br>• State of bus master enable bit in the PCI-compatible Command register.<br>• State of the Memory Space Enable bit in the PCI-compatible Command register. | • PCIEINTSTS0[1:0]<br>• PCIEINTSTS0[3:2] |
| intreq_pci*m*_sub | Link Equalization Request | Detection of Link Equalization Request | • PCIEINTSTS0[24] |
| intreq_pci*m*_sub | Target completion timeout | Detection of Target completion timeout | • PCIETRGTCMPLUT1[8] |
| intreq_pci*m*_dma | PCIEC DMA Interrupt | Occurrence of PCIEC DMA event<br>• Done: The DMA successfully completes the transfer.<br>• Abort: The DMA fails to complete the transfer, or an error occurs during the transfer. | • PCIEDMAINTSTS[15:0] |
| intreq_pci*m*_vndmsg | PCIEC Vendor Defined Message Interrupt | Occurrence of reception of Vendor defined message from remote device | • PCIEMSGRCPSTS[7] |

*m* = 0

Figure 104.41, Figure 104.42, Figure 104.43 and Figure 104.44 show the interrupts whose corresponding interrupt status and enable bits are located in the multiple registers.

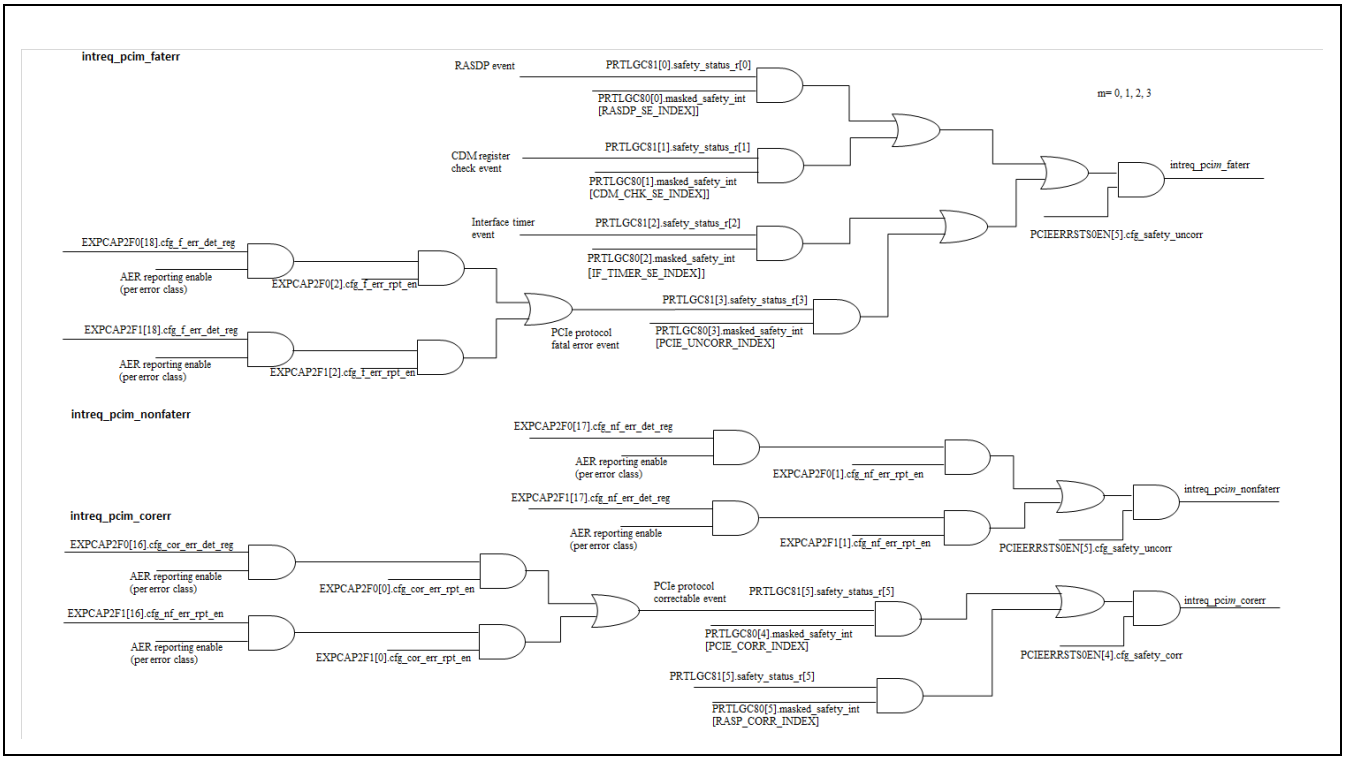



**Figure 104.41 Interrupts Involving Multiple Registers (1)**

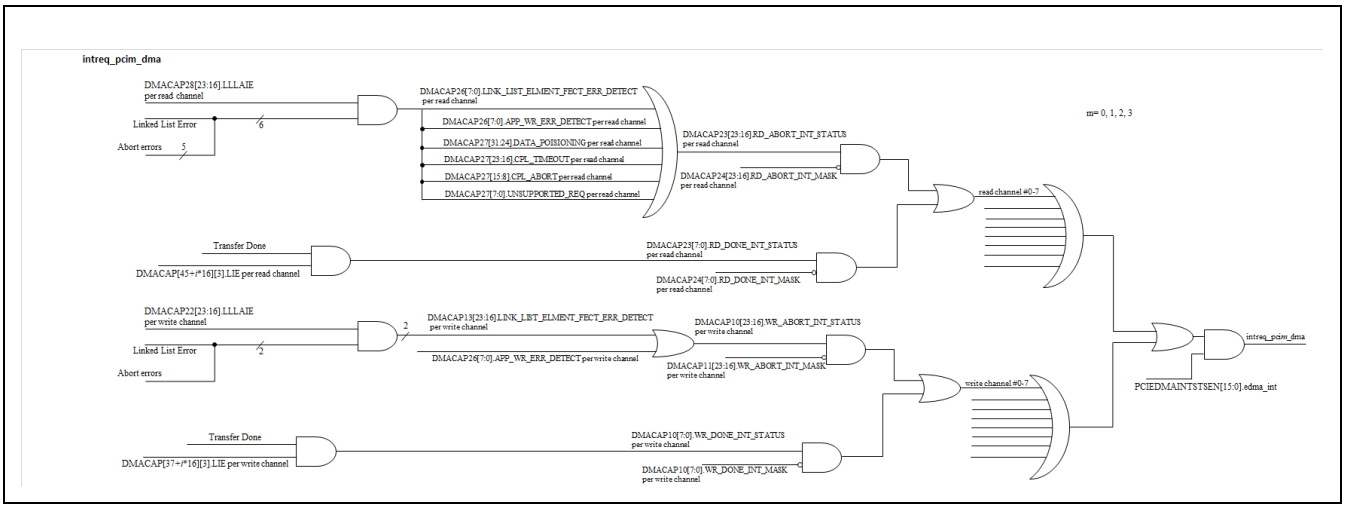



**Figure 104.42 Interrupts Involving Multiple Registers (2)**

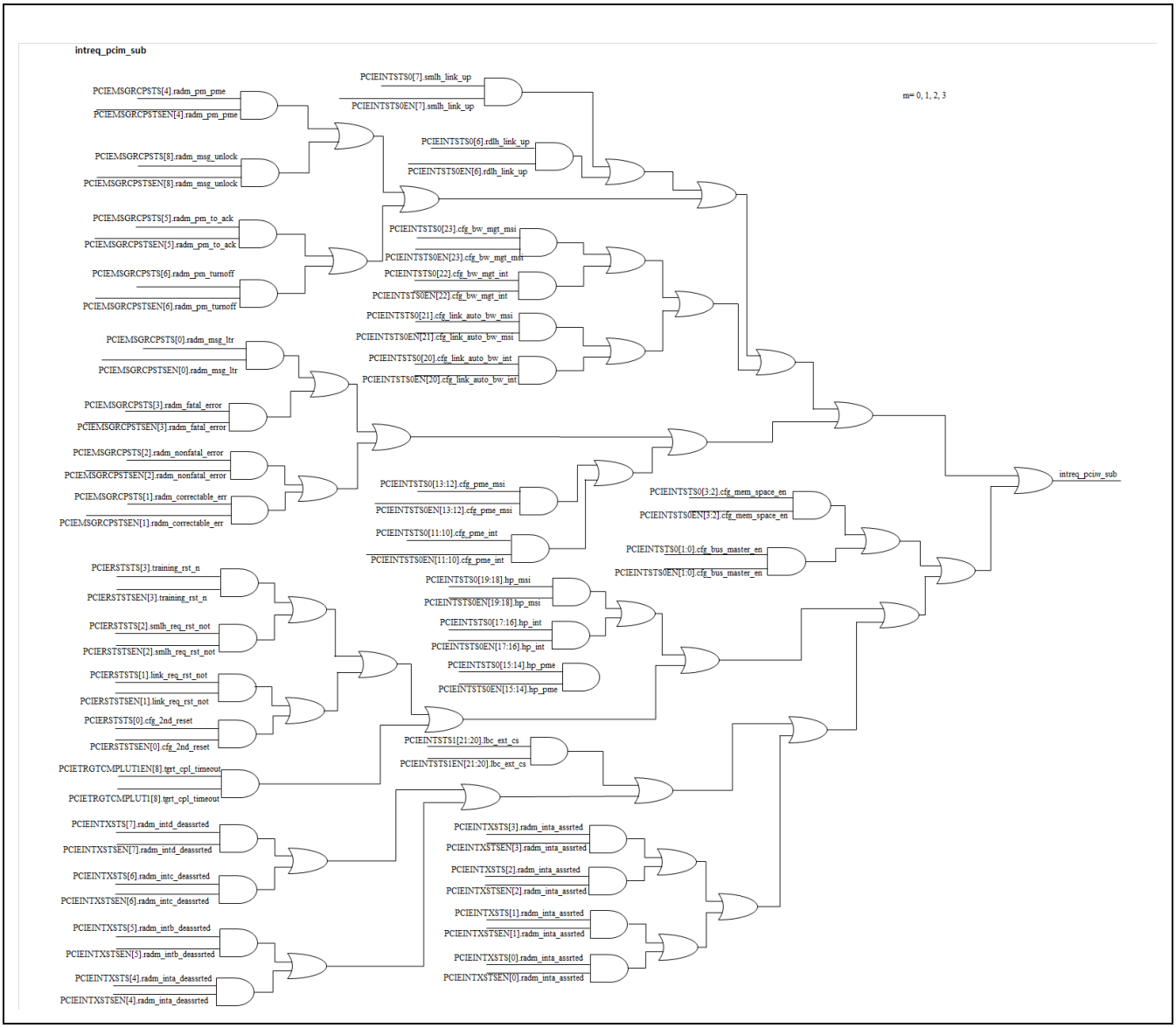



**Figure 104.43 Interrupts Involving Multiple Registers (3)**

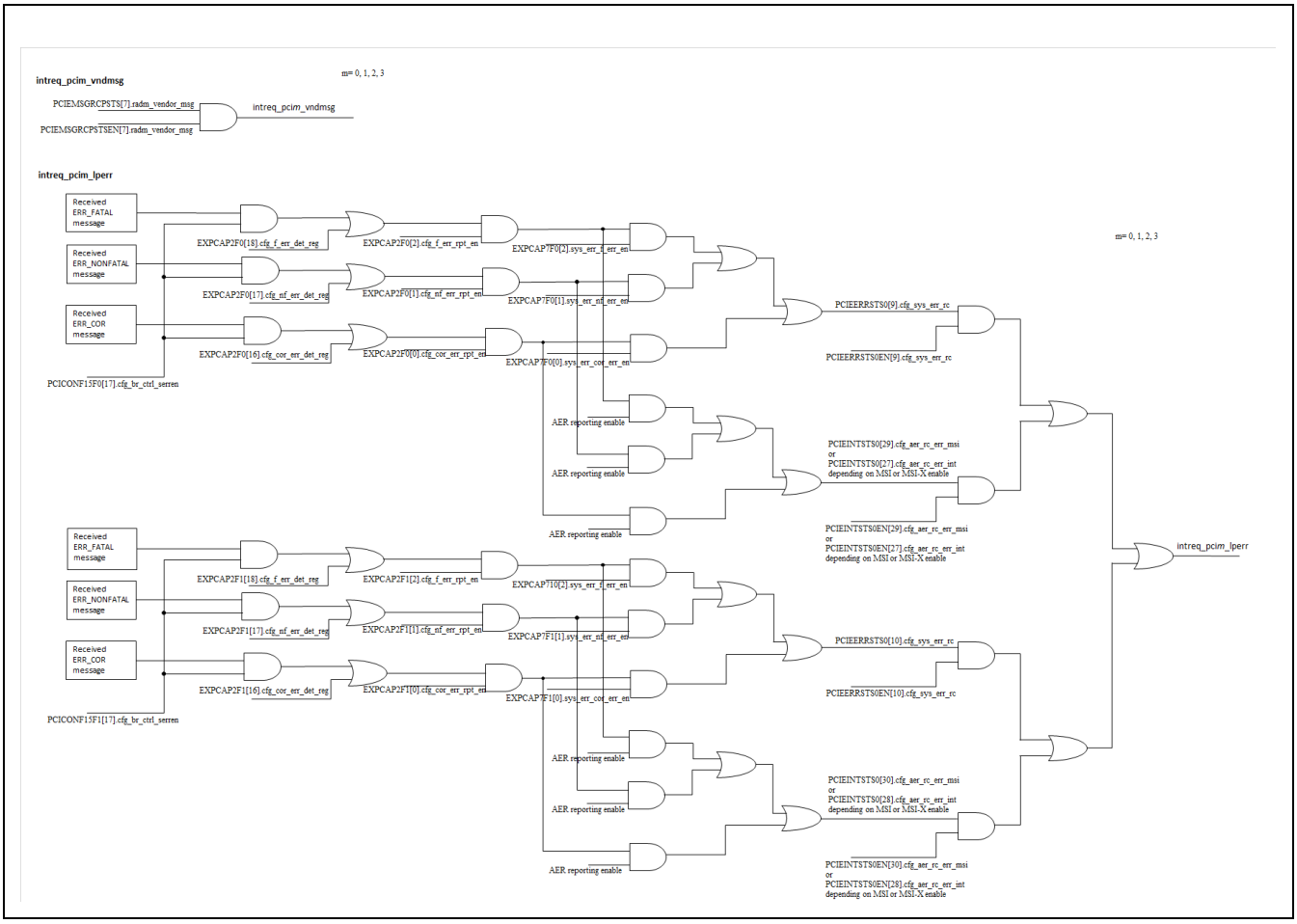



**Figure 104.44 Interrupts Involving Multiple Registers (4)**

### 104.3.18 Messages and MSI

PCIEC supports transmission and reception of the messages and MSI (Message Signaled Interrupt) defined in the PCI Express standard. The messages allow interrupt signaling, power management, error information notification. The MSI is used instead of INTx to signal interrupts. The following sections describe transmission and reception of the messages and MSI.

#### (1) Overview of Messages

PCIEC has the three types of message transmission and reception method.

The first type is automatically message transmission and receptions (Figure 104.45 (a) and Figure 104.46 (a)). The second type is message transmission and receptions by writing and reading the registers of PCIEC (Figure 104.45 (b) and Figure 104.46 (b)). The third types is message transmission and receptions by ATU conversion and received by reading the registers of PCIEC (Figure 104.45 (c) and Figure 104.46 (c)).

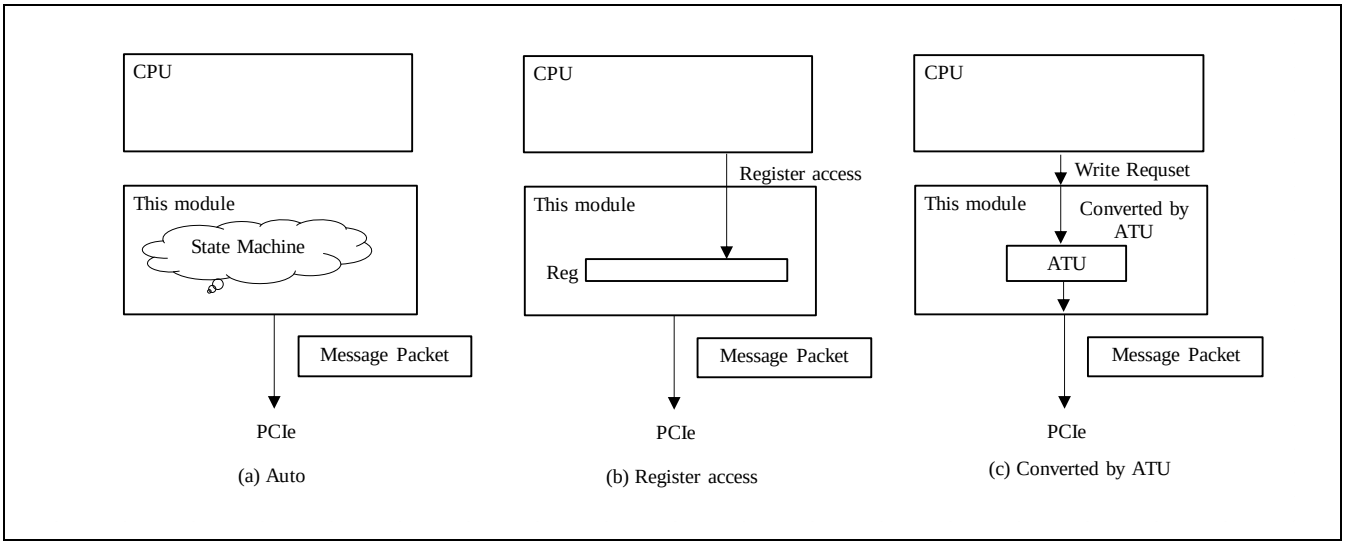



**Figure 104.45 Message Transmission Methods**

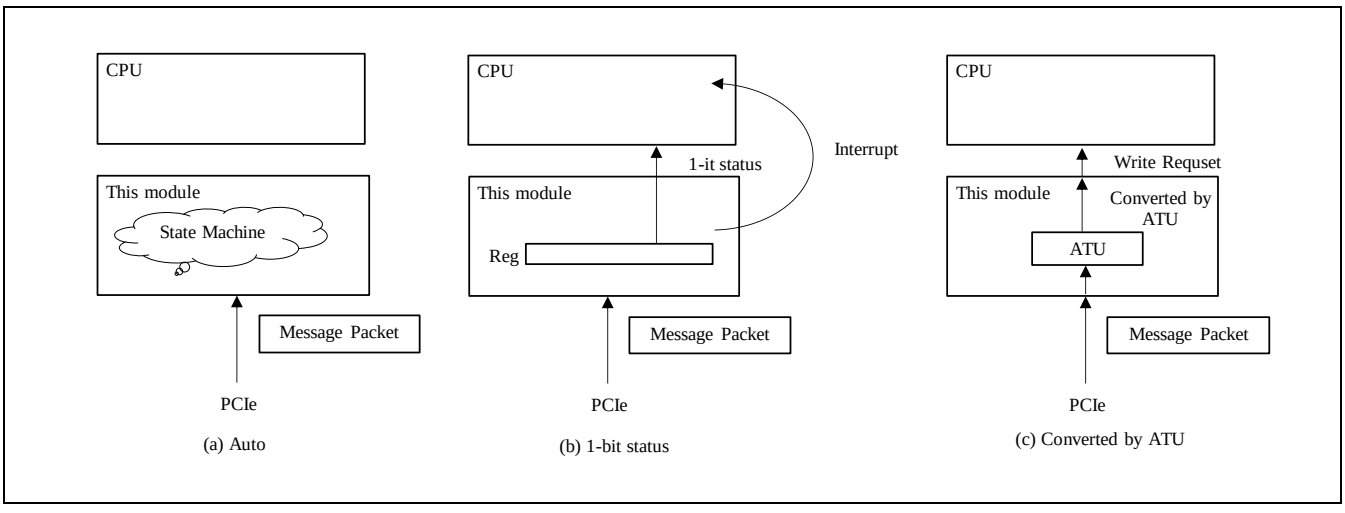



**Figure 104.46 Message Reception Methods**

### (2) Message Transmission

Messages can be transmitted by either of the following three methods.

#### (a) Automatic Message Transmission By Hardware

PCIEC automatically sends a necessary message when an error occurs, LTR clear event occur and PTM event occurs. A part of PM related messages (PME_TO_Ack) is also automatically sent. The following messages are automatically sent.

- ERR_COR: When transmission of the correctable error message is enabled, PCIEC is automatically sent if a correctable error occurs.
- ERR_NONFATAL: When transmission of the non-fatal error message is enabled, PCIEC is automatically sent if a non-fatal error occurs.
- ERR_FATAL: When transmission of the fatal error message is enabled, PCIEC is automatically sent if a fatal error occurs.
- PME_TO_Ack: When PME_Turn_Off message are received by EPs, PME_TO_Ack is automatically sent as response.
- Latency Tolerance Reporting (LTR) clear message: PCIEC automatically sends LTR message with the Requirement bits set to 0 for both Snoop and No-Snoop latencies (LTR clear message) when:
  - A function is directed to a non-D0 state through a write to PMCAP1F*n*[1:0]. Power State, and the LTR mechanism is active since the last DL_Down to DL_Up

  or
  - The LTR mechanism is disabled by clearing EXPCAP10[10].ltr_m_en_r. This field is cleared though a CFG write or function level reset (FLR).
  - By setting cfg_disable_ltr_clr_msg field of the PRTLGC58 register to 1, the automatic generation of the LTR clear message in the above situations can be disabled.

- PTM Request/Response/ResponseD: When PTMCAP2F0[0]. cfg_ptm_enable is set, PCIEC is automatically sent.

#### (b) Initiation of Message Transmission By Writing Through Software

PCIEC sends the PM related message, Vendor_Defined message type0. To send these messages, access the following registers.

- Power management message transmission (PME_* message transmission): Writing 1 to the bits corresponding to the messages in the power management message transmission control register allows transmission of the messages given below: PME_TOFF, and PM_PME.
- LTR message can be generated by writing to PCIELTRMSGCTRL0[0].app_ltr_msg_req to 1. Only use for application where the LTR is changing slowly or static.
- Set_Slot_Power_Limit message, triggered by writing to the Slot Capabilities (EXPCAP5F*i*) register through the DBI, or when LTSSM enters L0.
- Error signaling from the application (Msg). The controller generates error signaling message in response to application requests on the app_err* signal.

#### (c) Initiation of Message Transmission By ATU Conversion

PCIEC sends the LTR request message by ATU conversion. The PM related message, Vendor_Defined message type0 and INTx interrupt message can be transmitted by using this method. To send these message, program the ATU register and issue Write Request from internal bus(AXI). See 54.3.5 for more detail.

### (3) Message Reception

In PCIEC, reception of some messages is automatically handled by hardware. For the message reception processes not performed through hardware, refer to the status and process reception through software. Note that the messages except Vendor_Defined Message are required by the PCI Express standard to be transmitted and received with TC0. If such a message is received with the other TC, the PCIEC considers it as a malformed TLP (fatal error) thus performing error processing automatically.

#### (a) Automatic Message Reception By Hardware

A DSP must automatically send a Set_Slot_Power_Limit message when it enters L0, so the message is sent

and is received by the port itself.

- Set_Slot_Power_Limit

#### (b) Set 1-bit Status and Generating Interrupt

The module hardware detects reception of the following messages but does not automatically process them. Reception of these messages must be checked and processed by software as described below.

- PM_PME: when this message is received, PCIEMSGRCPSTS[4].radm_pm_pme is set. And corresponding enable bit PCIEMSGRCPSTSEN[4].radm_pm_pme is set. The intreq_pci*m*_sub is generated.
- Vendor_Defined Type0: when this message is received, PCIEMSGRCPSTS[7].radm_vendor_msg is set. And corresponding enable bit PCIEMSGRCPSTSEN[7].radm_vendor_msg is set. The intreq_pci*m*_vndmsg is generated.
- LTR request: when this message is received PCIEMSGRCPSTS[0].radm_msg_ltr is set. And corresponding enable bit PCIEMSGRCPSTSEN[0].radm_msg_ltr is set. The intreq_pci*m*_sub is generated.
- Assert_INTx (x is A to D): When this message is received in Root Port mode, PCIEINTXSTS[0,1,2,3].radm_intx_asserted are set (x = a,b,c,d). And the corresponding enable bit PCIEINTXSTSEN[0,1,2,3].radm_intx_asserted are set (x = a,b,c,d). The intreq_pci*m*_sub interrupt is generated. When received in Endpoint mode, this message is considered as a malformed TLP and error processing is performed.
- Deassert_INTx (x is A to D): When this message is received in Root Port mode, PCIEINTXSTS[4,5,6,7].radm_intx_dasserted are set (x = a,b,c,d) are set. And the corresponding enable bit PCIEINTXSTSEN[4,5,6,7].radm_intx_dasserted are set (x = a,b,c,d). The intreq_pci*m*_sub interrupt is generated. When received in Endpoint mode, this message is considered as a malformed TLP and error processing is performed.
- ERR_COR: When this message is received, the appropriate value is set to the configuration register to indicate reception of a correctable error PCIEMSGRCPSTS[1].radm_correctable_err and the corresponding enable bit PCIEMSGRCPSTSEN[1].radm_correctable_err is set. The intreq_pci*m*_sub interrupt is generated.
- ERR_NONFATAL: When this message is received, the appropriate value is set to the configuration register to indicate reception of a correctable error PCIEMSGRCPSTS[2].radm_nonfatal_err and the corresponding enable bit PCIEMSGRCPSTSEN[2].radm_nonfatal_err is set. The corresponding bit the intreq_pci*m*_sub interrupt is generated.
- ERR_FATAL: When this message is received, the appropriate value is set to the configuration register to indicate reception of a correctable error PCIEMSGRCPSTS[3].radm_fatal_err and the corresponding enable bit PCIEMSGRCPSTSEN[3].radm_fatal_err is set. The corresponding bit the intreq_pci*m*_sub interrupt is generated.

STATUS bits are defined for each specified message and 1 is set to the bit correcsponding to the received when a message is received. If the corresponding bit in the ENABLE and STATUS are set to 1, an interrupt will be generated.

**(c) Converted By ATU**

PCIEC converts the received message to Write Request (MEM or IO) and pass to internal bus (AXI) by ATU conversion. This method can be used for message cannot generated by software. All error-free MSG requests are decoded internally, signaled to the application logic..

**(4) Overview of MSI**

This section describes the MSI interrupt.

PCIEC supports MSI interrupt. Whether the PCI Express system should use INTx or MSI interrupts is determined by the Root Port during a configuration cycle. When PCIEC is to be used as a Root Port, whether INTx or MSI interrupt is used should be determined during a configuration cycle, and the results of the determination must be set in the configuration register.

When PCIEC is used as a Root Port, a maximum of 32 interrupt vectors can be used as MSI interrupts.

PCIEC supports both Per Vector Masking and the transmission of 64-bit address messages MSI.

### (5) MSI Transmission

MSI can be transmitted only in Endpoint mode. Figure 104.47 shows an MSI transmission flow.

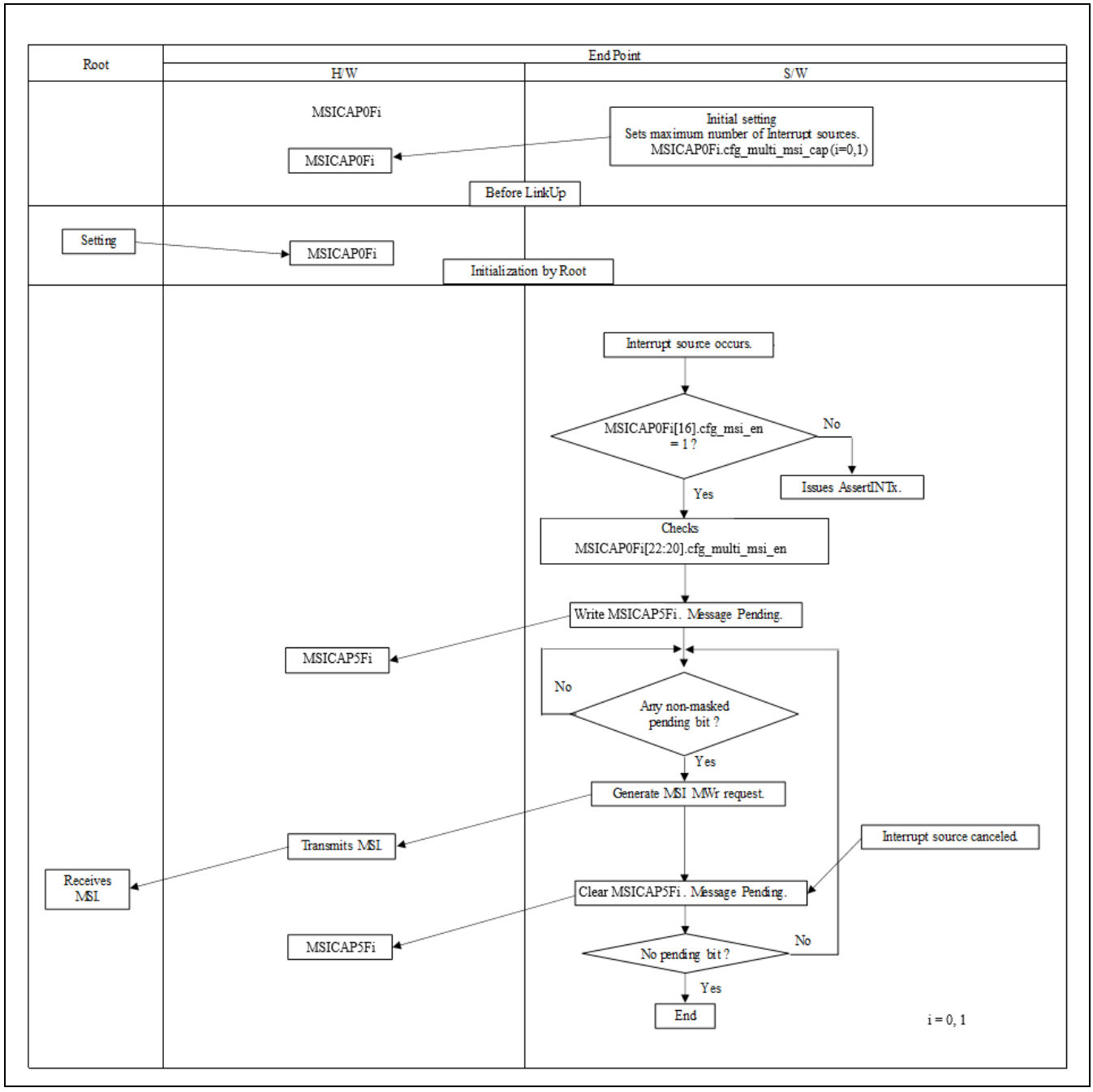



**Figure 104.47 MSI Transmission**

### (6) MSI Reception

MSI can be received only in Root Port mode. The following describes a MSI reception flow.

#### (a) Overview of MSI Reception

When PCIEC is used as a Root Port, whether INTx or MSI interrupts are used should be determined during a configuration cycle. If MSI interrupts are selected, PCIEC has MSI reception module to detect and terminate inbound MSI requests. Rather than propagating MSI MWr TLPs onto the AXI bus through the master interface, the MSI packets are captured, terminated in the AXI bridge, and an interrupt is signaled locally through intreq_pci*m*_sub. This interrupt remains asserted when any MSI interrupt is pending. It is only de-asserted when there is no MSI interrupt pending. PCIEC also can pass the MSI MWr request TLPs to the AXI bus by deactivating MSI reception module.

MSI reception module features are:

- Provides support for up to eight EPs. Each supported EP has a set of interrupt enable(PRTLGC27+3**i*), mask (PRTLGC28+3**i*), and status registers (PRTLGC29+3**i*).
- A maximum of 32 interrupts are supported per EP.

#### (b) Setting for MSI Reception Module

If MSI reception module is activated, the following setting should be performed.

Flow-1) The host CPU configures the MSI capabilities of all endpoints through CFG requests.

— Program the MSI Data Register (MSICAP3F*i* in the MSI capability) as follows to allow the MSI reception module to decode the interrupt source. Table 104.41 shows MSI Data Register Programming format.

**Table 104.41 MSI Data Register Programming**

| Bit | 15:8 | 7:5 | 4:0 |
|---|---|---|---|
| Description | Not Used | EP Number:<br>Allows each EP to be identified within the system. For example, EP#5 is programmed with B'101. | Interrupt Vector Number:<br>Identifies the interrupt source within each EP.<br>Programmed to B'00000.<br>Set by MSI generation logic to identify each interrupt source in real time.<br>Supports up to 32 Vectors. |

—

Flow-2) Program the MSI Capability Lower 32 Bits Address Register (MSICAP1F*i*) of every EP with one common MSI address.

Flow-3) The host CPU configures the MSI reception module regiseters through APB bus.

— Program the MSI reception module registers PRTLGC25 and PRTLGC26 with the same value as in MSICAP1F*i*.

— Read the MSI capability of each EP to determine the number of vectors enabled in each EP and uses this information to program the MSI reception module's interrupt enable register (PRTLGC27+3**i*) which allows up to 32 MSI interrupt vectors to be enabled for a given EP. (*i* = 0, 1, 2, 3, 4, 5, 6, 7)

— The host CPU should program EPs Interrupt Enable register in the MSI reception module appropriately according to the Multiple Message Enable field in an EPs MSI capability structure. For example, when the Multiple Message Enable is 3B'100 for EP#0 (it mean 16 interrupt vectors enabled), the host CPU should program PRTLGC27 with H'0000_FFFF.

#### (c) Processing of Detected Interrupts

The MSI controller decodes the MSI MWr data payload to determine which EP sent the MSI and to which interrupt vector it corresponds. When the decoded interrupt vector is enabled and not masked, then PCIEC sets the corresponding bit in PRTLGC29+3**i* and asserts intreq_pci*m*_sub. This signal remains asserted until the host CPU clears the status bit by writing 1 to the status bit. Writing 0 has no effect. When the decoded interrupt vector is enabled but is masked, then PCIEC sets the corresponding bit in interrupt status register but

it does not assert intreq_pci*m*_sub. This masking is local to the MSI reception module and is not part of Per Vector Masking (PVM) in any of the EPs. When no interrupt have been enabled in any of PRTLGC29+3*i*, then all MSI detection is disabled and valid MSI MWr request TLPs are not terminated in the bridge but are passed to the AXI bus. (*m* = 0, *i* = 0 to 7)

### 104.3.19 Compliance Test

Test compliance according to the PCI Express standard.

### 104.3.20 Precision Time Measurement (PTM)

Precision Time Measurement (PTM) enables precise coordination of events across multiple components with independent local time clocks. Ordinarily, such precise coordination would be difficult given that individual time clocks have differing notions of the value and rate of change of time. To work around this limitation, PTM enables components to calculate the relationship between their local times and a shared PTM Master Time: an independent time domain associated with a PTM Root.

The controller supports the optional PTM feature (CX_PTM_ENABLE =1) as defined in Section 6.22, Precision Time Measurement (PTM) Mechanism of the PCI Express Base Specification, Revision 4.0, Version 1.0. This enables the coordination of timing information across multiple devices with independent local time-bases.

The PTM synchronization is initiated from a downstream requester (USP) to an upstream responder (DSP).

The controller handles the entire timestamp update process without software intervention. Each current (t#n) and previous timestamp (t#n-1) value as well as other context information (clock) can be read from the VSEC registers (PTMRQCAP*/PTMRSCAP*) by software.

**Transmit and Receive Latencies**

A key operational goal of PTM is to stamp the time at which a message is transmitted from the physical boundary of the PCIe device. The Tx latency is the time that it takes a message to travel from the boundary of the controller to the boundary of the PCIe device. It is added to the time at which the message left the controller. The Rx latency is the time that it takes a message to travel from the boundary of the PCIe device to the boundary of the controller. It is subtracted from the time at which the message reaches the controller. The Latency registers may be written to at any time with a more accurate value if required.

For example:

- t1, t3 = Local PTM Clock time when message leaves controller + PTMRQCAP19F*i*.PTM_REQ_TX_LATENCY

- t2, t4 = Local PTM Clock time when message received by controller - PTMRQCAP20F*i*.PTM_REQ_RX_LATENCY

**Note:** The Tx latency includes all delays from the moment timestamp is captured, to the time the message reaches the port pins. The Rx latency includes all delays from the time the message reaches the port pins, to the time the timestamp is captured. These delays depend on the characteristics of the PHY used, skew on the link, and deviation between the reference clocks of the two ports. There may be a dependency on link width and the current link speed also.

The average latencies for all speeds and link widths may be used for Tx and Rx latency calculation. For best accuracy the Tx and Rx latency values should be updated whenever the link parameters change.

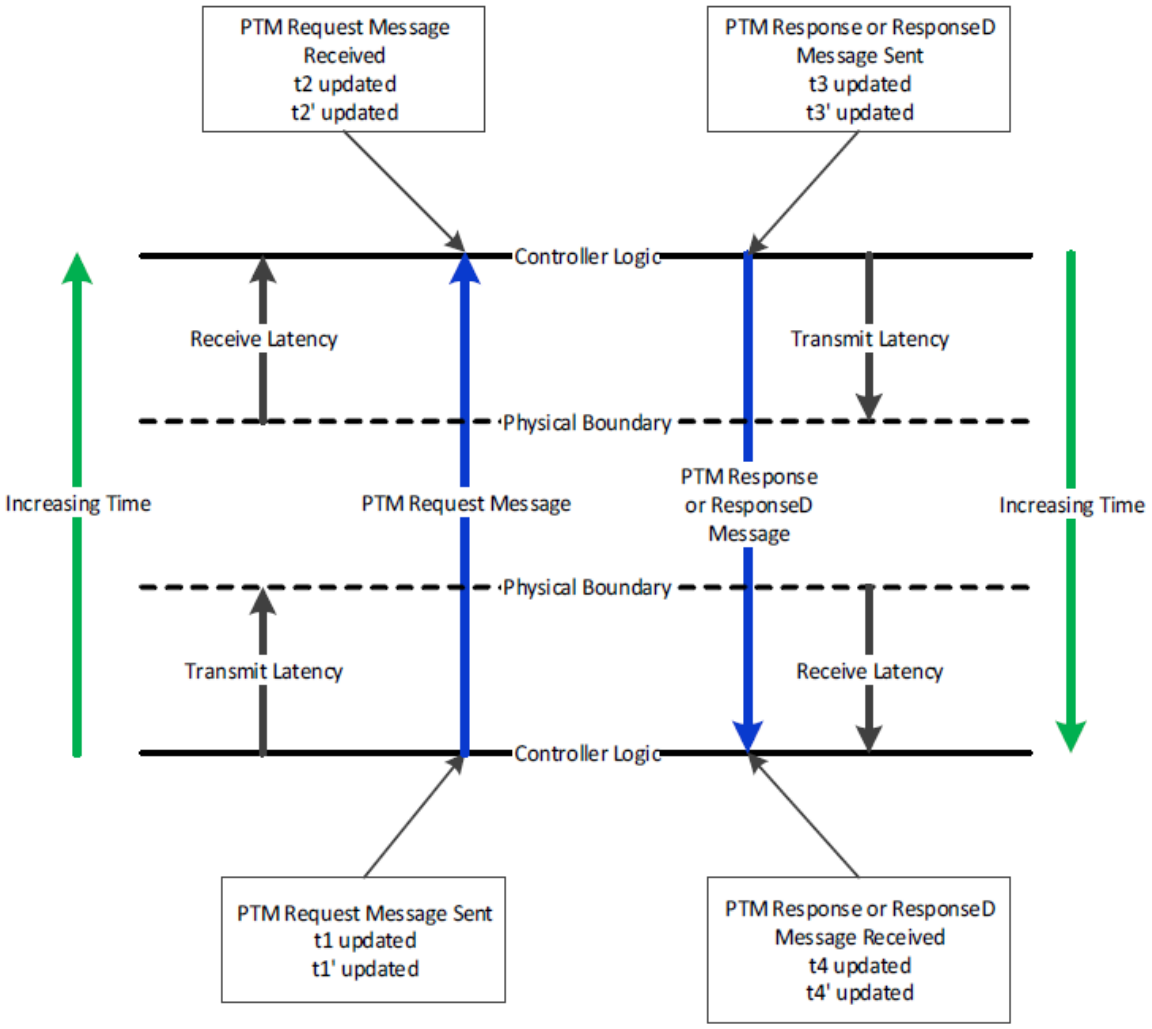



Figure 104.48 Tx and Rx Latencies

**PTM Timers**

The controller implements 1us and 100us timers as per the requirements of the PCI Express Base Specification,

Revision 4.0, Version 1.0. The controller also implements a 10 ms timer which is used for automatic update of PTM context by the Requester.

The controller implements 1us, 100us, and 10ms (used to invalidate local context) timers as per the requirements of the PCI Express Base Specification, Revision 4.0, Version 1.0. The 10 ms timer is also used for automatic update of PTM context by the Requester (USP).

### (1) Requester (USP) Features

■ Integrated hardware for scheduling of requests to start PTM dialogs

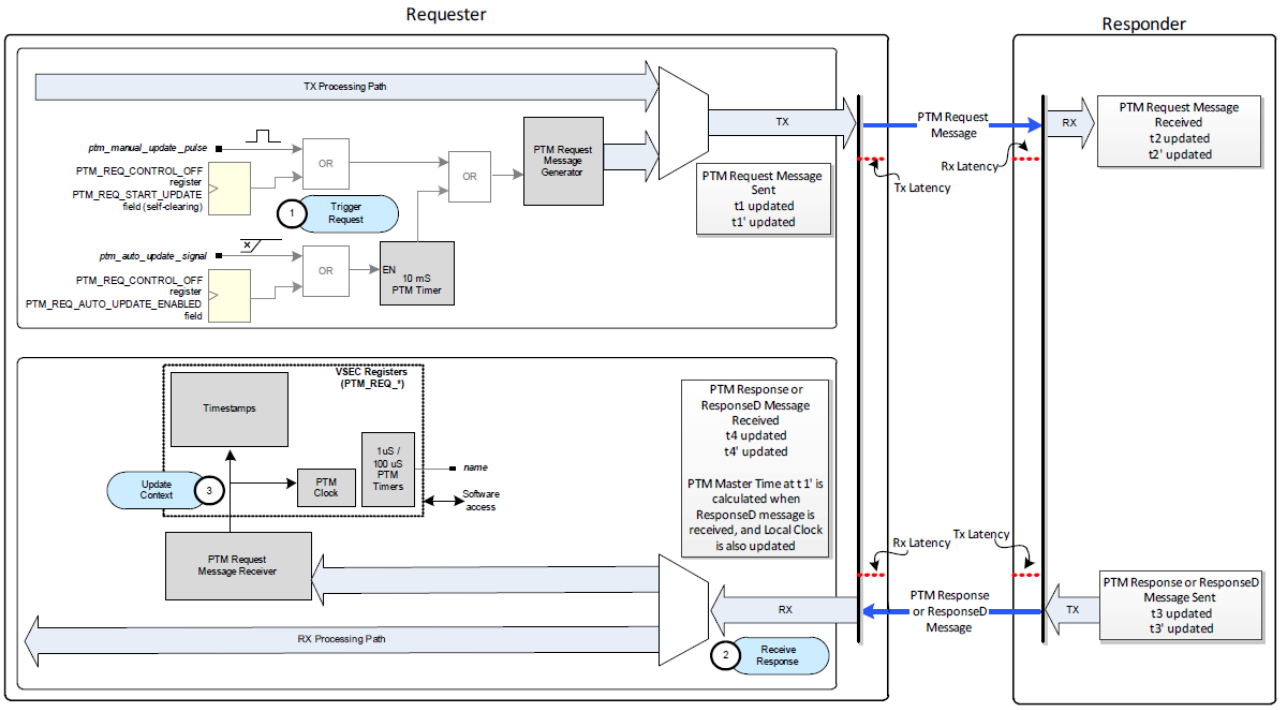



Figure 104.49 PTM Requester (USP)

■ Requester automatically updates PTM context (starting dialogs) when enabled using any or all of:

❑ Automatic trigger every 10ms (can be enabled through top-level pin or through a register)

❑ Manual trigger through top-level pin

❑ Manual trigger through a register write

■ Local PTM Clock

❑ Runs on core_clk, counts time in non-integer nanosecond, and its minimum granularity is defined by the core_clk frequency

❑ Auto-updated on PTM context refresh

❑ Can also be updated by software

❑ Can be read by software

❑ Available through top-level port

■ Context automatically invalidated when core_clk stops, or runs at the wrong frequency (for example, in L1 substates or when the link speed is changing), or when PTM is disabled by clearing the PTM_ENABLE bit in the PTMCAP2F0 register.

■ Context information stored in PTM Requester VSEC (PTMRQCAP*); one instance shared across all functions)

❑ Provides access to timestamps of most recent context refresh

❑ Provides notification when context is valid

❑ Programmable Tx and Rx latency values

### (2) Responder (DSP) Features

■ Integrated hardware for scheduling of PTM responses to dialog requests when context is valid

■ Timestamp generation when PTM Messages are detected

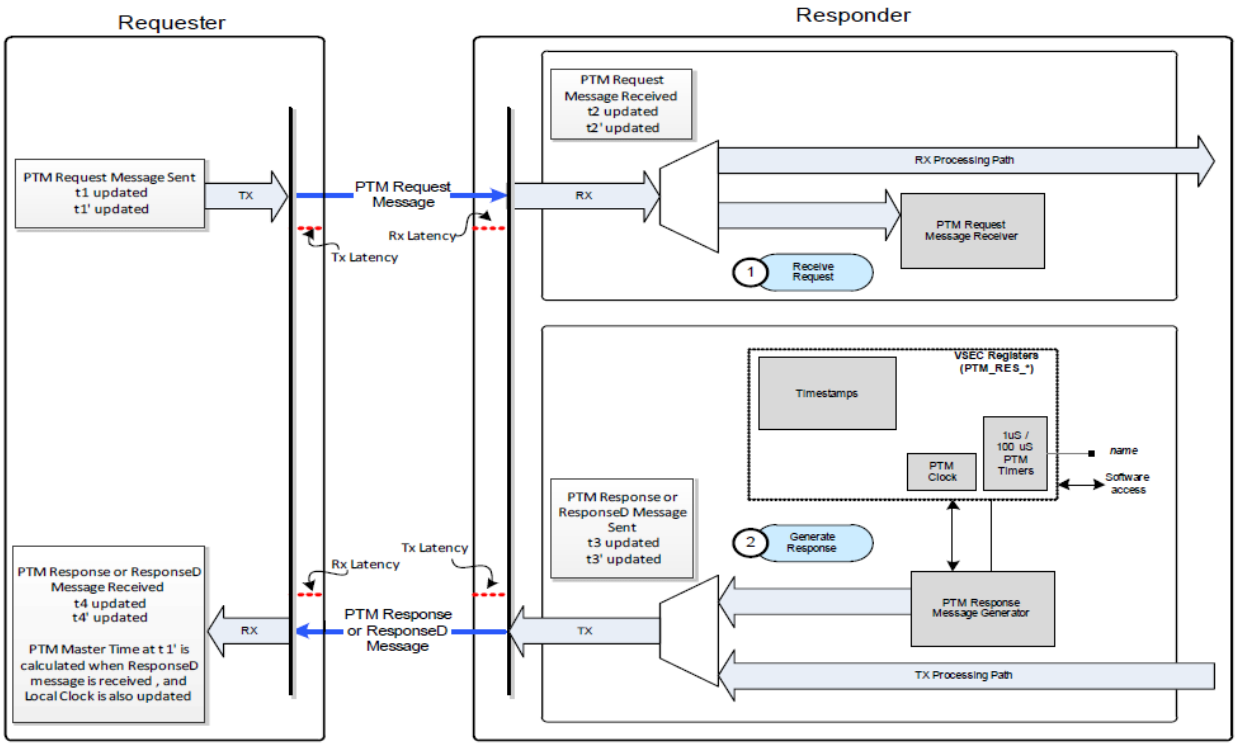



Figure 104.50 PTM Responder (DSP)

■ Local PTM Clock

❑ if CX_PTM_EXTERNAL_MASTER_TIME =1 an external master time can be used to set the clock value

Note: The external master time is latched when ptm_external_master_strobe signal is set to 1.

❑ Can also be updated by software

❑ Can be read by software

❑ Available through top-level port

■ Context automatically invalidated when core_clk stops, or runs at the wrong frequency (for example, in L1 substates or when the link speed is changing), or when PTM is disabled by clearing the PTM_ENABLE bit in the PTMCAP2F0 register. When the controller asserts ptm_responder_ rdy_to_validate signal, your application can re-validate the context by either,

❑ [CX_PTM_EXTERNAL_MASTER_TIME =1]: Updating the local PTM clock using the ptm_external_

master_time and ptm_external_master_strobe signals, which automatically asserts the ptm_context_valid signal.

❑ [DBI Access]: Updating the local PTM clock by writing to PTMRSCAP4F0 and PTMRSCAP5F0 registers, and then setting the PTM_RES_CCONTEXT_VALID field of PTMRSCAP2F0 to 1, which automatically asserts ptm_context_valid signal.

■ Root Support

❑ Ideally a PTM root behaves like a responder with permanently valid context. However,

- If the PTM root enters low power modes or changes link frequency, PTM context is invalidated (indicated by ptm_context_valid =0).

- Following PTM context invalidation, the application must re-validate PTM context, by using either the PTM master time interface signals or the PTM responder VSEC registers, as described in the previous list item.

❑ PTM messages must not contain TLP prefixes

■ Context information stored in PTM Responder VSEC (PTMRSCAP*)

❑ Provides access to timestamps of most recent context refresh

❑ Provides notification when context is valid

❑ Programmable Tx and Rx latency values.

### (3) Local PTM Clock Scaling

PTM clock scaling feature allows the local PTM clock to increment a programmable non-integer nanosecond amount in each controller clock period. This allows the PTM local time to remain coherent if there is a frequency deviation between PTM Requester and Responder. PTM clock scaling also allows the controller to use PTM when the clock period is a non-integer nanosecond multiple, for example when running at CCIX ESM data rates.

**PTM Requester**

To enable this feature set the PTM_REQ_SCALED_CLOCK_T_EN field of PTM_REQ_SCALED_CLOCK_T_OFF register to ‘1’ (this bit can be set only when ptm_context_valid is set). You can set a 24-bit value

consisting of an 8-bit integral part (nanoseconds) and 16-bit fractional part (1/216 nanoseconds) using

PTM_REQ_SCALED_CLOCK_T_INT and PTM_REQ_SCALED_CLOCK_T_FRAC fields of the

PTM_REQ_SCALED_CLOCK_T_OFF register, which represents the scaled local PTM clock period.

The clock scaling procedure is as follows:

■ Following the second and subsequent assertions of the ptm_clock_updated output signal, while

ptm_context_valid remains asserted, your application may read the PTM Requester Clock Correction

register (PTM_REQ_CLOCK_CORR_LSB_OFF).

■ If the correction is below a threshold expected by your the application, your application may calculate

a scaled local PTM clock period value, based on the current default clock period value, the

programmed auto-update interval, and the contents of the PTM Clock Correction register (PTM_REQ_-

CLOCK_CORR_LSB_OFF).

■ Your application may write the scaled PTM clock period to PTM Requester's Scaled PTM Clock Period

register (PTM_REQ_SCALED_CLOCK_T_OFF), which represents the scaled local PTM clock period. This

should happen as quickly as possible after the ptm_clock_updated pulse to avoid error accumulation.

**PTM Responder**

For the PTM Responder, your application fully controls PTM local clock updates. If the external master

clock runs from the same source clock as the PCIe controller, clock scaling is not needed. However, for applications where the external master and PCIe controller run on unrelated clock sources, it may be necessary to scale the local PTM clock period.

To enable this feature set the PTM_RES_SCALED_CLOCK_T_EN field of PTM_RES_SCALED_CLOCK_T_OFF register to '1' (this bit can only be set when ptm_context_valid is set). You can set a 24-bit value consisting of an 8-bit integral part (nanoseconds) and 16-bit fractional part ($1/2^{16}$ nanoseconds) using PTM_RES_SCALED_CLOCK_T_INT and PTM_RES_SCALED_CLOCK_T_FRAC fields of the PTM_RES_SCALED_CLOCK_T_OFF register, which represents the scaled local PTM clock period. The clock scaling procedure is as follows:

- Your application initializes the PTM Responder with a known time from an external master.
- The PTM Responder counts local PTM time using the Nominal PTM Clock Period, which your application can read from the PTM_RES_NOM_CLOCK_T_OFF register.
- Your application waits for a known time and reads the local PTM clock value again.
- Your application calculates the scaled PTM clock period using the deviation between the external master clock and the local PTM clock, and writes the scaled PTM clock period to PTM Responder's Scaled PTM Clock Period register (PTM_RES_SCALED_CLOCK_T_OFF), which represents the scaled local PTM clock period.

If the PTM context becomes invalid your application must reiterate the steps listed above.

### 104.3.21 Master interfaces of AXI

Based on the BDF information, the OSID is generated using the CNVIDn register as a conversion table. When comparing the BDF information, bits with 1 set in the CNVIDMSKn register are compared as don't cares. If none of the BDF information matches, assign the value of the CNVOSIDCTRL register as the OSID.

### 104.3.22 BAR Settings

The BAR settings for this product are shown below:

- BAR0:64-bit Resizable Memory BAR(default size:256MByte), Target_1
- BAR1:- (BAR1 cannot be used because BAR0 is set to 64bit.)
- BAR2:64-bit Resizable Memory BAR(default size:256MByte), Target_1
- BAR3:- (BAR3 cannot be used because BAR2 is set to 64bit.)
- BAR4:32-bit Fixed Mask I/O BAR 256Byte, Target_1
- BAR5: 32-bit Resizable Memory BAR(default size:1MByte), Target_0

Set the size of Resizable BAR in the following registers.
RESBAR_CTRL_REG_0_REG_F*i*.RESBAR_CTRL_REG_BAR_SIZE
RESBAR_CTRL_REG_2_REG_F*i*.RESBAR_CTRL_REG_BAR_SIZE
RESBAR_CTRL_REG_5_REG_F*i*.RESBAR_CTRL_REG_BAR_SIZE

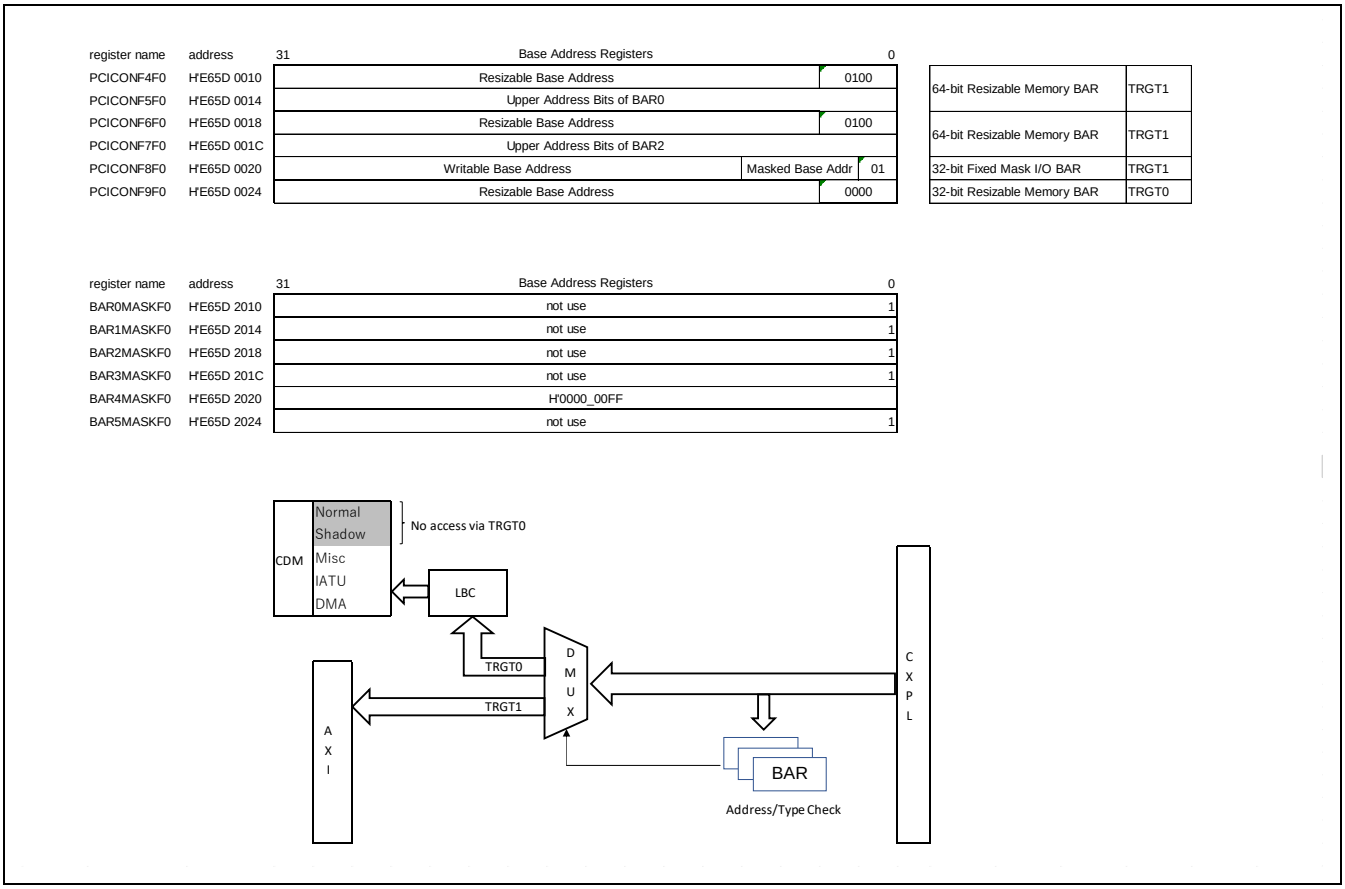



**Figure 104.51 Example Base Address Register Configuration**

### 104.3.23 FLR Signal Interface Timing

The PCI Express Base Specification requires the entire reset process to be completed within 100ms. The controller requires (4* 256 * core_clk) to reset the function. Figure 104.52 shows the timing flow for FLR of physical function x (PFx).

1. The host software writes to the Initiate FLR bit of PFx using a CfgWr request. It cannot use a local DBI write. The controller asserts the PCIERSTSTS.cfg_flr_pf_active[x] in response.
2. Your application terminates TLP transmission from PFxs application logic. During this time, your receive application logic should not halt the receive interface for TLPs directed to PFx.
3. When your application has stopped/reset the PFx application logic, set the PCIERSTCTRL3.app_flr_pf_done[x].
4. When the controller detects 1 on PCIERSTCTRL3.app_flr_pf_done[x], it initiates FLR of PFx.
5. When the reset is complete, the controller de-asserts PCIERSTSTS.cfg_flr_pf_active[x].

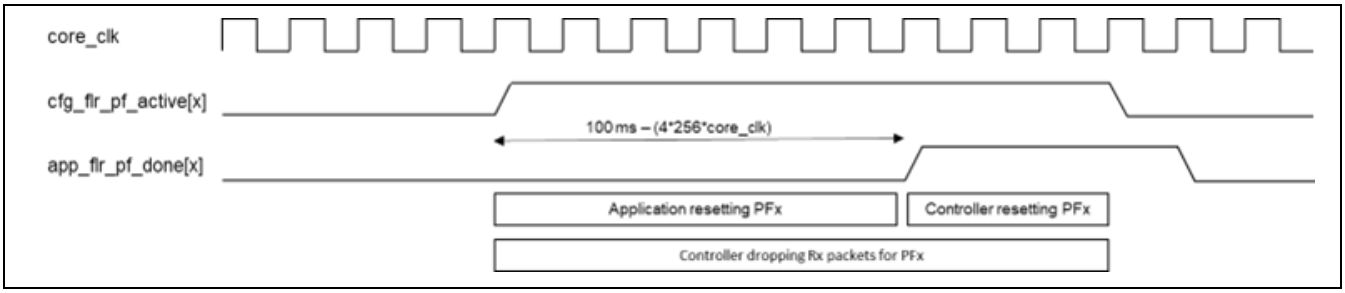



**Figure 104.52 FLR Timing Flow**

### 104.3.24 MSI-X by INTC

The PCIe controller recognizes the MSI/MSI-X interrupt from the AWADDR(AXI write address) of the PCIe/AXI 16 or 32bit Single Write transaction from each VF(Virtual Function) of EP(End Point).
In the case of the MSI/MSI-X interrupt, the PCIe controller overwrite the BDF(RequesterID) in WDATA[47:32] of AXI bus and notifies the INTC in AXI bus.
The INTC allocates the interrupt to the appropriate virtual OS from the BDF which received.

Address judgment method:

AXIINTCCONT.intc_enable & ((AWADDR[31:12],AWADDR[11:3]&AXIINTCCONT.intc_mask) == AXIINTCADDR.intc_address[31:3])

0: Normal Access
1: The MSI/MSI-X interrupt Access

## 104.4 Usage Notes

### 104.4.1 Write/Read Attributes of Configuration Registers

Although the registers in the configuration space have the register attributes defined by the PCI Express standard, these registers have different attributes when accessed by the internal bus. This section shows the attributes required for control.

**Table 104.42 Write/Read Attributes**

| Attributes of PCI Express Registers | Description | Attributes of PCIEC Registers | Example |
|---|---|---|---|
| HWInit | Hardware initialized bit. Initialized by hardware or software. | R for fields initialized by hardware. | EXPCAP0F*i*[24].SLTIMP (*i* = 0, 1) |
| | | RW for fields initialized by software. | EXPCAP3F*i*[31:24].Port Number (*i* = 0, 1) |
| RO | Read-only bit. Cannot be modified by software. | R for fields initialized by hardware. | PCICONF0F*i*[31:16].Device ID (*i* = 0, 1) |
| | | RW for fields initialized by software. | EXPCAP0F*i*[23:20].Device Port Type (*i* = 0, 1) |
| RW | Read/Write bit. Can be modified by software. | RW | PCICONF1F*i*[2].BME (*i* = 0, 1) |
| RW1C | Read-only status bit. Writing 1 clears the bit. Writing 0 has no effect. | RW1C | PCICONF1F*i*[24].MDPE (*i* = 0, 1) |
| ROS | Sticky-read-only bit. Cannot be modified by software.<br>Not initialized by hot reset. | Applies to no PCIEC registers. | — |
| RWS | Sticky-read write bit. Can be modified by software.<br>Not initialized by hot reset. | RWS | EXPCAP2F*i*[10].AUXPPME (*i* = 0, 1) |
| RW1CS | Sticky-read-only status bit. Writing 1 clears the bit. Writing 0 has no effect.<br>Not initialized by hot reset. | RW1CS | PMCAP1F*i*[15].PMEST (*i* = 0, 1) |
| RsvdP | Reserved bit. Read-only.<br>May be used as a RW bit in the future.<br>Reading the bit returns 0. | R | EXPCAP4F*i*[13:12] (*i* = 0, 1) |
| RsvdZ | Reserved bit. Read-only.<br>May be used as a RW1C bit in the future.<br>Reading the bit returns 0. | R | EXPCAP2F*i*[31:22] (*i* = 0, 1) |

The following shows which PCI Configuration Capability is supported by PCIEC.

| | | Offset | | Capability | Spec Requirements | Root Port Type01 | Endpoint Type00 |
|---|---|---|---|---|---|---|---|
| PCI Configuration Space | PCI3.0 Compatible Configuration Space | H'000<br>H'03F | | PCI Compatible Configuration Registers | Required | √ | √ |
| | | H'040 | | PCI Power Management Capability Structure | Required | √ | √ |
| | | | | MSI Capability Structure | Required if an interrupt resource is required | x | √ |
| | | | | PCI Express Capability Structure | Required | √ | √ |
| | | H'0FF | | | | | |
| | PCI Express Extended Configuration Space | H'0100 | | Virtual Channel Capability | Optional | x | x |
| | | | | Device Serial Number Capability | Optional | √ | √ |
| | | | | Advanced Error Reporting Capability | Optional | √ | √ |
| | | | | PCI Express Root Complex Link Declaration Capability | Optional | x | x |
| | | | | PCI Express Root Complex Internal Link Control Capability | Optional | x | x |
| | | | | Power Budgeting Capability | Optional | x | x |
| | | | | PCI Express Root Complex Event Collector Endpoint Association Capability | Optional | x | x |
| | | | | Multi-Function Virtual Channel Capability | Optional | x | x |
| | | | | Vendor-Specific Capability | Optional | √ | √ |
| | | | | RCRB Header Capability | Optional | x | x |
| | | | | Correlation Access Capability | Optional | x | x |
| | | | | Access Control Service | Optional | x | x |
| | | | | Alternative Routing ID Interpretation | Optional | x | x |
| | | | | Access Control Services | Optional | x | x |
| | | | | Single Root I/O Virtualization | Optional | x | x |
| | | | | Multi-Root I/O Virtualization | Optional | x | x |
| | | | | Multi Cast | Optional | x | x |
| | | | | Page Request Interface Page Request Extended Capability Structure | Optional | x | x |
| | | | | Resizable BAR | Optional | x | √ |
| | | | | Dynamic Power Allocation Capability | Optional | x | x |
| | | | | TPH Requester | Optional | x | x |
| | | | | Latency Tolerance Reporting Capability | Optional | √ | x |
| | | | | Secondary PCI Express Extended Capability | Required | √ | √ |
| | | | | Protocol Multiplexing | Optional | x | x |
| | | | | PASID Extended Capability | Optional | x | x |
| | | | | LNR Extended Capability | Optional | x | x |
| | | | | Downstream Port Containment | Optional | x | x |
| | | | | L1 PM Substates Extended Capability | Optional | √ | √ |
| | | | | Precision Time Management Capability | Optional | √ | √ |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | Function Readiness Status Queuing Extended Capablity | Optional | x | x |
| | | | | Readiness Time Reporting | Optional | x | x |
| | | | | Designated vendor-specific | Optional | x | x |
| | | | | VF Resizable BAR Capability | Optional | x | x |
| | | | | Data Link Feature Extended Capability | Required | √ | √ |
| | | | | Physical Layer 16.0 GT/s Extended Capability | Required | √ | √ |
| | | | | Lane Margining at Receiver Extended Capability | Required | √ | √ |
| | | | | Hierarchy ID | Optional | x | x |
| | | H'0FFF | | Native PCIe Enclosure Management Extended Capability | Optional | x | x |

### 104.4.2 Notes on Accessing Configuration Registers

- In Endpoint mode, writing to the configuration registers of this IP is prohibited except at initialization.
- ”PCI RW” in the column header of the table in the description section of each register indicates the attribute of configuration access from the PCI Express bus in Endpoint mode; it is invalid in Root Port mode. For the registers with the same attribute shared in Endpoint mode and Root Port, “—“ is indicated.
- Since the Endpoint configuration space may be accessed by the peer Root Port, it should be set appropriately. The registers implemented according to the Root Port configuration space are not accessed by Endpoint. Therefore, in Root Port mode, the user should appropriately set the registers. For details on register setting, refer to description of each register.

### 104.4.3 Power Management States

Do not stop inputting the reference clock signal in the power management states (L0s/L1/L2). Transition to the power management states (L0s/L1/L2) is made from L0.

The reference clock signal is stopped by PCIE*m*_CLKREQ_N (m = 0)in the power management states (L1 PM Substates). Transition to the power management states (L1 PM Substates) is made from L0.

### 104.4.4 Root Port Signaled Target Abort

When a function other than Function 0 in a Multi-function Device sends a request to the Root Port and the Root Port replies with Completion (CPL) whose status is Completer Abort (CA), the Signaled Target Abort in the Secondary Status Register is mistakenly generated with respect to the Requester ID rather than the Completer ID. Since the Root Port is a single function device where it only contains Function 0, a problem will occur if the Requester ID presents a function other than 0.

To make sure the problem does not occur, the Signaled Target Abort should be asserted with respect to the Function 0 and the Completer ID should be checked rather than the Requester ID.

### 104.4.5 Access DMA Configuration Registers

Feature Usage Note:

Ensure driver software does not generate TLPs with TPH/PH set when accessing DMA configuration registers.

### 104.4.6 Master Data Parity Error and Signaled System Error

・The controller does not correctly set the Master Data Parity Error field when the application reports a Poisoned TLP completion received error through PCIETXCNTRL1. app_err_bus[7].
・The controller sets the Signaled System Error register field to '1' for all PFs. It is only susceptible to the defect when a non-function-specific fatal or non-fatal error is detected.

Host Software Overhead

Instead of using the legacy error reporting method (The Master Data Parity Error), use the Advanced Error Reporting (AER) feature.

### 104.4.7 PM_PME message in L1 sub-state

The PCIe specification defines that the PM_PME messages must be retransmitted every 100ms +50%/-5%.
The PCIe controller does not meet this requirement when the link is in the L1 sub-state.
Note, Assert PCIEPWRMNGCTRL.app_xfer_pending when a PME message is triggered to exit L1 substate.
And de-assert PCIEPWRMNGCTRL.app_xfer_pending only after PCIEPWRMNGSTS1.pm_status is clear.

### 104.4.8 Usage notes for MRRS (Max Read Request Size)

Please set "Max Read Request Size" to 128 bytes or 256 bytes.
If "Max Read Request Size" is set to anything other than the above, the transferred data will not match the expected value.
[Target register]
PCI Express Capability Register 2 function i (EXPCAP2Fi) i = 0, 1 Register
bit14 to 12: Max Read Request Size

### 104.4.9 Usage notes for DMA transfer address

In DMA transfer, if the address is not 32byte aligned, it will not match the expected value.
To ensure correct transfer data, please set either A or B below:

A: The data size of a DMA transfer must be 480 bytes or less.
Non-Linked List Mode:
[Target register]
DMA Capabilities Registers [38+i*16] (DMACAP[38+i*16]) (i = 0 to 7) DMA_TRANSFER_SIZE (for write)
DMA Capabilities Registers [46+i*16] (DMACAP[46+i*16]) (i = 0 to 7) DMA_TRANSFER_SIZE (for read)

Linked List Mode:
Set "Transfer size" to 480 bytes or less, in the Linked List Descriptor in Local Memory.
For more details see Section 104.3.9 DMA Transfer (4) Linked List Mode.

B: Both the source address and destination address must be 32 bytes aligned.
Non-Linked List Mode:
[Target register]

DMA Capabilities Registers [39+i*16] (DMACAP[39+i*16]) (i = 0 to 7) SRC_ADDR_REG_LOW (for write)
DMA Capabilities Registers [41+i*16] (DMACAP[41+i*16]) (i = 0 to 7) DST_ADDR_REG_LOW (for write)
DMA Capabilities Registers [47+i*16] (DMACAP[47+i*16]) (i = 0 to 7) SRC_ADDR_REG_LOW (for read)
DMA Capabilities Registers [49+i*16] (DMACAP[49+i*16]) (i = 0 to 7) DST_ADDR_REG_LOW (for read)

Linked List Mode:
Set "SAR Low" and "DAR Low" to 32 bytes aligned address, in the Linked List Descriptor in Local Memory.
For more details see Section 104.3.9 DMA Transfer (4) Linked List Mode.

### 104.4.10 PCIE_CAP_LINK_TRAINING Register note

When LTSSM is in L1 state and software enables the RTRNLK field, the controller sets the EXPCAP4Fi register field after a delay.
System Usage Note:
Firmware to monitor both 'LTSSM' states on smlh_ltssm_state field of PCIEPWRMNGSTS0 and RTRNLK field of EXPCAP4Fi register bit and interrupt software when RTRNLK field of EXPCAP4Fi register bit is asserted before LTSSM enters recovery state and till it exits recovery state.

### 104.4.11 The L1 Exit Timer note

The L1 exit timer in L1.Idle expires in less than 100ms when GEN3_ZRXDC_NONCOMPL register field is set to one and L1 Substate/CPM is disabled.
System Usage Note:
Program the ECLKPM field in the EXPCAP4Fi register to '1'.

### 104.4.12 FLR Leads to Missing AXI Subordinate Responses in Multi-Function Devices When Operating in 5-Bit Tag Mode

System Usage Note:
When completing an FLR, stop AXI subordinate traffic to all functions and not only the one(s) being reset and wait for all outstanding requests to complete; after the FLR is complete, set the extended tag enable bit to 0 on the function(s) that was (were) reset before resuming AXI subordinate traffic and re-enabling HDMA channels (if present). If the HDMA is enabled, wait for any pending transaction to complete, and then disable all channels.

### 104.4.13 Incorrect Tag Generation in the Controller After Updating Extended Tag Requester Enable

System Usage Note:
After programming the ETAGE field of the EXPCAP2Fi register, issue AXI slave tags Non-Posted requests making sure only one request is outstanding at any given time; this flushes out any duplicate tags. The number of available tags after this procedure might be less than 32.

### 104.4.14 The Controller Drops TLP with ECRC Error when ECRC Check is Disabled

Feature Usage Note:
Program the ECRC check enable bit (ECRC_CHECK_EN) of the Advanced Error Capabilities and Control Register (AERCAP6Fi) to '0' before link-up.

## 104.5 Safety Mechanisms

PCIEC has no logics for FuSa into itself.

PCIEC applies End-to-End and SRAM ECC Protection as chip FuSa policy.

# 105. Serial Communication Interface with FIFO(SCIF)

## 105.1 Overview

This LSI has serial communication interfaces with FIFO buffers (SCIF) that handles asynchronous communication and clock synchronous serial communication. The SCIF has two 16-stage FIFO buffers separately for transmission and reception, which enables fast, efficient, and uninterrupted communication. The following table lists the functions of each interface.

The presence of the SCKn, RTSm#, and CTSm# pins depends on the product or channel. The synchronous mode and mode control functions are supported in the interface channel which has the pins related to these functions.

n = 0, 1, 3, 4
m = 0, 1, 3, 4

**Table 105.1 Channel Information**

| Name | Function | Pin | Base Address | Remarks |
|---|---|---|---|---|
| SCIF-0 | Asynchronous mode (modem control is enabled)<br>Clock synchronous mode | RX0, TX0, SCK0, RTS0#, and CTS0# | H'E6E6 0000 | Transmission/reception clock can be supplied externally from SCK0 pin. |
| | Asynchronous mode<br>Clock synchronous mode | RX0, TX0 and SCK0 | | |
| SCIF-1 | Asynchronous mode (modem control is enabled)<br>Clock synchronous mode | RX1, TX1, SCK1, RTS1#, and CTS1# | H'E6E6 8000 | Transmission/reception clock can be supplied externally from SCK1 pin. |
| | Asynchronous mode<br>Clock synchronous mode | RX1, TX1 and SCK1 | | |
| SCIF-3 | Asynchronous mode (modem control is enabled)<br>Clock synchronous mode | RX3, TX3, SCK3, RTS3#, and CTS3# | H'E6C5 0000 | Transmission/reception clock can be supplied externally from SCK3 pin. |
| | Asynchronous mode<br>Clock synchronous mode | RX3, TX3 and SCK3 | | |
| SCIF-4 | Asynchronous mode (modem control is enabled)<br>Clock synchronous mode | RX4, TX4, SCK4, RTS4#, and CTS4# | H'E6C4 0000 | Transmission/reception clock can be supplied externally from SCK4 pin. |
| | Asynchronous mode<br>Clock synchronous mode | RX4, TX4 and SCK4 | | |

### 105.1.1 Features

The SCIF has the following features.

- Asynchronous serial communication mode

  The SCIF performs serial data communication based on a character-by-character asynchronous system. This feature enables serial data communication with standard asynchronous communication chips that support Universal Asynchronous Receiver/Transmitter (UART) or Asynchronous Communication Interface Adapter (ACIA). There is a choice of eight serial data transfer formats.

  — Data length: 7 or 8 bits
  — Stop bit length: 1 or 2 bits
  — Parity: Even/odd/none
  — Receive error detection: Parity, framing, and overrun errors
  — Break detection:

    A break is detected when a framing error lasts for more than 1 frame length at Space 0 (low level).

    When a framing error occurs, a break can also be detected by reading the RX pin level directly from the serial port register (SCSPTR).
- Clock synchronous serial communication mode

  The SCIF performs serial data communication synchronized with a clock. This feature enables serial data communication with other LSIs that support synchronous communication. There is a single serial data communication format for clock synchronous serial communication.

  — Data length: 8 bits
  — Receive error detection: Overrun errors
- Full-duplex communication capability

  The SCIF has an independent transmitter and receiver that enable simultaneous transmission and reception. The transmitter and receiver both have a 16-stage FIFO buffer structure, enabling continuous serial data transmission and reception.
- On-chip baud rate generator, enabling any bit rate to be selected

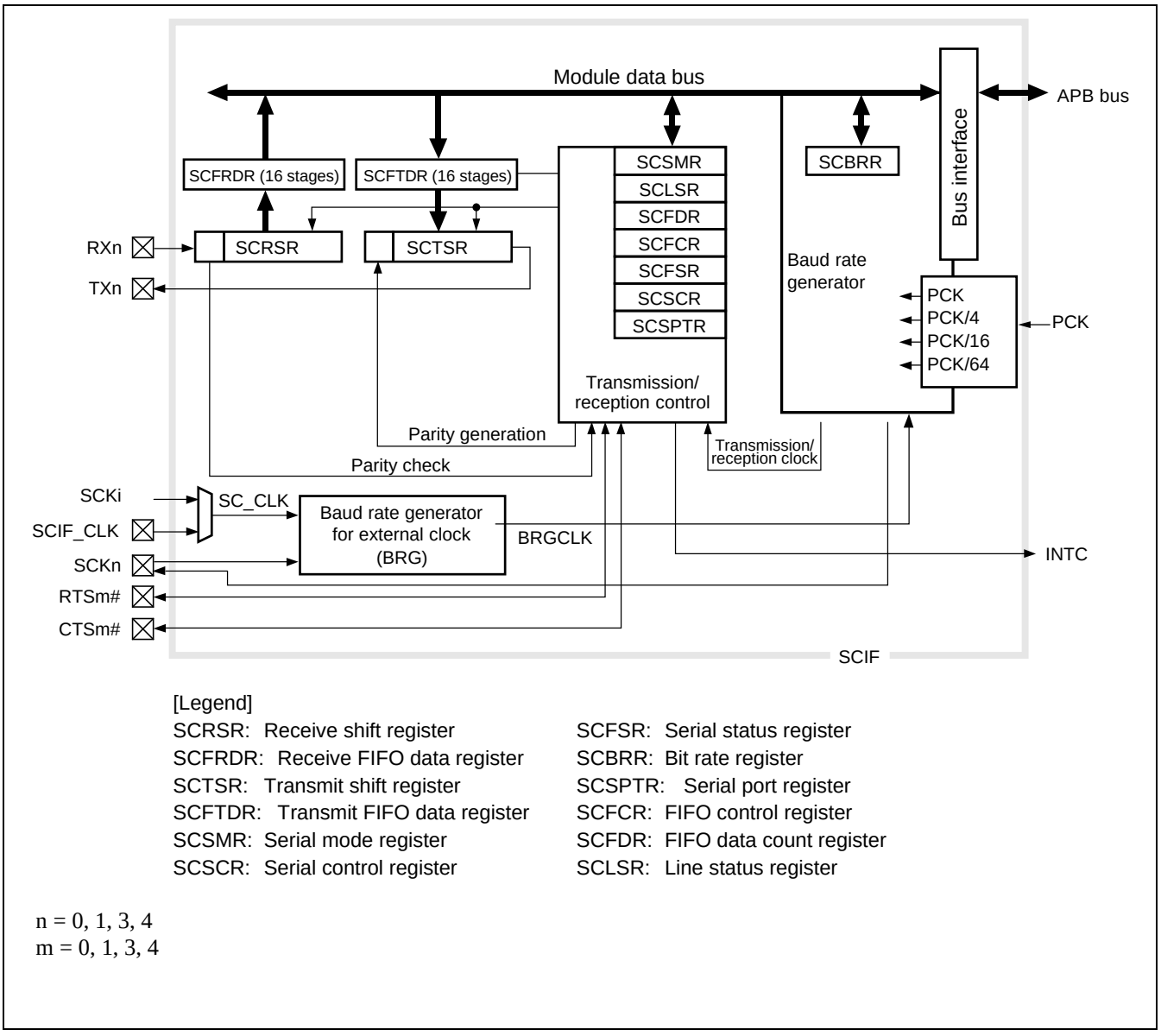
  The SCIF enables choice of a clock source for transmission/reception: a clock from the on-chip baud rate generator based on the internal clock or an external clock.
- Eight interrupt sources

  The SCIF has eight types of interrupt sources – receive-data-ready, receive-FIFO-data-full, break, transmit-FIFO-data-empty, transmit-end, receive-error, overrun-error and time-out and enables any of them to be requested independently.
- DMA data transfer

  When the transmit FIFO register is empty or the receive FIFO register has received data, issuing a DMA transfer request activates the DMA controller (DMAC) to execute a data transfer.
- The amount of data in the transmit/receive FIFO registers and the number of receive errors in receive data in the receive FIFO register are available.
- In asynchronous mode, a receive data ready (DR) or a timeout error (TO) can be detected during reception.

### 105.1.2 Block Diagram

Figure 105.1 shows the SCIF block diagram.




[Legend]

| | |
|---|---|
| SCRSR: Receive shift register | SCFSR: Serial status register |
| SCFRDR: Receive FIFO data register | SCBRR: Bit rate register |
| SCTSR: Transmit shift register | SCSPTR: Serial port register |
| SCFTDR: Transmit FIFO data register | SCFCR: FIFO control register |
| SCSMR: Serial mode register | SCFDR: FIFO data count register |
| SCSCR: Serial control register | SCLSR: Line status register |

n = 0, 1, 3, 4
m = 0, 1, 3, 4

**Figure 105.1 SCIF Block Diagram**

Clock for “Bus interface” = SASYNCPERD4φ
PCK = SASYNCPERD4φ
SCKi = SASYNCPERD1φ

### 105.1.3 External Pins

Table 105.2 shows the SCIF pin configuration. Pin functions can differ with the interface number. Other functions are also multiplexed on the same pins, so the multiplexed pin settings may restrict usage of the pins.

**Table 105.2 Pin Configuration**

| Name | Abbreviation | I/O | Function |
|---|---|---|---|
| Serial clock pin | SCK | I/O | Synchronous clock I/O |
| Receive data pin | RX | Input | Receive data input |
| Transmit data pin | TX | Output | Transmit data output |
| Modem control pin | CTS# | I/O | Transmission enabled |
| Modem control pin | RTS# | I/O | Transmission request |
| Baud rate generation clock pin* | SCIF_CLK | Input | Clock for input to the baud rate generator for the external clock |

These pins are made to function as serial pins by setting up SCIF operation using bit C/A# in SCSMR, bits TE, RE, CKE[1:0] in SCSCR, and bit MCE in SCFCR. SCSPTR of the SCIF can be used to handle the transmission and detection of break states.

Note: ch0/1/3 can use SCIF_CLK pin, ch4 can use SCIF_CLK2 pin.

### 105.1.4 Register Configuration

(include Safety Mechanism related registers)

Table 105.3 shows the registers in the SCIF.

**Table 105.3 Register Configuration**

| | Abbreviation | R/W | Offset from Base Address | Initial Value | Size |
|---|---|---|---|---|---|
| Serial mode register | SCSMR | R/W | H'00 | H'0000 | 16 |
| Bit rate register | SCBRR | R/W | H'04 | H'FF | 8 |
| Serial control register | SCSCR | R/W | H'08 | H'0000 | 16 |
| Transmit FIFO data register | SCFTDR | —/WB | H'0C | Undefined | 8 |
| Serial status register | SCFSR | R/WC0*1 | H'10 | H'0060 | 16 |
| Receive FIFO data register | SCFRDR | R | H'14 | Undefined | 8 |
| FIFO control register | SCFCR | R/W | H'18 | H'0000 | 16 |
| FIFO data count register | SCFDR | R | H'1C | H'0000 | 16 |
| Serial port register | SCSPTR | R/W | H'20 | H'00XX*3 | 16 |
| Line status register | SCLSR | R/WC0*2 | H'24 | H'0000 | 16 |
| Frequency division register | DL | R/W | H'30 | H'0000 | 16 |
| Clock Select register | CKS | R/W | H'34 | H'0000 | 16 |

Notes: 1. Only 0 can be written to bits 7 to 4, 1, and 0 to clear the flags. Bits 15 to 8, 3, and 2 are read-only bits and cannot be modified.
2. Only 0 can be written to bits 2 and 0 to clear the flags. Bits 15 to 3, and 1 are read-only bits and cannot be modified.
3. The initial values of SCSPTR bits 6, 4, 2, and 0 are undefined.

The table below lists the registers used in asynchronous mode, asynchronous mode with modem control, and clock synchronous mode. When setting up the registers, set the bits related to unsupported modes in each interface to their initial values. Otherwise, the SCIF may malfunction. Do not write to any registers other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from registers other than listed below are undefined.

**Table 105.4 Register Settings in Each Mode**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCFRDR | — | — | — | — | — | — | — | — | * | * | * | * | * | * | * | * |
| SCFTDR | — | — | — | — | — | — | — | — | * | * | * | * | * | * | * | * |
| SCSMR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | C | A | A | A | A | 0 | * | * |
| SCSCR | 0 | 0 | 0 | 0 | * | 0 | 0 | 0 | * | * | * | * | * | A | * | * |
| SCFSR | A | A | A | A | A | A | A | A | A | * | * | A | A | A | * | A |
| SCBRR | — | — | — | — | — | — | — | — | * | * | * | * | * | * | * | * |
| SCFCR | 0 | 0 | 0 | 0 | 0 | M | M | M | * | * | * | * | M | * | * | M |
| SCFDR | 0 | 0 | 0 | * | * | * | * | * | 0 | 0 | 0 | * | * | * | * | * |
| SCSPTR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M | M | M | M | C | C | * | * |
| SCLSR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | * |
| DL | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| CKS | * | * | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

[Legend]
* : Used in any mode.
A : Used in asynchronous mode.
M : Used in asynchronous mode (modem control is enabled) (in addition to A).
C : Used in clock synchronous mode.
0 : Reserved (the write value should always be 0.)
— : Undefined

### 105.1.5 Connected Module

Table 105.5 shows the connected modules to the SCIF.

**Table 105.5 Connected modules**

| Module name | Connected module name | Function of connected module |
|---|---|---|
| SCIF | AP-System Core | Access the Register |
| | CPG | Output Clocks |
| | PFC | Select External pins |
| | Module Standby | Control to stop clocks |
| | Software Reset | Execute software reset |
| | INTC | Control to interrupt |
| | SYS-DMAC | Control Direct Memory Access |

## 105.2 Register Description

(include Safety Mechanism related register description)

### 105.2.1 Receive Shift Register (SCRSR)

SCRSR is a register that receives serial data.

The SCIF sets serial data input to the SCRSR from the RX pin in the order received, starting with the LSB (bit 0), and converts it to parallel data. When one byte of data has been received, it is transferred to the receive FIFO data register SCFRDR, automatically.

SCRSR cannot be read from and written to by the CPU.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | — | — | — | — | — | — | — | — |

### 105.2.2 Receive FIFO Data Register (SCFRDR)

SCFRDR is a 16-stage FIFO register that stores received serial data.

When the SCIF has received one byte of serial data, it transfers the received data from the receive shift register (SCRSR) to SCFRDR for storage, and reception is thus completed. SCRSR is then ready for reception, and is capable of receiving up to 16 consecutive bytes of data before SCFRDR is full.

SCFRDR is a read-only register and cannot be modified by the CPU. Note that the read value will be undefined while there is no receive data in SCFRDR. When SCFRDR is full of receive data, subsequent serial data is lost.

SCFRDR is read as an undefined value after a power-on reset.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R |

### 105.2.3 Transmit Shift Register (SCTSR)

SCTSR is a register that transmits serial data.

To perform serial data transmission, the SCIF first transfers transmit data from the transmit FIFO data register (SCFTDR) to SCTSR, then sends the data to the TX pin starting with the LSB (bit 0). When transmission of one byte is completed, the SCIF transfers the next transmit data from SCFTDR to SCTSR automatically, then starts transmission.

SCTSR cannot be read from and written to directly by the CPU.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | — | — | — | — | — | — | — | — |

### 105.2.4 Transmit FIFO Data Register (SCFTDR)

SCFTDR is an 8-bit FIFO register of 16 stages that stores data for serial transmission.

If SCTSR is empty after transmit data has been written to SCFTDR, the SCIF transfers the data from SCFTDR to SCTSR and starts serial transmission.

SCFTDR is a write-only register and cannot be read from by the CPU. Writing further data to SCFTDR is no longer possible when it is full (contains 16 bytes of transmit data). Attempts at writing data to the register in this situation are ignored.

SCFTDR is read as an undefined value on a power-on reset.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB |

### 105.2.5 Serial Mode Register (SCSMR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | C/A# | CHR | PE | O/E# | STOP | — | CKS[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W |

SCSMR is a 16-bit register that sets the SCIF's serial transfer format and selects the baud rate generator clock source.

SCSMR can always be read from and written to by the CPU.

SCSMR is initialized to H'0000 by a power-on reset.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | C/A# | 0 | R/W | Communication Mode<br>Selects asynchronous mode or clock synchronous mode for the SCIF operation.<br>0: Asynchronous mode<br>1: Clock synchronous mode<br>Note: Whether clock synchronous mode can be selected or not depends on the interface. |
| 6 | CHR | 0 | R/W | Character Length<br>Selects 7 or 8 bits for asynchronous mode data length. In clock synchronous mode, the data length is fixed at 8 bits regardless of the CHR bit setting.<br>0: 8 bits<br>1: 7 bits*<br>Note: * When the 7-bit data is selected, the MSB (bit 7) in the transmit FIFO data register (SCFTDR) is not transmitted. |
| 5 | PE | 0 | R/W | Parity Enable<br>Determines whether parity bit is added in transmission or not, and parity bit is checked in reception in asynchronous mode or not. In clock synchronous mode, the parity bit is not added and checked regardless of the PE bit setting.<br>When bit PE is set to 1, the parity (even or odd) specified by bit O/E# is added to transmit data. In reception, the parity bit is checked for the parity (even or odd) specified by bit O/E#.<br>0: Disables parity bit addition and check.<br>1: Enables parity bit addition and check. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | O/E# | 0 | R/W | Parity Mode<br>Selects either even or odd parity to use in parity addition and check.<br>In asynchronous mode, the O/E# bit setting is valid only when bit PE is set to 1, enabling parity bit addition and check. In clock synchronous mode or when parity addition or check is disabled in asynchronous mode, the O/E# bit setting is invalid.<br>0: Even parity*1<br>1: Odd parity*2<br>Notes: 1. When even parity is set, the parity bit is added in transmission so that the total number of 1-bits in the transmit character plus the parity bit is even. In reception, a check is performed to see if the total number of 1-bits in the receive character plus the parity bit is even.<br>2. When odd parity is set, the parity bit is added in transmission so that the total number of 1-bits in the transmit character plus the parity bit is odd. In reception, a check is performed to see if the total number of 1-bits in the receive character plus the parity bit is odd. |
| 3 | STOP | 0 | R/W | Stop Bit Length<br>Selects 1 bit or 2 bits as the stop bit length in asynchronous mode.<br>The stop bit setting is valid only in asynchronous mode. Since the stop bit is not added in clock synchronous mode, the STOP bit setting is invalid in this mode.<br>In reception, only the first stop bit is checked, regardless of the STOP bit setting. If the second stop bit is 1, it is treated as a stop bit. If it is 0, it is treated as the start bit of the next transmit character.<br>0: 1 stop bit*1<br>1: 2 stop bits*2<br>Notes: 1. In transmission, high level of 1 bit (stop bit) is added to the end of a transmit character before it is sent.<br>2. In transmission, high level of 2 bits (stop bits) are added to the end of a transmit character before it is sent. |
| 2 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 1, 0 | CKS[1:0] | 00 | R/W | Clock Select 1 and 0<br>These bits select the clock source for the on-chip baud rate generator.<br>The clock source can be selected from PCK, PCK /4, PCK /16, and PCK /64, according to the setting of bits CKS1 and CKS0.<br>00: PCK<br>01: PCK /4<br>10: PCK /16<br>11: PCK /64 |

### 105.2.6 Serial Control Register (SCSCR)

SCSCR is a register that enables or disables transmission/reception by the SCIF, enables or disables interrupt requests, and selects transmission/reception clock source for the SCIF.

SCSCR can always be read from and written to by the CPU.

SCSCR is initialized to H'0000 by a power-on reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | TEIE | — | — | — | TIE | RIE | TE | RE | REIE | TOIE | CKE[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | TEIE | 0 | R/W | Transmit End Interrupt Enable<br>When a transmit-end request is enabled by the TIE bit, the TEIE bit selects the source of the transmit end interrupt request from the following:<br>• Setting the TDFE flag in SCFSR<br>Setting the TEND flag in SCFSR<br>0: The transmit FIFO data empty (TDFE) interrupt request is used.<br>1: The transmit end (TEND) interrupt request is used. |
| 10 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | TIE | 0 | R/W | Transmit Interrupt Enable<br>Enables or disables a transmit-FIFO-data-empty interrupt (TDFE) request when the TEIE bit in SCSCR is pulled 0, if all of the following conditions are satisfied:<br>• Serial transmit data has been transferred from SCFTDR to SCTSR.<br>• The number of data bytes in SCFTDR is equal to or less than the transmit trigger count.<br>• The TDFE flag in SCFSR is set to 1.<br>Enables or disables a transmit-end interrupt (TEND) request when the TEIE bit of SCSCR is set to 1, if both of the following conditions are satisfied:<br>• Transmission was ended because there is no valid data in SCFTDR when the last bit of the transmit character in SCTSR was transmitted.<br>• The TEND flag of SCFSR is set to 1.<br>0: When the TEIE bit is 0, disables transmit-FIFO-data-empty interrupt (TDFE) request.<br>When the TEIE bit is 1, disables transmit-end interrupt (TEND) request.<br>1: When the TEIE bit is 0, enables transmit-FIFO-data-empty interrupt (TDFE) request.<br>When the TEIE bit is 1, enables transmit-end interrupt (TEND) request. |
| 6 | RIE | 0 | R/W | Receive Interrupt Enable<br>Enables or disables a receive-FIFO-data-full interrupt request when the RDF flag in SCFSR is set to 1, a receive-data-ready interrupt request when the DR flag in SCFSR is set to 1, a receive-error interrupt request when the ER flag in SCFSR is set to 1, a break interrupt request when the BRK flag in SCFSR is set to 1, and an overrun error interrupt request when the ORER flag in SCLSR is set to 1.<br>0: Disables receive-FIFO-data-full interrupt (RDF) requests, receive-data-ready interrupt (DR) requests, receive-error interrupt (ER) requests, break interrupt (BRK) requests, and overrun error interrupt (ORER) requests.<br>1: Enables receive-FIFO-data-full interrupt (RDF) requests, receive-data-ready interrupt (DR) requests, receive-error interrupt (ER) requests, break interrupt (BRK) requests, and overrun error interrupt (ORER) requests. |
| 5 | TE | 0 | R/W | Transmit Enable<br>Enables or disables the start of SCIF serial transmission. The SCIF starts serial transmission when transmit data is written to the SCFTDR while TE is 1. Before setting TE to 1, set SCSMR and SCFCR to specify the transmission format and reset the transmit FIFO.<br>0: Disables transmission.<br>1: Enables transmission. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | RE | 0 | R/W | Receive Enable<br>Enables or disables the start of SCIF serial reception. When RE is 1, the SCIF starts serial reception by detecting a start bit in asynchronous mode or by detecting a synchronization clock input in clock synchronous mode. Before setting RE to 1, set SCSMR and SCFCR to specify the reception format and reset the receive FIFO.<br>0: Disables reception.*<br>1: Enables reception.<br>Note: Even if RE is cleared to 0, the DR, ER, BRK, RDF, FER, PER, TO and ORER flags are not affected, and retain their states. |
| 3 | REIE | 0 | R/W | Receive Error Interrupt Enable<br>Enables or Disables generation of receive-error interrupt (ER) requests, break interrupt (BRK) requests and overrun-error interrupt (ORER) requests.<br>0: Disables receive-error interrupt (ER) requests, break interrupt (BRK) requests and overrun-error interrupt (ORER) requests.*<br>1: Enables receive-error interrupt (ER) requests, break interrupt (BRK) requests and overrun-error interrupt (ORER) requests.<br>Note: * When REIE is 1, ER, BRK or ORER interrupt requests will occur even if RIE is cleared to 0. This setting is used to notify the interrupt controller of ER, BRK, and ORER interrupt requests during DMAC transfer. |
| 2 | TOIE | 0 | R/W | Timeout Interrupt Enable<br>Enables or disables generation of timeout interrupt (TO) requests when the TO flag in SCLSR is set to 1.<br>0: Disables timeout interrupts (TO).<br>1: Enables timeout interrupts (TO). |
| 1, 0 | CKE[1:0] | 00 | R/W | Clock Enable 1 and 0<br>These bits select the SCIF clock source and enables or disables the clock output from the SCK pin.<br>Whether the SCK pin functions as a serial clock output pin or a serial clock input pin is determined by combination of the CKE [1:0] bits settings. To specify synchronization clock output in clock synchronous mode, set C/A# to 1 in SCSMR then set CKE [1:0].<br>See Table 105.6 for the bit settings. |

**Table 105.6 Clock Selection**

| CKE[1:0] | Mode | Clock Source | SCK Pin |
|---|---|---|---|
| 00 | Asynchronous mode | Internal clock (PCK, PCK /4, PCK/16, PCK/64) | The SCK pin is not used.<br>The SCK pin functions as an input pin (Input signals are ignored). (Initial value) |
| | Clock synchronous mode | | The SCK pin outputs the synchronization clock. (Initial value) |
| 01 | Asynchronous mode | | The SCK pin outputs the clock (with a frequency 16 times the bit rate). |
| | Clock synchronous mode | | The SCK pin outputs the synchronization clock. |
| 10 | Asynchronous mode | Baud rate generator output for external clock or SCK | When SC_CLK is selected:<br>The SCK pin is an input pin (Input signals are ignored).<br>Set the SC_CLK frequency so that the frequency of BRGCLK is 16 times the bit rate. |
| | | | When SCK is selected:<br>The SCK pin inputs the clock (with a frequency 16 times the bit rate). |
| | Clock synchronous mode | SCK* | The SCK pin inputs the synchronization clock. |
| 11 | Prohibited | — | — |

Note: * It is not allowed to set synchronous communication using SC_CLK for input.

### 105.2.7 Serial Status Register (SCFSR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PER[3:0] | | | | FER[3:0] | | | | ER | TEND | TDFE | BRK | FER | PER | RDF | DR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/WC0 | R/WC0 | R/WC0 | R/WC0 | R | R | R/WC0 | R/WC0 |

SCFSR is a 16-bit register. The lower 8 bits are a status flag that indicates the operating status of the SCIF, and the upper 8 bits indicate the number of reception errors of data in the receive FIFO register.

SCFSR can always be read from and written to by the CPU. However, the flags ER, TEND, TDFE, BRK, RDF, and DR cannot be written by 1. The FER and PER flags are read-only flags and cannot be modified.

SCFSR is initialized to H'0060 by a power-on reset.

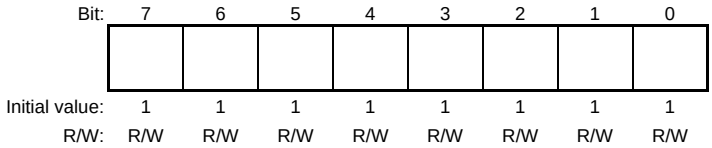
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 12 | PER[3:0] | All 0 | R | Parity Error Count<br>These bits indicate the number of parity errors of receive data stored in the receive FIFO data register (SCFRDR).<br>After the ER is set in SCFSR, the value represented by bits 15 to 12 indicates the number of parity errors. If a parity error occurs in all 16-byte receive data in SCFRDR, PER3 to PER0 are cleared to 0. |
| 11 to 8 | FER[3:0] | All 0 | R | Framing Error Count<br>These bits indicate the number of framing errors of receive data stored in the receive FIFO data register (SCFRDR).<br>After the ER bit is set in SCFSR, the value represented by bits 11 to 8 indicates the number of framing errors. If a framing error occurs in all 16-byte receive data in SCFRDR, FER3 to FER0 are cleared to 0. |
| 7 | ER | 0 | R/WC0 | Receive Error<br>Indicates that a framing error or a parity error has occurred in reception.*1 The ER flag is not affected by an error and retains its previous state when the RE bit is 0 in SCSCR.<br>If a receive error occurs, receive data will be transferred to SCFRDR and reception operation will be continued. Whether there is a receive error in data read from SCFRDR can be determined by the FER and PER bits in SCFSR.<br>0: Indicates that no framing or parity error has occurred in reception.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• 0 is written to ER.<br>1: Indicates that a framing error or a parity error has occurred in reception.<br>[Setting conditions]<br>• The SCIF checks whether the stop bit at the end of receive data is 1, but the stop bit is 0.*2<br>• The number of 1-bits in receive data plus the parity bit does not match the parity setting (even or odd) specified by the O/E# bit in SCSMR during reception.<br>Note: In the 2-stop-bit mode, only the first stop bit is checked that the value is 1; the second stop bit is not checked. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | TEND | 1 | R/WC0 | Indicates that transmission has been ended * because there was no valid data in SCFTDR when the last bit of the transmit character was transmitted.<br>0: Indicates that transmission is in progress.<br>[Clearing conditions]<br>• Transmit data is written to SCFTDR, and 0 is written to TEND.<br>• Data is written to SCFTDR by the DMAC.<br>1: Indicates that transmission has been ended.<br>[Setting conditions]<br>• A power-on reset is executed.<br>• The TE bit in SCSCR is 0.<br>• There is no transmit data in SCFTDR when the last bit of a 1-byte serial transmission character is transmitted.<br>Note: When transmit data is written to SCFTDR by the DMAC in clock synchronous mode, the TEND flag may not be cleared. Therefore, if the DMAC is used for transmission in clock synchronous mode, the TEND flag should be read using the following procedure.<br>1. Check that data transfer is completed in the DMAC.<br>2. Read the TEND flag.<br>3. If TEND = 1, clear the TEND flag to 0.<br>4. Read the TEND flag again.<br>5. Use the second read TEND flag. |
| 5 | TDFE | 1 | R/WC0 | Transmit FIFO Data Empty<br>Indicates that the SCIF has transferred data from SCFTDR to SCTSR, the number of data bytes in SCFTDR becomes equal to or less than the transmit trigger count specified by the TTRG1 and TTRG0 bits in SCFCR, and SCFTDR is ready to be written by new transmit data.<br>SCFTDR is a 16-byte FIFO register. The maximum number of bytes that can be written to when TDFE = 1 is “16 – [the transmit trigger count]”. If data exceeding this value is attempted to be written, the data will be ignored. The number of data bytes in SCFTDR is indicated by the upper bits of SCFDR.<br>If the number of data written in SCFTDR is equal to or less than the transmit trigger count, this bit will be set to 1 even if it is cleared to 0 after it is read as 1.<br>0: Indicates that the number of transmit data written to SCFTDR exceeds the transmit trigger count.<br>[Clearing conditions]<br>• Transmit data exceeding the specified transmit trigger count have been written to SCFTDR, and 0 is written to TDFE.<br>• Transmit data exceeding the specified transmit trigger count have been written to SCFTDR by the DMAC.<br>1: Indicates that the number of transmit data in SCFTDR is equal to or less than the transmit trigger count.<br>[Setting conditions]<br>• A power-on reset is executed.<br>• The number of transmit data in SCFTDR is equal to or less than the transmit trigger count after transmission. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | BRK | 0 | R/WC0 | Break Detect<br>Indicates that a receive data break signal has been detected. If a break signal is detected, receive data (H'00) transfer to SCFRDR is stopped. After the break is canceled and the receive signal returns to 1, the receive data transfer resumes.<br>0: Indicates that no break signal has been received.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• 0 is written to BRK.<br>1: Indicates that a break signal has been received.<br>[Setting condition]<br>• Data with a framing error is received, followed by the space "0" (low level) for at least one frame length. |
| 3 | FER | 0 | R | Framing Error<br>Indicates that a framing error has been found in the data that is to be read next from SCFRDR in asynchronous mode.<br>0: Indicates that there is no framing error in the receive data that is to be read from SCFRDR.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• There is no framing error in the data that is to be read next from SCFRDR.<br>1: Indicates that there is a framing error in the receive data that is to be read from SCFRDR.<br>[Setting condition]<br>• There is a framing error in the data that is to be read next from SCFRDR. |
| 2 | PER | 0 | R | Parity Error<br>This bit indicates that a parity error has been found in the data that is to be read next from SCFRDR in asynchronous mode.<br>0: Indicates that there is no parity error in the receive data that is to be read from SCFRDR.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• There is no parity error in the data that is to be read next from SCFRDR.<br>1: Indicates that there is a parity error in the receive data that is to be read from SCFRDR.<br>[Setting condition]<br>• There is a parity error in the data that is to be read next from SCFRDR. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | RDF | 0 | R/WC0 | Receive FIFO Data Full<br>Indicates that the received data has been transferred from SCRSR to SCFRDR, and the number of receive data bytes in SCFRDR becomes equal to or more than the receive trigger count specified by the RTRG1 and RTRG0 bits in SCFCR.<br>SCFRDR is a 16-byte FIFO register. When RDF = 1, data equal to or more than the number of receive trigger data bytes can be read. When SCFRDR is empty, SCFRDR is read as an undefined value. The number of receive data bytes in SCFRDR is indicated by the lower bits of SCFDR.<br>If the number of data in SCFRDR is equal to or more than the trigger count, this bit will be set to 1 even if it is cleared to 0. At this time, read receive data until the number of data in SCFRDR is less than the trigger count, read RDF as 1, and then clear RDF.<br>0: Indicates that the number of receive data bytes in SCFRDR is less than the specified receive trigger count.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• SCFRDR is read until the number of receive data bytes in SCFRDR is less than the receive trigger count, and 0 is written to RDF.<br>• SCFRDR is read by the DMAC until the number of receive data bytes in SCFRDR is less than the receive trigger count.<br>1: Indicates that the number of receive data bytes in SCFRDR is equal to or more than the specified receive trigger count.<br>[Setting condition]<br>• Receive data more than the receive trigger count have been stored in SDFRDR. |
| 0 | DR | 0 | R/WC0 | Receive Data Ready<br>Indicates that the number of data bytes in SCFRDR is less than the receive trigger count and that no further data has been received for at least 15 etu after the stop bit of the data received last in asynchronous mode. This bit is not set in clock synchronous mode.<br>0: Indicates that data is being received or has been successfully received, and there is no receive data in SCFRDR.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• All the receive data in SCFRDR has been read, and 0 is written to DR.<br>• All the receive data in SCFRDR has been read by the DMAC.<br>1: Indicates that no further receive data has been received.<br>[Setting condition]<br>• The receive FIFO data register (SCFRDR) has the number of data byte that is below the receive trigger number of data bytes and no further data has arrived for at least 15 etu after the stop bit of the data received last.<br>Note: etu: Elementary Time Unit (time for transfer of 1 bit)<br>Equivalent to 1.5 frames with an 8-bit, 1-stop-bit format. |

### 105.2.8 Bit Rate Register (SCBRR)



| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

SCBRR is an 8-bit register that sets the serial transmission/reception bit rate in accordance with the baud rate generator operating clock selected by the CKS1 and CKS0 bits in SCSMR. This baud rate generator is intended for PCK, PCK/4, PCK/16, and PCK/64. see section 105.3.3, Generation of Internal Clock.

For details on the baud rate generator for external clock, see section 105.3.4, Baud Rate Generator for External Clock (BRG).

SCBRR can always be read from and written to by the CPU except for during transfer.

SCBRR is initialized to H'FF by a power-on reset.

### 105.2.9 FIFO Control Register (SCFCR)

SCFCR is a register that resets data counts and sets the number of trigger data bytes for the transmit and receive FIFO registers. It also has a loopback test enable bit.

SCFCR can always be read from and written to by the CPU except for during transfer.

SCFCR is initialized to H'0000 by a power-on reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RSTRG[2:0] | | | RTRG[1:0] | | TTRG[1:0] | | MCE | TFRST | RF RST | LOO P |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 10 to 8 | RSTRG[2:0] | 000 | R/W | RTS Output Active Trigger<br>The RTS# signal is high when the number of receive data bytes stored in SCFRDR is equal to or more than the specified trigger number shown below:<br>000: 15<br>001: 1<br>010: 4<br>011: 6<br>100: 8<br>101: 10<br>110: 12<br>111: 14 |
| 7, 6 | RTRG[1:0] | 00 | R/W | Receive FIFO Data Count Trigger<br>These bits specify the number of receive data bytes that makes the RDF (receive data full) flag to be set in SCFSR.<br>The RDF flag is set when the number of receive data bytes in SCFRDR is equal to or more than the specified trigger number shown below: |

| RTRG[1:0] | Asynchronous Mode | Clock Synchronous Mode |
|---|---|---|
| 00 | 1 | 1 |
| 01 | 4 | 2 |
| 10 | 8 | 8 |
| 11 | 14 | 14 |

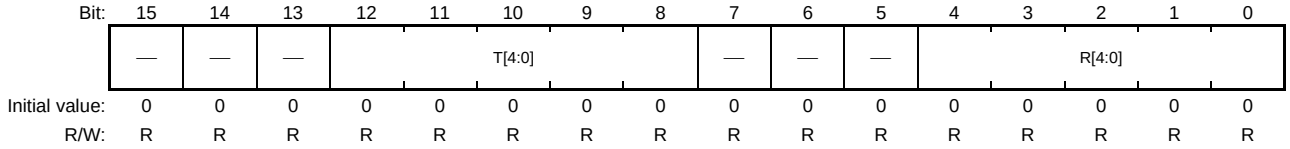
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5, 4 | TTRG[1:0] | 00 | R/W | Transmit FIFO Data Count Trigger<br>These bits specify the number of remaining transmit data bytes that makes the transmit FIFO data register empty (TDFE) flag to be set in SCFSR.<br>The TDFE flag is set when the number of transmit data bytes in SCFTDR is equal to or less than the specified trigger count shown below, after transmission:<br>00: 8 (8)<br>01: 4 (12)<br>10: 2 (14)<br>11: 0 (16)<br>Note: Values in parentheses indicate the empty space in SCFTDR in bytes. |
| 3 | MCE | 0 | R/W | Modem Control Enable<br>Enables or disables modem control signals CTS# and RTS#.<br>This bit should always be set to 0 in clock synchronous mode. Whether modem control can be selected or not depends on the interface.<br>0: Disables modem signals.*<br>1: Enables modem signals.<br>Note: CTS# and RTS# control ports. |
| 2 | TFRST | 0 | R/W | Transmit FIFO Data Register Reset<br>Enables or disables a transmit FIFO data register reset that empties the register.<br>0: Disables the reset.*<br>1: Enables the reset.<br>Note: The register is reset by a power-on reset. |
| 1 | RFRST | 0 | R/W | Receive FIFO Data Register Reset<br>Enables or disables a receive FIFO data register reset that empties the register.<br>0: Disables the reset.*<br>1: Enables the reset.<br>Note: The register is reset by a power-on reset. |
| 0 | LOOP | 0 | R/W | Loopback Test<br>Enables or disables the loopback test by internally connecting the transmit output pin (TX) and receive input pin (RX), and the RTS# pin and CTS# pin. If the loopback test is specified, the signal on the RTS# pin is internally conveyed to CTS# pin, and data transmission is automatically suspended when RTS# is asserted.<br>0: Disables the loopback test.<br>1: Enables the loopback test.<br>Even if loopback test is specified by this bit, SCSPTR.RTSDT and SCSPTR.CTSDT bits does not reflect signal on loopback path, because those registers shows signals on RTS#/CTS# pins. |

### 105.2.10 FIFO Data Count Register (SCFDR)

SCFDR is a 16-bit register that indicates the number of data bytes stored in SCFTDR and that in SCFRDR.

The upper 8 bits indicate the number of transmit data bytes in SCFTDR, and the lower 8 bits indicates the number of receive data bytes in SCFRDR.

SCFDR can always be read by the CPU.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 13 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 12 to 8 | T[4:0] | H'00 | R | These bits show the number of data bytes untransmitted and still stored in SCFTDR.<br>H'00 indicates that there is no transmit data in SCFTDR. H'10 indicates that SCFTDR is full of transmit data. |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 to 0 | R[4:0] | H'00 | R | These bits show the number of receive data stored in SCFRDR in bytes.<br>H'00 indicates that there is no receive data in SCFRDR. H'10 indicates that SCFRDR is full of receive data. |

### 105.2.11 Serial Port Register (SCSPTR)

SCSPTR controls multiplexed input/output and data on the serial communication interface (SCIF) ports. Bits 1 and 0 control breaks in serial transmission/reception by reading input data from the RX pin and writing output data to the TX pin. Bits 3 and 2 read input data from and write output data to the SCK pin. Bits 5 and 4 read input data from and write output data to the CTS# pin. Bits 7 and 6 read input data from and write output data to the RTS# pin.

SCSPTR is a 16-bit register that can always be read from and written to by the CPU.

All SCSPTR bits except bits 6, 4, 2, and 0 are initialized to 0 by a power-on reset. The values of bits 6, 4, 2, and 0 are undefined.

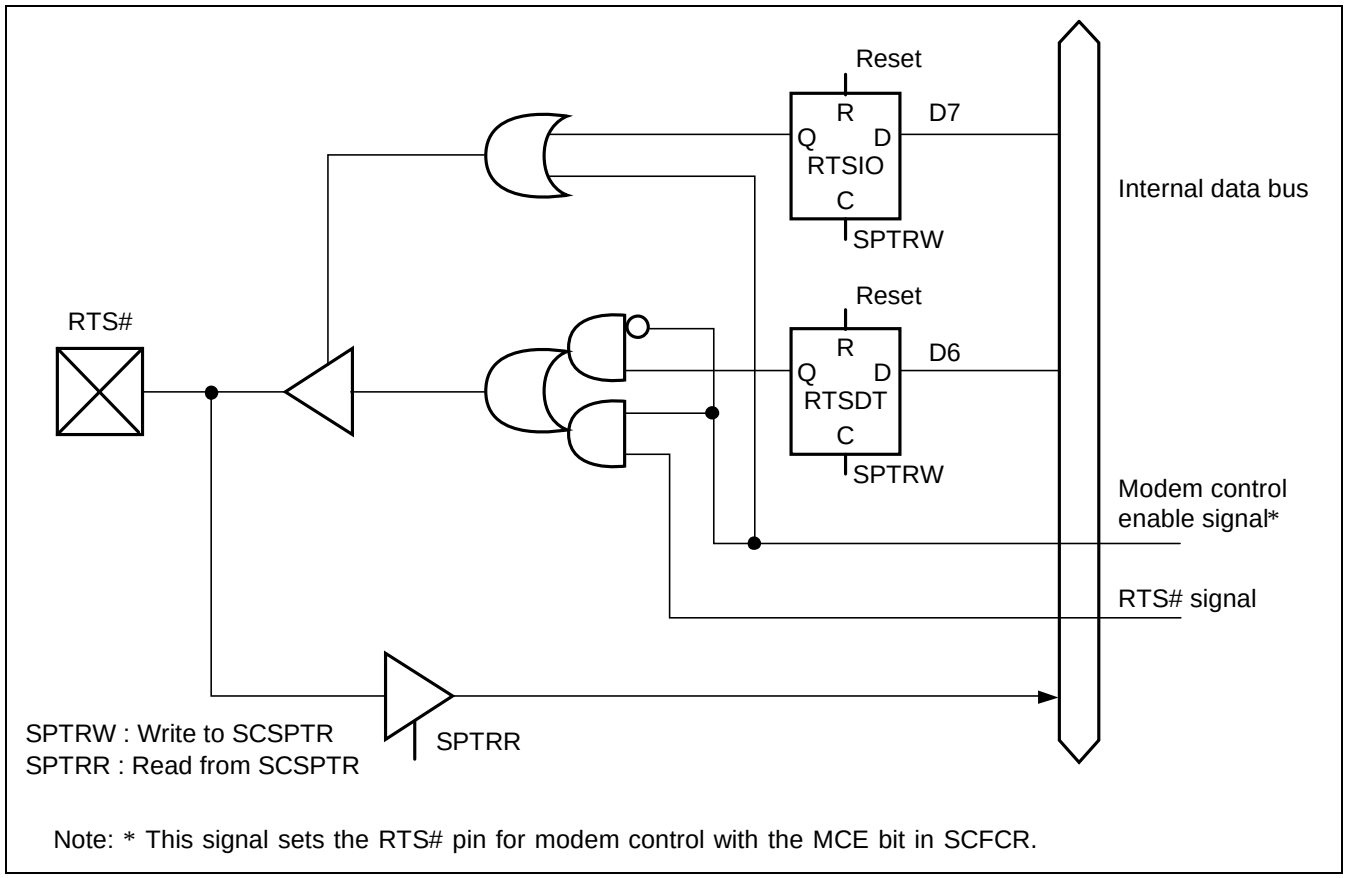
Note: Whether modem control can be selected or not depends on the interface.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RTSIO | RTSDT | CTSIO | CTSDT | SCKIO | SCKDT | SPB2IO | SPB2DT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | 0 | — | 0 | — | 0 | — |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | RTSIO | 0 | R/W | Serial Port RTS# Pin Input/output<br>Specifies input or output for the serial port RTS# pin. To actually set the RTS# pin as a port output pin to output the value set by the RTSDT bit, the MCE bit in SCFCR should be cleared to 0.<br>0: Indicates that this bit does not output the value of the RTSDT bit to the RTS# pin.<br>1: Indicates that this bit outputs the value of the RTSDT bit to the RTS# pin. |
| 6 | RTSDT | — | R/W | Serial Port RTS# Pin Data<br>Specifies the input/output data level of the serial port RTS# pin. Whether the pin is set for input or output is determined by the RTSIO bit. When the pin is set for output, the value of the RTSDT bit is output to the RTS# pin. Regardless of the value of the RTSIO bit, the value of the RTS# pin is read from the RTSDT bit.<br>The initial value of this bit is undefined after a power-on reset.<br>0: Indicates that the input/output data is low level.<br>1: Indicates that the input/output data is high level. |
| 5 | CTSIO | 0 | R/W | Serial Port CTS# Pin Input/output<br>Specifies input or output for the serial port CTS# pin. To actually set the CTS# pin as a port output pin to output the value set by the CTSDT bit, the MCE bit in SCFCR should be cleared to 0.<br>0: Indicates that the CTSDT bit value is not output to the CTS# pin.<br>1: Indicates that the CTSDT bit value is output to the CTS# pin. |

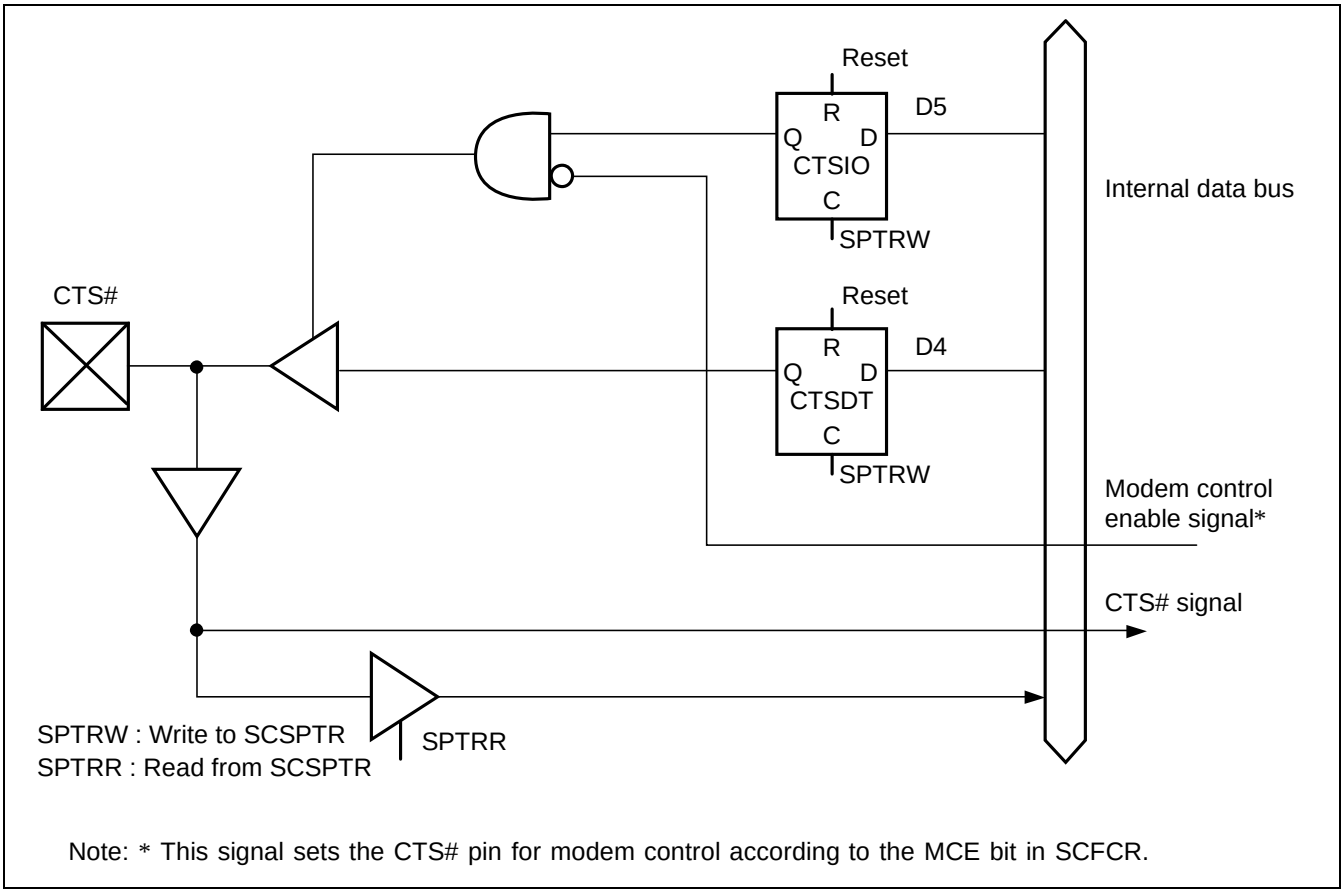
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | CTSDT | — | R/W | Serial Port CTS# Pin Data<br>Specifies the input/output data level of the serial port CTS# pin. Whether the pin is set for input or output is determined by the CTSIO bit. When the pin is set for output, the value of the CTSDT bit is output to the CTS# pin. Regardless of the value of the CTSIO bit, the value of the CTS# pin is read from the CTSDT bit.<br>The initial value of this bit is undefined after a power-on.<br>0: Indicates that the input/output data is low level.<br>1: Indicates that the input/output data is high level. |
| 3 | SCKIO | 0 | R/W | Serial Port Clock Pin Input/output<br>Specifies input or output for the serial port SCK pin. To actually set the SCK pin as a port output pin to output the value set by the SCKDT bit, the CKE [1:0] bits in SCSCR should be cleared to 0.<br>0: Indicates that the SCKDT bit value is not output to the SCK pin.<br>1: Indicates that the SCKDT bit value is output to the SCK pin. |
| 2 | SCKDT | — | R/W | Serial Port Clock Pin Data<br>Specifies the input/output data level of the serial port SCK pin. Whether the pin is set for input or output is determined by the SCKIO bit. When the pin is set for output, the value of the SCKDT bit is output to the SCK pin. Regardless of the value of the SCKIO bit, the value of the SCK pin is read from the SCKDT bit.<br>The initial value of this bit is undefined after a power-on reset.<br>0: Indicates that the input/output data is low level.<br>1: Indicates that the input/output data is high level. |
| 1 | SPB2IO | 0 | R/W | Serial Port Break Input/Output<br>Specifies the output condition of the serial port TX pin. To actually set the TX pin as a port output pin to output the value set by the SPB2DT bit, the TE bit in SCSCR should be cleared to 0.<br>0: Indicates that the SPB2DT bit value is not output to the TX pin.<br>1: Indicates that the SPB2DT bit value is output to the TX pin. |
| 0 | SPB2DT | — | R/W | Serial Port Break Data<br>Specifies the input level of the serial port RX pin and the output level of the TX pin. The TX pin output conditions are determined by the SPB2IO bit. When the TX pin is set for output, the value of the SPB2DT bit is output to the TX pin. Regardless of the value of the SPB2IO bit, the value of the RX pin is read from the SPB2DT bit.<br>The initial value of this bit is undefined after a power-on reset.<br>0: Indicates that the input/output data is low level.<br>1: Indicates that the input/output data is high level. |

Figure 105.2 to 105.6 show the block diagrams of the SCIF I/O ports.



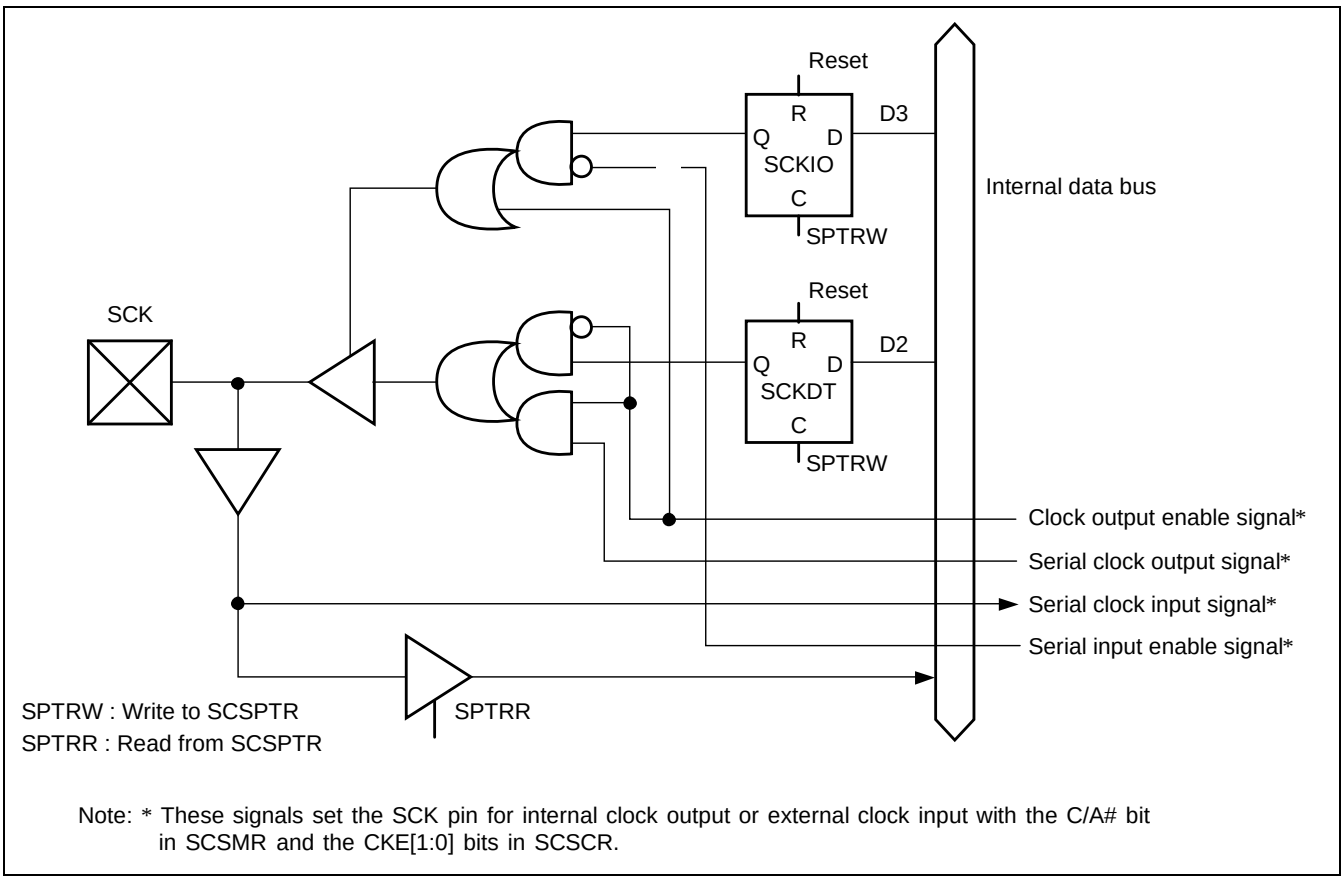

Note: * This signal sets the RTS# pin for modem control with the MCE bit in SCFCR.

**Figure 105.2 RTS# Pin**




Note: * This signal sets the CTS# pin for modem control according to the MCE bit in SCFCR.

**Figure 105.3 CTS# Pin**




Note: * These signals set the SCK pin for internal clock output or external clock input with the C/A# bit in SCSMR and the CKE[1:0] bits in SCSCR.

**Figure 105.4 SCK Pin**

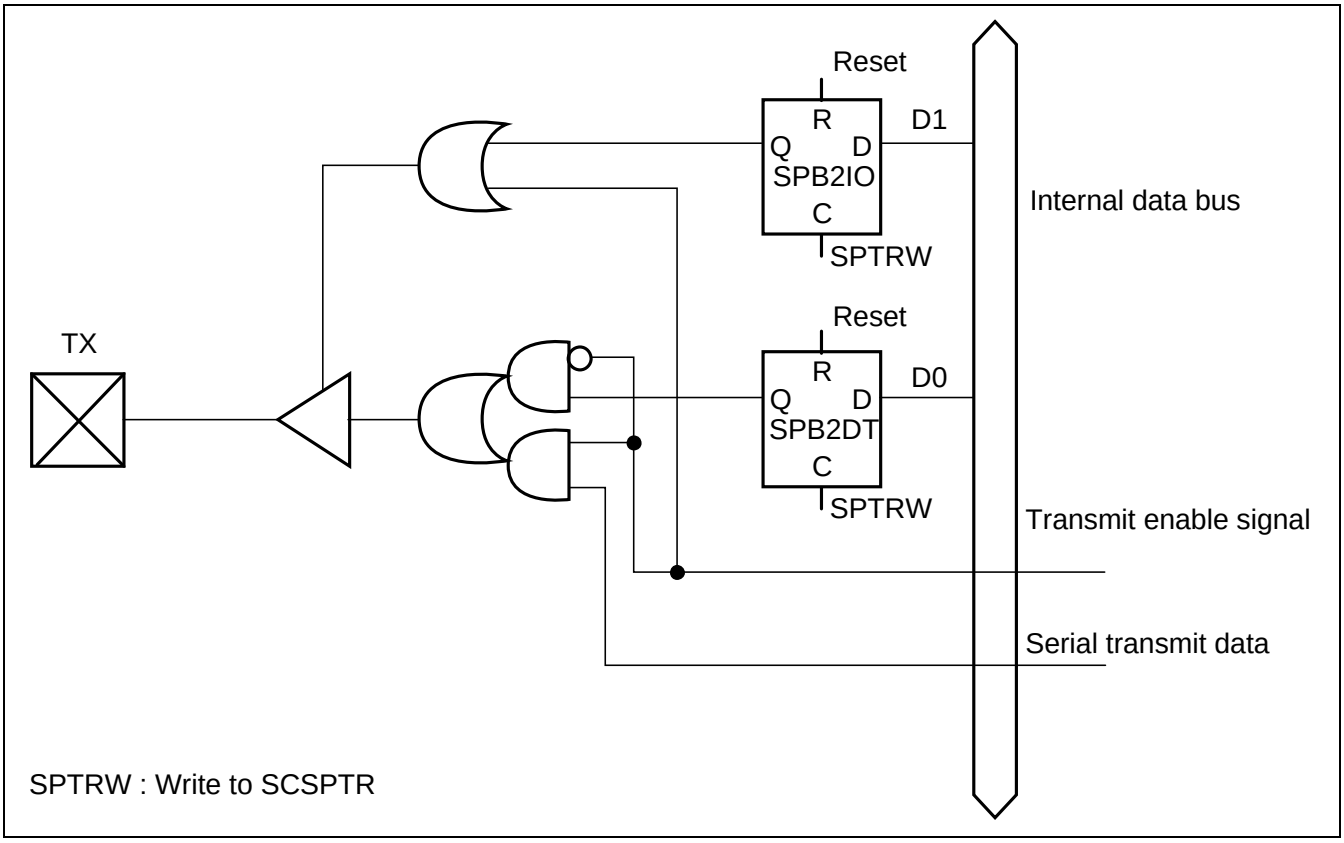



**Figure 105.5 TX Pin**

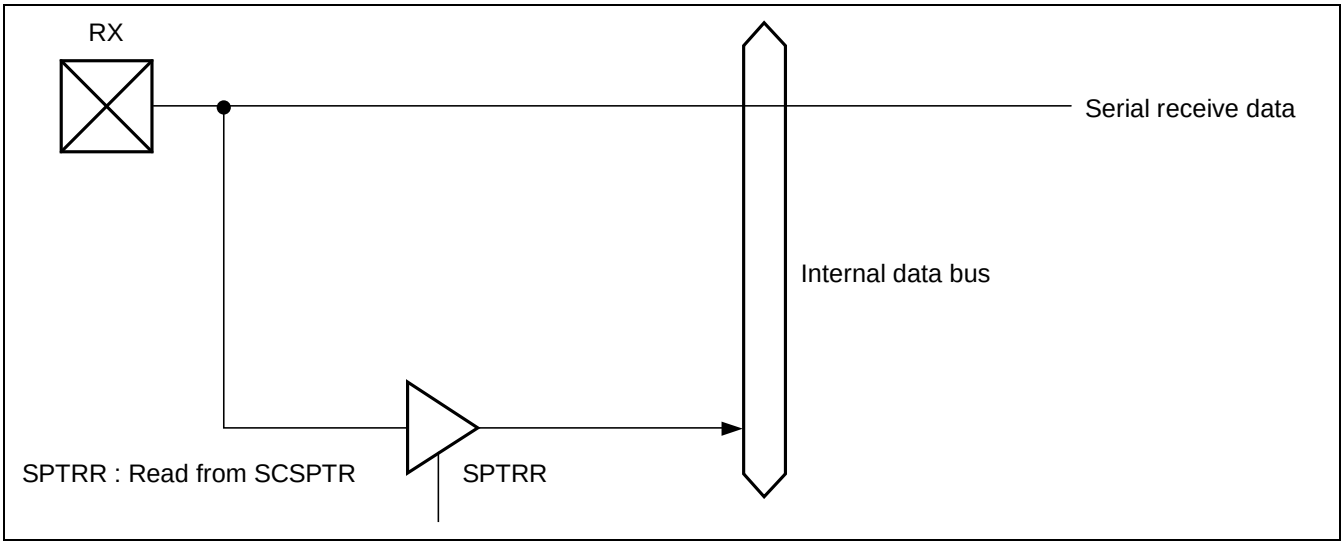



**Figure 105.6 RX Pin**

### 105.2.12 Line Status Register (SCLSR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TO | — | ORER |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/WC0 | R | R/WC0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | TO | 0 | R/WC0 | Timeout<br>Indicates that the number of data bytes in SCFRDR is less than the receive trigger count, and that no further data has been received for at least 15 etu after the stop bit of the last receive data in asynchronous mode. This bit is not set in clock synchronous mode.<br>0: Indicates that data is being received or has been successfully received and that there is no receive data in SCFRDR.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• All the receive data in the SCFRDR has been read, and 0 is written to TO.<br>1: Indicates that no further receive data has been received (receive timeout).<br>[Setting condition]<br>• The number of data bytes in SCFRDR is less than the receive trigger count and no further data has been received for at least 15 etu after the stop bit of the last receive data.*<br>Note: etu: Elementary Time Unit (time for transfer of 1 bit)<br>Equivalent to 1.5 frames with an 8-bit, 1-stop-bit format. |
| 1 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | ORER | 0 | R/WC0 | Overrun Error<br>Indicates that an overrun error has occurred in reception and abnormal termination is caused.<br>If an overrun error occurs, the receive data prior to the overrun error is retained in SCFRDR and the data received subsequently is discarded.<br>Any subsequent serial reception is disabled while the ORER flag is 1.<br>To resume data reception after clearing the ORER flag, be sure to first read (or clear) data in the receive FIFO and handle the error, then clear the ORER flag.<br>0: Indicates that data is being received or has been successfully received.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• 0 is written to ORER.<br>1: Indicates that an overrun error has occurred in reception.<br>[Setting condition]<br>• The next serial reception has been completed while SCFRDR is full of 16-byte data.<br>Note: When bit RE in SCSCR is cleared to 0, the ORER flag is not affected and its previous state is retained. |

### 105.2.13 Frequency Division Register (DL)

This register specifies the value of frequency division for the frequency divided clock generated by the BRG.

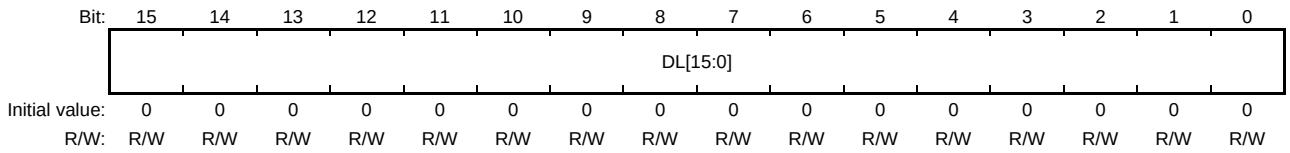
This register supports a 16-bit binary format that allows specifying a value in the rage of 1 to 65535.

Setting all 0s in this register makes the BRG output the frequency divided clock at the low level.

The value of frequency division is given by the following formula:

The value of frequency division = (clock input frequency)/ (required baud rate × 16)

Table 105.9 shows how to use the baud rate generator with a 3.6864-MHz crystal resonator.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | DL[15:0] | All 0 | R/W | Specifies a division value of frequency clock generated in BRG.<br>The value settings enabled in the range of 1 to 65535. |

### 105.2.14 Clock Select Register (CKS)

This register switches the output between the frequency divided clock and specifies a source clock for the external baud rate.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CKS | XIN | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | CKS | 0 | R/W | This bit switches the output between the frequency divided clock (SC_CLK) and external clock (SCK).<br>0: Selects the frequency divided clock.<br>1: Selects the external clock. |
| 14 | XIN | 0 | R/W | Selects the baud rate generator clock source for the external clock between SCIF_CLK and SCKi.<br>0: Selects the external clock (SCIF_CLK).<br>1: Selects the internal clock (SCKi). |
| 13 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Only 0 should be written to these bits. |

## 105.3 Operation

### 105.3.1 Operation in Asynchronous Mode

In asynchronous mode, the SCIF performs serial communication, in which data is transmitted/received in character units using the attached start bit indicating the start of communication and stop bit indicating the end of communication.

Figure 105.7 shows the general format for asynchronous serial communication.

In asynchronous serial communication, the transmission line is usually kept marked (high level). The SCIF monitors the transmission line, and when it finds a space (low level), it regards the space as a start bit and starts serial communication.

One character in serial communication consists of a start bit (low level), followed by transmit/receive data (LSB-first; from the lowest bit), a parity bit (high or low level), and a stop bit (high level).

During reception in asynchronous mode, the SCIF performs synchronization at the falling edge of the start bit. Communication data is acquired at the center of each bit because the SCIF samples data at the eighth pulse of the clock which has a frequency of 16 times the bit rate.

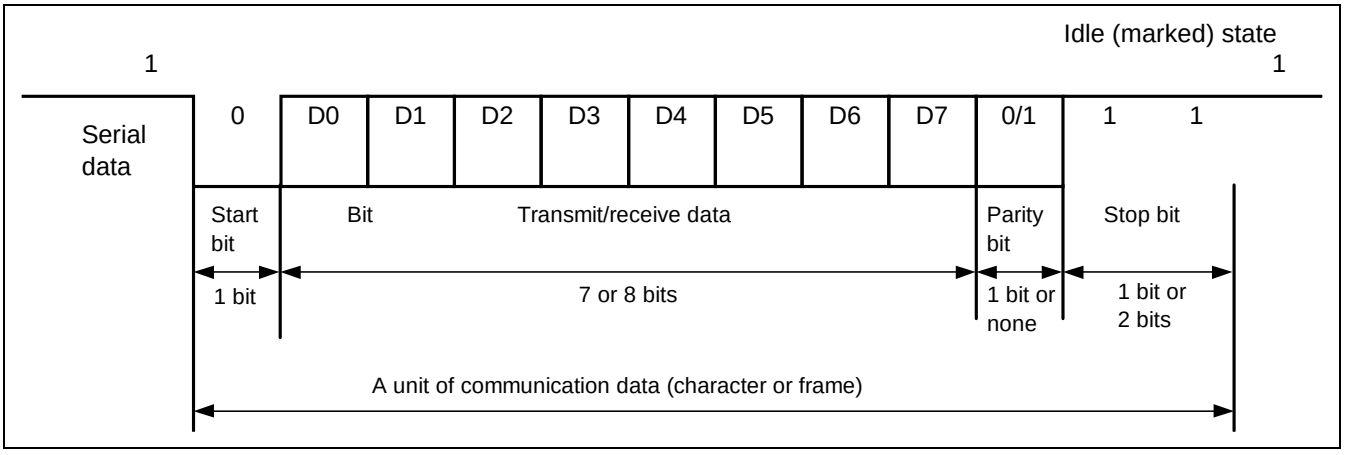



**Figure 105.7 Data Format in Asynchronous Mode (Example of 8-Bit Data with Parity and Two Stop Bits)**

### (1) Transmission/Reception Format

Table 105.7 shows available data transfer formats. The SCIF supports 8 transfer formats, which can be specified by SCSMR.

**Table 105.7 Serial Transmission/Reception Formats (Asynchronous Mode)**

| SCSMR Settings | | | Serial Transmission/Reception Format and Frame Length | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHR | PE | STOP | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 0 | 0 | 0 | S | 8-bit data | | | | | | | | STOP | | |
| | | 1 | S | 8-bit data | | | | | | | | STOP | STOP | |
| | 1 | 0 | S | 8-bit data | | | | | | | | P | STOP | |
| | | 1 | S | 8-bit data | | | | | | | | P | STOP | STOP |
| 1 | 0 | 0 | S | 7-bit data | | | | | | | STOP | | | |
| | | 1 | S | 7-bit data | | | | | | | STOP | STOP | | |
| | 1 | 0 | S | 7-bit data | | | | | | | P | STOP | | |
| | | 1 | S | 7-bit data | | | | | | | P | STOP | STOP | |

[Legend]
S: Start bit
STOP: Stop bit
P: Parity bit

### (2) Clock

The transfer clock can be selected from the following two clocks using the CKE [1:0] bits in SCSCR:

- Internal clock generated by the on-chip baud rate generator
- External clock generated by an external clock baud rate generator

### (3) Data Transmission/Reception

#### (a) Initialization of SCIF (Asynchronous Mode)

Before transmitting/receiving data or changing the operating mode or communication format, the SCIF should be initialized using the sample flowchart for SCIF initialization shown in Figure 105.8.

[Notes]

Clearing the TE bit to 0 initializes SCTSR. However, SCFSR, SCFTDR, and SCFRDR contents are retained even if the TE and RE bits are cleared to 0.

The TE bit should be cleared to 0 after all transmit data has been sent and the TEND flag has been set in SCFSR. The TE bit can be cleared to 0 during transmission, but the data being transmitted will enter the marked state after clearing. In addition, before setting the TE bit to 1 to restart the transmission, set the TFRST bit to 1 in SCFCR to reset SCFTDR.

When an external clock is used, do not stop the clock during operation or initialization. If stopped, the operation will be unreliable. Furthermore, when the baud rate generator for external clock is also to be used, be sure to make settings for it before starting initialization of the SCIF.

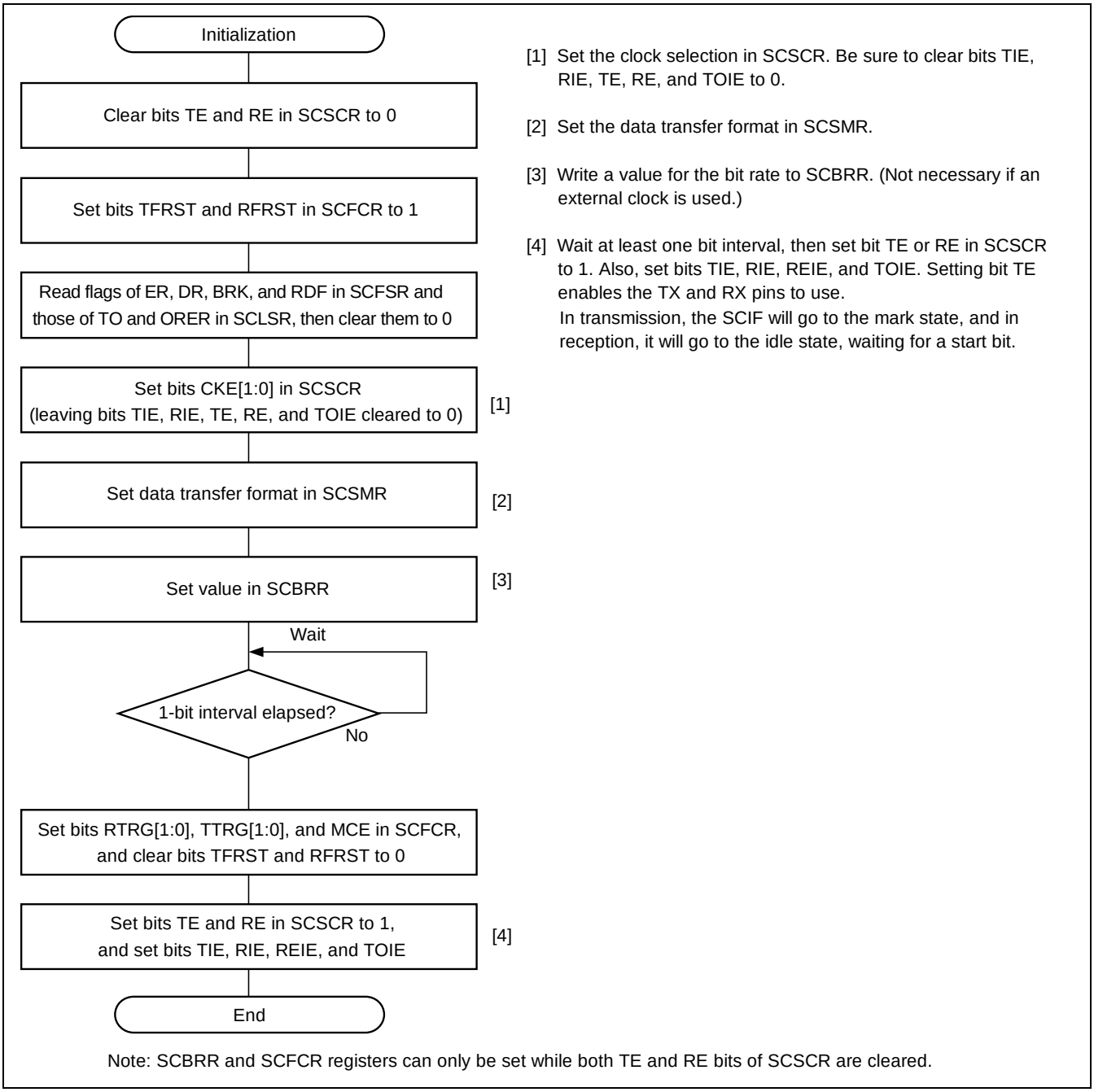



**Figure 105.8 Sample Flowchart for Initializing the SCIF**

**(b) Serial Data Transmission (Asynchronous Mode)**

Figure 105.9 shows a sample flowchart for serial transmission.

After the SCIF transmission operation is enabled, serial data transmission can be performed using the following procedure:

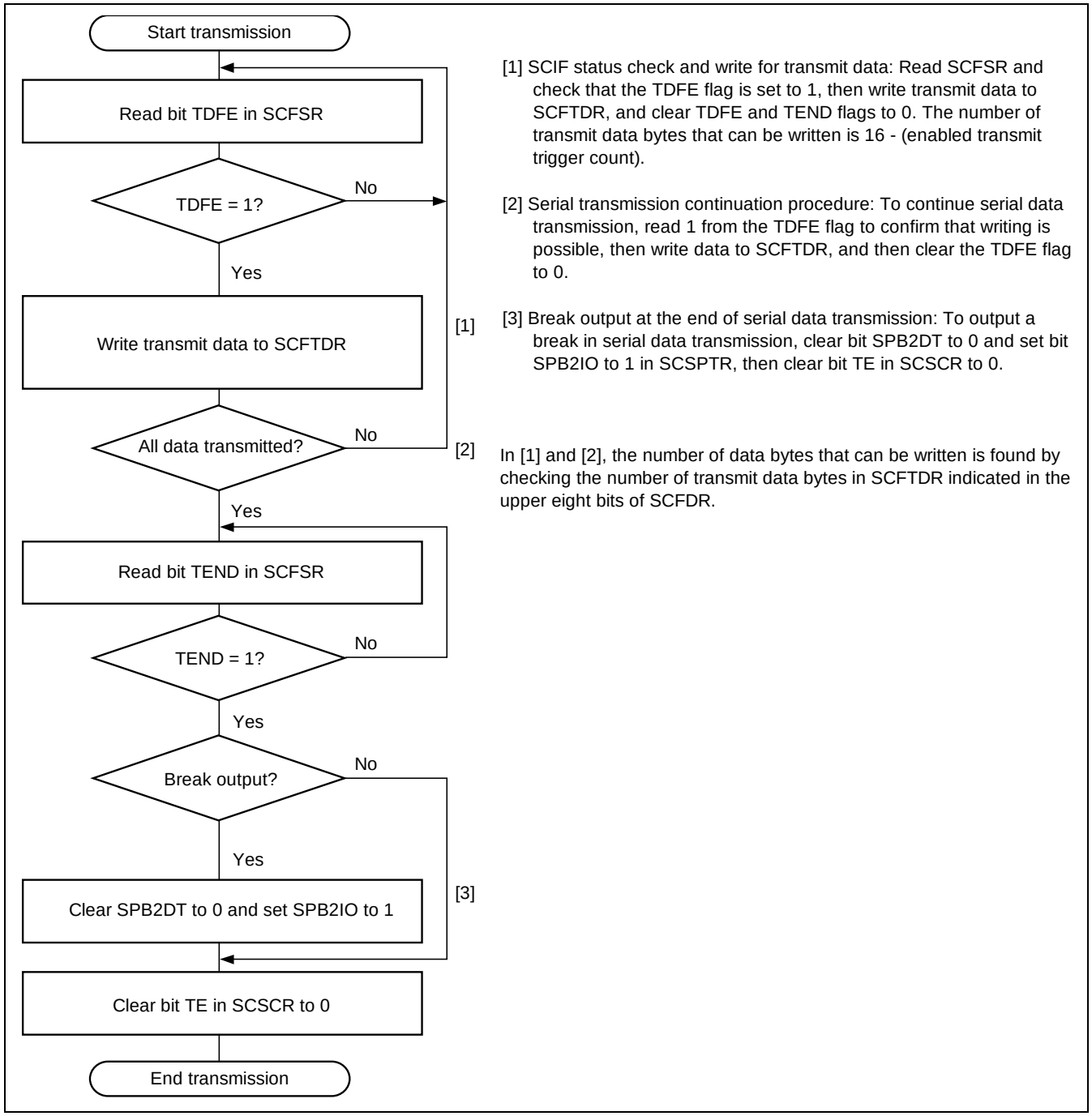



**Figure 105.9 Sample Flowchart for Serial Transmission**

In serial transmission, the SCIF operates as follows:

1. When data is written to SCFTDR, the SCIF transfers the data from SCFTDR to SCTSR and starts transmitting. Confirm that the TDFE flag in SCFSR is set to 1 before writing transmit data to SCFTDR. The number of data bytes that can be written is at least (16- (transmit trigger count)).
2. When data is transferred from SCFTDR to SCTSR and the SCIF starts transmission, consecutive transmission is performed until there is no transmit data left in SCFTDR. When the number of transmit data bytes in SCFTDR is equal to or less than the transmit trigger count specified in SCFCR, the TDFE flag is set. If the TIE and TEIE bits in SCSCR are set to 1 and 0, respectively at this time, a transmit-FIFO-data-empty interrupt (TDFE) request occurs. The serial transmit data is sent from the TX pin in the following order:
   A. Start bit: One 0-bit is output.
   B. Transmit data: 8-bit or 7-bit data is output in LSB-first order.
   C. Parity bit: One parity bit (even or odd parity) is output.
      A format that does not output a parity bit can also be selected.
   D. Stop bit(s): One or two stop bits are output.
   E. Marked state: 1 is output continuously until the start bit that starts the next transmission is sent.
3. The SCIF checks transmit data in SCFTDR when sending the stop bit. If there is data in it, the SCIF transfers the data from SCFTDR to SCTSR, sends the stop bit, and then starts serial transmission of the next frame. If there is no transmit data, the SCIF sets the TEND flag to 1 in SCFSR and sends out the stop bit, and then the marked state is entered to output 1 continuously. At this time, if the TIE and TEIE bits in SCSCR are set to 1, a transmit-end interrupt (TEND) request occurs.

Figure 105.10 shows an example of transmission in asynchronous mode.

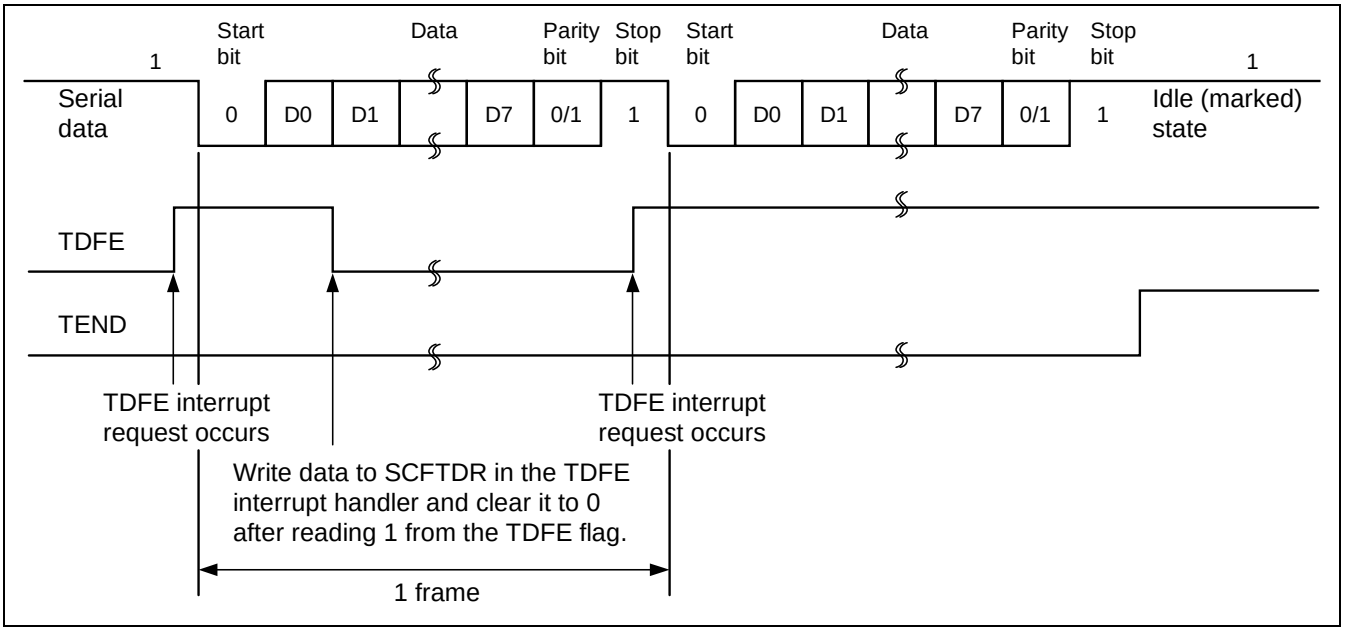



**Figure 105.10 Sample SCIF Transmission Operation (Example of 8-Bit Data with Parity and One Stop Bit)**

4. When modem control is enabled, transmission can be stopped or resumed in accordance with the CTS# input value. When CTS# is set to 1 during transmission, the marked state is entered after one frame of data transmission is ended. Setting CTS# to 0 restarts outputting the next transmit data from the start bit. Figure 105.11 shows an example of the operation with modem control enabled.

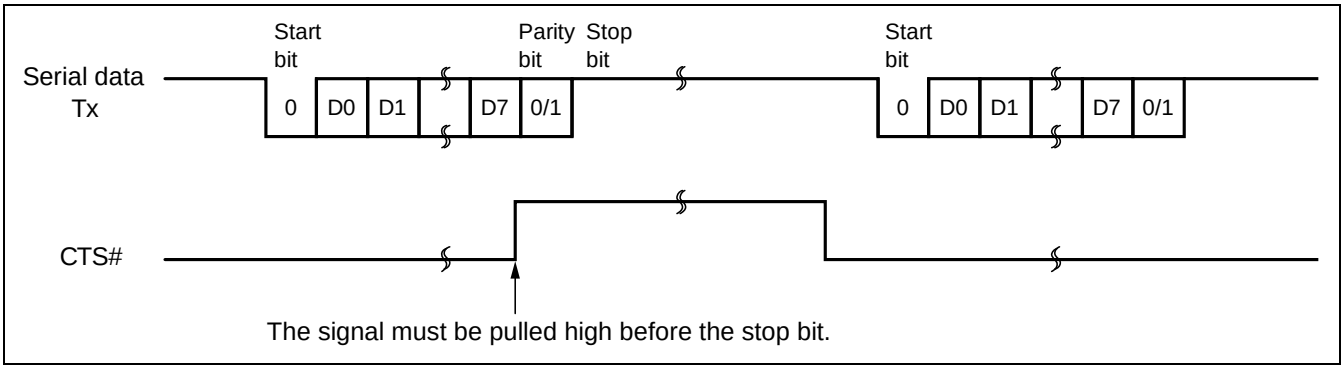



**Figure 105.11 Sample Operation with Modem Control Enabled (CTS#)**

### (c) Serial Data Reception (Asynchronous Mode)

Figure 105.12 and 105.13 show sample flowcharts for serial reception.

After the SCIF reception operation is enabled, serial data reception can be performed using the following procedure:

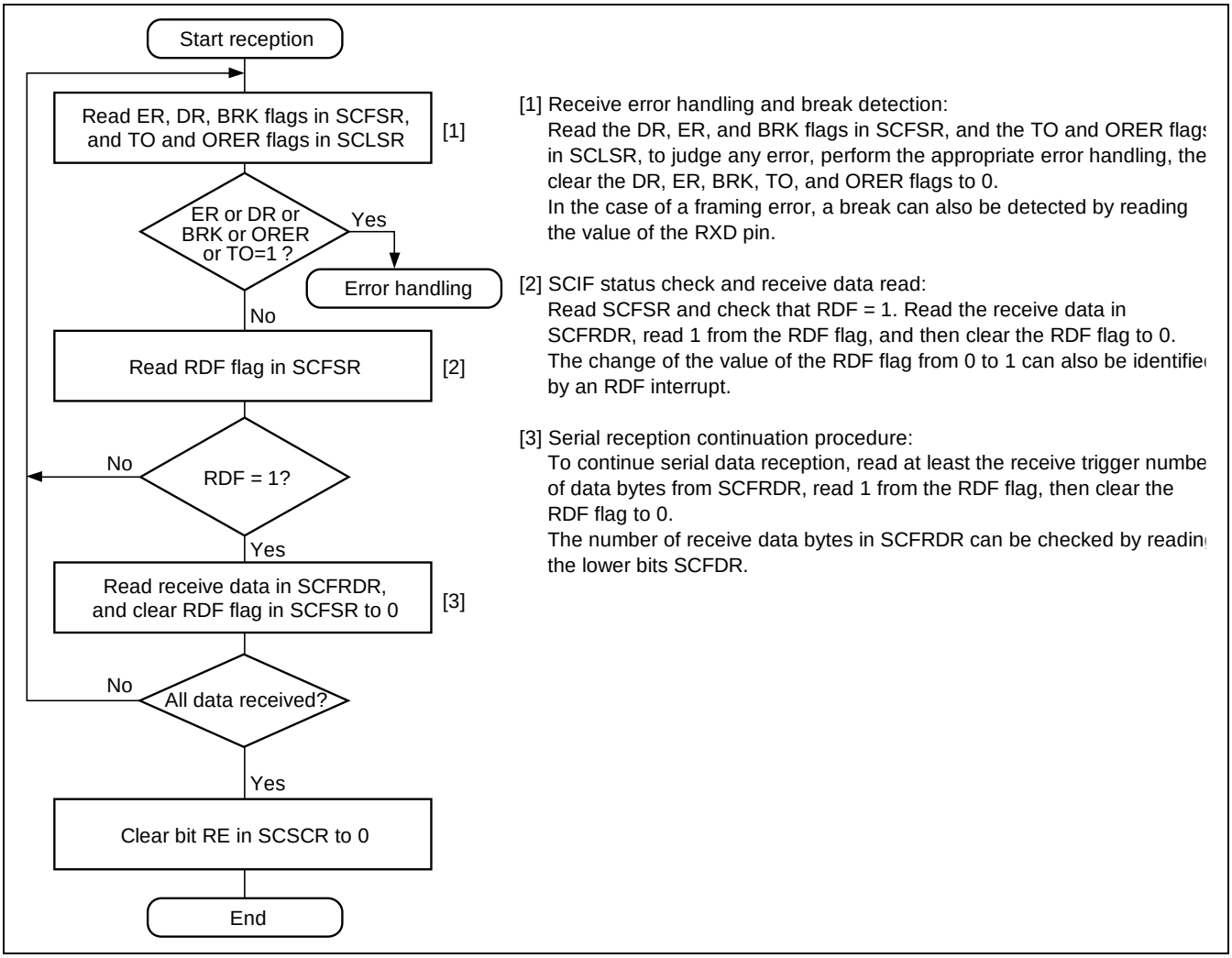
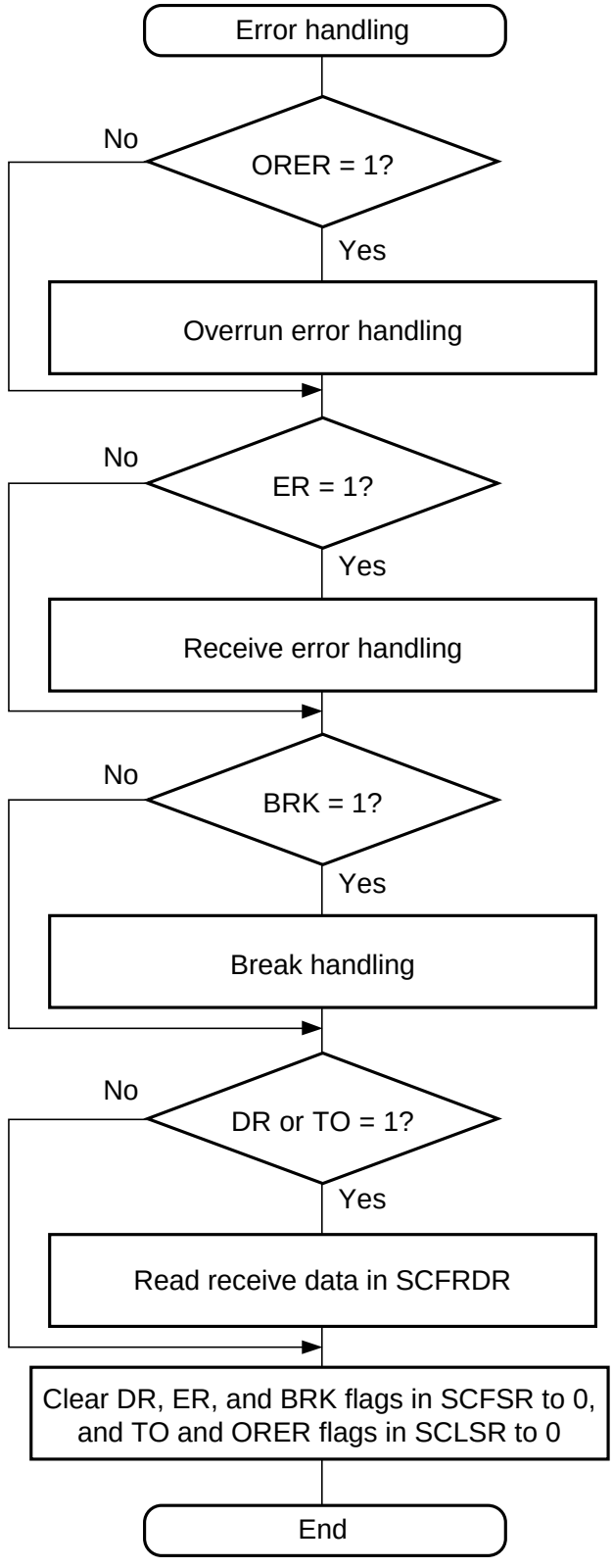



**Figure 105.12 Sample Flowchart for Serial Reception (1)**




Whether a framing error or parity error has occurred in the receive data that is to be read from SCFRDR can be determined by checking bits FER and PER in SCFSR.

**Figure 105.13 Sample Flowchart for Serial Reception (2)**

In serial reception, the SCIF operates as follows:

1. The SCIF monitors the transmission line, and when detecting the start bit 0, it performs internal synchronization and starts reception.
2. The SCIF stores the received data in SCRSR in LSB-to-MSB order.
3. The SCIF receives the parity bit and stop bit.
   After receiving these bits, the SCIF performs the following checks:
   A. Stop bit: The SCIF checks whether the stop bit is 1.
      If there are two stop bits, it checks only the first stop bit.
   B. Receive data: The SCIF checks that receive data can be transferred from the receive shift register (SCRSR) to SCFRDR.
   C. Overrun error: The SCIF checks that the ORER flag is 0, indicating that no overrun error has occurred.
   D. Break state: The SCIF checks that the BRK flag is 0, indicating that the break state is not set.
   If the SCIF can confirm the conditions of (b), (c), and (d), it stores the receive data in SCFRDR.

   Note: The SCIF continues to receive data even when a parity error or a framing error occurs.

4. If the RDF flag changes to 1 while the RIE bit in SCSCR is 1, a receive-FIFO-data-full interrupt (RDF) request occurs.
   If the DR flag changes to 1 while the RIE bit in SCSCR is 1, a receive-data-ready interrupt (DR) request occurs.
   If the TO flag changes to 1 while the TOIE bit in SCSCR is 1, a timeout interrupt (TO) request occurs.
   If the ER flag changes to 1 while the RIE or REIE bit in SCSCR is 1, a receive-error interrupt (ER) request occurs.
   If the BRK flag changes to 1 while the RIE or REIE bit in SCSCR is 1, a break interrupt (BRK) request occurs.
   If the ORER flag changes to 1 while the RIE or REIE bit in SCSCR is 1, an overrun-error interrupt (ORER) request occurs.

Figure 105.14 shows an example of reception in asynchronous mode.

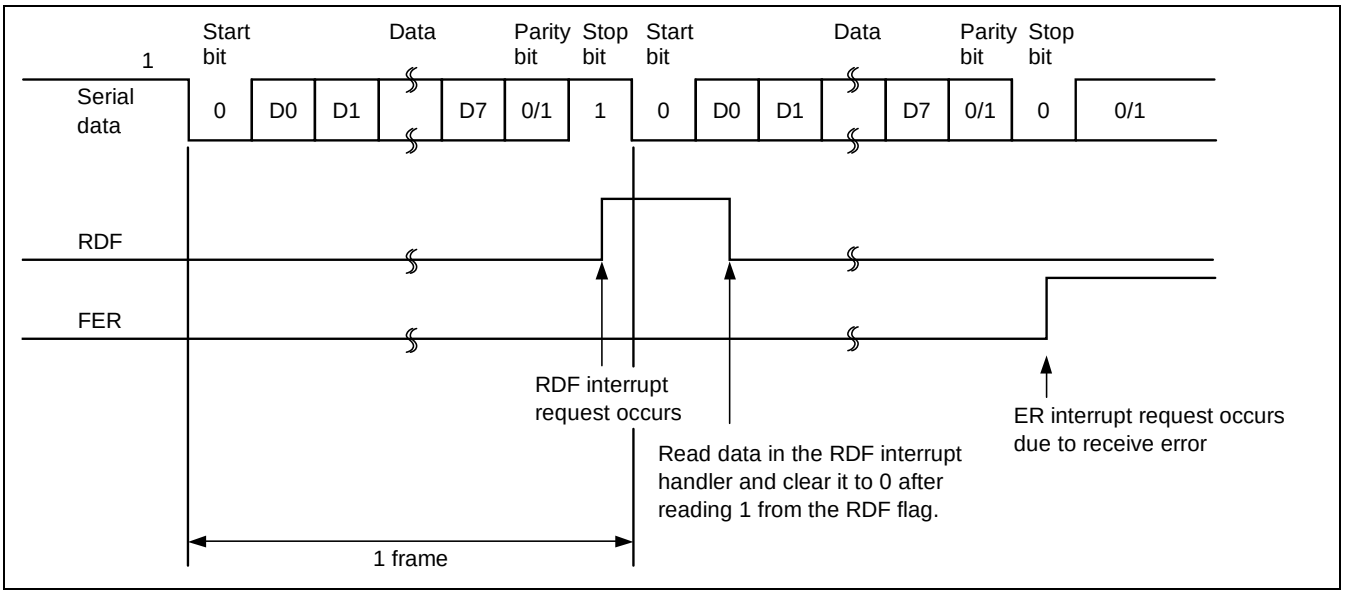



**Figure 105.14 Sample SCIF Receive Operation (Example of 8-Bit Data with Parity and One Stop Bit)**

5. When modem control is enabled, the RTS# signal is output when SCFRDR is empty. When RTS# is 0, data can be received. When RTS# is set to 1, the number of data bytes in SCFRDR is equal to or more than the RTS# output active trigger count.

Figure 105.15 shows an example of the operation with modem control enabled.

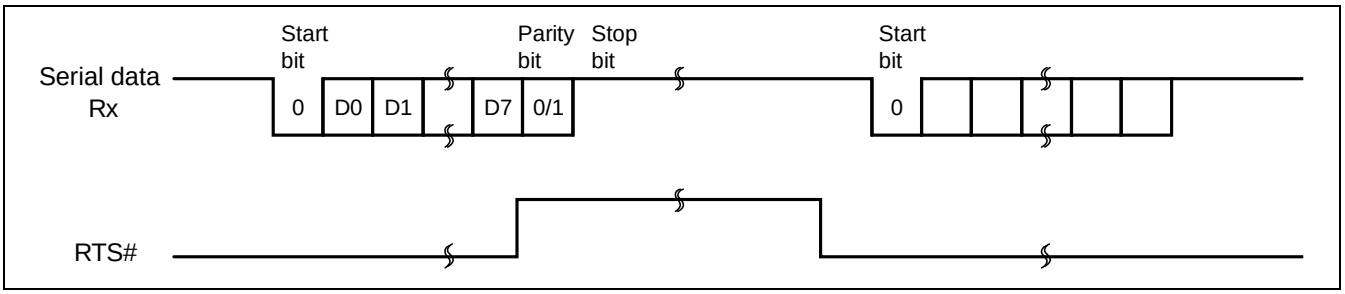



**Figure 105.15 Example of the Operation with Modem Control Enabled (RTS#)**

### 105.3.2 Operation in Clock Synchronous Mode

Clock synchronous mode, in which data is transmitted and received in synchronization with clock pulses, is suitable for fast serial communication.

Figure 105.16 shows the general format for clock synchronous serial communication. In the clock synchronous serial communication, data on the communication line is output between one falling edge of the synchronization clock and the next falling edge. The data decision is done at the rising edge of the clock.

In serial communication, each character is output starting with the LSB and ending with the MSB. After the MSB is output, the communication line retains the state of the last data.

In clock synchronous mode, the SCIF receives data in synchronization with a rising edge of the synchronization clock.

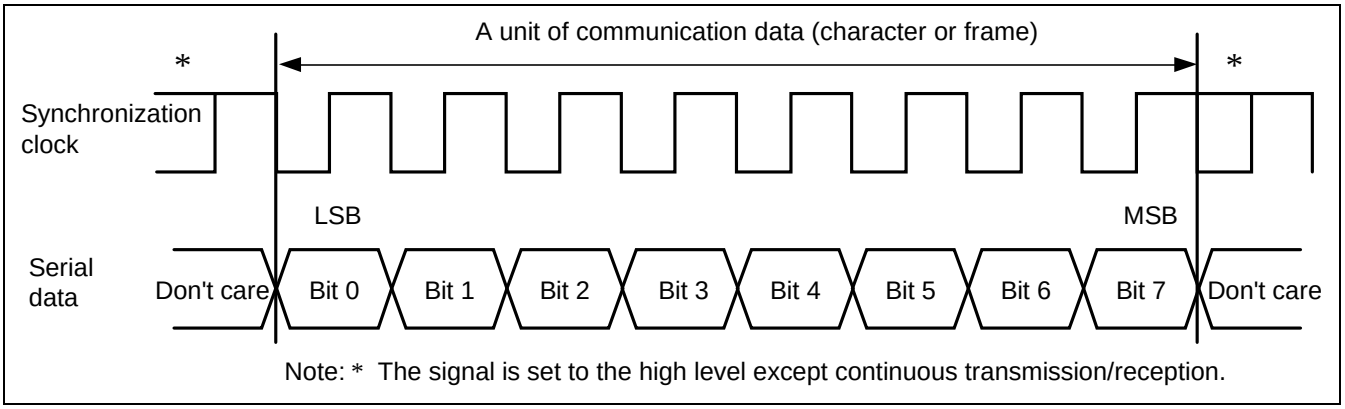



**Figure 105.16 Data Format for Clock Synchronous Communication**

#### (1) Data Transfer Format

The data transfer format is fixed to 8 bits. No parity bit can be added.

#### (2) Clock

The clock can be selected from the following two clocks using the C/A# bit in SCSMR and the CKE [1:0] bits in SCSCR:

- Internal clock generated by the on-chip baud rate generator
- External synchronization clock input at the SCK pin

When the SCIF is operated on an internal clock, the synchronization clock is output from the SCK pin. Eight synchronization clock pulses are output in one-character transfer, and when no transfer is performed the clock is fixed high. When only reception operation is performed, selecting an internal clock outputs clock pulses until the number of data bytes in the receive FIFO reaches the specified receive trigger count, while the RE bit in SCSCR is 1.

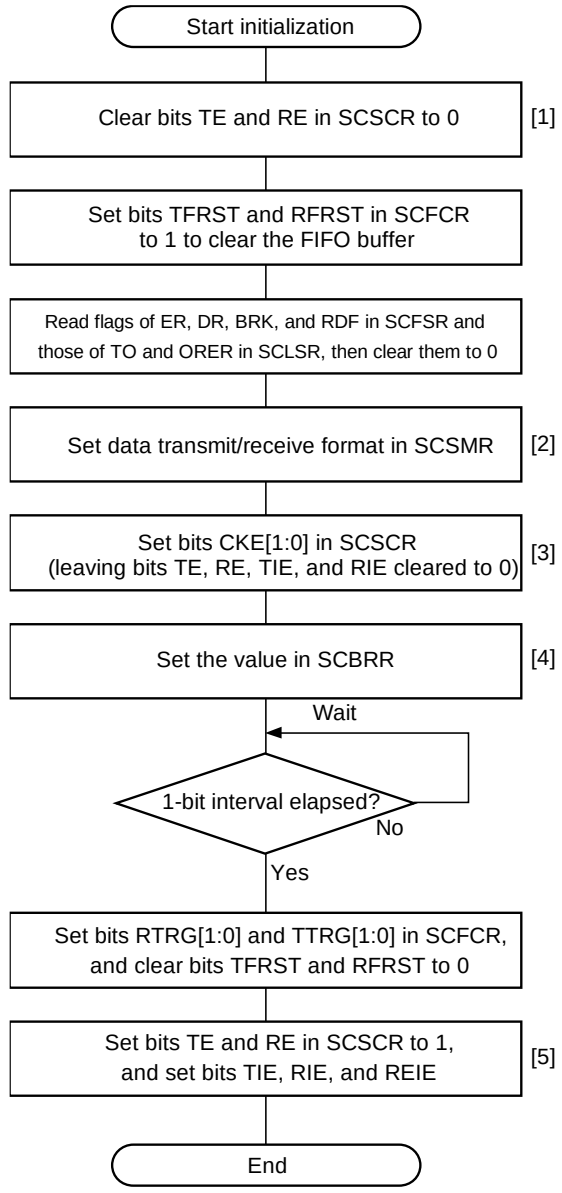
#### (3) Data Transmission/Reception

#### (a) Initialization of SCIF (Synchronous Mode)

Before transmitting/receiving data or changing the operating mode or communication format, the TE and RE bits in SCSCR to 0 and then the SCIF should be initialized using the sample flowchart for SCIF initialization shown in Figure 105.17.

[Note]

Clearing the TE bit to 0 initializes SCTSR. However, clearing the RE bit to 0 does not affect the RDF, PER, FER, and ORER flags and SCFRDR contents.




[1] Leave bits TE and RE cleared to 0 until the initialization almost ends.

[2] Set the data transfer format in SCSMR.

[3] Set bits CKE[1:0].

[4] Write the value of the bit rate to SCBRR. This is not required if an external clock is used. Wait at least one-bit interval after writing the value for the bit rate.

[5] Set bit TE or RE in SCSCR to 1. Also set bits TIE, RIE, and REIE to enable the TX, RX, and SCK pins to use.
In transmission, the TX pin will go to the marked state. In reception in clock synchronous mode with the synchronization clock output (clock master) selected, a clock starts to be output from the SCK pin at this point.

Note: SCBRR and SCFCR registers can only be set while both TE and RE bits of SCSCR are cleared.

**Figure 105.17 Sample Flowchart for SCIF Initialization**

**(b) Serial Data Transmission (Clock Synchronous mode)**

Figure 105.18 shows a sample flowchart for serial transmission.

After the SCIF transmission operation is enabled, serial data transmission can be performed using the following procedure:

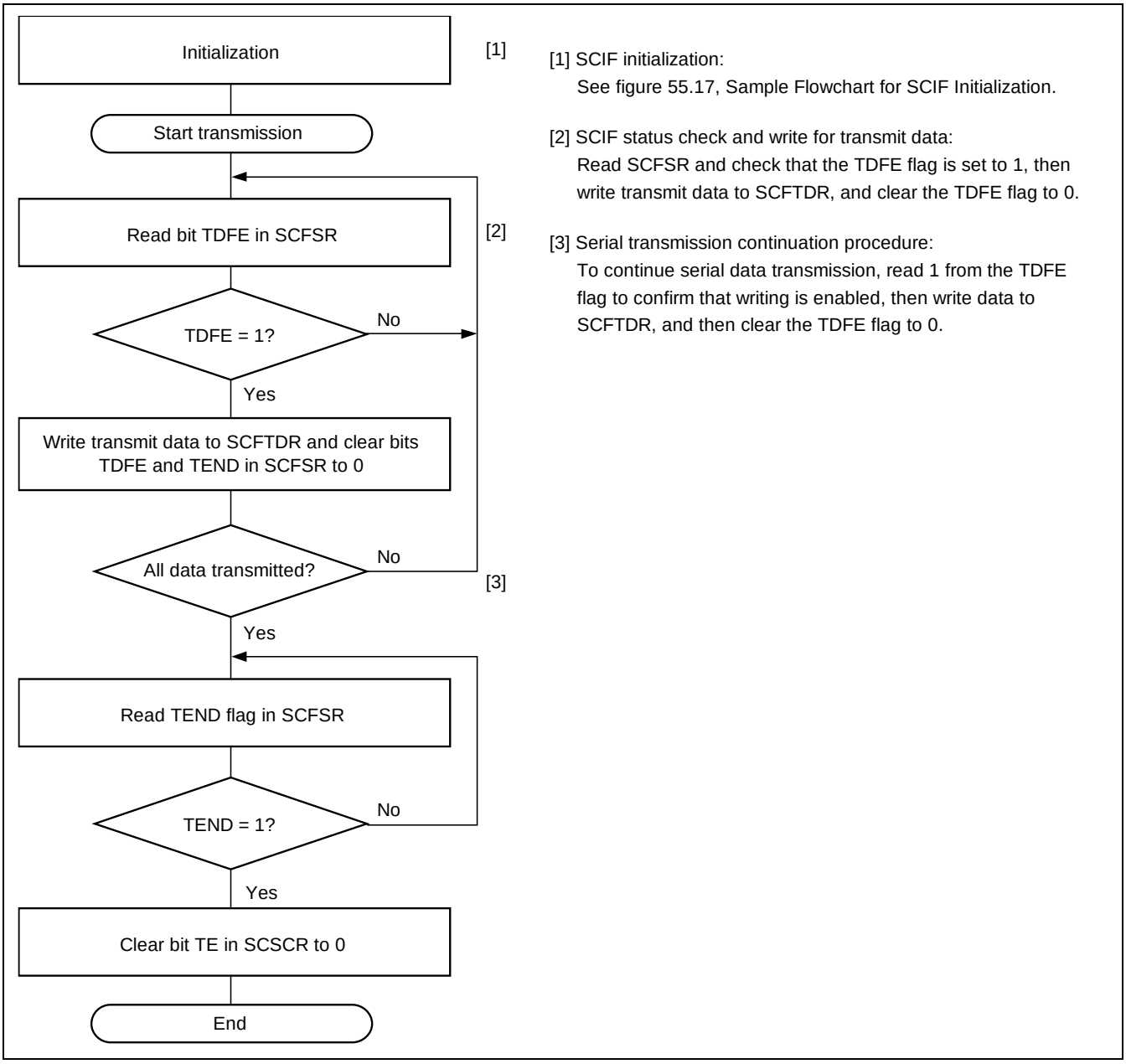



**Figure 105.18 Sample Flowchart for Serial Transmission**

In serial data transmission, the SCIF operates as described below:

1. When data is written to SCFTDR, the SCIF transfers the data from SCFTDR to SCTSR to start transmission. Confirm that the TDFE flag in SCFSR is set to 1 before writing transmit data to SCFTDR. The number of data bytes that can be written is at least (16- (the transmit trigger count)).
2. After the SCIF transfers data from SCFTDR to SCTSR and starts data serial transmission, the SCIF consecutively transmits it until there is no transmit data left in SCFTDR. When the number of transmit data bytes in SCFTDR is equal to or less than the transmit trigger count set in SCFCR, the TDFE flag is set. If the TIE and TEIE bits in SCSCR are set to 1 and 0, respectively at this time, a transmit-FIFO-data-empty interrupt (TDFE) request occurs. When the external clock is selected, data is output in synchronization with the input clock. The serial transmit data is output at the TX pin in the LSB-first order.
3. The SCIF checks transmit data in SCFTDR when sending the last bit. If there is transmit data, the SCIF transfers the data to SCTSR, and starts serial transmission of the next frame. If there is no transmit data left, the TEND flag in SCFSR is set to 1, and the transmit data pin (TX pin) retains its state after the last bit is sent. At this time, if the TIE and TEIE bits in SCSCR are set to 1, a transmit-end interrupt (TEND) request occurs.
4. After serial transmission ends, the SCK pin is fixed high.

Note: In clock synchronous mode, when transmit data is written to SCFTDR by the DMAC, the TEND flag may not be cleared. Therefore, if the DMAC is used for transmission in clock synchronous mode, read the TEND flag in the following method.
   1. Confirm that data transfer is completed in the DMAC.
   2. Read the TEND flag.
   3. Clear the TEND flag to 0 if TEND = 1.
   4. Read the TEND flag again.
   5. Use the second-read TEND flag.

Figure 105.19 shows an example of SCIF serial transmission in clock synchronous mode.

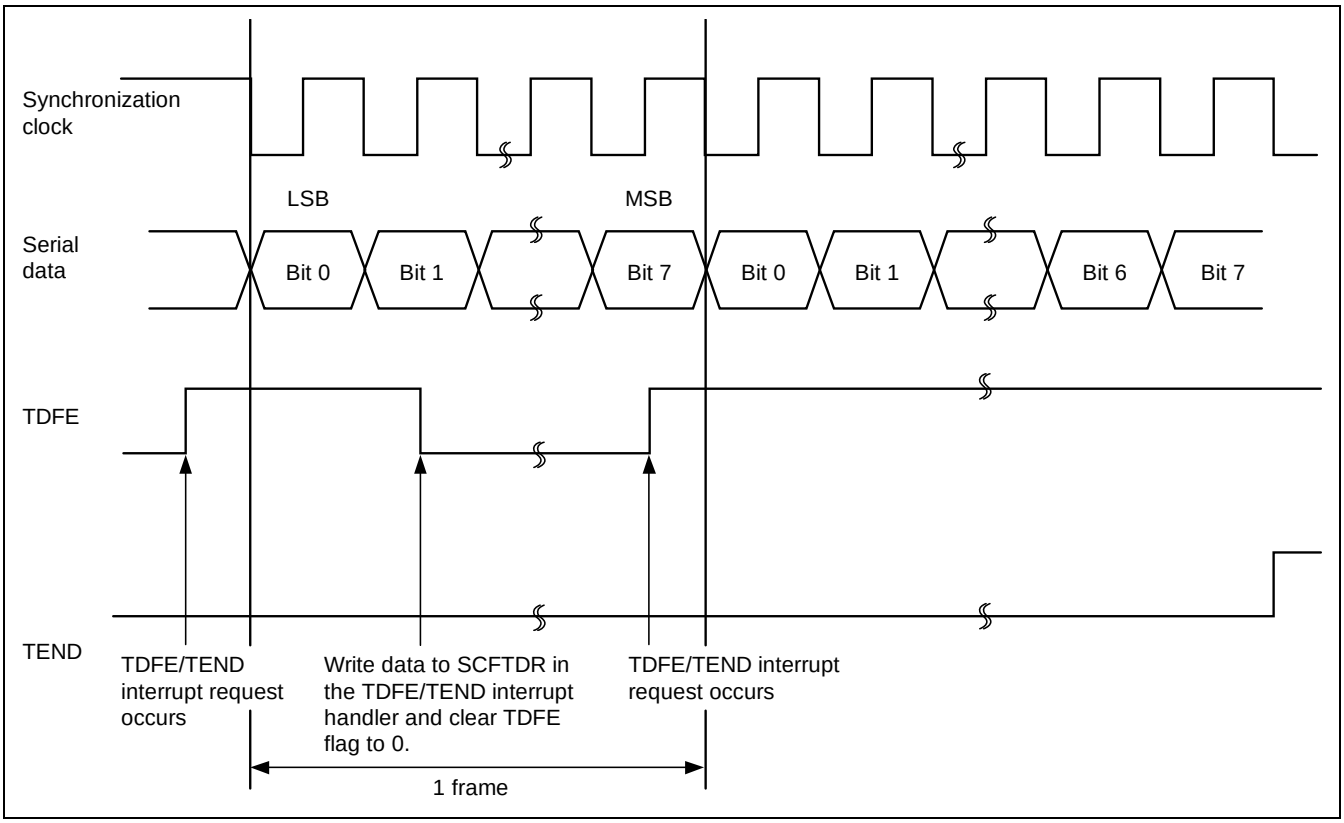



**Figure 105.19 Example of SCIF Serial Transmission in Clock Synchronous Mode**

**(c) Serial Data Reception (Clock Synchronous Mode)**

Figure 105.20 and 105.21 show sample flowcharts for serial reception.

The SCIF reception should be enabled before taking the following steps for serial data transmission.

When switching operating mode from asynchronous mode to clock synchronous mode without initializing the SCIF, check that the ORER, PER3 to PER0, and FER3 to FER0 flags are cleared to 0.

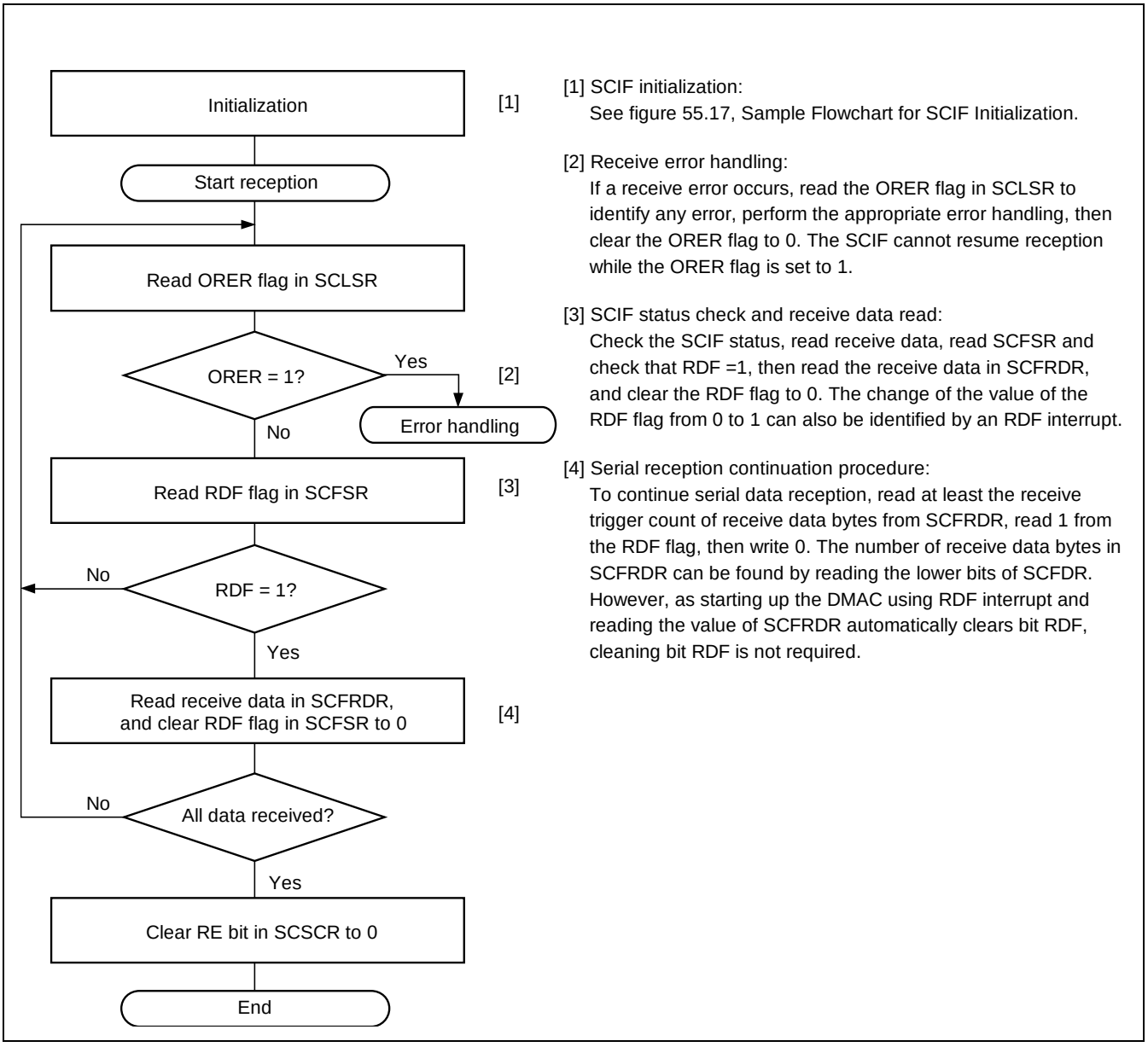



**Figure 105.20 Sample Flowchart for Serial Data Reception**

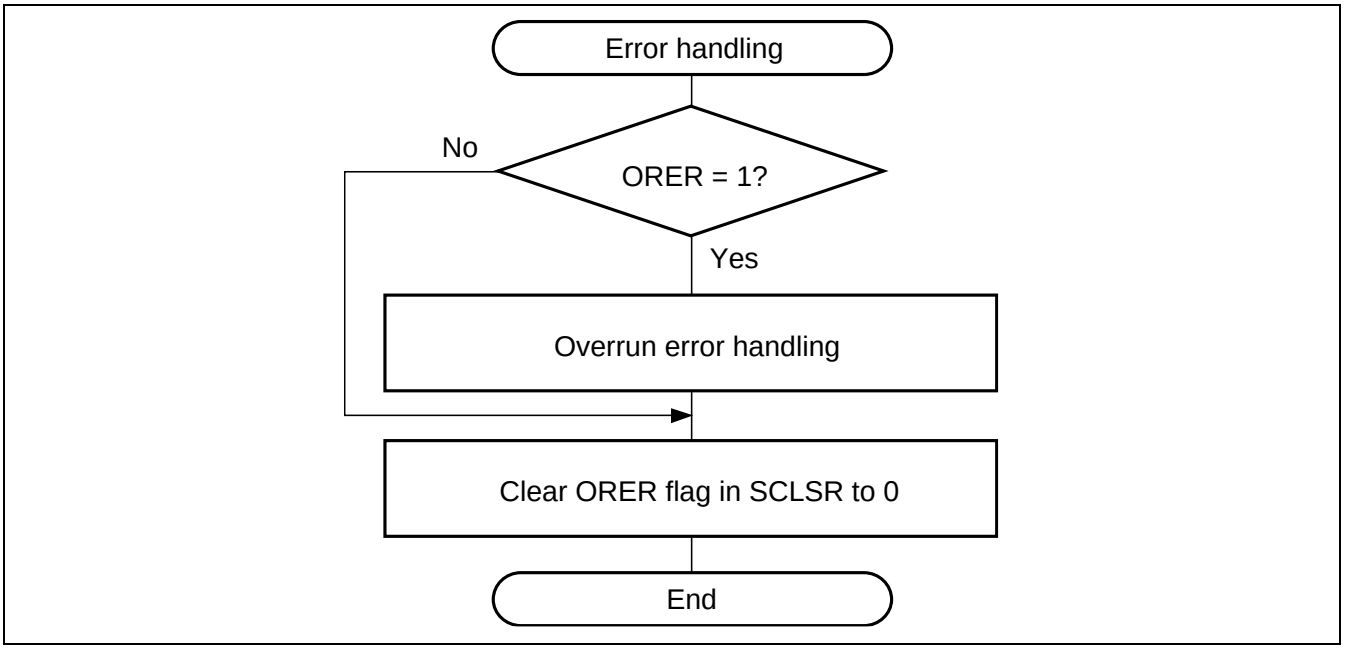



**Figure 105.21 Sample Flowchart for Serial Data Reception**

In serial data reception, the SCIF operates as described below:

1. The SCIF starts receiving data in synchronization with synchronization clock input or output.
2. The SCIF stores receive data in SCRSR in LSB-to-MSB order.
   After receiving the data, the SCIF first checks if the receive data can be transferred from SCRSR to SCFRDR, then starts storing the receive data in SCFRDR.
   If the SCIF detects an overrun error, the SCIF cannot receive subsequent data.
3. If the RDF flag changes to 1 while the RIE bit in SCSCR is 1, a receive-FIFO-data-full interrupt (RDF) request occurs. If the ORER flag changes to 1 while the RIE or REIE bit in SCSCR is 1, a break interrupt (BRI) request occurs.

Figure 105.22 shows an example of SCIF reception in clock synchronous mode.

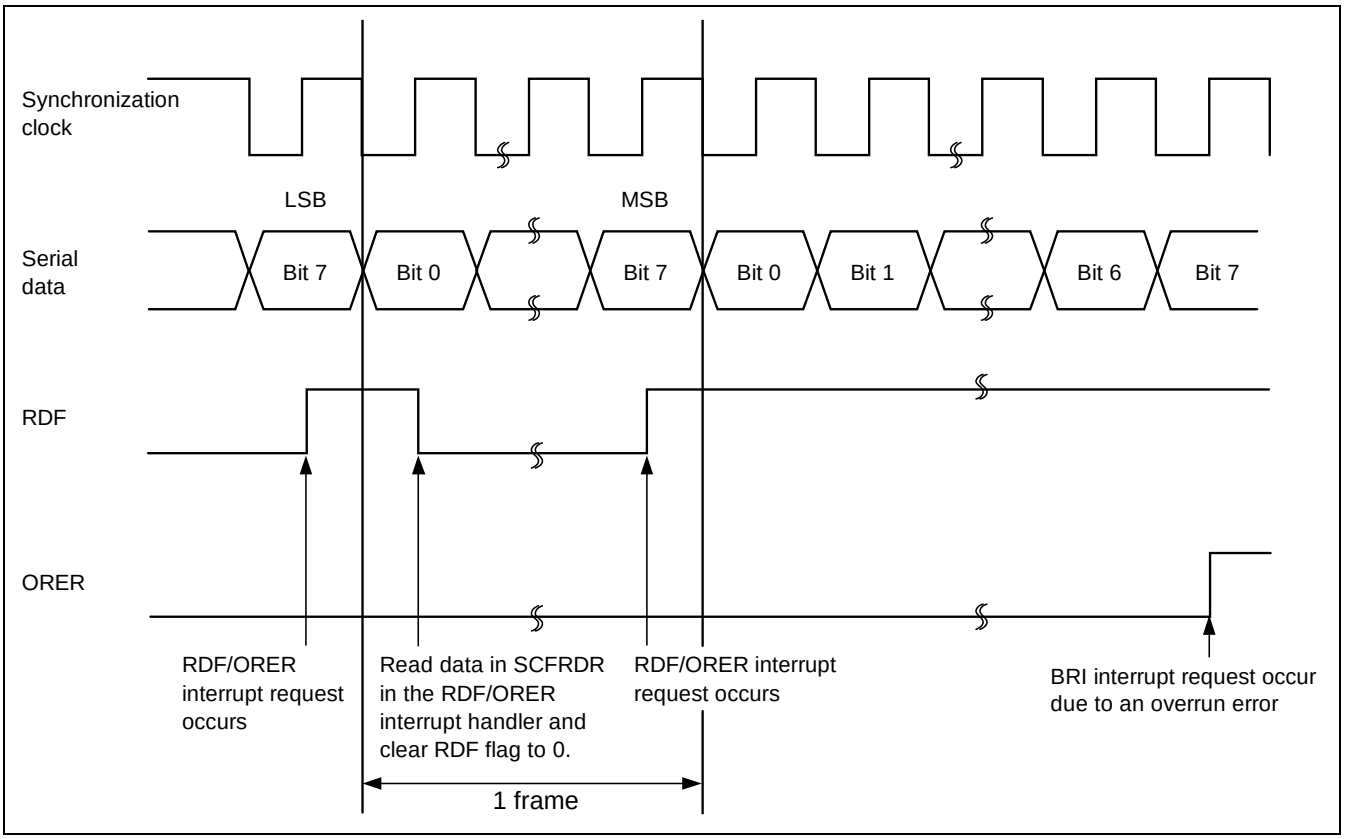
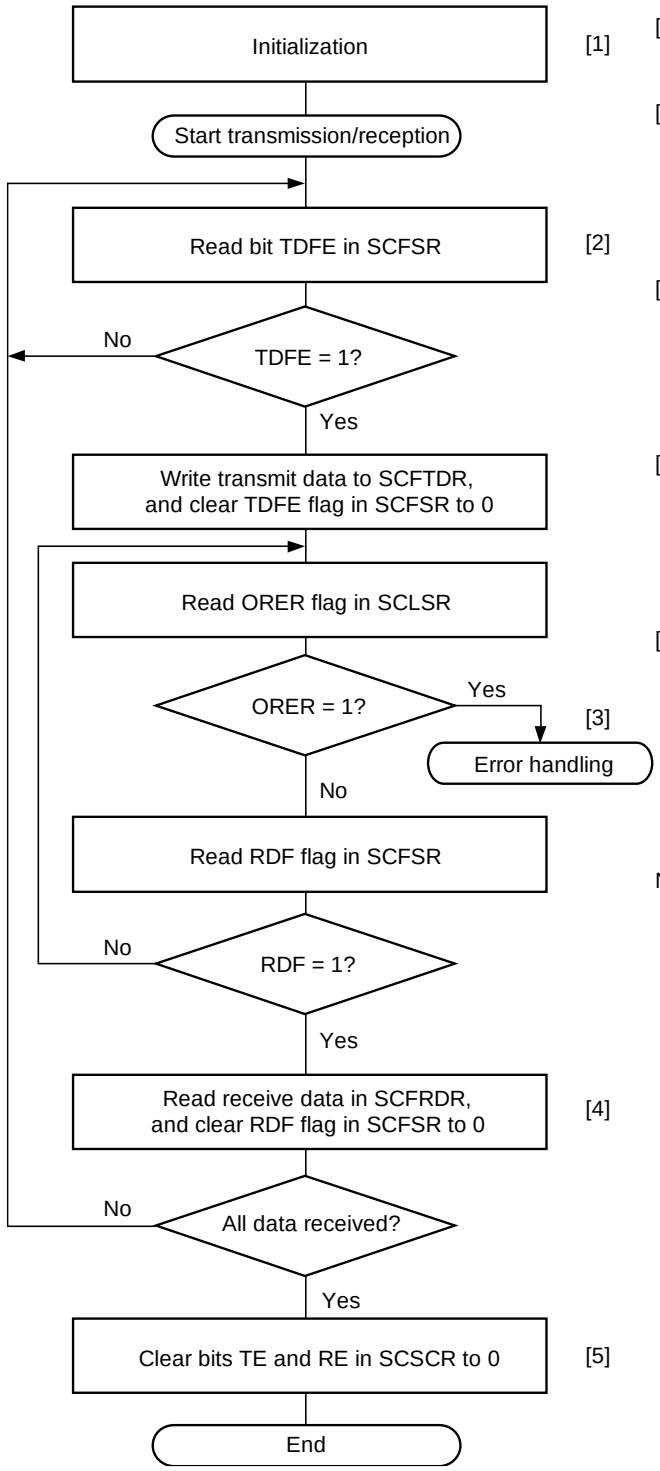



**Figure 105.22 Example of SCIF Reception in Clock Synchronous Mode**

**(d) Simultaneous Serial Data Transmission/Reception (Clock Synchronous Mode)**

Figure 105.23 shows a sample flowchart for simultaneous serial data transmission/reception.

The SCIF transmission and reception should be enabled before taking the following steps for simultaneous serial data transmission/reception.




[1] SCIF initialization:
See figure 55.17, Sample Flowchart for SCIF Initialization.

[2] SCIF status check and write for transmit data:
Read SCFSR and check that the TDFE flag is set to 1, then write transmit data to SCFTDR, and clear the TDFE flag to 0. The change of the value of the TDFE flag from 0 to 1 can also be identified by a TDFE/TEND interrupt.

[3] Receive error handling:
If a receive error occurs, read the ORER flag in SCLSR, perform the appropriate error handling, then clear the ORER flag to 0. The SCIF cannot resume reception while the ORER flag is set to 1.

[4] SCIF status check and read for receive data:
Read SCFSR and check that RDF =1, then read the receive data in SCFRDR, and clear the RDF flag to 0. The change of the value of the RDF flag from 0 to 1 can also be identified by an RDF interrupt.

[5] Serial transmission and reception continuation procedure:
To continue serial data transmission and reception, read RDF flag and SCFRDR, and finish clearing the RDF flag to 0 before receiving the MSB in the current frame.
Similarly, read 1 from the TDFE flag to confirm that writing is enabled before transmitting the MSB in the current frame.
Then write data to SCFTDR and clear the TDFE flag to 0.

Note : When switching from transmission or reception to simultaneous transmission and reception, clear bits TE and RE to 0, and then set them to 1 simultaneously.

**Figure 105.23 Sample Flowchart for Simultaneous Serial Data Transmission/Reception**

### 105.3.3 Generation of Internal Clock

Internal clock generated by the on-chip baud rate generator.
Baud rate is determined by the SCBRR and CKS[1:0] bit in SCSMR.

[Asynchronous mode]

$$N = \frac{PCK}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

[Clock synchronous mode]

$$N = \frac{PCK}{8 \times 2^{2n-1} \times B} \times 10^6 - 1$$

B: Bit rate (bit/s)

N: SCBRR setting for the baud rate generator ($0 \leq N \leq 255$) (which satisfies the electrical characteristics)

PCK: Peripheral module operating frequency (MHz)

n: Baud rate generator input clock (n = 0, 1, 2, 3)
(See the Table 105.8 for the relation between n and the clock.)

**Table 105.8 SCSMR Settings**

| n | Clock | SCSMR.CKS[1:0] Setting |
|---|---|---|
| 0 | PCK | 00 |
| 1 | PCK/4 | 01 |
| 2 | PCK/16 | 10 |
| 3 | PCK/64 | 11 |

The bit rate error in asynchronous mode is determined by the following equation:

$$\text{error (\%)} = \left\{ \frac{PCK \times 10^6}{(N+1) \times B \times 64 \times 2^{2n-1}} - 1 \right\} \times 100$$

### 105.3.4 Baud Rate Generator for External Clock (BRG)

#### (1) Overview

The SCIF incorporates a baud rate generator for external clock (abbreviated as BRG, hereafter). The BRG supplies a sampling clock (BRGCLK) to the SCIF core by dividing the external clock SC_CLK (selectable between SCIF_CLK and SCKi) by 1 to 216 – 1. In addition, the BRG switches the output between the external clock SCK and divided clock.

Figure 105.24 shows a block diagram of the BRG.

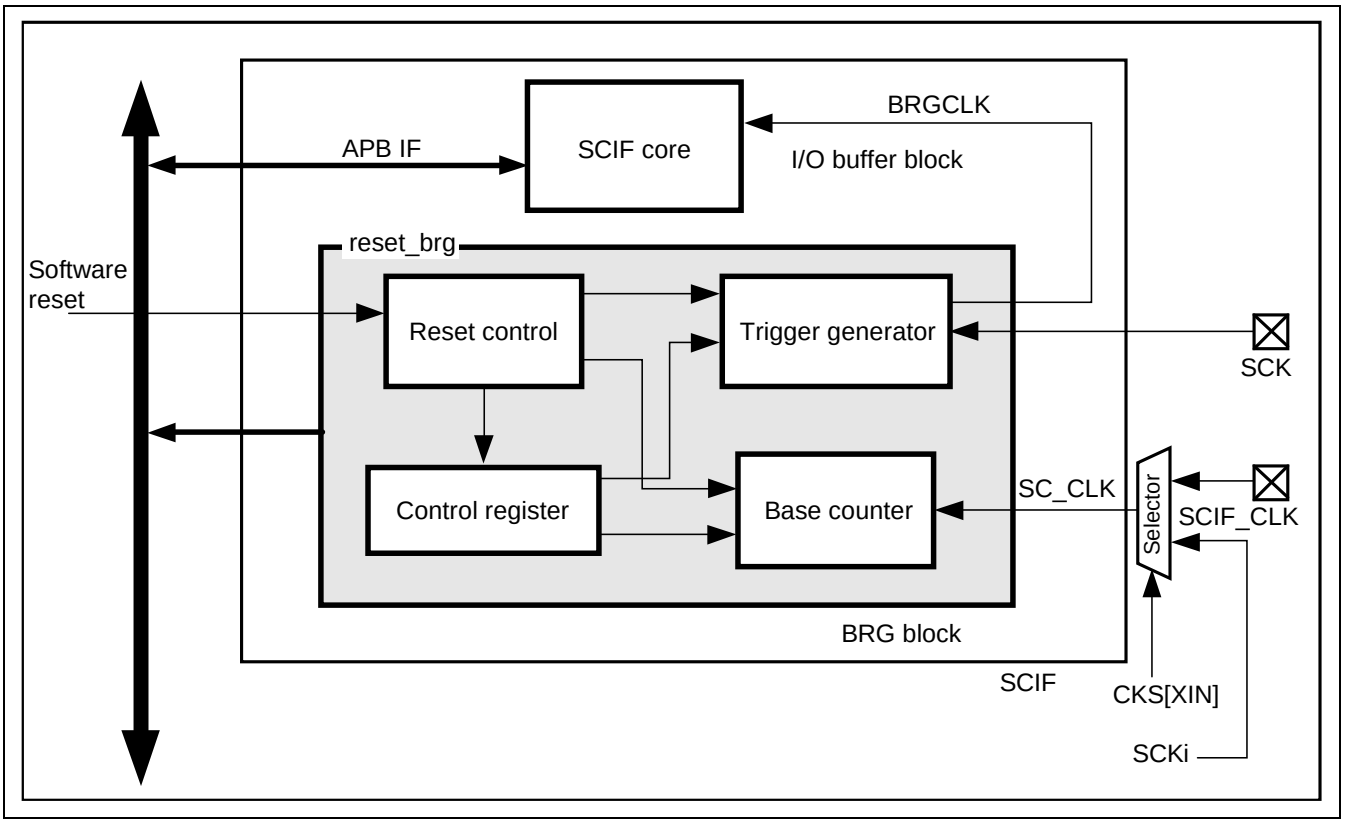



**Figure 105.24 BRG Block Diagram**

(a) Reset Controller:

The reset controller handles resetting the control register, base counter, and trigger generator.

(b) Control Register:

The control register has the frequency division register and clock select register.

(c) Base Counter:

The base counter is a 16-bit CLK synchronization counter that is used to determine the timing for generating a frequency divided clock.

(d) Trigger Generator:

The trigger generator generates rising-edge/falling-edge triggers for a frequency divided clock with the timing according to values of the frequency division register and base counter. The triggers are used to generate the frequency divided clock. In addition, the trigger generator switches the output between the SCK (external clock input) and frequency divided clock.

**(2) Registers**

To set baud rate requires setting to Frequency Division Register (DL) and Clock Select Register (CKS).

**(3) Frequency Division Register (DL)**

To set a division value of frequency requires setting to Frequency Division Register (DL).

Refer to Frequency Division Register (DL).

Table 105.9 shows reference values of Frequency Division Register (DL).

**Table 105.9** Baud Rate (3.6864-MHz clock)

| Baud Rate | Value of Frequency Division | Error Rate (*) |
|---|---|---|
| 50 | 4608 | — |
| 75 | 3072 | — |
| 110 | 2095 | –0.022 |
| 134.5 | 1713 | 0.001 |
| 150 | 1536 | — |
| 300 | 768 | — |
| 600 | 384 | — |
| 1200 | 192 | — |
| 1800 | 128 | — |
| 2000 | 115 | 0.174 |
| 2400 | 96 | — |
| 3600 | 64 | — |
| 4800 | 48 | — |
| 7200 | 32 | — |
| 9600 | 24 | — |
| 14400 | 16 | — |
| 19200 | 12 | — |
| 38400 | 6 | — |
| 76800 | 3 | — |
| 115200 | 2 | — |

Note: —: Indicates that the error rate is 0.

### (4) Clock Select Register (CKS)

To set output between the frequency divided clock and specifies a source clock for the external baud rate requires setting to Clock Select Register (CKS).

Refer to Clock Select Register (CKS).

### (5) Restrictions in BRG

**Notes on Setting Frequency Division Register**

1. For the initial setting of the register after a reset, at least one bit of waiting period is required to secure the clock stabilization time.
   (Example) One bit period when DL = 2

   $3.68\ (\text{MHz}) \times 1/2 \times 1/16 = 0.115\ (\text{MHz}) \rightarrow 8695\ (\text{ns})$

2. For modifying the register value after the setting stated in <1> above, at least one bit of waiting period at the maximum bit rate (DL = '65535') is required.
   The SCIF registers and BRG registers should be set as the following table:

- Asynchronous mode (SC_CLK external input)

| SCIF | | BRG | |
|---|---|---|---|
| **Register Name** | **Setting Value** | **Register Name** | **Setting Value** |
| SCSMR.C/A# | B’0 | CKS | H’0000 |
| SCSCR.CKE[1:0] | B’10 | DL | H’0001 to H’FFFF |

- Asynchronous mode (SCK external input)

| SCIF | | BRG | |
|---|---|---|---|
| **Register Name** | **Setting Value** | **Register Name** | **Setting Value** |
| SCSMR.C/A# | B’0 | CKS | H’8000 |
| SCSCR.CKE[1:0] | B’10 | DL | Don’t care |

- Clock synchronous mode (external input)

| SCIF | | BRG | |
|---|---|---|---|
| **Register Name** | **Setting Value** | **Register Name** | **Setting Value** |
| SCSMR.C/A# | B’1 | CKS | H’8000 |
| SCSCR.CKE[1:0] | B’10 | DL | Don’t care |

- Asynchronous mode (SC_CLK internal input)

| SCIF | | BRG | |
|---|---|---|---|
| **Register Name** | **Setting Value** | **Register Name** | **Setting Value** |
| SCSMR.C/A# | B’0 | CKS | H’4000 |
| SCSCR.CKE[1:0] | B’10 | DL | H’0001 to H’FFFF |

3. The register settings for the baud rate generator for external clock should be made before starting initialization of the SCIF.

### 105.3.5 SCIF Interrupt Sources and the DMAC

If the DMAC is used for transmission/reception, set and enable the DMAC before setting the SCIF.

**Transmission Interrupts and DMA Transfer:**

If the TDFE/TEND flag in SCFSR is set to 1 when the TDFE/TEND interrupt is enabled by the TIE bit, a TDFE/TEND interrupt request and a transmit-FIFO-data-empty DMA transfer request will occur. If the TDFE/TEND flag is set to 1 when TDFE/TEND interrupt is disabled by the TIE bit, only the transmit-FIFO-data-empty DMA transfer request will occur. (A transmit-FIFO-data-empty DMA transfer request is generated when the TDFE flag is set while TEIE is 0, or when the TEND flag is set while TEIE is 1. DMA transfer requests are not affected by the TEIE bit.)

When TDFE/TEND interrupt requests are enabled, the interrupt requests are cleared by the DMAC regardless of the interrupt handling program.

**Reception Interrupts and DMA Transfer:**

If the RDF/DR flag in SCFSR is set to 1 when RDF/DR interrupt is enabled by the RIE bit, an RDF/DR interrupt request occurs. If the RDF/DR flag is set to 1, a receive-FIFO-data-full DMA transfer request occurs. If the RDF/DR flag is set to 1 when RDF/DR interrupt is disabled by the RIE bit, and only a receive-FIFO-data-full DMA transfer request occurs and DMAC can be activated to perform data transfer.

Setting the RIE bit in SCSCR to 0 and the REIE it to 1 generates the ER/BRK/ORER interrupt requests without generating RDF/DR interrupt requests. When the BRK flag in SCFSR or the ORER flag in SCLSR is set to 1, BRK/ORER interrupt requests occur.

If the TO flag is set to 1 in SCLSR when TO interrupts are enabled by the TOIE bit, TO interrupt requests occur.

DR/TO interrupt requests generated by setting the DR or TO flag to 1 or receive-FIFO-data-full DMA transfer requests occur only in asynchronous mode.

When DR/TO interrupt requests are enabled to be issued, interrupt requests generated by the DR flag are cleared by the DMAC regardless of the interrupt handling program, however, those generated by the TO flag are not cleared by the DMAC. Therefore, the TO flag interrupt requests need to be cleared with the interrupt handling program. (The DR and TO flags are set at the same time, but cleared separately.)

**Table 105.10 SCIF Interrupt Sources**

| Interrupt Source | DMAC Activation | Priority on Reset Release |
|---|---|---|
| Interrupts generated by receive error flag (ER) | Disabled | High |
| Interrupts generated by receive-FIFO-data-full (RDF), receive-data-ready (DR) or timeout (TO) * | Enabled | ↕ |
| Interrupts generated by break (BRK) or overrun error (ORER) | Disabled | |
| Interrupts generated by transmit FIFO data empty (TDFE) | Enabled | Low |

Note: * RXI interrupts by means of the DR or TO flag are available only in asynchronous mode.

## 105.4 Usage Notes

### 105.4.1 Break Detection and Operation

Break signals can also be detected by reading the RX pin value directly when a framing error (FER) is detected. In the break state, the input values from the RX pin are all 0s. So, the parity error flag (PER) may be set after the FER flag is set to 1.

Although the SCIF stops receive data transfer to SCFRDR after detecting a break, it continues data reception.

### 105.4.2 Sending a Break Signal

The input/output condition and level of the TX pin are determined by the SPB2IO and SPB2DT bits in SCSPTR. This enables to send a break signal.

The pin does not function as the TX pin from the initialization of the serial transmitter to setting of the TE bit (enabling transmission). In this period, the marked state is substituted by the value of the SPB2DT bit. Therefore, the SPB2IO and SPB2DT bits should be set to 1 (designating output and high level) beforehand.

To send a break signal during serial transmission, clear the SPB2DT bit to 0 (low level), and then clear the TE bit to 0 (halting transmission). When the TE bit is cleared, the transmitter is initialized regardless of the current transmission state, and the TX pin outputs 0.

### 105.4.3 Data Sampling Timing and Reception Margin in Asynchronous Mode

In asynchronous mode, the SCIF operates on the base clock with a frequency 16 times the bit rate.

In reception, the SCIF performs the internal synchronization by sampling the fall edge of the start bit using the base clock. In addition, the SCIF takes receive data at the rising edge of the eighth pulse on the base clock.

Figure 105.25 shows the timing of this operation.

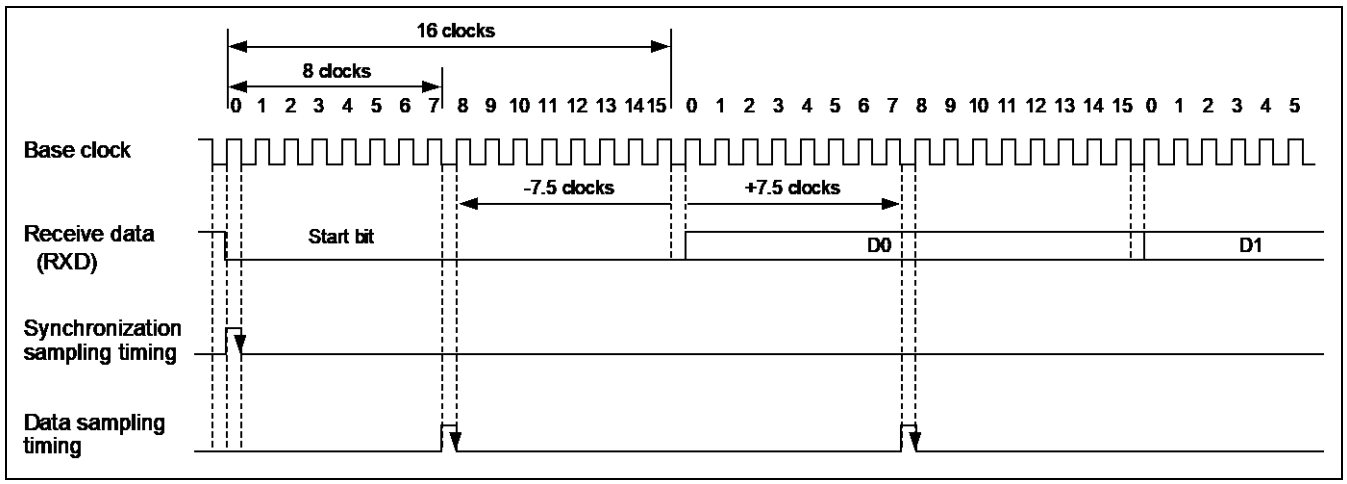



**Figure 105.25 Timing Chart of Receive Data Sampling**

The reception margin in asynchronous mode is given by formula (1).

$$M = \left| (0.5 - \frac{1}{2N}) - (L - 0.5)F - \frac{|D - 0.5|}{N}(1 + F) \right| \times 100\% \quad \text{..... Expression (1)}$$

M: Receive margin (%)

N: Ratio of bit rate to clock (N = 16)

D: Clock duty (D = 0 to 1.0)

L: Frame length (L = 9 to 12)

F: Absolute value of clock frequency deviation

Assuming F = 0 and D = 0.5 for expression (1), the reception margin obtained with expression (2) is 46.875% as shown below:

Assuming D = 0.5 and F = 0

M = (0.5 − 1/ (2 × 16)) × 100%

= 46.875% ……expression (2)

This is a theoretical value. A reasonable margin allowed in system designs is 20% to 30%.

### 105.4.4 Reception Margin and Baud Rate Error

The value of 46.875% obtained by the above expression (2) indicates the reception margin when the baud rate error is 0 (F = 0). If there is no error in the reception and transmission baud rates, reception is possible even with misalignment of approx. 1/2 bit. If there is an error in the reception and transmission baud rates, the errors are accumulated up to the stop bit reception, which reduces the reception margin. The allowable baud rate error can be obtained by modifying the F in expression (1). When D= 0.5:

F = {(15/32 − M)/ (L − 0.5)} × 100 (%)……expression (3)

By using expression (3), the relationship between the allowable error and reception margin with the frame length = 12 can be summarized as follows:

| Allowed Error (%) | Reception Margin (%) |
|---|---|
| 4.07 | 0 |
| 3.64 | 5 |
| 3.20 | 10 |
| 2.33 | 20 |
| 1.46 | 30 |

## 105.5 Safety Mechanisms

SCIF module has no mechanism of FuSa.

# 106. High Speed Serial Communication Interface with FIFO (HSCIF)

## 106.1 Overview

The fourth generation R-Car series products have a high speed serial communication interface with built-in FIFO buffers (high-speed serial communication interface with FIFO: HSCIF) that handles asynchronous communication. The HSCIF has two 128-stage FIFO buffers separately for transmission and reception, which enables fast, efficient, and uninterrupted communication. R-Car V4M has 4 channels of HSCIF. All functions of each channel are same.

### 106.1.1 Features

The HSCIF has the following features.

- Asynchronous serial communication mode

  The HSCIF performs serial data communication based on a character-by-character asynchronous system. This feature enables serial data communication with standard asynchronous communication chips that support universal asynchronous receiver/transmitter (UART) or asynchronous communication interface adapter (ACIA). There is a choice of eight serial data transfer formats.
  - Data length: 7 or 8 bits
  - Stop bit length: 1 or 2 bits
  - Parity: Even/odd/none
  - Receive error detection: Parity, framing, and overrun errors
  - Break detection:

    A break is detected when a framing error lasts for more than 1 frame length at space 0 (low level).

    When a framing error occurs, a break can also be detected by reading the HRX pin level directly from the serial port register (HSSPTR).
  - Sampling rate: Variable (integer number from 8 to 32)
- Capable of full-duplex communication

  The HSCIF has an independent transmitter and receiver that enable simultaneous transmission and reception. The transmitter and receiver both have a 128-stage FIFO buffer structure, enabling continuous serial data transmission and reception.
- On-chip baud rate generator, enabling any bit rate to be selected

  The HSCIF enables choice of a clock source for transmission/reception: a clock from the on-chip baud rate generator based on the internal clock or an external clock.
- Eight interrupt sources

  The HSCIF has eight types of interrupt sources: receive-data-ready, receive-FIFO-data-full, break detection, transmit-FIFO-data-empty, transmit-end, receive-error, overrun-error and time-out, and enables any of them to be requested independently.
- DMA data transfer

  When the transmit FIFO register is empty or the receive FIFO register has received data, issuing a DMA transfer request activates the DMA controller (DMAC) to execute a data transfer.
- Modem control functions (HRTS# and HCTS#) are stored.
- The amount of data in the transmit/receive FIFO registers and the number of receive errors in receive data in the receive FIFO register are available.

A receive data ready (DR) or a timeout error (TO) can be detected during reception.

### 106.1.2 Block Diagram

Figure 106.1 shows the HSCIF block diagram.

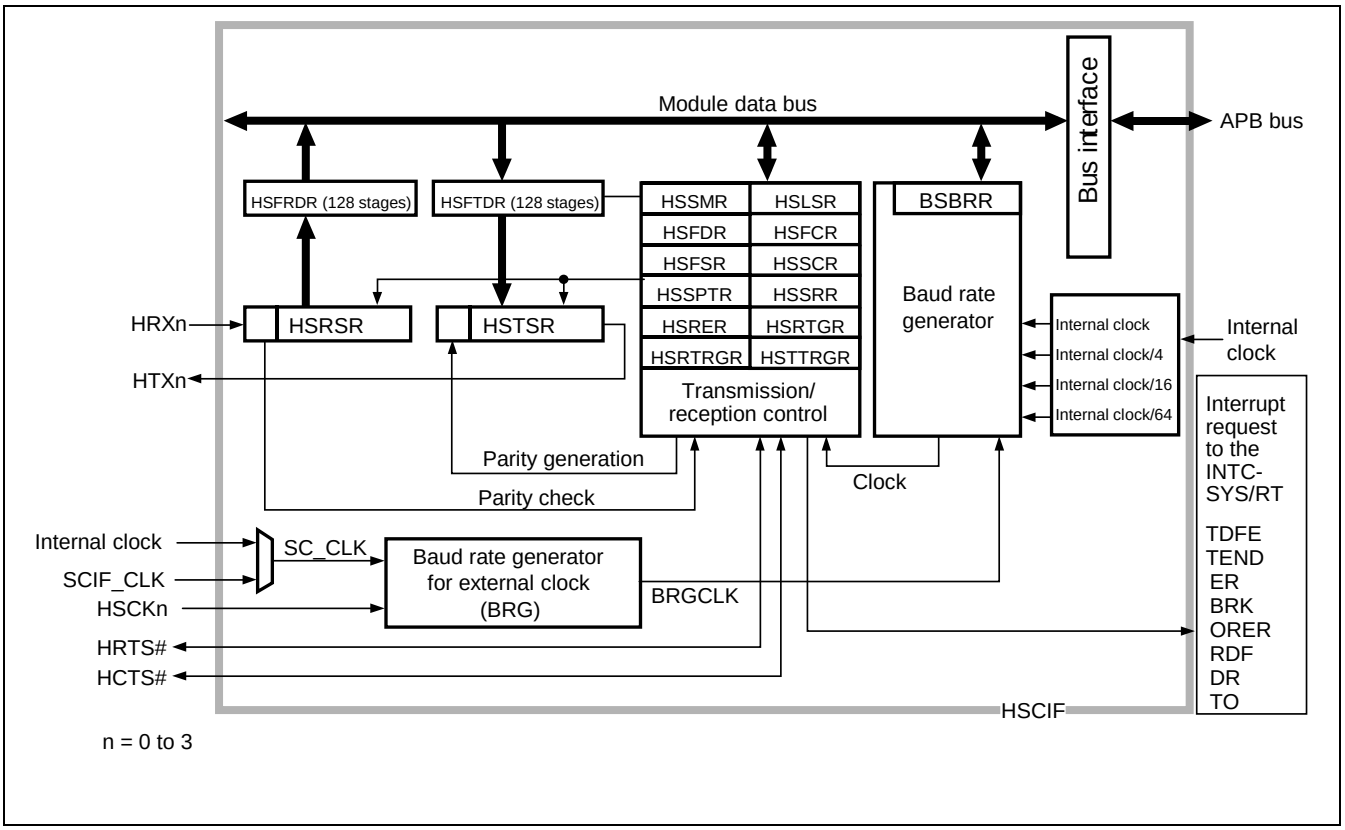



**Figure 106.1 HSCIF Block Diagram**

[Legend]

HSRSR: Receive shift register
HSFRDR: Receive FIFO data register
HSTSR: Transmit shift register
HSFTDR: Transmit FIFO data register
HSSMR: Serial mode register
HSSCR: Serial control register
HSFSR: Serial status register
HSBRR: Bit rate register
HSSPTR: Serial port register
HSFCR: FIFO control register
HSFDR: FIFO data count register
HSLSR: Line status register
HSSRR: Sampling rate register
HSRER: Serial error register
HSRTGR: RTS output active trigger register
HSRTRGR: Receive FIFO data count trigger register
HSTTRGR: Transmit FIFO data count trigger register

Clock for “Bus interface” = SASYNCPERD2φ
Internal Clock = SASYNCPERD1φ

### 106.1.3 External Pins

Table 106.1 shows the HSCIF pin configuration. These pins are multiplexed in other functions, so that the usage of the pins are may be restricted depending on the multiplexed pin settings.

**Table 106.1 Pin Configuration**

| Name | Abbreviation | Function | I/O | Descriptions |
|---|---|---|---|---|
| HSCK0 | HSCK | Serial clock pin | Input | Clock Input |
| HSCK1 | | Serial clock pin | Input | Clock Input |
| HSCK2 | | Serial clock pin | Input | Clock Input |
| HSCK3 | | Serial clock pin | Input | Clock Input |
| HRX0 | HRX | Receive data pin | Input | Receive data input |
| HRX1 | | Receive data pin | Input | Receive data input |
| HRX2 | | Receive data pin | Input | Receive data input |
| HRX3 | | Receive data pin | Input | Receive data input |
| HTX0 | HTX | Transmit data pin | Output | Transmit data output |
| HTX1 | | Transmit data pin | Output | Transmit data output |
| HTX2 | | Transmit data pin | Output | Transmit data output |
| HTX3 | | Transmit data pin | Output | Transmit data output |
| HCTS0# | HCTS# | Modem control pin | I/O | Transmission enabled |
| HCTS1# | | Modem control pin | I/O | Transmission enabled |
| HCTS2# | | Modem control pin | I/O | Transmission enabled |
| HCTS3# | | Modem control pin | I/O | Transmission enabled |
| HRTS0# | HRTS# | Modem control pin | I/O | Transmission request |
| HRTS1# | | Modem control pin | I/O | Transmission request |
| HRTS2# | | Modem control pin | I/O | Transmission request |
| HRTS3# | | Modem control pin | I/O | Transmission request |
| SCIF_CLK(ch0,1,3)<br>SCIF_CLK2(ch2) | SCIF_CLK | Baud rate generation clock pin | Input | Clock for input to the baud rate generator for the external clock |

Note: These pins are made to function as serial pins by setting up HSCIF operation using bits TE, RE, CKE[1:0] in HSSCR, and bit MCE in HSFCR. HSSPTR of the HSCIF can be used to handle the transmission and detection of break states.

### 106.1.4 Register Configuration

(include Safety Mechanism related registers)

Table 106.2 shows the registers in the HSCIF. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined.

**Table 106.2 Register Configuration**

| Register Name | Abbreviation | R/W | Value after Reset | Offset From Base Address | Access Size |
|---|---|---|---|---|---|
| Serial mode register | HSSMR | R/W | H'0000 | H'00 | 16 |
| Bit rate register | HSBRR | R/W | H'FF | H'04 | 8 |
| Serial control register | HSSCR | R/W | H'0000 | H'08 | 16 |
| Transmit FIFO data register | HSFTDR | —/WB | Undefined | H'0C | 8 |
| Serial status register | HSFSR | R/WC0*1 | H'0060 | H'10 | 16 |
| Receive FIFO data register | HSFRDR | R | Undefined | H'14 | 8 |
| FIFO control register | HSFCR | R/W | H'0000 | H'18 | 16 |
| FIFO data count register | HSFDR | R | H'0000 | H'1C | 16 |
| Serial port register | HSSPTR | R/W | Undefined*3 | H'20 | 16 |
| Line status register | HSLSR | R/WC0*2 | H'0000 | H'24 | 16 |
| Frequency division register | DL | R/W | H'00 | H'30 | 16 |
| Clock select register | CKS | R/W | H'00 | H'34 | 16 |
| Sampling rate register | HSSRR | R/W | H'000F | H'40 | 16 |
| Serial error register | HSRER | R | H'0000 | H'44 | 16 |
| RTS output active trigger register | HSRTGR | R/W | H'000F | H'50 | 16 |
| Receive FIFO data count trigger register | HSRTRGR | R/W | H'0001 | H'54 | 16 |
| Transmit FIFO data count trigger register | HSTTRGR | R/W | H'0008 | H'58 | 16 |

Notes: 1. Only 0 can be written to clear the flags. Bits 15 to 8, 3, and 2 are read-only bits and cannot be modified.
2. Only 0 can be written to clear the flags. Bits 15 to 3, and 1 are read-only bits and cannot be modified.
3. The initial values of HSSPTR bits 6, 4, 2, and 0 are undefined.

| Channel No. | Base Address |
|---|---|
| 0 | H'E654 0000 |
| 1 | H'E655 0000 |
| 2 | H'E656 0000 |
| 3 | H'E66A 0000 |

## 106.2 Register Description

(include Safety Mechanism related register description)

### 106.2.1 Receive Shift Register (HSRSR)

HSRSR is a register that receives serial data.

The HSCIF sets serial data input to the HSRSR from the HRX pin in the order received, starting with the LSB (bit 0), and converts it to parallel data. When one byte of data has been received, it is transferred to the receive FIFO register HSFRDR, automatically.

HSRSR cannot be read from and written to by the CPU.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | — | — | — | — | — | — | — | — |

### 106.2.2 Receive FIFO Data Register (HSFRDR)

HSFRDR is a 128-stage FIFO register that stores received serial data.

When HSCIF has received one byte of serial data, it transfers the received data from the receive shift register (HSRSR) to HSFRDR for storage, and reception is thus completed. HSRSR is then ready for reception, and is capable of receiving up to 128 consecutive bytes of data before HSFRDR is full.

HSFRDR is a read-only register and cannot be modified by the CPU. Note that the read value will be undefined while there is no receive data in HSFRDR. When HSFRDR is full of receive data, subsequent serial data is lost.

HSFRDR is read as an undefined value after a power-on reset.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R |

### 106.2.3 Transmit Shift Register (HSTSR)

HSTSR is a register that transmits serial data.

To perform serial data transmission, the HSCIF first transfers transmit data from the transmit fifo data register (HSFTDR) to HSTSR, and then sends the data to the HTX pin starting with the LSB (bit 0). When transmission of one byte is completed, the HSCIF transfers the next transmit data from HSFTDR to HSTSR automatically, then starts transmission.

HSTSR cannot be read from and written to directly by the CPU.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | — | — | — | — | — | — | — | — |

### 106.2.4 Transmit FIFO Data Register (HSFTDR)

HSFTDR is an 8-bit FIFO register of 128 stages that stores data for serial transmission.

If HSTSR is empty after transmit data has been written to HSFTDR, the HSCIF transfers the data from HSFTDR to HSTSR and starts serial transmission.

HSFTDR is a write-only register and cannot be read from by the CPU. Writing further data to HSFTDR is no longer possible when it is full. Attempts at writing data to the register in this situation are ignored.

HSFTDR is read as an undefined value on a power-on reset.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — |
| R/W: | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB |

### 106.2.5 Serial Mode Register (HSSMR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CHR | PE | O/E# | STOP | — | CKS[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R/W | R/W |

HSSMR is a 16-bit register that sets the HSCIF's serial transfer format and selects the baud rate generator clock source.

HSSMR can always be read from and written to by the CPU.

HSSMR is initialized to H'0000 by a power-on reset.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 | CHR | 0 | R/W | Character Length<br>Selects 7 or 8 bits for data length. When the 7-bit data is selected, the MSB (bit 7) in the transmit FIFO data register (HSFTDR) is not transmitted.<br>0: 8 bits<br>1: 7 bits |
| 5 | PE | 0 | R/W | Parity Enable<br>Determines whether parity bit is added in transmission or not, and parity bit is checked in reception or not. When bit PE is set to 1, the parity (even or odd) specified by bit O/E# is added to transmit data. In reception, the parity bit is checked for the parity (even or odd) specified by bit O/E#.<br>0: Disables parity bit addition and check.<br>1: Enables parity bit addition and check. |
| 4 | O/E# | 0 | R/W | Parity Mode<br>Selects either even or odd parity to use in parity addition and check.<br>The O/E# bit setting is valid only when bit PE is set to 1, enabling parity bit addition and check.<br>0: Even parity<br>1: Odd parity<br>When even parity is set, the parity bit is added in transmission so that the total number of 1-bit in the transmit character plus the parity bit is even. In reception, a check is performed to see if the total number of 1-bit in the receive character plus the parity bit is even.<br>When odd parity is set, the parity bit is added in transmission so that the total number of 1-bit in the transmit character plus the parity bit is odd. In reception, a check is performed to see if the total number of 1-bit in the receive character plus the parity bit is odd. |
| 3 | STOP | 0 | R/W | Stop Bit Length<br>Selects 1 bit or 2 bits as the stop bit length.<br>In reception, only the first stop bit is checked, regardless of the STOP bit setting. If the second stop bit is 1, it is treated as a stop bit. If it is 0, it is treated as the start bit of the next transmits character.<br>0: 1 stop bit*1<br>1: 2 stop bits*2<br>Notes: 1. In transmission, high level of 1 bit (stop bit) is added to the end of a transmit character before it is sent.<br>2. In transmission, high level of 2 bits (stop bits) are added to the end of a transmit character before it is sent. |
| 2 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1, 0 | CKS[1:0] | 00 | R/W | Clock Select 1 and 0<br>These bits select the clock source for the on-chip baud rate generator.<br>The clock source can be selected from Internal clock, Internal clock/4, Internal clock/16, and Internal clock/64, according to the setting of bits CKS [1:0].<br>00: Internal clock<br>01: Internal clock/4<br>10: Internal clock/16<br>11: Internal clock/64 |

### 106.2.6 Serial Control Register (HSSCR)

HSSCR is a register that enables or disables transmission/reception by the HSCIF, enables or disables interrupt requests, and selects transmission/reception clock source for the HSCIF.

HSSCR can always be read from and written to by the CPU.

HSSCR is initialized to H'0000 by a power-on reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TOT[1:0] | | — | — | TEIE | — | — | — | TIE | RIE | TE | RE | REIE | TOIE | CKE[1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R | R | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15, 14 | TOT[1:0] | 00 | R/W | Set the time for a data ready (DR) or a timeout (TO) to be set in asynchronous mode.<br>00: 15 etu*<br>01: 31 etu<br>10: 47 etu<br>11: 63 etu<br>Note: * ETU: Elementary Time Unit (time for transfer of one bit) Equivalent to 1.5 frames with an 8-bit, 1-stop-bit format. |
| 13, 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 | TEIE | 0 | R/W | Transmit End Interrupt Enable<br>When a transmit-end request is enabled by the TIE bit, the TEIE bit selects the source of the transmit end interrupt request from the following:<br>• Setting the TDFE flag in HSFSR<br>• Setting the TEND flag in HSFSR<br>0: The transmit FIFO data empty (TDFE) interrupt request is used.<br>1: The transmit end (TEND) interrupt request is used. |
| 10 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | TIE | 0 | R/W | Transmit Interrupt Enable<br>Enables or disables a transmit-FIFO-data-empty interrupt (TDFE) request when the TEIE bit in HSSCR is pulled 0, if all of the following conditions are satisfied:<br>• Serial transmit data has been transferred from HSFTDR to HSTSR.<br>• The number of data bytes in HSFTDR is equal to or less than the transmit trigger count.<br>• The TDFE flag in HSFSR is set to 1.<br>Enables or disables a transmit-end interrupt (TEND) request when the TEIE bit of HSSCR is set to 1, if both of the following conditions are satisfied:<br>• Transmission was ended because there is no valid data in HSFTDR when the last bit of the transmit character in HSTSR was transmitted.<br>• The TEND flag of HSFSR is set to 1.<br>0: When the TEIE bit is 0, disables transmit-FIFO-data-empty interrupt (TDFE) request.<br>When the TEIE bit is 1, disables transmit-end interrupt (TEND) request.<br>1: When the TEIE bit is 0, enables transmit-FIFO-data-empty interrupt (TDFE) request.<br>When the TEIE bit is 1, enables transmit-end interrupt (TEND) request. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | RIE | 0 | R/W | Receive Interrupt Enable<br>Enables or disables a receive-FIFO-data-full interrupt request when the RDF flag in HSFSR is set to 1, a receive-data-ready interrupt request when the DR flag in HSFSR is set to 1, a receive-error interrupt request when the ER flag in HSFSR is set to 1, a break interrupt request when the BRK flag in HSFSR is set to 1, and an overrun error interrupt request when the ORER flag in HSLSR is set to 1.<br>0: Disables receive-FIFO-data-full interrupts (RDF) requests, receive-data-ready interrupt (DR) requests, receive-error interrupt (ER) requests, break interrupt (BRK) requests, and overrun error interrupt (ORER) requests.<br>1: Enables receive-FIFO-data-full interrupt (RDF) requests, receive-data-ready interrupt (DR) requests, receive-error interrupt (ER) requests, break interrupt (BRK) requests, and overrun error interrupt (ORER) requests. |
| 5 | TE | 0 | R/W | Transmit Enable<br>Enables or disables the start of HSCIF serial transmission. The HSCIF starts serial transmission when transmit data is written to the HSFTDR while TE is 1. Before setting TE to 1, set HSSMR and HSFCR to specify the transmission format and reset the transmit FIFO.<br>0: Disables transmission.<br>1: Enables transmission. |
| 4 | RE | 0 | R/W | Receive Enable<br>Enables or disables the start of HSCIF serial reception. When RE is 1, the HSCIF starts serial reception by detecting a start bit. Before setting RE to 1, set HSSMR and HSFCR to specify the reception format and reset the receive FIFO.<br>0: Disables reception.*<br>1: Enables reception.<br>Note: * Even if RE is cleared to 0, the DR, ER, BRK, RDF, FER, PER, TO and ORER flags are not affected, and retain their states. |
| 3 | REIE | 0 | R/W | Receive Error Interrupt Enable<br>Enables or Disables generation of receive-error interrupt (ER) requests, break interrupt (BRK) requests and overrun-error interrupt (ORER) requests.<br>0: Disables receive-error interrupt (ER) requests, break interrupt (BRK) requests and overrun-error interrupt (ORER) requests.*<br>1: Enables receive-error interrupt (ER) requests, break interrupt (BRK) requests and overrun-error interrupt (ORER) requests.<br>Note: * When REIE is 1, ER, BRK or ORER interrupt requests will occur even if RIE is cleared to 0. This setting is used to notify the interrupt controller of ER, BRK, and ORER interrupt requests during DMAC transfer. |
| 2 | TOIE | 0 | R/W | Timeout Interrupt Enable<br>Enables or disables generation of timeout interrupt (TO) requests when the TO flag in HSLSR is set to 1.<br>0: Disables timeout interrupts (TO).<br>1: Enables timeout interrupts (TO). |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1, 0 | CKE[1:0] | 00 | R/W | Clock Enable 1 and 0<br>See Table 106.3 for the bit settings. |

**Table 106.3 Clock Selection**

| CKE[1] | CKE[0] | Clock Source | HSCK Pin |
|---|---|---|---|
| 0 | 0 | Internal clock (Internal clock, Internal clock/4, Internal clock/16, and Internal clock/64) | The HSCK pin is not used.<br>The HSCK pin functions as an input pin (input signals are ignored). (Initial value) |
| 0 | 1 | Prohibited | — |
| 1 | 0 | Baud rate generator output for external clock or HSCK | When SC_CLK is selected:<br>The HSCK pin is an input pin (input signals are ignored).<br>Set the SC_CLK frequency so that the frequency of BRGCLK is multiplied by the sampling rate. |
| | | | When HSCK is selected:<br>The HSCK pin inputs the clock (with the bit rate multiplied by the sampling rate). |
| | 1 | Prohibited | |

Note: It is not allowed to set synchronous communication using SC_CLK for input.

### 106.2.7 Serial Status Register (HSFSR)

HSFSR is a 16-bit register. The lower 8 bits are status flags that indicate the operating status of the HSCIF, and the upper 8 bits are all reserved.

HSFSR can always be read from and written to by the CPU. However, the flags ER, TEND, TDFE, BRK, RDF, and DR cannot be written by 1. The FER and PER flags are read-only flags and cannot be modified.

HSFSR is initialized to H'0060 by a power-on reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ER | TEND | TDFE | BRK | FER | PER | RDF | DR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/WC0 | R/WC0 | R/WC0 | R/WC0 | R | R | R/WC0 | R/WC0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | ER | 0 | R/WC0 | Receive Error<br>Indicates that a framing error or a parity error has occurred in reception. The ER flag is not affected by an error and retains its previous state when the RE bit is 0 in HSSCR.<br>If a receive error occurs, receive data will be transferred to HSFRDR and reception operation will be continued. Whether there is a receive error in data read from HSFRDR can be determined by the FER and PER bits in HSFSR.<br>0: Indicates that no framing or parity error has occurred in reception.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• 0 is written to ER.<br>1: Indicates that a framing error or a parity error has occurred in reception.<br>[Setting conditions]<br>• The HSCIF checks whether the stop bit at the end of receive data is 1, but the stop bit is 0.*<br>• The number of 1-bit in receive data plus the parity bit does not match the parity setting (even or odd) specified by the O/E# bit in HSSMR during reception.<br>Note: * In the 2-stop-bit mode, only the first stop bit is checked that the value is 1; the second stop bit is not checked. |
| 6 | TEND | 1 | R/WC0 | Transmit End<br>Indicates that transmission has been ended because there was no valid data in HSFTDR when the last bit of the transmit character was transmitted.<br>0: Indicates that transmission is in progress.<br>[Clearing conditions]<br>• Transmit data is written to HSFTDR, and 0 is written to TEND.<br>• Data is written to HSFTDR by the DMAC.<br>1: Indicates that transmission has been ended.<br>[Setting conditions]<br>• A power-on reset is executed.<br>• The TE bit in HSSCR is 0.<br>• There is no transmission data in HSFTDR when the last bit of a 1-byte serial transmission character is transmitted. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | TDFE | 1 | R/WC0 | Transmit FIFO Data Empty<br>Indicates that the HSCIF has transferred data from HSFTDR to HSTSR, the number of data bytes in HSFTDR becomes equal to or less than the transmit trigger count specified by the HSTTRGR, and HSFTDR is ready to be written by new transmit data.<br>HSFTDR is a 128-byte FIFO register. The maximum number of bytes that can be written to when TDFE = 1 is "128 – [the transmit trigger count]". If data exceeding this value is attempted to be written, the data will be ignored. The number of data bytes in HSFTDR is indicated by the upper bits of HSFDR.<br>If the number of data written in HSFTDR is equal to or less than the transmit trigger count, this bit will be set to 1 even if it is cleared to 0 after it is read as 1.<br>0: Indicates that the number of transmit data written to HSFTDR exceeds the transmit trigger count.<br>[Clearing conditions]<br>• Transmit data exceeding the specified transmit trigger count have been written to HSFTDR, and 0 is written to TDFE.<br>• Transmit data exceeding the specified transmit trigger count have been written to HSFTDR by the DMAC.<br>1: Indicates that the number of transmit data in HSFTDR is equal to or less than the transmit trigger count.<br>[Setting conditions]<br>• A power-on reset is executed.<br>• The number of transmit data in HSFTDR is equal to or less than the transmit trigger count after transmission. |
| 4 | BRK | 0 | R/WC0 | Break Detect<br>Indicates that a receive data break signal has been detected. If a break signal is detected, receive data (H'00) transfer to HSFRDR is stopped. After the break is canceled and the receive signal returns to 1, the receive data transfer resumes.<br>0: Indicates that no break signal has been received.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• 0 is written to BRK.<br>1: Indicates that a break signal has been received.<br>[Setting condition]<br>• Data with a framing error is received, followed by the space "0" (low level) for at least one frame length. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | FER | 0 | R | Framing Error<br>Indicates that a framing error has been found in the data that is to be read next from HSFRDR.<br>0: Indicates that there is no framing error in the receive data that is to be read from HSFRDR.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• There is no framing error in the data that is to be read next from HSFRDR.<br>1: Indicates that there is a framing error in the receive data that is to be read from HSFRDR.<br>[Setting condition]<br>• There is a framing error in the data that is to be read next from HSFRDR. |
| 2 | PER | 0 | R | Parity Error<br>This bit indicates that a parity error has been found in the data that is to be read next from HSFRDR.<br>0: Indicates that there is no parity error in the receive data that is to be read from HSFRDR.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• There is no parity error in the data that is to be read next from HSFRDR.<br>1: Indicates that there is a parity error in the receive data that is to be read from HSFRDR.<br>[Setting condition]<br>• There is a parity error in the data that is to be read next from HSFRDR. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | RDF | 0 | R/WC0 | Receive FIFO Data Full<br>Indicates that the received data has been transferred from HSRSR to HSFRDR, and the number of receive data bytes in HSFRDR becomes equal to or more than the receive trigger count specified by the HSRTRGR.<br>HSFRDR is a 128-byte FIFO register. When RDF = 1, data equal to or more than the number of receive trigger data bytes can be read. When HSFRDR is empty, HSFRDR is read as an undefined value. The number of receive data bytes in HSFRDR is indicated by the lower bits of HSFDR.<br>If the number of data in HSFRDR is equal to or more than the trigger count, this bit will be set to 1 even if it is cleared to 0. At this time, read receive data until the number of data in HSFRDR is less than the trigger count, read RDF as 1, and then clear RDF.<br>0: Indicates that the number of receive data bytes in HSFRDR is less than the specified receive trigger count.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• HSFRDR is read until the number of receive data bytes in HSFRDR is less than the receive trigger count, and 0 is written to RDF.<br>• HSFRDR is read by the DMAC until the number of receive data bytes in HSFRDR is less than the receive trigger count.<br>1: Indicates that the number of receive data bytes in HSFRDR is equal to or more than the specified receive trigger count.<br>[Setting condition]<br>• Receive data more than the receive trigger count have been stored in HSFRDR. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | DR | 0 | R/WC0 | Receive Data Ready<br>Indicates that the receive FIFO data register (HSFRDR) contains fewer bytes than the trigger number for reception and no further data have arrived over at least the time corresponding to the setting of HSSCR[15:14]* since the stop bit for the byte to have been received.<br>0: Indicates that data is being received or has been successfully received, and there is no receive data in HSFRDR.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• All the receive data in HSFRDR has been read, and 0 is written to DR.<br>• All the receive data in HSFRDR has been read by the DMAC.<br>1: Indicates that no further receive data has been received.<br>[Setting condition]<br>The receive FIFO data register (HSFRDR) contains fewer bytes than the trigger number for reception and no further data have arrived over at least the time corresponding to the setting of HSSCR[15:14]* since the stop bit for the byte to have been received.<br>Note: * When the setting is 00, the time is 15 etu. This is equivalent to 1.5 frames in an 8-bit, 1-stop-bit format.<br>etu: Elementary Time Unit (time for transfer of one bit) |

### 106.2.8 Bit Rate Register (HSBRR)

HSBRR is an 8-bit register that sets the serial transmission/reception bit rate in accordance with the baud rate generator operating clock selected by the CKS [1:0] bits in HSSMR. This baud rate generator is intended for Internal clock, Internal clock/4, Internal clock/16, and Internal clock/64, see section 106.3.2, Generation of Internal Clock.

For details on the baud rate generator for external clock, see section 106.3.3 Baud Rate Generator for External Clock (BRG).

HSBRR can always be read from and written to by the CPU except for during transfer.

HSBRR is initialized to H'FF by a power-on reset.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

### 106.2.9 Frequency Division Register (DL)

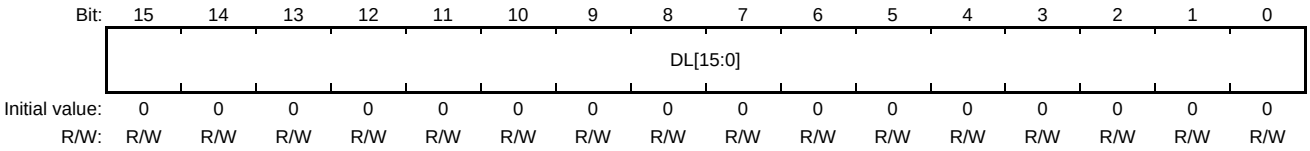
This register specifies the value of frequency division for the frequency divided clock generated by the BRG.

This register supports a 16-bit binary format that allows specifying a value in the range of 1 to 65535.

Setting all 0s in this register makes the BRG output the frequency divided clock at the low level.

The value of frequency division is given by the following formula:

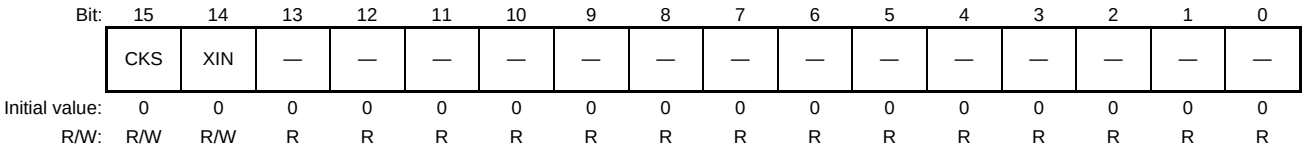
The value of frequency division = (clock input frequency)/ (required baud rate × sampling rate)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | DL[15:0] | All 0 | R/W | Specifies a division value of frequency clock generated in BRG.<br>The setting value is enabled in the range of 1 to 65535. |

### 106.2.10 Clock Select Register (CKS)

This register switches the output between the frequency divided clocks and specifies a source clock for the external baud rate.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | CKS | 0 | R/W | This bit switches the output between the frequency divided clock (SC_CLK) and external clock (HSCK).<br>0: Selects the frequency divided clock.<br>1: Selects the external clock. |
| 14 | XIN | 0 | R/W | Selects the baud rate generator clock source for the external clock between SCIF_CLK and Internal clock.<br>0: Selects the external clock (SCIF_CLK).<br>1: Selects the internal clock (Internal clock). |
| 13 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 106.2.11 FIFO Control Register (HSFCR)

HSFCR is a register that resets data counts for transmit and receive FIFO registers. It also has a modem control and a loopback test enable bit.

HSFCR can always be read from and written to by the CPU except for during transfer.

HSFCR is initialized to H'0000 by a power-on reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | MCE | TFRST | RFRST | LOOP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

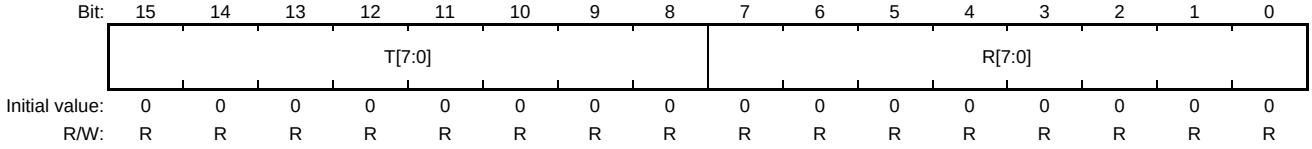
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 3 | MCE | 0 | R/W | Modem Control Enable<br>Enables or disables modem control signals HCTS# and HRTS#.<br>0: Disables modem signals.*<br>1: Enables modem signals.<br>Note: * HCTS# and HRTS# control ports. |
| 2 | TFRST | 0 | R/W | Transmit FIFO Data Register Reset<br>Enables or disables a transmit FIFO data register reset that empties the register.<br>0: Disables the reset.*<br>1: Enables the reset.<br>Note: * The register is reset by a power-on reset. |
| 1 | RFRST | 0 | R/W | Receive FIFO Data Register Reset<br>Enables or disables a receive FIFO data register (HSFRDR) reset that empties the register.<br>0: Disables the reset.*<br>1: Enables the reset.<br>Note: * The register is reset by a power-on reset. |
| 0 | LOOP | 0 | R/W | Loopback Test<br>Enables or disables the loopback test by internally connecting the transmit output pin (HTX) and receive input pin (HRX), and the HRTS# pin and HCTS# pin.<br>0: Disables the loopback test.<br>1: Enables the loopback test. |

### 106.2.12 FIFO Data Count Register (HSFDR)

HSFDR is a 16-bit register that indicates the number of data bytes stored in HSFTDR and that in HSFRDR.

The upper 8 bits indicate the number of transmit data bytes in HSFTDR, and the lower 8 bits indicates the number of receive data bytes in HSFRDR.

HSFDR can always be read by the CPU.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 8 | T[7:0] | H'00 | R | These bits indicate the number of data bytes un-transmitted and still stored in HSFTDR.<br>H'00 indicates that there is no transmit data in HSFTDR, and H'80 indicates that HSFTDR is full of transmit data. |
| 7 to 0 | R[7:0] | H'00 | R | These bits indicate the number of receive data stored in HSFRDR.<br>H'00 indicates that there is no receive data in HSFRDR, and H'80 indicates that HSFRDR is full of receive data. |

### 106.2.13 Serial Port Register (HSSPTR)

HSSPTR controls multiplexed input/output and data on the high speed serial communication interface (HSCIF) ports. Bits 1 and 0 control breaks in serial transmission/reception by reading input data from the HRX pin and writing output data to the HTX pin. Bits 3 and 2 read input data from and write output data to the HSCK pin. Bits 5 and 4 read input data from and write output data to the HCTS# pin. Bits 7 and 6 read input data from and write output data to the HRTS# pin.

HSSPTR is a 16-bit register that can always be read from and written to by the CPU.

All HSSPTR bits except bits 6, 4, 2, and 0 are initialized to 0 by a power-on reset. The values of bits 6, 4, 2, and 0 are undefined.

Note: Whether modem control can be selected or not depends on the channel.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RTSIO | RTSDT | CTSIO | CTSDT | SCKIO | SCKDT | SPB 2IO | SPB 2DT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | 0 | — | 0 | — | 0 | — |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | RTSIO | 0 | R/W | Serial Port – RTS Port Input/output<br>Specifies input or output for the serial port HRTS# pin. To actually set the HRTS# pin as a port output pin to output the value set by the RTSDT bit, the MCE bit in HSFCR should be cleared to 0.<br>0: Indicates that this bit does not output the value of the RTSDT bit to the HRTS# pin.<br>1: Indicates that this bit outputs the value of the RTSDT bit to the HRTS# pin. |
| 6 | RTSDT | — | R/W* | Serial Port – RTS Port Data<br>Specifies the input/output data level of the serial port HRTS# pin. Whether the pin is set for input or output is determined by the RTSIO bit. When the pin is set for output, the value of the RTSDT bit is output to the HRTS# pin. Regardless of the value of the RTSIO bit, the value of the HRTS# pin is read from the RTSDT bit.<br>The initial value of this bit is undefined after a power-on reset.<br>0: Indicates that the input/output data is low level.<br>1: Indicates that the input/output data is high level. |
| 5 | CTSIO | 0 | R/W | Serial Port – CTS Port Input/output<br>Specifies input or output for the serial port HCTS# pin. To actually set the HCTS# pin as a port output pin to output the value set by the CTSDT bit, the MCE bit in HSFCR should be cleared to 0.<br>0: Indicates that the CTSDT bit value is not output to the HCTS# pin.<br>1: Indicates that the CTSDT bit value is output to the HCTS# pin. |
| 4 | CTSDT | — | R/W* | Serial Port – CTS Port Data<br>Specifies the input/output data level of the serial port HCTS# pin. Whether the pin is set for input or output is determined by the CTSIO bit. When the pin is set for output, the value of the CTSDT bit is output to the HCTS# pin. Regardless of the value of the CTSIO bit, the value of the HCTS# pin is read from the CTSDT bit.<br>The initial value of this bit is undefined after a power-on reset.<br>0: Indicates that the input/output data is low level.<br>1: Indicates that the input/output data is high level. |
| 3 | SCKIO | 0 | R | Serial Port – Clock Port Input |
| 2 | SCKDT | — | R | Serial Port – Clock Port Data<br>The initial value of this bit is undefined after a power-on reset. |
| 1 | SPB2IO | 0 | R/W | Serial Port – Break Input/output<br>Specifies the output condition of the serial port HTX pin. To actually set the HTX pin as a port output pin to output the value set by the SPB2DT bit, the TE bit in HSSCR should be cleared to 0.<br>0: Indicates that the SPB2DT bit value is not output to the HTX pin.<br>1: Indicates that the SPB2DT bit value is output to the HTX pin. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | SPB2DT | — | R/W* | Serial Port – Break Data<br>Specifies the input level of the serial port HRX pin and the output level of the HTX pin. The HTX pin output conditions are determined by the SPB2IO bit. When the HTX pin is set for output, the value of the SPB2DT bit is output to the HTX pin. Regardless of the value of the SPB2IO bit, the value of the HRX pin is read from the SPB2DT bit.<br>The initial value of this bit is undefined after a power-on reset.<br>0: Indicates that the input/output data is low level.<br>1: Indicates that the input/output data is high level. |

Note: * The value written to these bits can't be read because the value of the pin is read.

### 106.2.14 Line Status Register (HSLSR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TO | — | ORER |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/WC0 | R | R/WC0 |

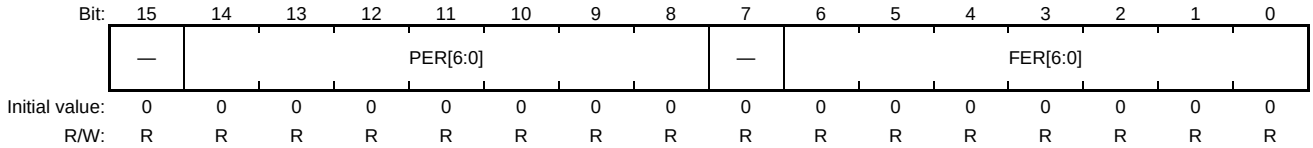
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | TO | 0 | R/WC0 | Timeout<br>Indicates that the receive FIFO data register (HSFRDR) contains fewer bytes than the trigger number for reception and no further data have arrived over at least the time corresponding to the setting of HSSCR[15:14]* since the stop bit for the byte to have been received.<br>0: Indicates that data is being received or has been successfully received and that there is no receive data in HSFRDR.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• All the receive data in the HSFRDR has been read, and 0 is written to TO.<br>1: Indicates that no further receive data has been received (receive timeout).<br>[Setting condition]<br>• The receive FIFO data register (HSFRDR) contains fewer bytes than the trigger number for reception and no further data have arrived over at least the time corresponding to the setting of HSSCR[15:14]* since the stop bit for the byte to have been received.*<br>Note: * When the setting is 00, the time is 15 etu. This is equivalent to 1.5 frames in an 8-bit, 1-stop-bit format.<br>etu: Elementary Time Unit (time for transfer of one bit) |
| 1 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 0 | ORER | 0 | R/WC0 | Overrun Error<br>Indicates that an overrun error has occurred in reception and abnormal termination is caused.<br>If an overrun error occurs, the receive data prior to the overrun error is retained in HSFRDR and the data received subsequently is discarded.<br>Any subsequent serial reception is disabled while the ORER flag is 1.<br>To resume data reception after clearing the ORER flag, be sure to first read (or clear) data in the receive FIFO and handle the error, then clear the ORER flag.<br>0: Indicates that data is being received or has been successfully received.<br>[Clearing conditions]<br>• A power-on reset is executed.<br>• 0 is written to ORER.<br>1: Indicates that an overrun error has occurred in reception.<br>[Setting condition]<br>• The next serial reception has been completed while HSFRDR is full of 128-byte data.<br>Note: When bit RE in HSSCR is cleared to 0, the ORER flag is not affected and its previous state is retained. |

### 106.2.15 Sampling Rate Register (HSSRR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRE | SRDE | — | — | SRHP[3:0] | | | | — | — | — | SRCYC[4:0] | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | SRE | 0 | R/W | Sampling Rate Register Enable (SRE)<br>0: Set the SRCYC [4: 0] bits to 15 (initial value).<br>1: Validates the setting of the SRCYC [4:0] bits. |
| 14 | SRDE | 0 | R/W | Sampling Point Register Enable (SRDE)<br>0: Invalidates the setting of the SRHP [3:0] bits and the sampling point will be (S+1)/2 for an odd sampling rate (S) and S/2 for an even sampling rate.<br>1: Validates the setting of the SRHP [3: 0] bits. |
| 13, 12 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 11 to 8 | SRHP[3:0] | 0000 | R/W | Sampling Point Register (SRHP)<br>The sampling point can be moved by setting the SDRE bit to 1 and setting a value in these bits. Normally, the sampling point is the point of S/2 or (S+1)/2 for a sampling rate of S. By setting a signed 4-bit integer in these bits, the sampling point can be shifted by the amount of the specified sampling clock cycles. This will improve the receive margin.<br>When setting a value in these bits, take notice that the sampling point does not become a negative value or it does not exceed the sampling rate. The shifted sampling point must satisfy the setup margin and hold margin. |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 to 0 | SRCYC[4:0] | 01111 | R/W | Bits 4 to 0: Sampling Rate Register (SRCYC)<br>Data transfer at a desired sampling rate can be enabled by setting the SRE bit to 1 and setting a value in these bits. Set a value of "S - 1" in these bits for a sampling rate of S. Note that the sampling rate must be from 8 to 32 (a value from 7 to 31 can be set in these bits). Set these bits to 15 (initial value) when the SRE bit is set to 0. |

### 106.2.16 Serial Error Register (HSRER)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 14 to 8 | PER[6:0] | H'00 | R | Parity Error Count<br>These bits indicate the number of data items in which a parity error occurred in the receive data stored in the receive FIFO data register (HSFRDR).<br>After the ER bit in HSFSR is set, the value in bits 14 to 8 will be the number of data items in which a parity error occurred. If all 128 bytes of receive data in HSFRDR have parity errors, bits PER [6:0] will have the value 0. |
| 7 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 to 0 | FER[6:0] | H'00 | R | Framing Error Count<br>These bits indicate the number of data items in which a framing error occurred in the receive data stored in the receive FIFO data register (HSFRDR).<br>After the ER bit in HSFSR is set, the value in bits 6 to 0 will be the number of data items in which a framing error occurred. If all 128 bytes of receive data in HSFRDR have framing errors, bits<br>FER [6:0] will have the value 0. |

### 106.2.17 RTS Output Active Trigger Register (HSRTGR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | RSTRG[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

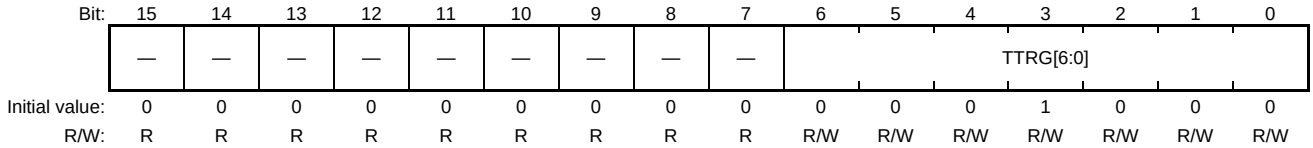
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 to 0 | RSTRG[6:0] | H'0F | R/W | RTS Output Active Trigger Count<br>The HRTS# signal goes high when the number of receive data items stored in the receive FIFO data register (HSFRDR) exceeds the value set in these bits. The initial value is 15. |

### 106.2.18 Receive FIFO Data Count Trigger Register (HSRTRGR)

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | RTRG[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 to 0 | RTRG[6:0] | H'01 | R/W | Receive FIFO Data Count Trigger<br>These bits set the receive data item count at which the receive data full (RDF) flag in the serial status register (HSFSR) is set. The RDF flag is set when the number of receive data items stored in the receive FIFO data register (HSFRDR) equals or exceeds the value set in these bits. The initial value is 1. |

### 106.2.19 Transmit FIFO Data Count Trigger Register (HSTTRGR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 6 to 0 | TTRG[6:0] | H'08 | R/W | Transmit FIFO Data Count Trigger<br>These bits set the un-transmitted data item count at which the transmit FIFO data register empty (TDFE) flag in the serial status register (HSFSR) is set. The TDFE flag is set when the number of transmit data items in the transmit FIFO data register (HSFTDR) falls under the value set in these bits due to transmit operations. The initial value is 8. |

## 106.3 Operation

### 106.3.1 Operation in Asynchronous Serial Communication Mode

In asynchronous serial communication mode, the HSCIF performs serial communication, in which data is transmitted or received in character units using the attached start bit indicating the start of communication and stop bit indicating the end of communication.

Figure 106.2 shows the general format for asynchronous serial communication.

In asynchronous serial communication, the transmission line is usually kept marked (high level). The HSCIF monitors the transmission line, and when it finds a space (low level), it regards the space as a start bit and starts serial communication.

One character in serial communication consists of a start bit (low level), followed by transmit/receive data (LSB-first; from the lowest bit), a parity bit (high or low level), and a stop bit (high level).

During reception in asynchronous mode, the HSCIF performs synchronization at the falling edge of the start bit. Transferred data is acquired at the center of each bit because the HSCIF samples on *S/2*th cycles of the clock which has a frequency of S times the bit rate, when the sampling rate as set in the sampling rate register (HSSRR) is S (if the sampling-rate setting is an odd number, the data is sampled on every *(S+1)/2*th pulse).

In addition, when the setting of the SRDE bit makes the setting of the sampling point bits effective, the point where the bits are latched can be intentionally moved from the centers of each bit.

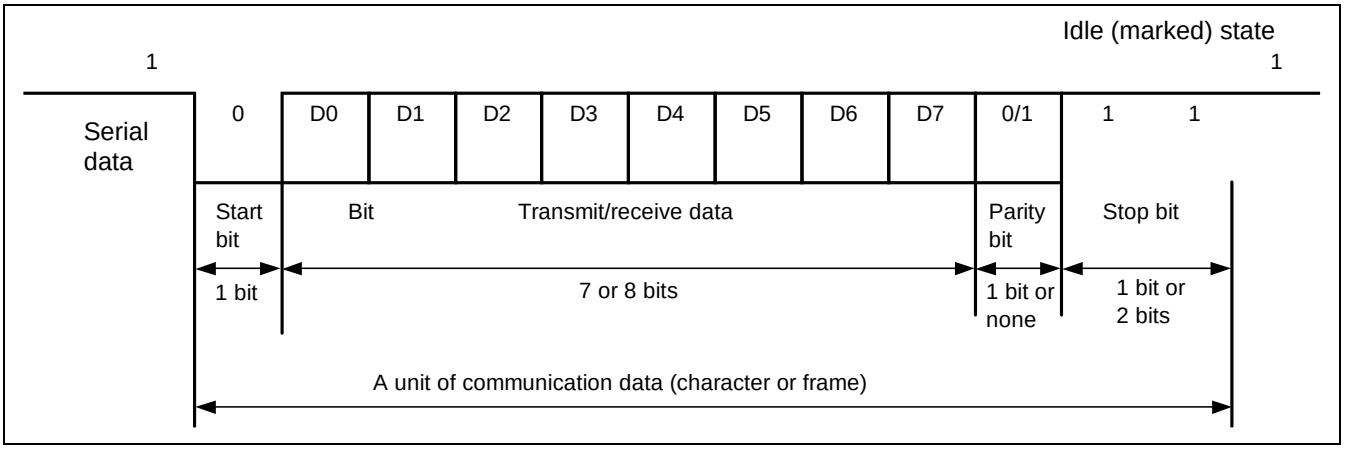



**Figure 106.2 Data Format in Asynchronous Mode (Example of 8-Bit Data with Parity and Two Stop Bits)**

### (1) Transmission/reception format

Table 106.4 shows available data transfer formats. The HSCIF supports 8 transfer formats, which can be specified by HSSMR.

**Table 106.4 Serial Transmission/Reception Formats (Asynchronous Mode)**

| SCSMR settings | | | Serial transmission/reception format and frame length | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHR | PE | STOP | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 0 | 0 | 0 | S | 8-bit data | | | | | | | | STOP | | |
| 0 | 0 | 1 | S | 8-bit data | | | | | | | | STOP | STOP | |
| 0 | 1 | 0 | S | 8-bit data | | | | | | | | P | STOP | |
| 0 | 1 | 1 | S | 8-bit data | | | | | | | | P | STOP | STOP |
| 1 | 0 | 0 | S | 7-bit data | | | | | | | | STOP | | |
| 1 | 0 | 1 | S | 7-bit data | | | | | | | | STOP | STOP | |
| 1 | 1 | 0 | S | 7-bit data | | | | | | | | P | STOP | |
| 1 | 1 | 1 | S | 7-bit data | | | | | | | | P | STOP | STOP |

[Legend]
S : Start bit
STOP : Stop bit
P : Parity bit

### (2) Clock

The transfer clock can be selected from the following two clocks using the CKE[1:0] bits in HSSCR:

- Internal clock generated by the on-chip baud rate generator
- External clock generated by an external clock baud rate generator

### (3) Data transmission/reception

#### (a) Initialization of HSCIF (asynchronous mode)

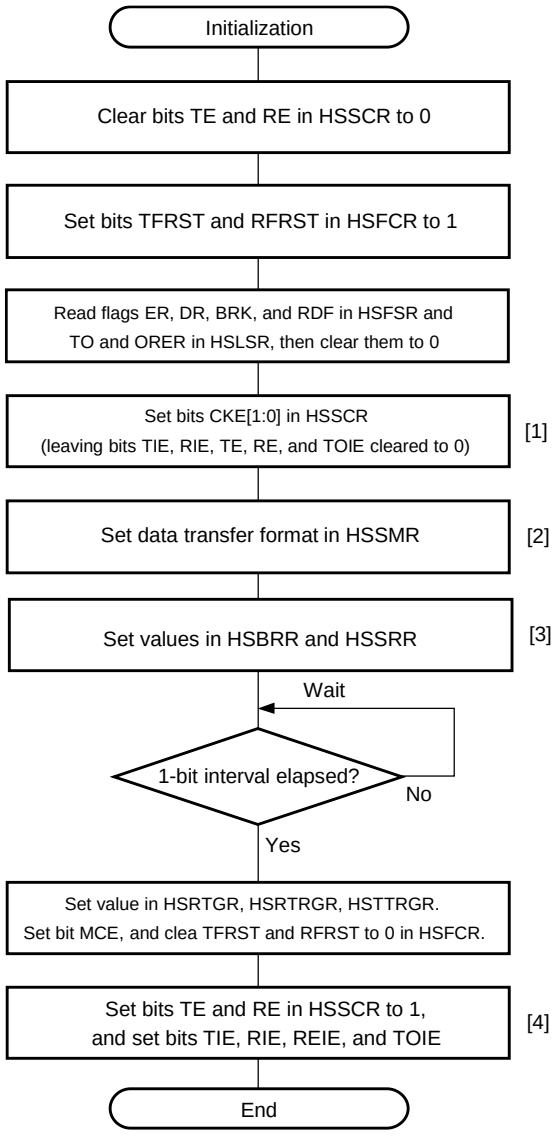
Before transmitting/receiving data or changing the operating mode or communication format, the HSCIF should be initialized using the sample flowchart for HSCIF initialization shown in Figure 106.3.

[Notes]

Clearing the TE bit to 0 initializes HSTSR. However, HSFSR, HSFTDR, and HSFRDR contents are retained even if the TE and RE bits are cleared to 0.

The TE bit should be cleared to 0 after all transmit data has been sent and the TEND flag has been set in HSFSR. The TE bit can be cleared to 0 during transmission, but the data being transmitted will enter the marked state after clearing. In addition, before setting the TE bit to 1 to restart the transmission, set the TFRST bit to 1 in HSFCR to reset HSFTDR.

When an external clock is used, do not stop the clock during operation or initialization. If stopped, the operation will be unreliable. Furthermore, when the baud rate generator for external clock is also to be used, be sure to make settings for it before starting initialization of the HSCIF.




[1] Set the clock selection in HSSCR. Be sure to clear bits TIE, RIE, TE, RE, and TOIE to 0.

[2] Set the data transfer format in HSSMR.

[3] Write a value for the bit rate to HSBRR. (Not necessary if an external clock is used.) Set values for the sampling rate and sampling point to HSSRR.

[4] Wait at least one bit interval, then set bit TE or RE in HSSCR to 1. Also, set bits TIE, RIE, REIE, and TOIE. Setting bit TE enables the HTX and HRX pins to use.
In transmission, the HSCIF will go to the mark state, and in reception, it will go to the idle state, waiting for a start bit.

Note: HSBRR and HSFCR registers can only be set while both TE and RE bits of HSSCR are cleared.

**Figure 106.3 Sample Flowcharts for Initializing the HSCIF**

**(b) Serial data transmission (asynchronous mode)**

Figure 106.4 shows a sample flowchart for serial transmission.

After the HSCIF transmission operation is enabled, serial data transmission can be performed using the following procedure:

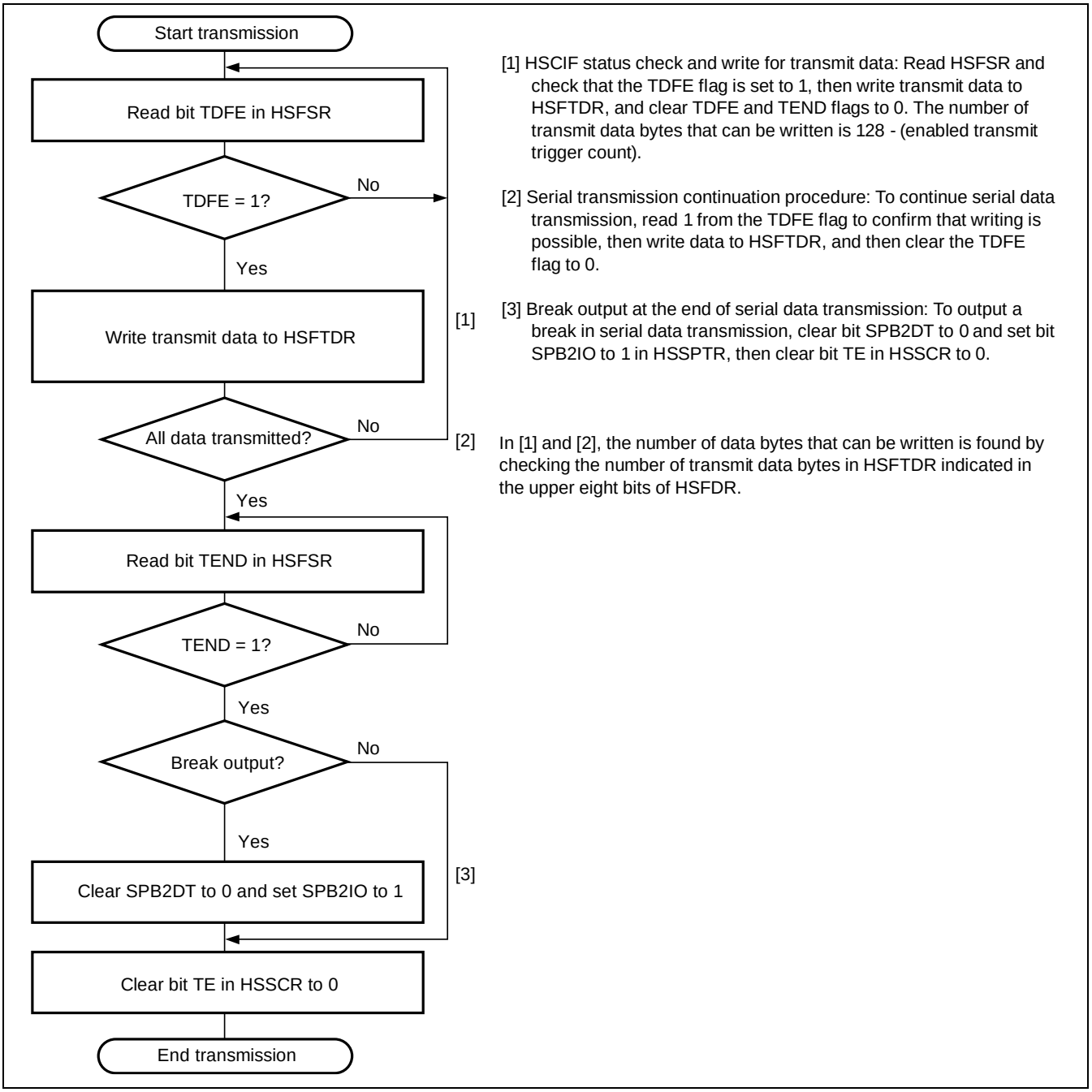



**Figure 106.4 Sample Flowcharts for Serial Transmission**

In serial transmission, the HSCIF operates as follows:

1. When data is written to HSFTDR, the HSCIF transfers the data from HSFTDR to HSTSR and starts transmitting. Confirm that the TDFE flag in HSFSR is set to 1 before writing transmit data to HSFTDR. The number of data bytes that can be written is at least 128 - (transmit trigger count).

2. When data is transferred from HSFTDR to HSTSR and the HSCIF starts transmission, consecutive transmission is performed until there is no transmit data left in HSFTDR. When the number of transmit data bytes in HSFTDR is equal to or less than the transmit trigger count specified in HSTTRGR, the TDFE flag is set. If the TIE and TEIE bits in HSSCR are set to 1 and 0, respectively at this time, a transmit-FIFO-data-empty interrupt (TDFE) request occurs. The serial transmit data is sent from the HTX pin in the following order:
   A. Start bit: One 0-bit is output.
   B. Transmit data: 8-bit or 7-bit data is output in LSB-first order.
   C. Parity bit: One parity bit (even or odd parity) is output.
      A format that does not output a parity bit can also be selected.
   D. Stop bit(s): One or two stop bits are output.
   E. Marked state: 1 is output continuously until the start bit that starts the next transmission is sent.
3. The HSCIF checks transmit data in HSFTDR when sending the stop bit. If there is data in it, the HSCIF transfers the data from HSFTDR to HSTSR, sends the stop bit, and then starts serial transmission of the next frame. If there is no transmit data, the HSCIF sets the TEND flag to 1 in HSFSR and sends out the stop bit, and then the marked state is entered to output 1 continuously. At this time, if the TIE and TEIE bits in HSSCR are set to 1, a transmit-end interrupt (TEND) request occurs.

Figure 106.5 shows an example of transmission in asynchronous mode.

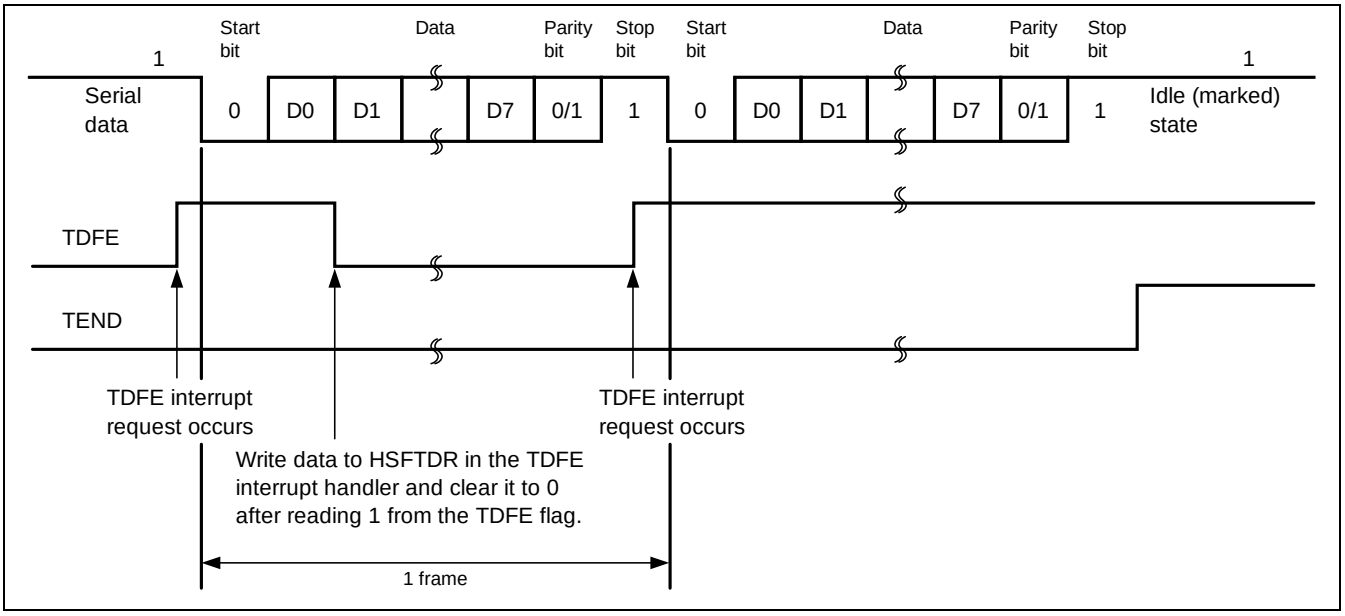



**Figure 106.5 Sample HSCIF Transmission Operation (Example of 8-Bit Data with Parity and One Stop Bit)**

4. When modem control is enabled, transmission can be stopped or resumed in accordance with the HCTS# input value. When HCTS# is set to 1 during transmission, the marked state is entered after one frame of data transmission is ended. Setting HCTS# to 0 restarts outputting the next transmit data from the start bit. Figure 106.6 shows an example of the operation with modem control enabled.

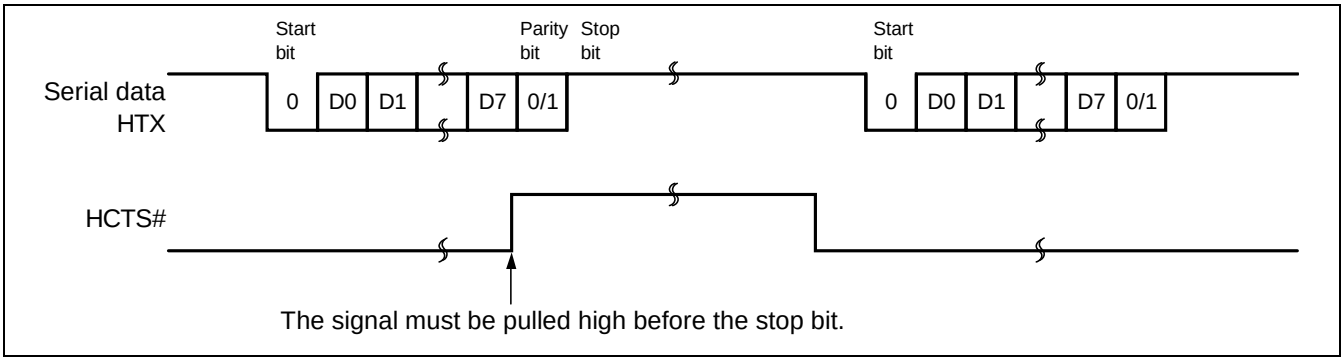



**Figure 106.6 Sample Operation with Modem Control Enabled (HCTS#)**

### (c) Serial data reception (asynchronous mode)

Figure 106.7 and Figure 106.8 show sample flowcharts for serial reception.

After the HSCIF reception operation is enabled, serial data reception can be performed using the following procedure:

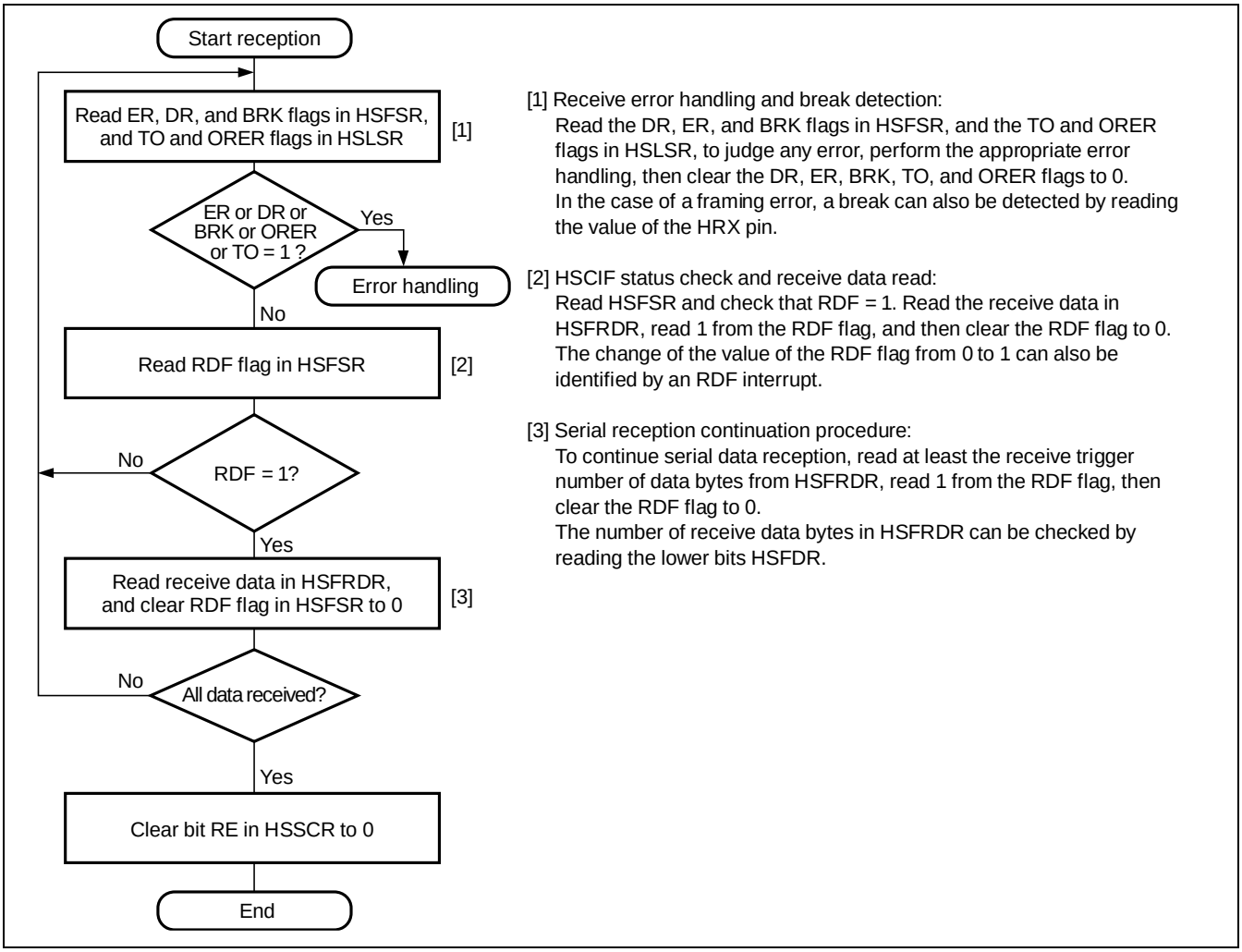
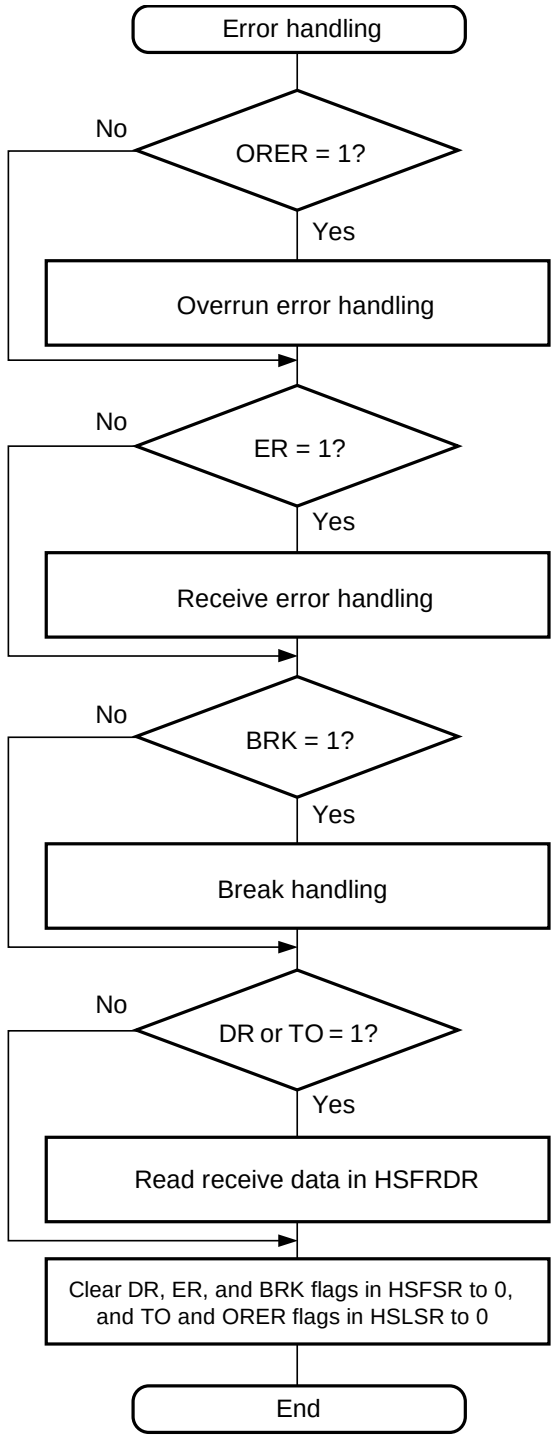



**Figure 106.7 Sample Flowcharts for Serial Reception (1)**




Whether a framing error or parity error has occurred in the receive data that is to be read from HSFRDR can be determined by checking bits FER and PER in HSRER.

**Figure 106.8 Sample Flowcharts for Serial Reception (2)**

In serial reception, the HSCIF operates as follows:

1. The HSCIF monitors the transmission line, and when detecting the start bit 0, it performs internal synchronization and starts reception.
2. The HSCIF stores the received data in HSRSR in LSB-to-MSB order.
3. The HSCIF receives the parity bit and stop bit.
   After receiving these bits, the HSCIF performs the following checks:

A. Stop bit: The HSCIF checks whether the stop bit is 1.
   If there are two stop bits, it checks only the first stop bit.
B. Receive data: The HSCIF checks that receive data can be transferred from the receive shift register (HSRSR) to HSFRDR.
C. Overrun error: The HSCIF checks that the ORER flag is 0, indicating that no overrun error has occurred.
D. Break state: The HSCIF checks that the BRK flag is 0, indicating that the break state is not set.

If the HSCIF can confirm the conditions of (b), (c), and (d), it stores the receive data in HSFRDR.

Note: The HSCIF continues to receive data even when a parity error or a framing error occurs.

4. If the RDF flag changes to 1 while the RIE bit in HSSCR is 1, a receive-FIFO-data-full interrupt (RDF) request occurs.
   If the DR flag changes to 1 while the RIE bit in HSSCR is 1, a receive-data-ready interrupt (DR) request occurs.
   If the TO flag changes to 1 while the TOIE bit in HSSCR is 1, a timeout interrupt (TO) request occurs.
   If the ER flag changes to 1 while the RIE or REIE bit in HSSCR is 1, a receive-error interrupt (ER) request occurs.
   If the BRK flag changes to 1 while the RIE or REIE bit in HSSCR is 1, a break interrupt (BRK) request occurs.
   If the ORER flag changes to 1 while the RIE or REIE bit in HSSCR is 1, an overrun-error interrupt (ORER) request occurs.

Figure 106.9 shows an example of reception in asynchronous mode.

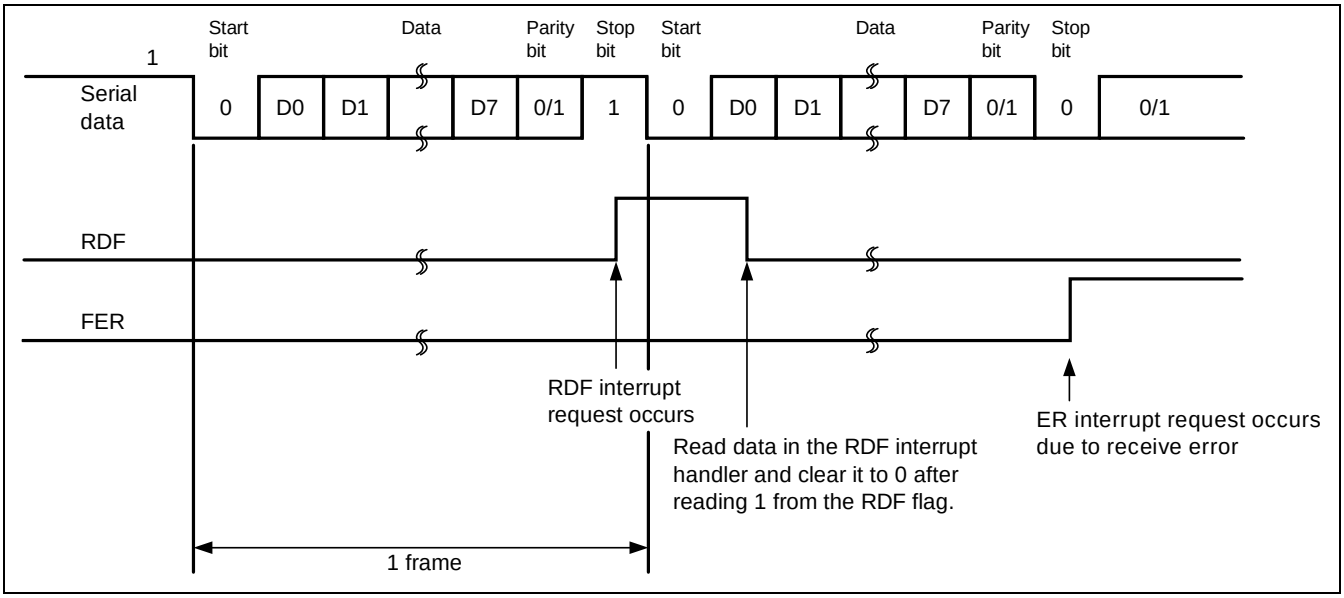



**Figure 106.9 Sample HSCIF Receive Operation (Example of 8-Bit Data with Parity and One Stop Bit)**

5. When modem control is enabled, the HRTS# signal is output when HSFRDR is empty. When HRTS# is 0, data can be received. When HRTS# is set to 1, the number of data bytes in HSFRDR is equal to or more than the RTS output active trigger count (HSRTGR.RSTRG).

Figure 106.10 shows an example of the operation with modem control enabled.

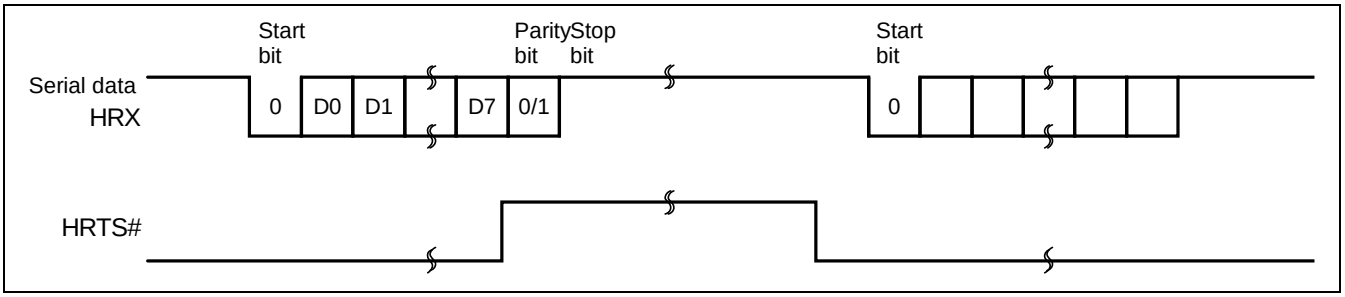



**Figure 106.10 Example of the Operation with Modem Control Enabled (HRTS#)**

### 106.3.2 Generation of Internal Clock

Internal clock is generated by the on-chip baud rate generator.
Baud rate is determined by the HSBRR and CKS[1:0] bit in HSSMR.

[Asynchronous mode]

$$N = \frac{\text{Internal clock}}{Sr \times 2^{2n+1} \times B} \times 10^6 - 1$$

B: Bit rate (bit/s)

N: HSBRR setting for the baud rate generator ($0 \leq N \leq 255$) (which satisfies the electrical characteristics)

Internal clock: AXI bus clock operating frequency (MHz)

n: Baud rate generator input clock (n = 0, 1, 2, 3)
(See the Table 106.5 for the relation between n and the clock.)

Sr: Sampling rate (8 to 32)

**Table 106.5 HSSMR Settings**

| | Baud Rate Generator | HSSMR Setting | |
|---|---|---|---|
| **n** | **Input Clock** | **CKS[1]** | **CKS[0]** |
| 0 | Internal clock | 0 | 0 |
| 1 | Internal clock/4 | 0 | 1 |
| 2 | Internal clock/16 | 1 | 0 |
| 3 | Internal clock/64 | 1 | 1 |

The bit rate error in asynchronous mode is determined by the following equation:

$$\text{error (\%)} = \left\{ \frac{\text{Internal clock} \times 10^6}{(N+1) \times B \times Sr \times 2^{2n+1}} - 1 \right\} \times 100$$

### 106.3.3 Baud Rate Generator for External Clock (BRG)

#### (1) Overview

The HSCIF incorporates a baud rate generator for external clock (abbreviated as BRG, hereafter). The BRG supplies a sampling clock (BRGCLK) to the HSCIF core by dividing the external clock SC_CLK (selectable between SCIF_CLK and Internal clock) by 1 to 216 – 1. In addition, the BRG switches the output between the external clock HSCK and divided clock.

Figure 106.11 shows a block diagram of the BRG.

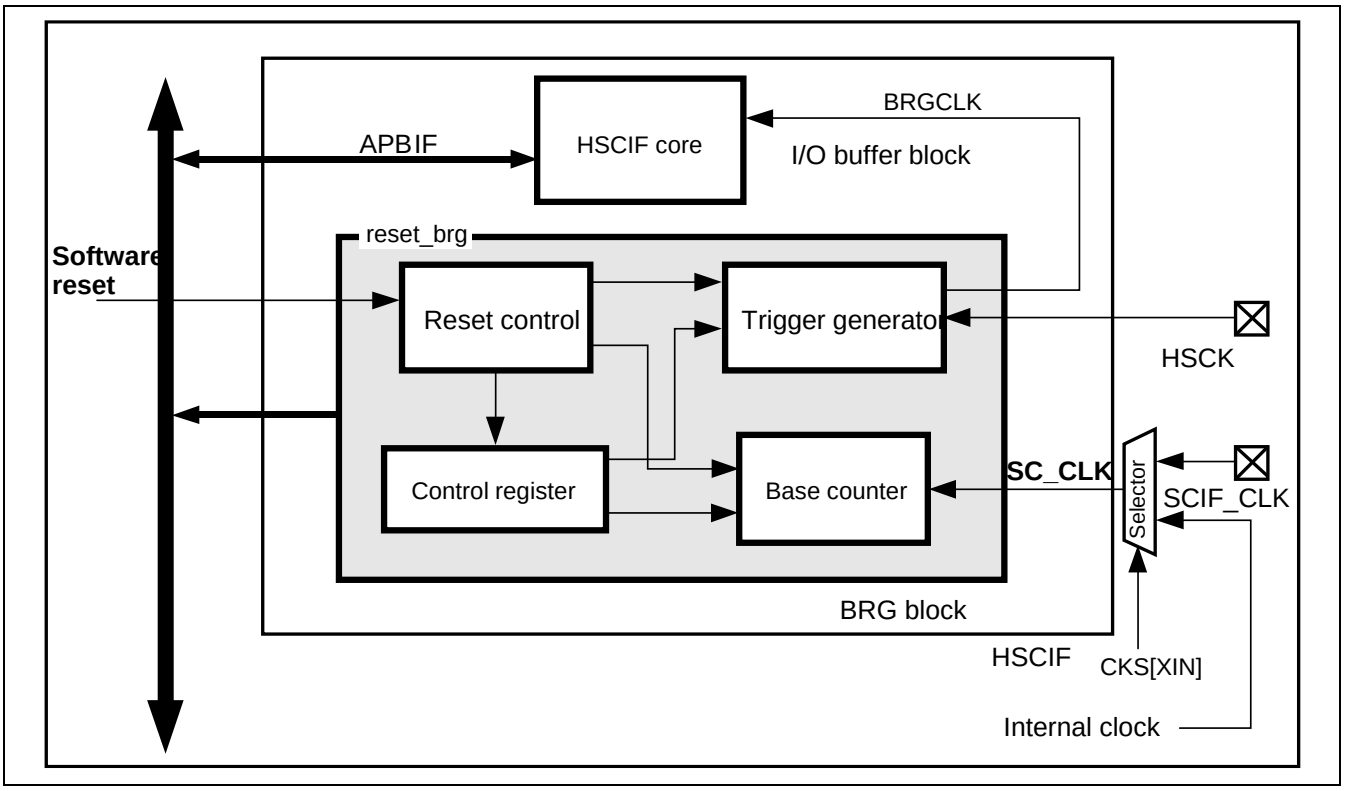



**Figure 106.11 BRG Block Diagram**

(a) Reset controller:

The reset controller handles resetting the control register, base counter, and trigger generator.

(b) Control register:

The control register has the frequency division register and clock select register.

(c) Base counter:

The base counter is a 16-bit CLK synchronization counter that is used to determine the timing for generating a frequency divided clock.

(d) Trigger generator:

The trigger generator generates rising-edge/falling-edge triggers for a frequency divided clock with the timing according to values of the frequency division register and base counter. The triggers are used to generate the frequency divided clock. In addition, the trigger generator switches the output between the HSCK (external clock input) and frequency divided clock.

#### (2) Registers

To set baud rate requires setting to Frequency Division Register(DL) and Clock Select Register(CKS)

### (3) Frequency Division Register (DL)

To set a division value of frequency requires setting to Frequency-Division-Register(DL).

Refer to Frequency-Division-Register(DL).

Table 106.6 shows reference values of Frequency-Division-Register(DL).

Table 106.6 and Table 106.7 show how to use the baud rate generator with a3.6864-MHz crystal resonator, and a 26-MHz crystal resonator.

**Table 106.6 Baud Rate (3.6864-MHz clock)**

| Baud Rate | Value of Frequency Division | Error Rate (*) |
|---|---|---|
| 50 | 4608 | — |
| 75 | 3072 | — |
| 110 | 2095 | –0.022 |
| 134.5 | 1713 | 0.001 |
| 150 | 1536 | — |
| 300 | 768 | — |
| 600 | 384 | — |
| 1200 | 192 | — |
| 1800 | 128 | — |
| 2000 | 115 | 0.174 |
| 2400 | 96 | — |
| 3600 | 64 | — |
| 4800 | 48 | — |
| 7200 | 32 | — |
| 9600 | 24 | — |
| 14400 | 16 | — |
| 19200 | 12 | — |
| 38400 | 6 | — |
| 76800 | 3 | — |
| 115200 | 2 | — |

Note: —: Indicates that the error rate is 0.

**Table 106.7 Baud Rate (26-MHz clock)**

| Baud Rate | Value of Frequency Division | Sampling Rate | Error Rate* |
|---|---|---|---|
| 9600 | 129 | 21 | 0.025 |
| 19200 | 52 | 26 | -0.160 |
| 38400 | 26 | 26 | -0.160 |
| 57600 | 15 | 30 | -0.309 |
| 115200 | 9 | 25 | -0.309 |
| 230400 | 4 | 28 | -0.756 |
| 460800 | 2 | 28 | -0.756 |
| 921600 | 1 | 28 | -0.756 |
| 1843200 | 1 | 14 | -0.756 |
| 3250000 | 1 | 8 | — |

Note: * "—" Indicates that the error rate is 0.

### (4) Clock Select Register (CKS)

To set output between the frequency divided clock and specifies a source clock for the external baud rate requires setting to Clock Select Register (CKS).

Refer to Clock Select Register (CKS).

### (5) Restrictions in BRG

#### (a) Notes on setting frequency division register

1. For the initial setting of the register after a reset, at least one bit of waiting period is required to secure the clock stabilization time.

   (Example) One bit period when DL = 2

   3.68 (MHz) × 1/2 × 1/16 = 0.115 (MHz) → 8695 (ns)

2. For modifying the register value after the setting stated in <1> above, at least one bit of waiting period at the maximum bit rate (DL = '65535') is required.

   The HSCIF registers and BRG registers should be set as the following table:

- Asynchronous mode (SC_CLK external input)

| HSCIF | | BRG | |
|---|---|---|---|
| **Register Name** | **Setting Value** | **Register Name** | **Setting Value** |
| HSSCR.CKE[1:0] | B'10 | CKS | H'0000 |
| | | DL | H'1 to H'FFFF |

- Asynchronous mode (HSCK external input)

| HSCIF | | BRG | |
|---|---|---|---|
| **Register Name** | **Setting Value** | **Register Name** | **Setting Value** |
| HSSCR.CKE[1:0] | B'10 | CKS | H'8000 |
| | | DL | Don't care |

- Asynchronous mode (SC_CLK internal input)

| HSCIF | | BRG | |
|---|---|---|---|
| **Register Name** | **Setting Value** | **Register Name** | **Setting Value** |
| HSSCR.CKE[1:0] | B'10 | CKS | H'4000 |
| | | DL | H'1 to H'FFFF |

3. The register settings for the baud rate generator for external clock should be made before starting initialization of the HSCIF.

### 106.3.4 HSCIF Interrupt Sources and the DMAC

If the DMAC is used for transmission/reception, set and enable the DMAC before setting the HSCIF.

**Transmission interrupts and DMA transfer:**

If the TDFE/TEND flag in HSFSR is set to 1 when the TDFE/TEND interrupt is enabled by the TIE bit, a TDFE/TEND interrupt request and a transmit-FIFO-data-empty DMA transfer request will occur. If the TDFE/TEND flag is set to 1 when TDFE/TEND interrupt is disabled by the TIE bit, only the transmit-FIFO-data-empty DMA transfer request will occur. (A transmit-FIFO-data-empty DMA transfer request is generated when the TDFE flag is set while TEIE is 0, or when the TEND flag is set while TEIE is 1. DMA transfer requests are not affected by the TEIE bit.)

When TDFE/TEND interrupt requests are enabled, the interrupt requests are cleared by the DMAC regardless of the interrupt handling program.

**Reception interrupts and DMA transfer:**

If the RDF/DR flag in HSFSR is set to 1 when RDF/DR interrupt is enabled by the RIE bit, an RDF/DR interrupt request occurs. If the RDF/DR flag is set to 1, a receive-FIFO-data-full DMA transfer request occurs. If the RDF/DR flag is set to 1 when RDF/DR interrupt is disabled by the RIE bit, and only a receive-FIFO-data-full DMA transfer request occurs and DMAC can be activated to perform data transfer.

Setting the RIE bit in HSSCR to 0 and the REIE bit to 1 generates the ER/BRK/ORER interrupt requests without generating RDF/DR interrupt requests. When the BRK flag in HSFSR or the ORER flag in HSLSR is set to 1, BRK/ORER interrupt requests occur.

If the TO flag is set to 1 in HSLSR when TO interrupts are enabled by the TOIE bit, TO interrupt requests occur.

When DR/TO interrupt requests are enabled to be issued, interrupt requests generated by the DR flag are cleared by the DMAC regardless of the interrupt handling program, however, those generated by the TO flag are not cleared by the DMAC. Therefore, the TO flag interrupt requests need to be cleared with the interrupt handling program. (The DR and TO flags are set at the same time, but cleared separately.)

**Table 106.8 HSCIF Interrupt Sources**

| Interrupt Source | DMAC Activation |
|---|---|
| Interrupts generated by receive error flag (ER) | Disabled |
| Interrupts generated by receive-FIFO-data-full (RDF), receive-data-ready (DR) or timeout (TO) | Enabled |
| Interrupts generated by break (BRK) or overrun error (ORER) | Disabled |
| Interrupts generated by transmit FIFO data empty (TDFE) | Enabled |

## 106.4 Usage Notes

### (1) Break detection and operation

Break signals can also be detected by reading the HRX pin value directly when a framing error (FER) is detected. In the break state, the input values from the HRX pin are all 0s. So, the parity error flag (PER) may be set after the FER flag is set to 1.

Although the HSCIF stops receive data transfer to HSFRDR after detecting a break, it continues data reception.

### (2) Sending a break signal

The input/output condition and level of the HTX pin are determined by the SPB2IO and SPB2DT bits in HSSPTR. This enables to send a break signal.

The pin does not function as the HTX pin from the initialization of the serial transmitter to setting of the TE bit (enabling transmission). In this period, the marked state is substituted by the value of the SPB2DT bit. Therefore, the SPB2IO and SPB2DT bits should be set to 1 (designating output and high level) beforehand.

To send a break signal during serial transmission, clear the SPB2DT bit to 0 (low level), and then clear the TE bit to 0 (halting transmission). When the TE bit is cleared, the transmitter is initialized regardless of the current transmission state, and the HTX pin outputs 0.

### (3) Data sampling timing and reception margin in asynchronous mode

The HSCIF operates on the base clock with a frequency multiplied by the number which is set as a sampling rate for the bit rate.

In reception, the HSCIF performs the internal synchronization by sampling the falling edge of the start bit using the base clock. In addition, the HSCIF takes receive data at the rising edge of the S/2 pulse (when S is an even number) or (S+1)/2 pulse (when S is an odd number) on the base clock when sampling rate is S.

Figure 106.12 shows the timing of this operation when S = 16.

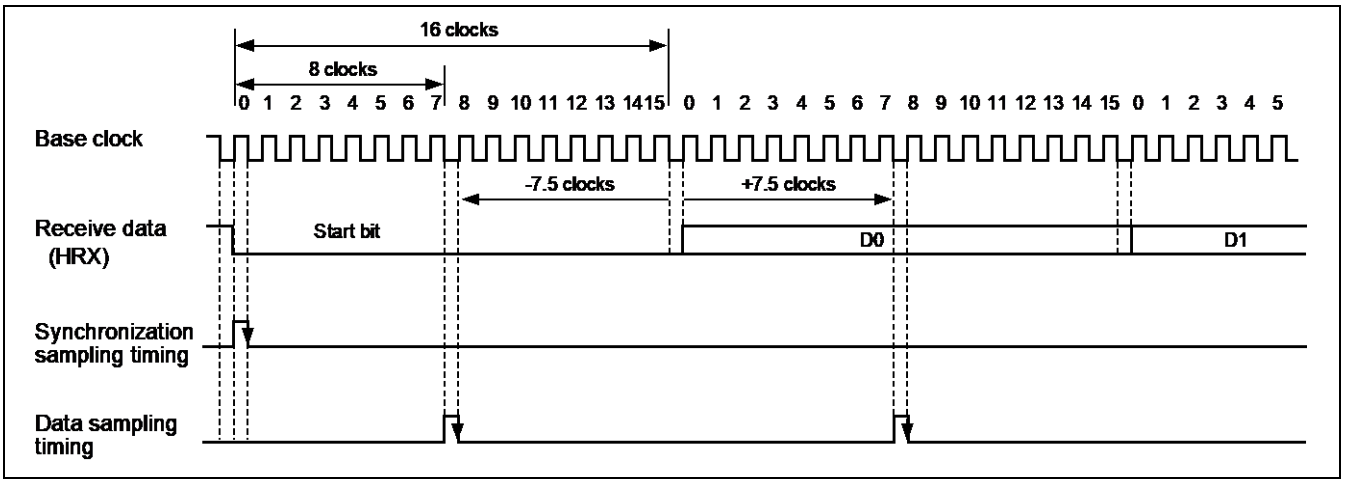



**Figure 106.12 Timing Chart of Receive Data Sampling**

The reception margin in asynchronous mode is given by formula (1).

$$M = \left| (0.5 - \frac{1}{2N}) - (L - 0.5)F - \frac{|D - 0.5|}{N}(1 + F) \right| \times 100\% \quad \cdots\cdots \text{Expression (1)}$$

M: Receive margin (%)

N: Ratio of bit rate to clock (N = sampling rate)

D: Clock duty (D = 0 to 1.0)

L: Frame length (L = 9 to 12)

F: Absolute value of clock frequency deviation

Assuming F = 0, D = 0.5 and sampling rate = 16 for expression (1), the reception margin obtained with expression (2) is 46.875% as shown below:

Assuming D = 0.5 and F = 0

$$M = (0.5 - 1/(2 \times 16)) \times 100\%$$

$$= 46.875\% \ldots\ldots \text{expression (2)}$$

This is a theoretical value. A reasonable margin allowed in system designs is 20% to 30%.

**(4) Reception margin and baud rate error**

The value of 46.875% obtained by the above expression (2) indicates the reception margin when the baud rate error is 0 (F = 0). If there is no error in the reception and transmission baud rates, reception is possible even with misalignment of approx. 1/2 bit. If there is an error in the reception and transmission baud rates, the errors are accumulated up to the stop bit reception, which reduces the reception margin. The allowable baud rate error can be obtained by modifying the F in expression (1). When D = 0.5:

$$F = \{(15/32 - M)/(L - 0.5)\} \times 100\ (\%) \ldots\ldots \text{expression (3)}$$

By using expression (3), the relationship between the allowable error and reception margin with the frame length = 12 can be summarized as follows:

| Allowed Error (%) | Reception Margin (%) |
|---|---|
| 4.07 | 0 |
| 3.64 | 5 |
| 3.20 | 10 |
| 2.33 | 20 |
| 1.46 | 30 |

**(5) Usage method of SRHP bits**

The method for using the SRHP bits in the sampling rate register is described below.

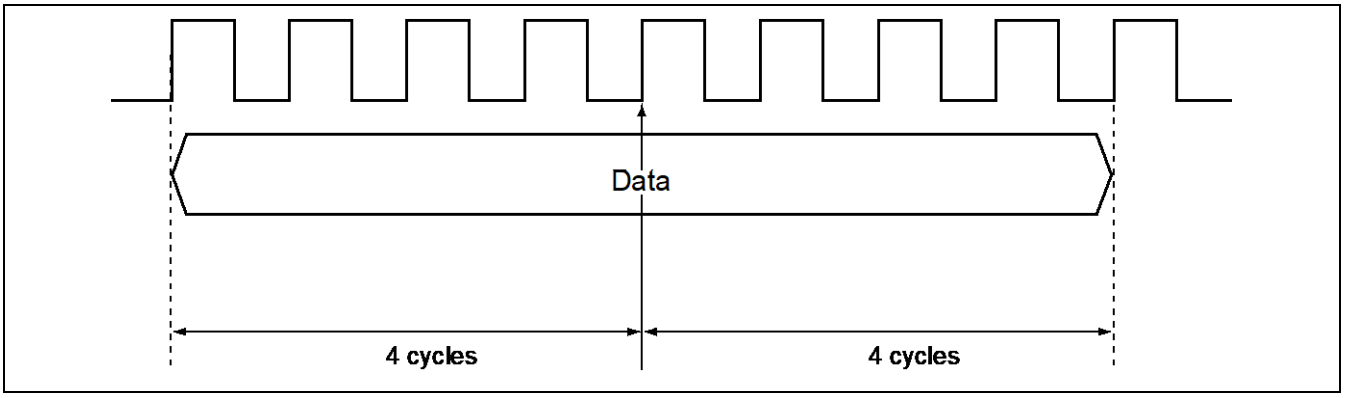



**Figure 106.13 Sampling with Invalid SRHP Bits**

Figure 106.13 shows a sampling example in which the SRHP bits are invalid (sampling rate = 8). In this case, the HSCIF samples data at half of the sampling rate, which is at the rising edge of the fourth pulse of the clock. This allows a setup margin and hold margin of 50% each to be provided.

However, if the ratio between the baud rate and sampling clock is not 1:1, either the setup margin or hold margin is omitted in data reception of a single frame. If the setup margin is omitted, the hold margin increases. On the other hand, if the hold margin is omitted, the setup margin increases. Figure 106.14 and Figure 106.15 show examples.

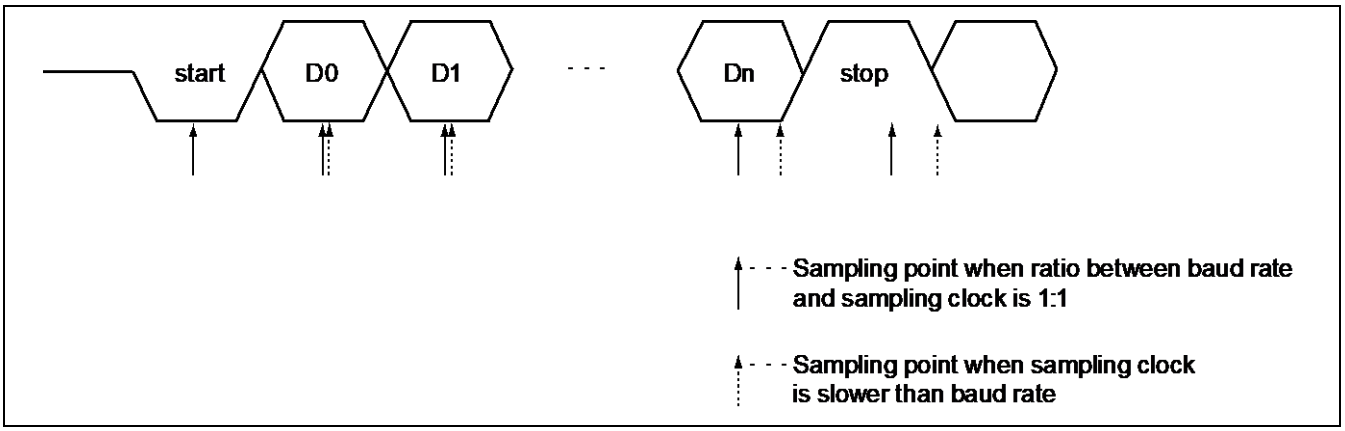



**Figure 106.14 Sampling Point with Sampling Clock Slower Than Baud Rate**

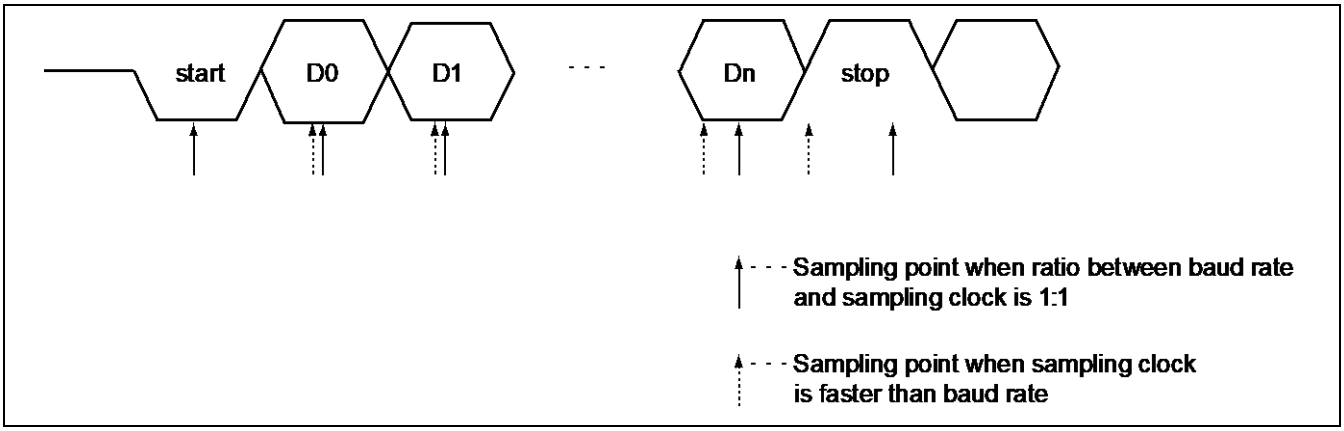



**Figure 106.15 Sampling Point with Sampling Clock Faster Than Baud Rate**

The ratio between the baud rate and sampling clock can be used to determine which margin among the setup margin and hold margin is to be omitted and which is to be increased. Based on this, the margin within a single frame can be increased by intentionally increasing the margin to be omitted in advance. The sampling point can be moved by setting a value in the SRHP bits. Figure 106.16 shows an example.

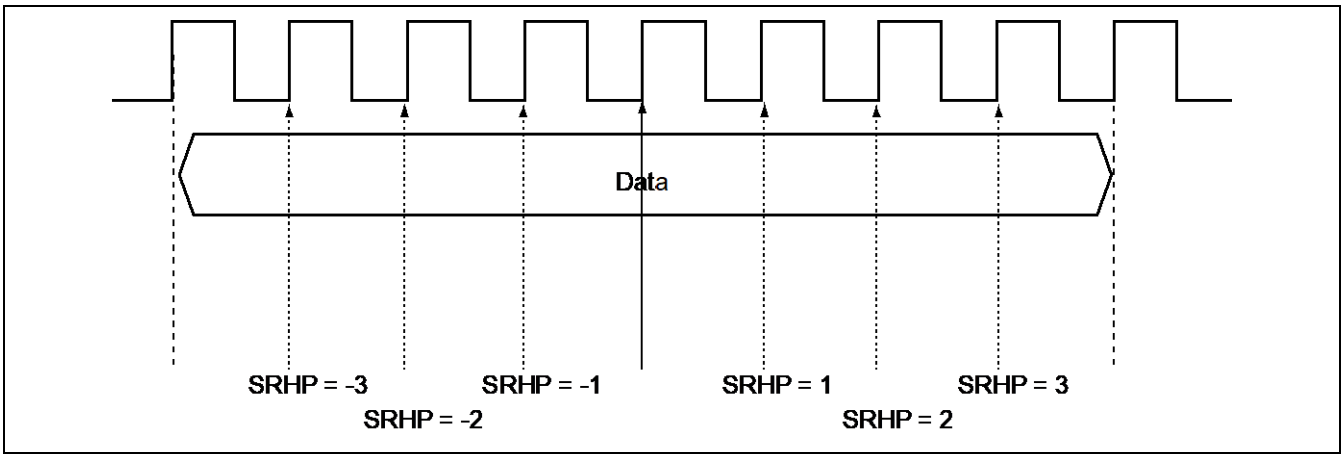



**Figure 106.16 Sampling Point Moved by SRHP Bits**

**(6) Sampling rate and bit rate settings and respective margin**

To set the baud rate in the HSCIF, in addition to setting the clock division ratio, the two registers, bit rate register (HSBRR) and sampling rate register (HSSRR), need to be set. These registers must be set so that the margin calculated by expression (1) in section 106.4 (3) will be a sufficient value.

The bit rate error is small when the value obtained by dividing the divided clock frequency used in the HSCIF with the sampling rate is near the desired baud rate. If there are many combinations of bit rate and sampling rate with the same bit rate error, choosing a combination with a large sampling rate provides a large margin. This is because when the bit rate error is constant in expression (1) in section 106.4 (3), that is, when the "absolute value of clock frequency deviation (F)" is constant, the margin (M) increases in accordance with the sampling rate (N).

If the sampling rate is made larger in a combination of bit rate and sampling rate with different bit rate errors, the bit rate error is increased. Accordingly, the "absolute value of clock frequency deviation (F)" on the right side in expression (1) in section 106.4 (3) gets larger and the margin (M) smaller.

Refer to the following procedure when selecting the sampling rate and bit rate settings.

1. Obtain the bit rate with the smallest bit rate error for sampling rates from 8 to 32.

2. Using expression (1) in section 106.4 (3), calculate the margin for each combination of sampling rate and bit rate which was obtained in step 1.

3. Select the combination with the largest margin among the combinations of sampling rate and bit rate.

## 106.5 Safety Mechanisms

HSCIF module has no mechanism of FuSa.

# 107. I2C Bus Interface (I2C)

## 107.1 Overview

### 107.1.1 Features

Fourth generation R-Car series products have up to four I2C bus interfaces conformant with the Philips Semiconductors (now NXP Semiconductors) I2C bus (Inter-IC bus) specification. However, the configuration of registers that control the I2C bus differs partly from the NXP Semiconductors register configuration. The I2C bus interface has the following features. Note that "module clock (MOD_CLK)" refers to the peripheral module clock in this section.

Module clock is S0D6_PERφ for all channels.

- The I2C interfaces support the I2C bus specification from NXP Semiconductors
- Multi-master compatible
- Seven- or ten-bit address compatible master
- Seven-bit slave address
- Fast mode compatible
- Variable clock frequencies
- DMA transfer support for Master/Slave Transmission Master/Slave Reception.
- Supporting Fast mode+

Figure 107.1 shows a block diagram of the I2C bus interface.

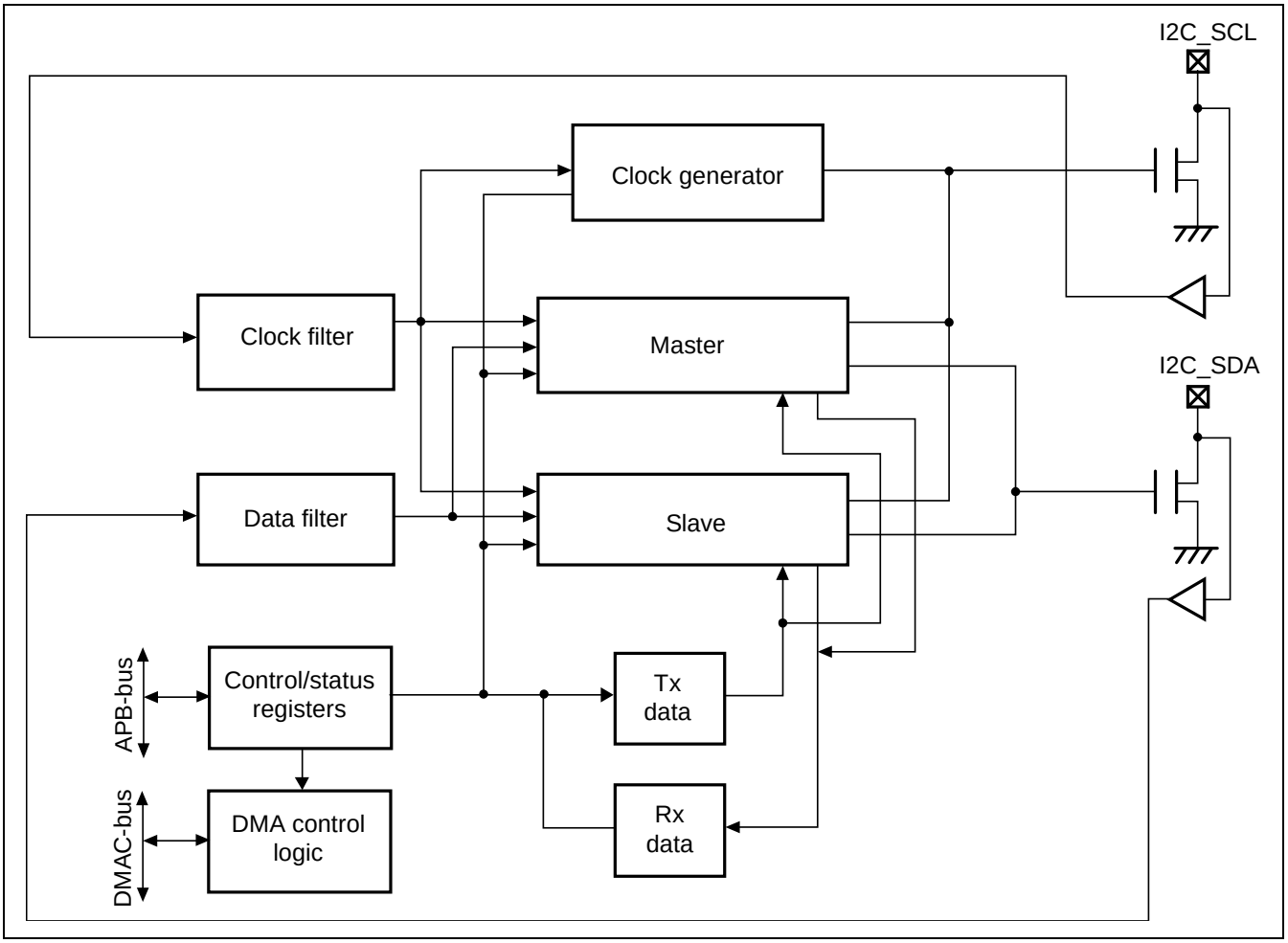



**Figure 107.1 Block Diagram of I2C Bus Interface**

### 107.1.2 External Pins

Table 107.1 lists the pins used in the I2C bus interface.

**Table 107.1 I2C Bus Interface**

| Channel Number | Pin Name | I/O | Function | Output Buffer Type*1 |
|---|---|---|---|---|
| 0 | SCL0 | I/O | SCL: | OD |
| | SDA0 | | I2C serial clock input/output pin | |
| 1 | SCL1 | | SDA: | OD |
| | SDA1 | | I2C serial data input/output pin | |
| 2 | SCL2 | | | OD |
| | SDA2 | | | |
| 3 | SCL3 | | | OD |
| | SDA3 | | | |

Note:1 Output buffer type: "OD" is open drain buffer. OD assigned channel pins are selected 1.8 V I/O or 3.3V I/O by 7.2.7 POWER Condition Control Register (POCn) register. (Not 3.3V tolerant IO cell)

The I2C buffer in this LSI is not compliant with the 5V-input. When turning off the I/O power source of this LSI, turn off the power source of the pull-up resistors connected to the I2C pins.

### 107.1.3 Register Configuration

(include Safety Mechanism related registers)

The base address for each channel of registers of the I2C bus interface is as follows:

- I2C0: H'E650 0000
- I2C1: H'E650 8000
- I2C2: H'E651 0000
- I2C3: H'E66D 0000

Each channel of the I2C bus interface has the registers below.

**Table 107.2 Register Configuration**

| Register Name | Abbreviation | R/W | Offset Address from Base Address | Initial Value | Size |
|---|---|---|---|---|---|
| Slave control register | ICSCR | R/W | H'00 | H'0000 0000 | 32 |
| Master control register | ICMCR | R/W | H'04 | H'0000 0000*3 | 32 |
| Slave status register | ICSSR | R/(W)*1 | H'08 | H'0000 0000 | 32 |
| Master status register | ICMSR | R/(W)*2 | H'0C | H'0000 0000 | 32 |
| Slave interrupt enable register | ICSIER | R/W | H'10 | H'0000 0000 | 32 |
| Master interrupt enable register | ICMIER | R/W | H'14 | H'0000 0000 | 32 |
| Clock control register | ICCCR | R/W | H'18 | H'0000 0000 | 32 |
| Slave address register | ICSAR | R/W | H'1C | H'0000 0000 | 32 |
| Master address register | ICMAR | R/W | H'20 | H'0000 0000 | 32 |
| Receive data register | ICRXD | R | H'24 | H'0000 0000 | 32 |
| Transmit data register | ICTXD | W | H'24 | H'0000 0000 | 32 |
| Clock control register 2 | ICCCR2 | R/W | H'28 | H'0000 0000 | 32 |
| SCL mask control register | ICMPR | R/W | H'2C | H'0000 0000 | 32 |
| SCL high control register | ICHPR | R/W | H'30 | H'0000 0000 | 32 |
| SCL low control register | ICLPR | R/W | H'34 | H'0000 0000 | 32 |
| DMA enable register | ICDMAER | R/W | H'3C | H'0000 0000 | 32 |
| First bit setup cycle register | ICFBSCR | R/W | H'38 | H'0000 0004 | 32 |

Notes: Do not write to any addresses other than listed above. Operations cannot be guaranteed if writing is attempted.
Values read from addresses other than those listed above are undefined.
Although the actual register name and its abbreviation have channel number as a suffix respectively, it is omitted in this section (the number "2" of ICCCR2 is not channel number).
Operation cannot be guaranteed if the register is not accessed as a longword (32bit)

1. Only 0 can be written to bits 4 to 0 to clear the flags.
2. Only 0 can be written to bits 6 to 0 to clear the flags.
3. Bits 30, 29, 22, 21, 14, 13, 6, and 5 are undefined.

## 107.2 Register Description

Registers in the I2C are allocated to and arranged in the address space of the internal bus.

Legend for Register Description

Initial value: Register value after a reset

—: Undefined value

R/W: Readable/writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

R/WC1: Readable/writable. Writing 1 initializes the bit. Writing 0 is ignored.

R: Read-only. When the bit is reserved, the write value should always be 0. Writing 1 to these bits can cause a malfunction of I2C.

—/W: Write-only. The read value is undefined.

### 107.2.1 Slave Control Register (ICSCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | SCSS | SDBS | SIE | GCAE | FNA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | □ | All 0 | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 to 5 | □ | All 0 | R | Reserved<br>The write value should always be 0. |
| 4 | SCSS | 0 | R/W | Slave Clock Stretch Select<br>This bit is used to select the timing of Clock Stretch.<br>0: Clock Stretch is in front of ACK/NACK<br>1: Clock Stretch is next to ACK/NACK |
| 3 | SDBS | 0 | R/W | Slave Data Buffer Select<br>This bit is used to select the data buffer. Select the single-buffer mode supported in this module.<br>When this bit is set to 1, the single-buffer mode is selected. SCL will be held low from the timing when the receive data register acquires the data packet until the SDR flag is cleared.<br>0: Setting prohibited<br>1: Single-buffer mode |
| 2 | SIE | 0 | R/W | Slave Interface Enable<br>This bit must be set for the slave operation. If this bit is low, the slave interface is reset. |
| 1 | GCAE | 0 | R/W | General Call Acknowledgement Enable<br>When slave devices are to issue an acknowledgement in response to a general call address sent from a master, this bit must be set to 1. |
| 0 | FNA | 0 | R/W | Forced Non Acknowledgement<br>In the slave receive mode, the level of this bit is sent to the transmitting device as the acknowledge signal. This bit is set to 0 during the period that the data packet is being received, and set to 1 on completion of data reception.<br>Forced non acknowledgement is returned to the master during slave reception.<br>When the slave has received the last byte of data in a data packet, the slave communicates with the master by sending a NACK, meaning that the acknowledgement is not driven. The master issues a stop on the bus after receiving a NACK. The setting of this bit does not affect the acknowledgement of the slave address. |

### 107.2.2 Slave Status Register (ICSSR)

Bits 0 to 4 among the status bits in the slave status register are cleared by writing 0 to the respective status bit positions. The individual bits are held 1 until 0 is written to (other than the GCAR and STM bits).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | GCAR | STM | SSR | SDE | SDT | SDR | SAR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W* | R/W* | R/W* | R/W* | R/W* |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 | — | 0 | R | Reserved<br>The write value should always be 0. |
| 6 | GCAR | 0 | R | General Call Address Received<br>Indicates that the address received from the bus is a general call address (H'00). This status bit does not cause an interrupt.<br>This bit is automatically cleared by hardware when the SIE bit (bit 2 in the slave control register) is set to 0 or when the SSR bit (bit 4 in this register) is set to 1. |
| 5 | STM | 0 | R | Slave Transmit Mode<br>Indicates whether the current slave transmit mode is read or write. When this bit is set to 0, the mode is write (slave reception). When this bit is set to 1, the mode is read (slave transmission). This status bit does not cause an interrupt.<br>This bit is automatically cleared by hardware when the SIE bit (bit 2 in the slave control register) is set to 0 or when the SSR bit (bit 4 in the slave status register) is set to 1. |
| 4 | SSR | 0 | R/W* | Slave Stop Received<br>A stop condition has been output on the bus. This status bit becomes active after the rising edge of SDA during reception of a stop bit. |
| 3 | SDE | 0 | R/W* | Slave Data Empty<br>Indicates that data to be transmitted has been loaded into the shift register. At the start of byte data transmission, the contents of the ICTXD register are loaded into a shift register ready for outputting data on the bus. This status bit indicates that data has been loaded and the ICTXD register is again ready for further data. This status bit becomes active on the falling edge of SCL before reception of the first data bit.<br>- When bit SDE = 1 & bit TSDMAE = 0, the interrupt SDE is issued to CPU, so the transmit data register is to accept further data.<br>- When bit SDE = 1 & bit TSDMAE = 1, enable DMA transfer request to DMAC.<br>During the single-buffer mode, this bit must be reset every time new data has been written to ICTXD. If not, the slave will hold SCL low after the current data transfer has finished to stop the bus.<br>During DMA transmit mode, the SDE bit is also set after data is loaded into the shift register. In this case, this bit is cleared by DMA logic control. This mechanism to assure the continuous data transfer without software control. For the last byte data transmitted, this bit is cleared by software. |
| 2 | SDT | 0 | R/W* | Slave Data Transmitted<br>A byte of data has been transmitted to the bus. This bit becomes active after the falling edge of SCL during reception of the last data bit. |
| 1 | SDR | 0 | R/W* | Slave Data Received<br>A byte of data has been received from the bus and is ready for reading from the receive data register. This bit becomes active after the falling edge of SCL during reception of the last data bit. During the single-buffer mode, this bit must be reset after data has been read from the ICRXD register.<br>- When bit SDR = 1 & bit RSDMAE = 0, the interrupt SDR is issued to CPU, so the transmit data register is to accept further data.<br>- When bit SDR = 1 & bit RSDMAE = 1, enable DMA transfer request to DMAC.<br>When SDBS is set to 1, SCL will be held low from the timing when the receive data register acquires the data packet until the SDR flag is cleared.<br>During DMA receive mode, the SDR bit is also set after data is received into RXD register. In this case, this bit is cleared by DMA logic control. This mechanism to assure the continuous data transfer without software control. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | SAR | 0 | R/W* | Slave Address Received<br>Indicates that the slave has recognized its own address on the bus (defined by the contents of the slave address register). If the general call acknowledgement enable bit is enabled in the slave control register, then this status bit is also set to 1 even if the address on the bus is a general call address. In this case, the GCAR bit in this register is used to determine whether or not the address is a general call address. The STM bit indicates whether the access is read (high) or write (low). This status becomes active after the falling edge of SCL during reception of the last address bit. The slave holds the SCL signal low from the start of the ACK phase until the software resets this status bit. |

Note: * This bit can be read from or written to. Writing 0 clears this bit to 0 and writing 1 is ignored.

### 107.2.3 Slave Interrupt Enable Register (ICSIER)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | SSRE | SDEE | SDTE | SDRE | SARE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 to 5 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 4 | SSRE | 0 | R/W | Slave Stop Received Interrupt Enable<br>0: Disables the SSR interrupt.<br>1: Enables the SSR interrupt. |
| 3 | SDEE | 0 | R/W | Slave Data Empty Interrupt Enable<br>0: Disables the SDE interrupt.<br>1: Enables the SDE interrupt. |
| 2 | SDTE | 0 | R/W | Slave Data Transmitted Interrupt Enable<br>0: Disables the SDT interrupt.<br>1: Enables the SDT interrupt. |
| 1 | SDRE | 0 | R/W | Slave Data Received Interrupt Enable<br>0: Disables the SDR interrupt.<br>1: Enables the SDR interrupt. |
| 0 | SARE | 0 | R/W | Slave Address Received Interrupt Enable<br>0: Disables the SAR interrupt.<br>1: Enables the SAR interrupt. |

### 107.2.4 Slave Address Register (ICSAR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | SADD0_6 | SADD0_5 | SADD0_4 | SADD0_3 | SADD0_2 | SADD0_1 | SADD0_0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 | — | 0 | R | Reserved<br>The write value should always be 0. |
| 6 to 0 | SADD0_6 to SADD0_0 | All 0 | R/W | Slave Address<br>This is the unique 7-bit address allocated to the slave on the I2C bus. The slave interface compares this address with the first seven bits transmitted as the slave address, at the beginning of a data packet transmission. |

### 107.2.5 Master Control Register (ICMCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | — | — | 0 | 0 | 0 | 0 | 0 | 0 | — | — | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | MDBS | FSCL | FSDA | OBPC | MIE | TSBE | FSB | ESG |
| Initial value: | 0 | — | — | 0 | 0 | 0 | 0 | 0 | 0 | — | — | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0, except bits 30, 29, 22, 21, 14, and 13 in which case, initial value is undefined. | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 | MDBS | 0 | R/W | Master Data Buffer Select<br>This bit is used to select the data buffer mode. Select the single-buffer mode supported in this module. When this bit is set to 1, the single-buffer mode is selected. SCL will be held low from the timing when the receive data register acquires the data packet until the MDR flag is cleared.<br>0: Setting prohibited<br>1: Single-buffer mode |
| 6 | FSCL | — | R/W | Forced SCL<br>This bit controls the status of the I2C_SCL pin (reading reflects the current status of the I2C_SCL pin). When the OBPC bit is set, this bit directly controls the SCL line on the bus.<br>During a read cycle, the level on this bit (which includes the reset level) will change depending on the level on I2C_SCL since it directly reflects the level on the I2C_SCL. |
| 5 | FSDA | — | R/W | Forced SDA<br>This bit controls the status of the I2C_SDA pin (reading reflects the busy status level on the I2C bus). When the OBPC bit is set, then this bit directly controls the SDA line on the bus.<br>During a read cycle, the level on this bit (which includes the reset level) will show the busy status of the I2C bus (1 for busy; 0 for not busy). |
| 4 | OBPC | 0 | R/W | Override Bus Pin Control<br>When this bit is set to 1, the FSDA and FSCL bits in this register control SDA and SCL directly. |
| 3 | MIE | 0 | R/W | Master Interface Enable<br>When this bit is set to 1, the master interface is enabled.<br>The master interface is disabled when this interface loses in arbitration of bus mastership even MIE bit indicates 1. If 0 is not subsequently written to the bit, the master interface is restored on detection of a STOP condition on the I2C bus. Regarding the state following a loss in arbitration of bus mastership, see the description of the MAL bit in ICMSR. |
| 2 | TSBE | 0 | R/W | Start Byte Transmission Enable<br>The write value for this bit should always be 0. |
| 1 | FSB | 0 | R/W | Forced Stop onto the Bus<br>When this bit is set to 1, the master transmits a STOP condition on the bus at the end of the current transfer. If ESG is also set, the master immediately transmits a START condition and begins transmitting a new data packet. If ESG is not set, the master enters the idle state. |
| 0 | ESG | 0 | R/W | Enable Start Generation<br>When this bit is set to 1, the master starts transmission of a data packet. If the bus is idle when ESG is set, the master transmits a START condition on the bus and then transmits the slave address. If the master is transferring data when ESG is set, at the end of that data byte transfer, the master transmits a repeated START condition before transmitting the slave address. When transmitting a data packet, the software must reset this bit when the slave address has been transmitted, otherwise a repeated START condition is transmitted after every transmission is completed. |

### 107.2.6 Master Status Register (ICMSR)

The status bits (bits 0 to 6) in the master status register are cleared by writing 0 to the respective status bit positions. The individual status bits are held 1 until a reset by writing 0 to the appropriate bit position.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | MNR | MAL | MST | MDE | MDT | MDR | MAT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* | R/W* |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 | — | 0 | R | Reserved<br>The write value should always be 0 |
| 6 | MNR | 0 | R/W* | Master NACK Received<br>When this bit is set to 1, this bit indicates that the master has received a NACK response (the SDA line is high during the acknowledge cycle on the bus) to either an address or data transmission. **<br>To restart the address transmission from idle state on Master, this bit must be cleared. |
| 5 | MAL | 0 | R/W* | Master Arbitration Lost<br>In a multi-master system, when this bit is set to 1, it indicates that the master has lost arbitration to other masters on the bus.<br>At this point, the master interface is disabled then Master enters the idle state again. |
| 4 | MST | 0 | R/W* | Master Stop Transmitted<br>When this bit is set to 1, it indicates that the master has sent a STOP condition on the bus. A STOP condition can be sent either as a result of the setting of the forced stop bit in the control register, or from a NACK being received from a slave during a slave receive data packet. |
| 3 | MDE | 0 | R/W* | Master Data Empty<br>At the start of a byte data transmission, the contents of the transmit data register are loaded into a shift register ready for transmitting on the bus. This status bit becomes active:<br>- When bit MDE = 1, MAT = 0 & bit TMDMAE = 0, the interrupt MDE is issued to CPU, so the transmit data register is to accept further data.<br>- When bit MDE = 1, MAT = 0 & bit TMDMAE = 1, enable DMA transfer request to DMAC.<br>During master transmission, the MDE bit is also set at the same time as the MAT bit is set after transmission of the slave address. In this case, you need to clear the MDE and MAT bits after the ICMCR's ESG bit is cleared. The clearing will restart the data transmission.<br>During DMA transmit mode, the MDE bit is also set after data is loaded into the shift register. In this case, this bit is cleared by DMA logic control. This mechanism to assure the continuous data transfer without software control. For the last byte data transmitted, this bit cleared by software. |
| 2 | MDT | 0 | R/W* | Master Data Transmitted<br>Byte data has been sent to the slave on the bus. This status bit becomes active after the falling edge of SCL during reception of the last data bit. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | MDR | 0 | R/W* | Master Data Received<br>Byte data has been received from the bus and is in the receive data register. This status bit becomes active after the falling edge of SCL during reception of the last data bit. During single-buffer mode, this status bit must be reset after data has been read from the receive data register.<br>- When bit MDR = 1, MAT = 0 & bit RMDMAE = 0, the interrupt MDR is issued to CPU, so the receive data register is to accept further data.<br>- When bit MDR = 1, MAT = 0 & bit RMDMAE = 1, enable DMA transfer request to DMAC.<br>When MDBS is set to 1, SCL will be held low from the timing when the receive data register acquires the data packet until the MDR flag is cleared.<br>During master reception mode, the MDR bit is also set at the same time as the MAT bit is set after transmission of the slave address. In this case, you must clear the MDR and MAT bits after the ICMCR's ESG bit is cleared. Clearing will start the data reception.<br>During DMA receive mode, the MDR bit is also set after data is received into RXD register. In this case, this bit is cleared by DMA logic control. This mechanism to assure the continuous data transfer without software control. |
| 0 | MAT | 0 | R/W* | Master Address Transmitted<br>The master has been transmitted the slave address byte of a data packet. This bit becomes active after the falling edge of SCL during reception of the ACK bit following transmission of the address. |

Note: * This bit can be read from or written to. Writing 0 clears this bit to 0 and writing 1 is ignored.

Note: ** In the case “ICMCR.ESG = 1”, STOP condition will never sent from Master I/F whenever ICMSR.MNR get asserted.

### 107.2.7 Master Interrupt Enable Register (ICMIER)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | MNRE | MALE | MSTE | MDEE | MDTE | MDRE | MATE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 | — | 0 | R | Reserved<br>The write value should always be 0. |
| 6 | MNRE | 0 | R/W | Master NACK Received Interrupt Enable<br>0: Disables the MNR interrupt.<br>1: Enables the MNR interrupt. |
| 5 | MALE | 0 | R/W | Master Arbitration Lost Interrupt Enable<br>0: Disables the MAL interrupt.<br>1: Enables the MAL interrupt. |
| 4 | MSTE | 0 | R/W | Master Stop Transmitted Interrupt Enable<br>0: Disables the MST interrupt.<br>1: Enables the MST interrupt. |
| 3 | MDEE | 0 | R/W | Master Data Empty Interrupt Enable<br>0: Disables the MDE interrupt.<br>1: Enables the MDE interrupt. |
| 2 | MDTE | 0 | R/W | Master Data Transmitted Interrupt Enable<br>0: Disables the MDT interrupt.<br>1: Enables the MDT interrupt. |
| 1 | MDRE | 0 | R/W | Master Data Received Interrupt Enable<br>0: Disables the MDR interrupt.<br>1: Enables the MDR interrupt. |
| 0 | MATE | 0 | R/W | Master Address Transmitted Interrupt Enable<br>0: Disables the MAT interrupt.<br>1: Enables the MAT interrupt. |

### 107.2.8 Master Address Register (ICMAR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SADD1_6 | SADD1_5 | SADD1_4 | SADD1_3 | SADD1_2 | SADD1_1 | SADD1_0 | STM1 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 to 1 | SADD1_6 to SADD1_0 | All 0 | R/W | Slave Address<br>These bits are the address of the slave which the master communicates with. |
| 0 | STM1 | 0 | R/W | Slave Transfer Mode<br>This bit specifies the mode in which the slave operates.<br>Bit STM1 sets the operating mode (transmit or receive mode) of the slave, which is an external slave device whose address matches the slave address (SADD1) sent from the master. The slave device is automatically set to transmit/receive mode by hardware on reception of the STM1 signal.<br>When this bit is set to 1, it indicates a read operation. When this bit is cleared to 0, it indicates a write operation. |

### 107.2.9 Clock Control Register (ICCCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | SCGD | | | | | | CDF | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved<br>The read value is always 0. The write value should always be 0. |
| 8 to 3 | SCGD | All 0 | R/W | SCL Clock Generation Divider<br>When operation is in master mode, the SCL clock is generated from the internal clock by using SCGD as the ratio. This is also the operating clock in slave-mode operation while SCL is being held low to stop the bus after an overflow has occurred. SCGD must be specified in both master and slave modes. The formula expressing the relationship is:<br>Equation 2: SCL rate calculation<br>SCLfreq = I2Cck/(20 + SCGD × 8 + F[(tICF + tr + IntDelay) × I2Cck])<br>I2Cck: I2C internal clock frequency<br>tICF: I2C SCL falling time (depending on external load)<br>tr: I2C SCL rising time (depending on external load)<br>IntDelay: LSI internal delay corresponds to output buffer type.<br>Open drain buffer: 50 ns (typ.), 110 ns (max.)<br>F[n]: n rounded down to an integer<br>Suggested settings for CDF and SCGD for CPU speeds and the two I2C bus speeds are given in Table 107.3. |
| 2 to 0 | CDF | All 0 | R/W | Clock Division Factor<br>The internal clock used in most blocks in the I2C module is a divided module clock. The internal I2C clock is generated from the module clock using the CDF as the division ratio:<br>Equation 1: I2C internal clock frequency calculation<br>I2Cck = MOD_CLK/(1 + CDF)<br>MOD_CLK: module clock frequency<br>Ensure that minimum values for setup and hold times of the SDA signal relative to the SCL signal on the bus are satisfied.<br>The clock frequency is to ensure that the glitch filtering will operate with glitches of up to 50 ns as described in the fast mode I2C specification.<br>Note: CDF must be set so that the clock frequency (I2Cck) is lower than 20 MHz. |

**Table 107.3 Recommended Settings for CDF and SCGD*1 (Normal mode)**

- Recommended register values for open drain buffer

| | 100 kHz | | 400 kHz | |
|---|---|---|---|---|
| **Module Clock Frequency** | **CDF(*2)** | **SCGD(*3)** | **CDF(*2)** | **SCGD(*3)** |
| 133.33 MHz | 6 | 21 | 6 | 3 |
| Error | | -0.80% | | -0.80% |

- Recommended register values for LVTTL buffer (low drive only)

| | 100 kHz | | 400 kHz | |
|---|---|---|---|---|
| **Module Clock Frequency** | **CDF(*2)** | **SCGD(*3)** | **CDF(*2)** | **SCGD(*3)** |
| 133.33 MHz | 6 | 21 | 6 | 3 |
| Error | | -0.28% | | +1.31% |

Notes: 1. These recommended values are for the SCL rate with an external load that produces tICF = 3 ns and tr = 150 ns.
Recalculate to obtain the actual values after measuring tICF and tr under actual conditions of usage.
2. Recommended value for CDF bit of ICCCR.
3. Recommended value for SCGD bit of ICCCR.

### 107.2.10 Receive and Transmit Data Registers (ICRXD and ICTXD)

Reading from or writing to these registers access different physical internal registers. When data is to be transmitted, the contents of the shift register are loaded via TXD. After data has been received into the shift register from the I2C bus, it is then loaded into RXD.

**(1) Receive data register (ICRXD)**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RXD | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, bits 31 to 24, 23 to 16, and 15 to 8 reflect the values of bits 7 to 0. The write value should always be 0. |
| 7 to 0 | RXD | All 0 | R | Read Receive Data<br>Data received by master or slave. |

**(2) Transmit data register (ICTXD)**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TXD | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 7 to 0 | TXD | All 0 | W | Write Transmit Data<br>Data to be transmitted by master or slave. |

### 107.2.11 Clock Control Register 2 (ICCCR2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | FMPE | — | — | — | — | CDFD | HLSE | SME |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, the value in bits 7 to 0 is set to bits 31 to 24, bits 23 to 16, and bits 15 to 8. The write value should always be 0. |
| 7 | FMPE | 0 | R/W | FM+ Enable<br>When this bit is set to 1, the Fast Mode+ of OD buffer is enable.<br>0: OD buffer is Standard / Fast mode.<br>1: OD buffer is Fast mode+. |
| 6 to 3 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 2 | CDFD | 0 | R/W | CDF Disable<br>When this bit is set to 1, the module clock is used for internal clocks except the clock filter. For SCL rate computation, see equation 3 in Note 2. The clock filter always operates at a clock frequency divided by the CDF value. |
| 1 | HLSE | 0 | R/W | HIGH/LOW Separate Control Enable<br>When this bit is set to 1, the SCGD setting is ignored and SCL is generated with the division ratio set by ICHPR during a high period and with the division ratio set by ICLPR during a low period. For SCL rate computation, see equation 3 in Note 2. The program should be prepared so that ICHPR and ICLPR should be written to before setting this bit. |
| 0 | SME | 0 | R/W | SCL Mask Enable<br>When this bit is set to 1, a change of internal SCL is ignored after an SCL external input pin changes during the time equal to the number of internal clock cycles specified by ICMPR. For SCL rate computation, see equation 3 in Note 2. The program should be prepared so that ICMPR should be written to before setting this bit. |

Notes: 1. The control bits should be set to "CDFD = 0, HLSE = 0, and SME = 0" or "CDFD = 1, HLSE = 1, and SME = 1". Otherwise, operation cannot be guaranteed. ICCCR2 should be written to prior to writing ICCCR.

2. Equation 3: SCL Rate Computation (CDFD = 1, HLSE = 1, and SME = 1)

Rise: SMD ≧ F[tr * MOD_CLK] + 4 * (CDF + 1)

Fall : SMD ≧ F[(IntDelay + tICF) * MOD_CLK] + 4 * (CDF + 1)

If both Rise and Fall conditions can be satisfied:

SCL freq = MOD_CLK/(8 + 2 * SMD + SCLD + SCHD)

If the Rise or Fall conditions cannot be met:

SCL freq = MOD_CLK/(8 + (F[ (tr + “rise internal delay”) * MOD_CLK] + 3 * (CDF + 1)) +
(F[ (IntDelay + tICF + “fall internal delay”) * MOD_CLK] + 3 * (CDF + 1) ) + SCHD + SCLD)

rise/fall internal delay: Min: 0 ns Max: (CDF + 1) / MOD_CLK ns

if specified SMD size satisfies either Rise or Fall side condition, satisfied side calculation can be replaced to SMD.

For example, Rise side condition can be satisfied, (F[(tr + "rise internal delay") * MOD_CLK] + 3 * (CDF + 1)) can be replaced to SMD.

MOD_CLK: Module Clock Frequency

SMD: SCL masked period (set by the SCL mask control register)

SCLD: I2C SCL low clock period (set by the SCL HIGH control register)

SCHD: I2C SCL high clock period (set by the SCL LOW control register)

tICF: I2C SCL falling time (depending on external load)

tr: I2C SCL rising time (depending on external load)

IntDelay: LSI internal delay corresponds to output buffer type.
Open drain buffer: 50 ns (typ.), 110 ns (max.)
If ICCR2.FMPE=1: 3 ns (min.), 29 ns (typ.), 55 ns (max.).

F[n]: n rounded down to an integer

Table 107.4 shows recommended register values for two types of I2C bus speeds.

**Table 107.4 Recommended Register Values when CDFD = 1, HLSE = 1, and SME = 1 (Duty variable mode)**

- Recommended Register Values for open drain buffer

| | 100 kHz | | | | | 400 kHz | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Module Clock Frequency** | **CDF (*1)** | **SCGD (*2)** | **SMD (*3)** | **SCHD (*4)** | **SCLD (*5)** | **CDF (*1)** | **SCGD (*2)** | **SMD (*3)** | **SCHD (*4)** | **SCLD (*5)** |
| 133.33 MHz | 6 (*6) | 0 | 50 | 607 | 626 | 6 (*6) | 0 | 50 | 108 | 125 |
| Error | | | -0.56% | | | | | -2.24% | | |

- Recommended Register Values for open drain buffer at Fast mode+ (When rise/fall internal delay = 0 ns (min))

| | 1 MHz | | | | |
|---|---|---|---|---|---|
| **Module Clock Frequency** | **CDF (*1)** | **SCGD (*2)** | **SMD (*3)** | **SCHD (*4)** | **SCLD (*5)** |
| 133.33 MHz | 6 (*6) | 0 | 21 | 30 | 43 |
| Error | | | -2.67% | | |

Note: tICF = 20ns, tr = 35ns, and IntDelay = 110ns(open drain buffer) /55ns(ICCCR2.FMPE=1)are used to calculate Error value.

Notes:
1. Recommended value for CDF bit of ICCCR.
2. Recommended value for SCGD bit of ICCCR.
3 Recommended value for SMD bit of ICMPR.
4. Recommended value for SCHD bit of ICHPR.
5. Recommended value for SCLD bit of ICLPR.
6. In case of connecting to device, whose $t_{HD;DAT}$ is smaller than 100ns, specifying 0 as CDF is recommended.

### 107.2.12 SCL Mask Control Register (ICMPR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SMD | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved<br>When these bits are read, the value in bits 7 to 0 is set to bits 31 to 24, bits 23 to 16, and bits 15 to 8. The write value should always be 0. |
| 7 to 0 | SMD | All 0 | R/W | SCL Mask Division<br>When SME = 1, a change of internal SCL is ignored after the external SCL changes during the time equal to the number of internal clock cycles specified by SMD. The value set in SMD should be smaller than SCHD and SCLD. |

### 107.2.13 SCL HIGH Control Register (ICHPR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SCHD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 15 to 0 | SCHD | All 0 | R/W | SCL HIGH Clock Division<br>When HLSE is 1 and internal SCL is driven high, the clock generated using the SCHD internal clock division ratio is output.<br>The period over the SCL signal is driven high can be calculated by the formula below.<br>Period over SCL is driven high = 1/MOD_CLK × (4 + SMD + SCHD) |

### 107.2.14 SCL LOW Control Register (ICLPR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SCLD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 15 to 0 | SCLD | All 0 | R/W | SCL LOW Clock Division<br>When HLSE is 1 and internal SCL is driven low, the clock generated using the SCLD internal clock division ratio is output.<br>The period over the SCL signal is driven low can be calculated by the formula below.<br>Period over SCL is driven low = 1/MOD_CLK × (4 + SMD + SCLD) |

### 107.2.15 DMA Enable Register (ICDMAER)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MDMACTSZ[7:0] | | | | | | | | RMDMATSZ[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TMDMATSZ[7:0] | | | | | | | | TMDM ACE | RMDM ACE | — | — | RSD MAE | TSD MAE | RMD MAE | TMD MAE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | MDMACTSZ | All 0 | R/W | DMA Master Continuous mode Transfer Size<br>Specify the number of transfer counts.<br>Transfer counts equals the number of DMAC descriptor number for this transfer.<br>This register should be set same value of DMACHCRB.DCNT[7:0] for this DMA transfer.<br>Minimum number is H'00. (Use 1 descriptor only)<br>Maximum number is H'FF. (Use 256 descriptor) |
| 23 to 16 | RMDMATSZ | All 0 | R/W | DMA Master Continuous Received mode 1unit Transfer Size register.<br>Specify byte number of 1 unit transfer size.<br>For example, Start/End bit is sent every 4byte, specify H'04.<br>Minimum size setting is "H'01". (Transfer of 1 unit is Once)<br>Maximum size setting is "H'00". (Transfer of 1 unit is 256 times) |
| 15 to 8 | TMDMATSZ | All 0 | R/W | DMA Master Continuous Transmitted mode 1unit Transfer Size register.<br>Specify byte number of 1 unit transfer size.<br>For example, Start/End bit is sent every 4byte, specify H'04.<br>Minimum size setting is "H'01". (Transfer of 1 unit is Once)<br>Maximum size setting is "H'00". (Transfer of 1 unit is 256 times) |
| 7 | TMDMACE | 0 | R/W | DMA Master Continuous Transmitted Enable<br>0: Disables DMA Continuous Transmitted mode<br>1: Enables DMA Continuous Transmitted mode |
| 6 | RMDMACE | 0 | R/W | DMA Master Continuous Received Enable<br>0: Disables DMA Continuous Received mode<br>1: Enables DMA Continuous Received mode |
| 5 to 4 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 3 | RSDMAE | 0 | R/W | DMA Slave Received Enable<br>0: Disables DMA Slave Received Mode.<br>1: Enables DMA Slave Received Mode. |
| 2 | TSDMAE | 0 | R/W | DMA Slave Transmitted Enable<br>0: Disables DMA Slave Transmitted Mode.<br>1: Enables DMA Slave Transmitted Mode. |
| 1 | RMDMAE | 0 | R/W | DMA Master Received Enable<br>0: Disables DMA Master Received Mode.<br>1: Enables DMA Master Received Mode. |
| 0 | TMDMAE | 0 | R/W | DMA Master Transmitted Enable<br>0: Disables DMA Master Transmitted Mode.<br>1: Enables DMA Master Transmitted Mode. |

### 107.2.16 First Bit Setup Cycle Register (ICFBSCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SRFBSC_4 | SRFBSC_3 | SRFBSC_2 | SRFBSC_1 | SRFBSC_0 | — | — | — | STFBSC_4 | STDBSC_3 | STFBSC_2 | STFBSC_1 | STFBSC_0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | FBSC_4 | FBSC_3 | FBSC_2 | FBSC_1 | FBSC_0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 28 to 24 | SRFBSC_4 to SRFBSC_0 | All 0 | R/W | Setting the delay time of the 1st data bit between SDA and SCL.<br>This bit is available at ICSCR.SCSS = B'1 and ICSSR.STM = B'0. |
| 23 to 21 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 20 to 16 | STFBSC_4 to STFBSC_0 | All 0 | R/W | Setting the delay time of the 1st data bit between SDA and SCL.<br>This bit is available at ICSCR.SCSS = B'1 and ICSSR.STM = B'1. |
| 15 to 5 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 4 to 0 | FBSC_4 to FBSC_0 | B'0 0100 | R/W | Setting the delay time of the 1st data bit between SDA and SCL |

Note: The delay time of the 1st data bit between SDA and SCL was calculated as below Table 107.5:
Figure 107.2 shows, Timing Chart of 1st bit setup delay cycle.

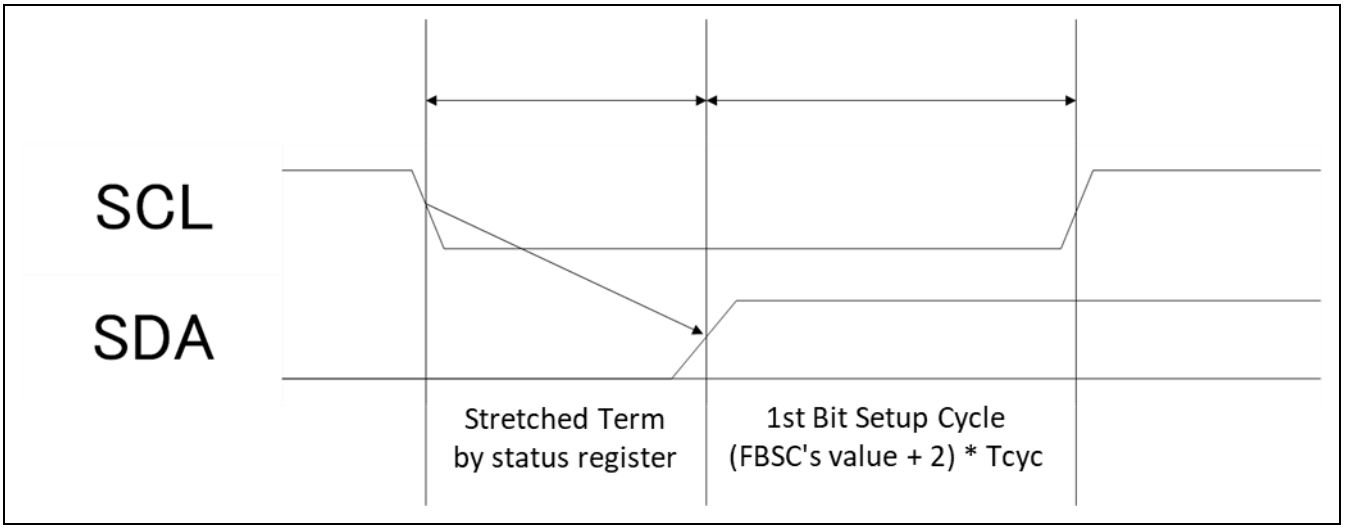



**Figure 107.2 Timing Chart of 1st bit setup delay cycle**

**Table 107.5 First Bit Setup Cycle Value (FBSC_4to0, STFBSC_4to0, and SRFBSC_4to0)**

| 1st Bit setup Cycle | FBSC's value | 1st Bit setup Cycle | FBSC's value |
|---|---|---|---|
| 02 × Tcyc | B'0 0000 | 18 × Tcyc | B'1 0000 |
| 03 × Tcyc | B'0 0001 | 19 × Tcyc | B'1 0001 |
| 04 × Tcyc | B'0 0010 | 20 × Tcyc | B'1 0010 |
| 05 × Tcyc | B'0 0011 | 21 × Tcyc | B'1 0011 |
| 06 × Tcyc | B'0 0100 | 22 × Tcyc | B'1 0100 |
| 07 × Tcyc | B'0 0101 | 23 × Tcyc | B'1 0101 |
| 08 × Tcyc | B'0 0110 | 24 × Tcyc | B'1 0110 |
| 09 × Tcyc | B'0 0111 | 25 × Tcyc | B'1 0111 |
| 10 × Tcyc | B'0 1000 | 26 × Tcyc | B'1 1000 |
| 11 × Tcyc | B'0 1001 | 27 × Tcyc | B'1 1001 |
| 12 × Tcyc | B'0 1010 | 28 × Tcyc | B'1 1010 |
| 13 × Tcyc | B'0 1011 | 29 × Tcyc | B'1 1011 |
| 14 × Tcyc | B'0 1100 | 30 × Tcyc | B'1 1100 |
| 15 × Tcyc | B'0 1101 | 31 × Tcyc | B'1 1101 |
| 16 × Tcyc | B'0 1110 | 32 × Tcyc | B'1 1110 |
| 17 × Tcyc | B'0 1111 | 33 × Tcyc | B'1 1111 |

Note: In Normal mode, Tcyc is period of I2Cck.
In Duty variable mode, Tcyc is period of MOD_CLK.

## 107.3 Operation

### 107.3.1 Data and Clock Filters

These blocks filter out glitches on signals coming from the I2C bus. Glitches up to one internal clock period in width are rejected (for details on the internal clock frequency, see section 107.2.9, Clock Control Register (ICCCR)). This is for the faster I2C bus rate (400 kHz) but does not violate the slower I2C bus rate specification.

These blocks also resynchronize bus signals with the internal clock.

### 107.3.2 Clock Generator

The clock generator has two functions. Firstly, it generates the SCL (I2C bus clock) according to commands of the master or slave interface. Secondly, it controls the internal clock rate, used by filtering blocks and the master and slave interfaces. This clock functions as a clock enable signal of the registers in these blocks.

### 107.3.3 Master/Slave Interfaces

These two interfaces run independently and in parallel. The master interface controls the transmission of address and data on the I2C bus. The slave interface monitors the I2C bus and takes part in transmissions if its programmed address is seen on the bus. The interfaces communicate with the control/status registers independently. There is only one interrupt line output from the I2C module. The interrupt source is either the master or the slave.

### 107.3.4 Software Status Interlocking

To make the software interface with the I2C module as robust and thus simple as possible, various interlocking status mechanisms have been included in the operation of the master and slave interfaces. The status bits involved are:

**(1) MDR and SDR**

MDR or SDR is set to 1 when data is received. Clear the status after reading the receive data register. If data is received while MDR or SDR is set, the hardware recognizes that unread data remains in the receive data register and automatically holds SCL at the low level to suspend data transmission. In this case, transmission can be resumed by clearing the status after reading the receive data.

Consequently, when receiving data continuously, be sure to clear the status bit (MDR or SDR) after reading the receive data register.

**(2) MDE and SDE**

If the MDE or SDE status bit is still set when data in the transmit data register is to be transmitted on the I2C bus by the slave or master, the SCL line must be held low until the MDE or SDE status bit is reset. The MDE or SDE status bit being set indicates that the data currently held in the transmit data register has already been transmitted on the I2C bus.

The software must clear this status bit when it writes to the transmit data register after transmission of the next data byte has become possible. This is not required for the first byte of data to be transmitted on the bus.

**(3) MAL**

When the master loses arbitration, the MAL bit (of the master status register) is set and the master interface is disabled. At this point, master mode is invalid and the I2C bus interface enters the slave mode. When master interface is restored, MAL bit must be cleared before the address re-transmission.

**(4) SAR**

The SAR status bit is set when the slave identifies its address on the I2C bus. At this point the slave interface forces the SCL line low until the SAR status bit is reset.

This is particularly important when a slave transmit is about to take place on the bus, and the slave will transmit the data from the transmit data register. The software responds to the SAR status by writing the required data into the transmit data register and then resetting the SAR status bit. This allows the slave interface to continue the access.

Unless the SAR bit is in use, when the slave is about to receive data, the software might still be reading out data that was loaded in previous access from the receive data register. In this case the valid data still held in the receive data register is overwritten. However, this is avoided using the SAR status bit. After the software has read data in the receive data register, reset the SAR bit (if it is set). Then overwriting the receive data register is avoided.

### 107.3.5 I2C Bus Data Format

Figure 107.3 shows a timing chart for the I2C bus interface. Table 107.6 describes the meaning of each symbol in Figure 107.3.

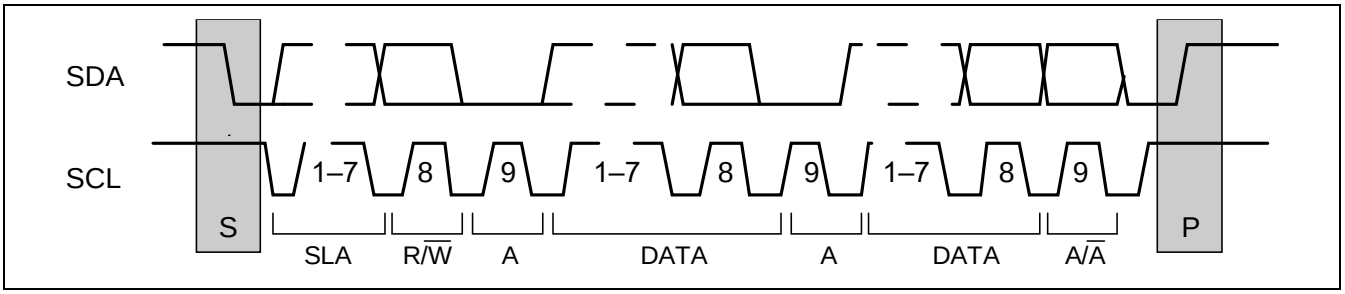



**Figure 107.3 I2C Bus Timing**

**Table 107.6 Description on Symbols of I2C Bus Data Format**

| Symbol | Description |
|---|---|
| S | Indicates a start condition. A master device changes SDA from high to low while SCL is high level. |
| SLA | Indicates a slave address. A slave address is used when a master device selects a slave device. |
| R/W | Indicates the direction of data transmission. If the R/W bit is 1, the data flows from the slave to the master device. If the bit is 0, the data flows from the master to the slave device. |
| A | Indicates data acknowledge. Data receiving device makes SDA low level (the slave device returns a data acknowledge signal in master transmission mode, and vice versa). |
| DATA | Indicates transmit or receive data. The data length is eight bits, which are transferred in the MSB first. |
| P | Indicates a stop condition. A master device changes SDA from low to high while SCL is high. |

### 107.3.6 7-Bit Address Format

Figure 107.4 shows the format of data transfer from a master to a slave device (master data transmit format). Figure 107.5 shows the data transfer format (master data receive format) when a master device reads the second and the following byte data from a slave device.

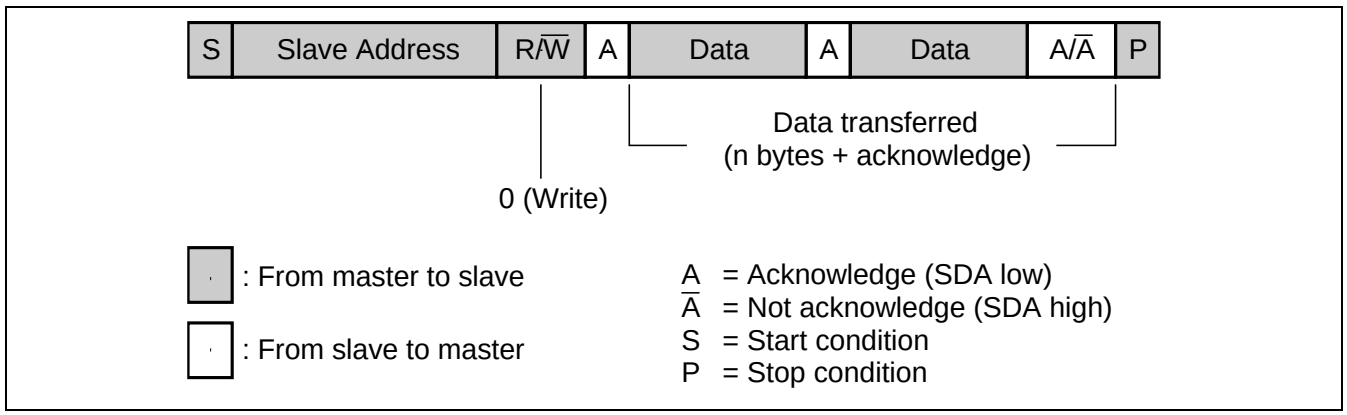



**Figure 107.4 Master Data Transmit Format**

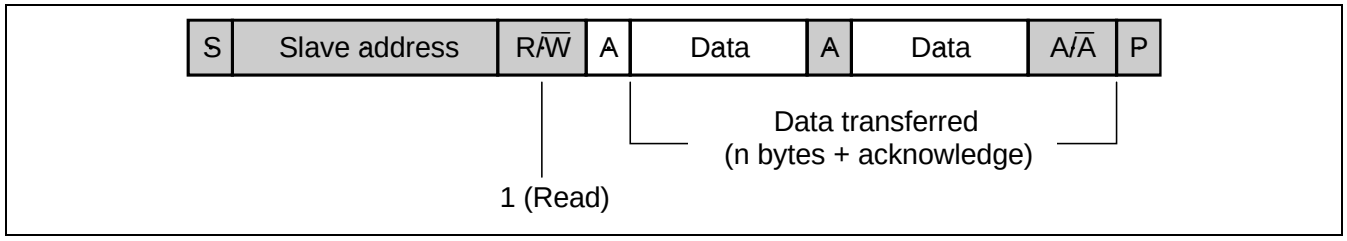
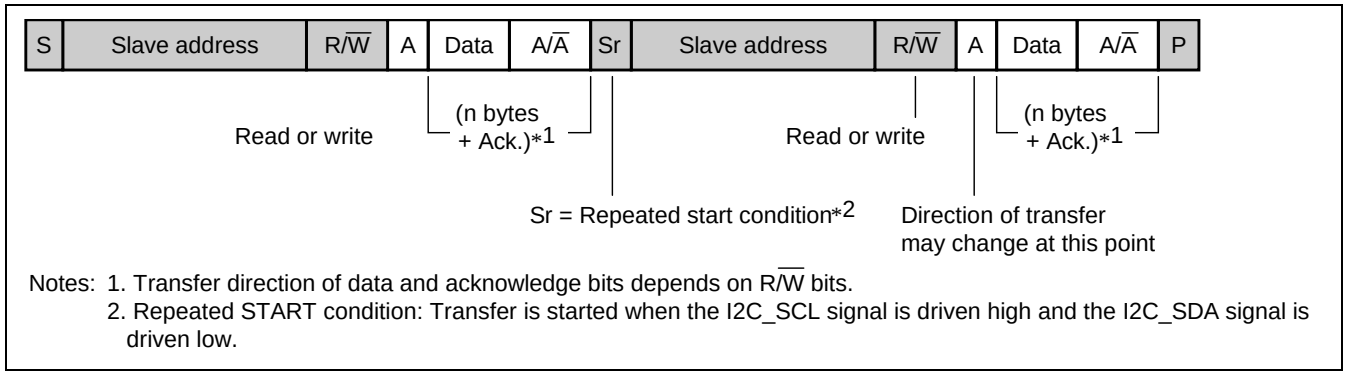



**Figure 107.5 Master Data Receive Format**

Figure 107.6 shows the combined format when the data transfer direction changes during one transfer. When changing the direction after the first transfer, the repeated START condition (Sr), slave address and R/ bits are transmitted. In this case, the R/ bit is set to the direction opposite to the first transfer direction. The repeated START condition is issued by the master at the end of a transmit or receive cycle if the enable start generation bit in the master control register has been set.




Notes: 1. Transfer direction of data and acknowledge bits depends on R/W bits.
2. Repeated START condition: Transfer is started when the I2C_SCL signal is driven high and the I2C_SDA signal is driven low.

**Figure 107.6 Combination Transfer Format of Master Transfer**

### 107.3.7 10-Bit Address Format

Description is given below on the 10-bit address transfer format supported in master mode. This format has three transfer methods as the 7-bit address transfer format.

Figure 107.7 shows the data transmit format. The set value in the master address register is output in one byte following the first START condition (S). The value set in the transmit data register (TXD) is transmitted as a slave address in the second byte. Data on and after the third byte is transferred in the same way as the 7-bit address data.

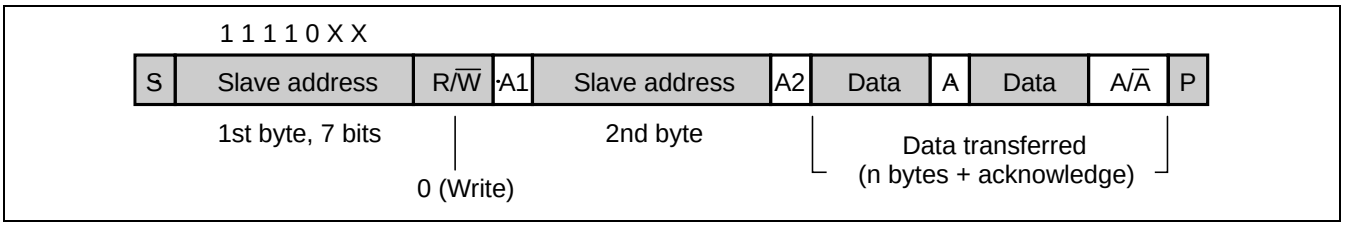



**Figure 107.7 10-Bit Address Data Transmit Format**

Figure 107.8 shows the data receive format. Two bytes of an address is transmitted in the same way as in the data transmit format. Then, repeated START condition (Sr) is transmitted and the value set in the address register is output. At this time, STM1 must be set to 1 (receive mode). Data is transferred in the same way as in the 7-bit address data receive format.

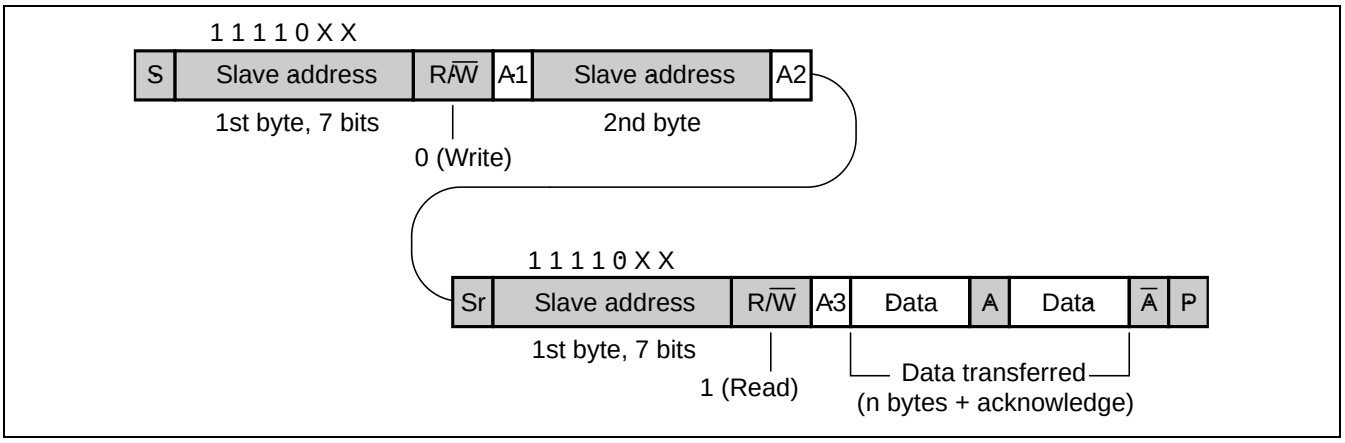



**Figure 107.8 10-Bit Address Data Receive Format**

Figure 107.9 shows the data transmit/receive combined format.

In the data transmit/receive combined format, data is transmitted after an address is transmitted with the first two bytes. Then, the repeated START condition (Sr) is transmitted instead of STOP condition (P). After Sr is transmitted, the procedure is the same as that in the data receive format.

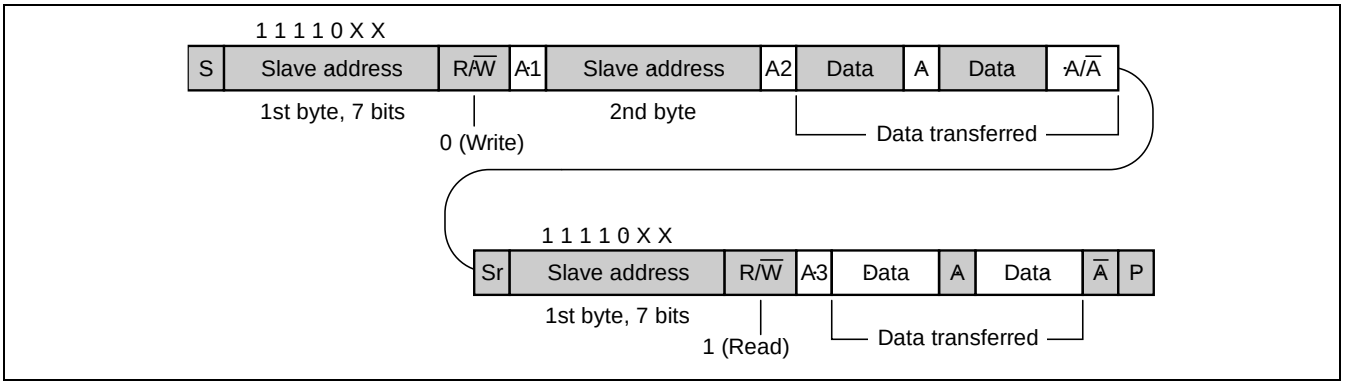



**Figure 107.9 10-Bit Address Transmit/Receive Combined Format**

### 107.3.8 Master Transmit Operation

The procedure and operations for transmission in master transmit mode are described below. Figure 107.10 shows the timing chart in master transmit mode. Setting the ICMCR.MDBS bit allows the I2C to operate in single-buffer mode.

1. Set up the initial state by setting the ICMAR, ICTXD, ICCCR, and ICMIER. Note that the setting of the clock control register must be in accord with the rate of transmission.
2. Monitor the ICMCR.FSDA. Confirm that this bit is low, meaning that other I2C devices are not using the bus. After confirmation, set the ICMCR.MIE (bit 3) and ICMCR.ESG (bit 0) bits to 1 to start master transmission.
3. After the transmit START condition, slave address, and data transfer direction bits are transmitted, an interrupt due to the ICMSR.MAT and ICMSR.MDE bits is generated at the timing of (1) in Figure 107.10. At this time, clear the ICMCR.ESG bit to 0. To suspend the data transmission, the master device will hold SCL low until the ICMSR.MDE bit is cleared.
4. Data is transmitted in units of nine bits: 8-bit data and 1-bit ACK. An interrupt of MDE (bit 3) is generated at the ninth clock before data transfer (at the timing of (2) in Figure 107.11, Figure 107.12). An interrupt of ICMSR.MDT (bit 2) is generated at the eighth clock after 1-byte data transfer (at the timing of (3) in Figure 107.11, Figure 107.12). Clear ICMSR.MDE to 0 after setting transmit data.
5. When this I2C receives NACK from the slave device, ICMSR.MNR is set to 1.
   After that, this I2C makes STOP condition and ICMSR.MST is set to 1.
6. To end of transmission, need to set ICMCR.FSB = 1.
   ICMCR.FSB must be set by MDE interruption at final byte data transmission.
   ICMSR.MST (master stop transmitted, at the timing of (4) in Figure 107.12) is checked by interruption or polling.

Signal level changes of (1) to (4) in Figure 107.10, Figure 107.11 and Figure 107.12 are generated after the falling edge of the clock.

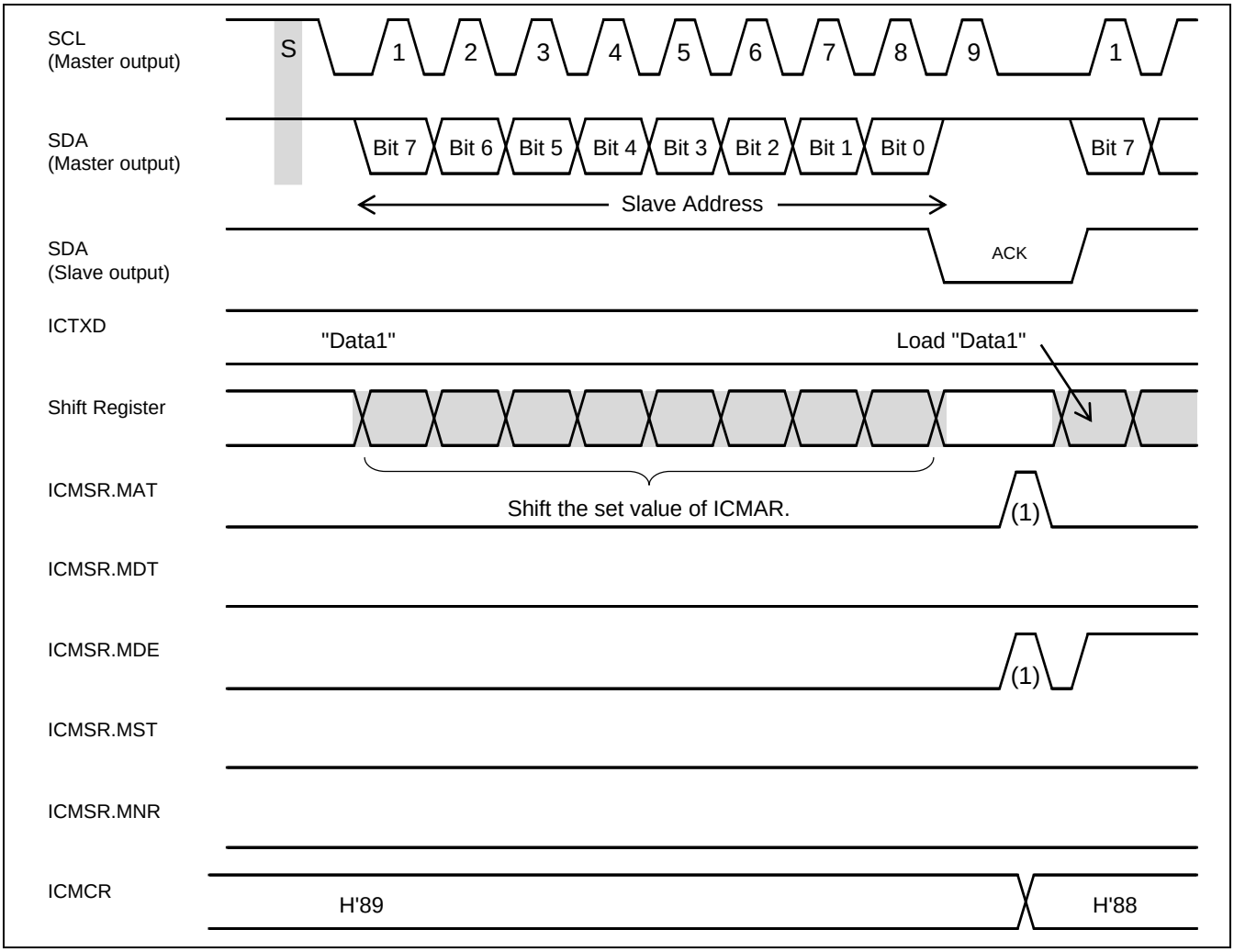



**Figure 107.10 Data Transmit Mode Operation Timing**

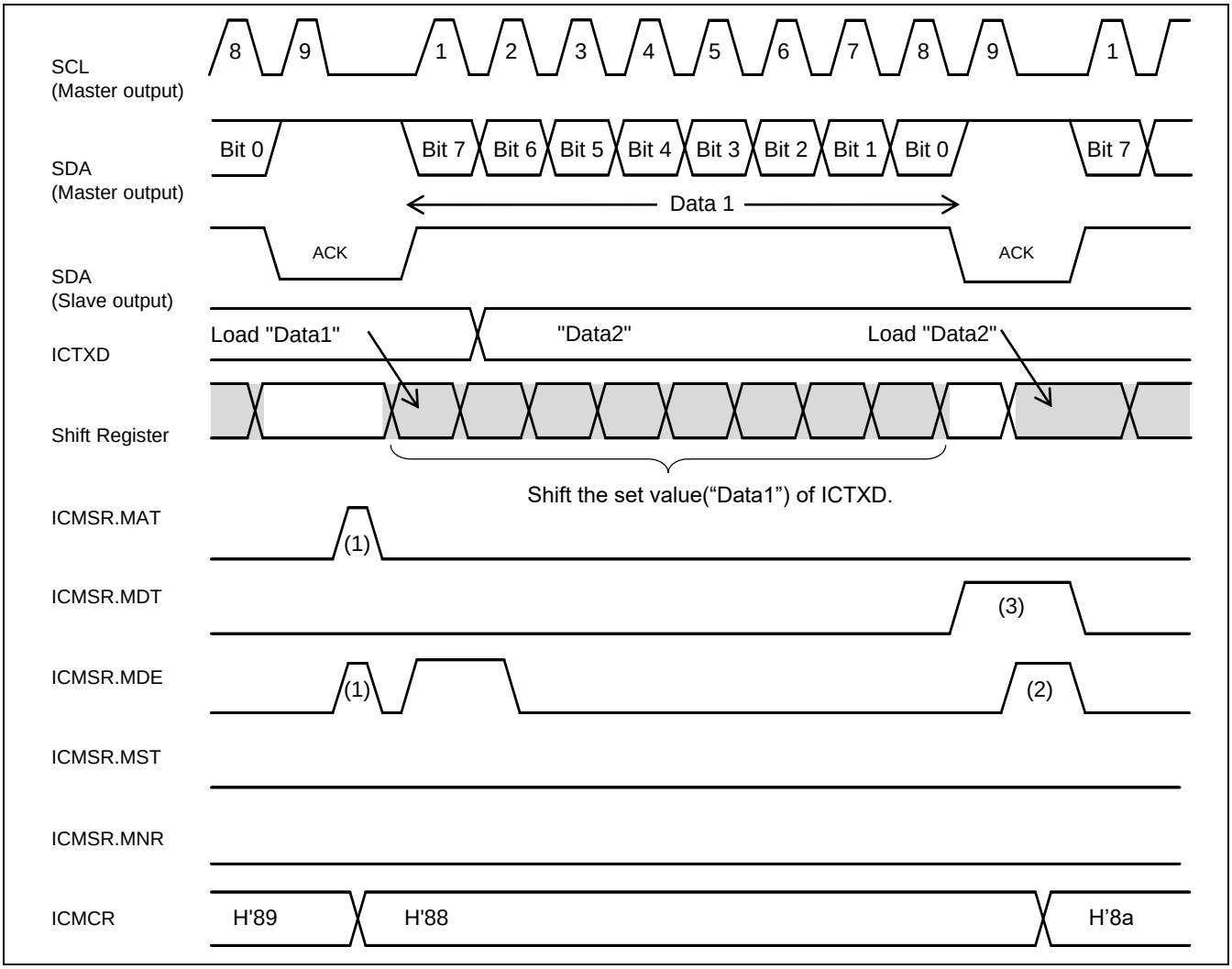



**Figure 107.11 Data Transmit Mode Operation Timing**

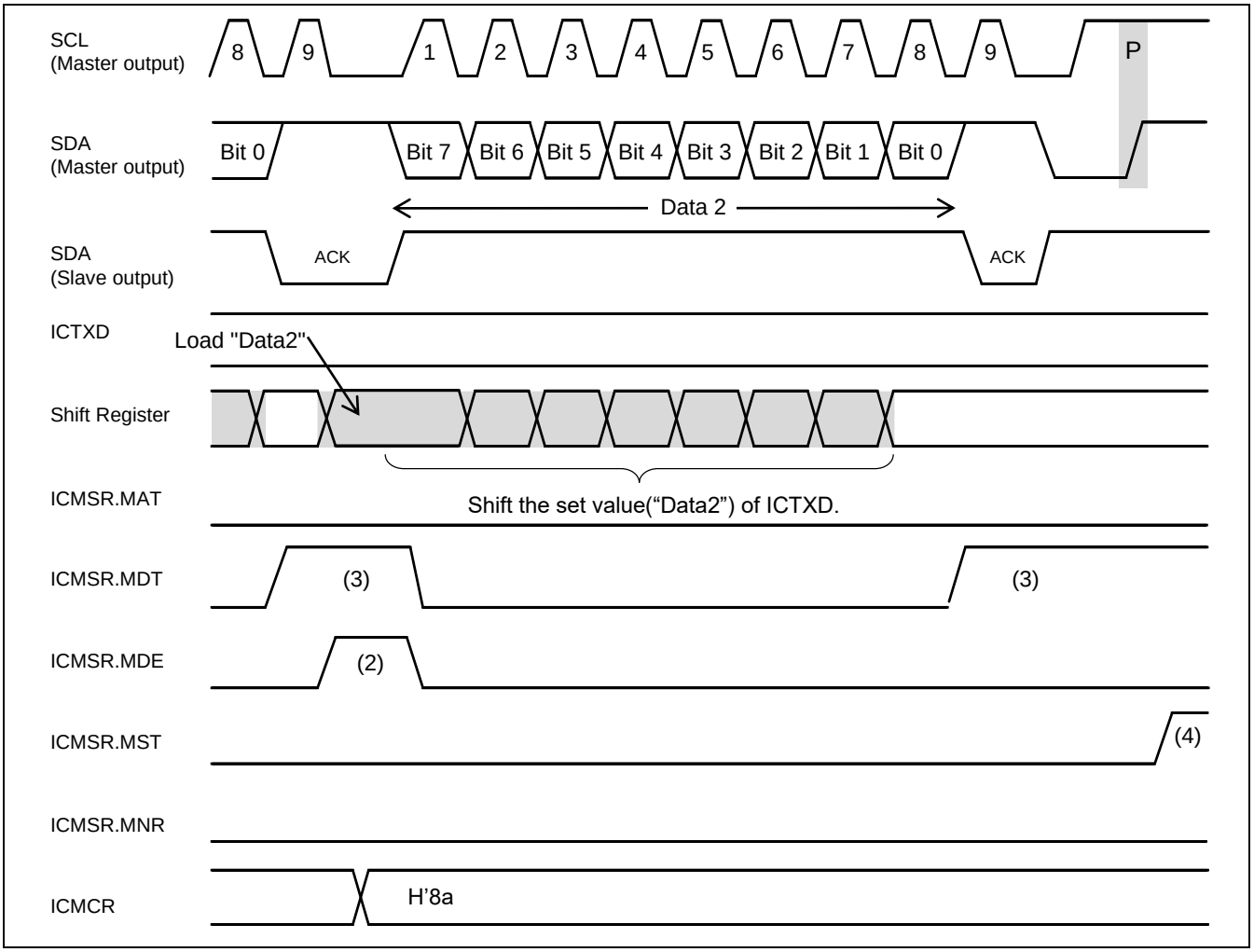



**Figure 107.12 Data Transmit Mode Operation Timing**

### 107.3.9 Master Receive Operation

The data receive procedure and operation in master receive mode are described below. Figure 107.13 shows the timing chart in master receive mode. Setting the ICMCR.MDBS bit allows the I2C to operate in single-buffer mode.

1. In master receive mode, as to transmit of a slave address and a 1-bit signal indicating the data transfer direction, operation is the same as that in master transmit mode. At this time, set the data transfer direction to 1 (reception).
2. The slave device automatically enters the data transmit mode according to the signal that indicates the data transfer direction, and transmits 1-byte data in synchronization with the SCL clock output from the master device. The master device generates an interrupt of ICMSR.MDR (bit 1) at the eighth clock (at the timing of (1) in Figure 107.13 , Figure 107.14).

   Clear the ICMSR.MDR bit after the master device reads ICRXD. If this processing is delayed, the master device extends the SCL period to suspend data transmission (at the timing of (2) in Figure 107.13 , Figure 107.14).
3. To end data transfer, set ICMCR.FSB (bit 1) of the master device and output STOP condition. In order to stop the communication after predetermined number of byte data is transferred, ICMCR.FSB bit needs to be set before the last byte data transfer is started. After confirming reception of the last byte, even when the master receiver has completed the reception transaction.

Signal level changes of (1) to (2) in Figure 107.13 and Figure 107.14 are generated after the falling edge of the clock.

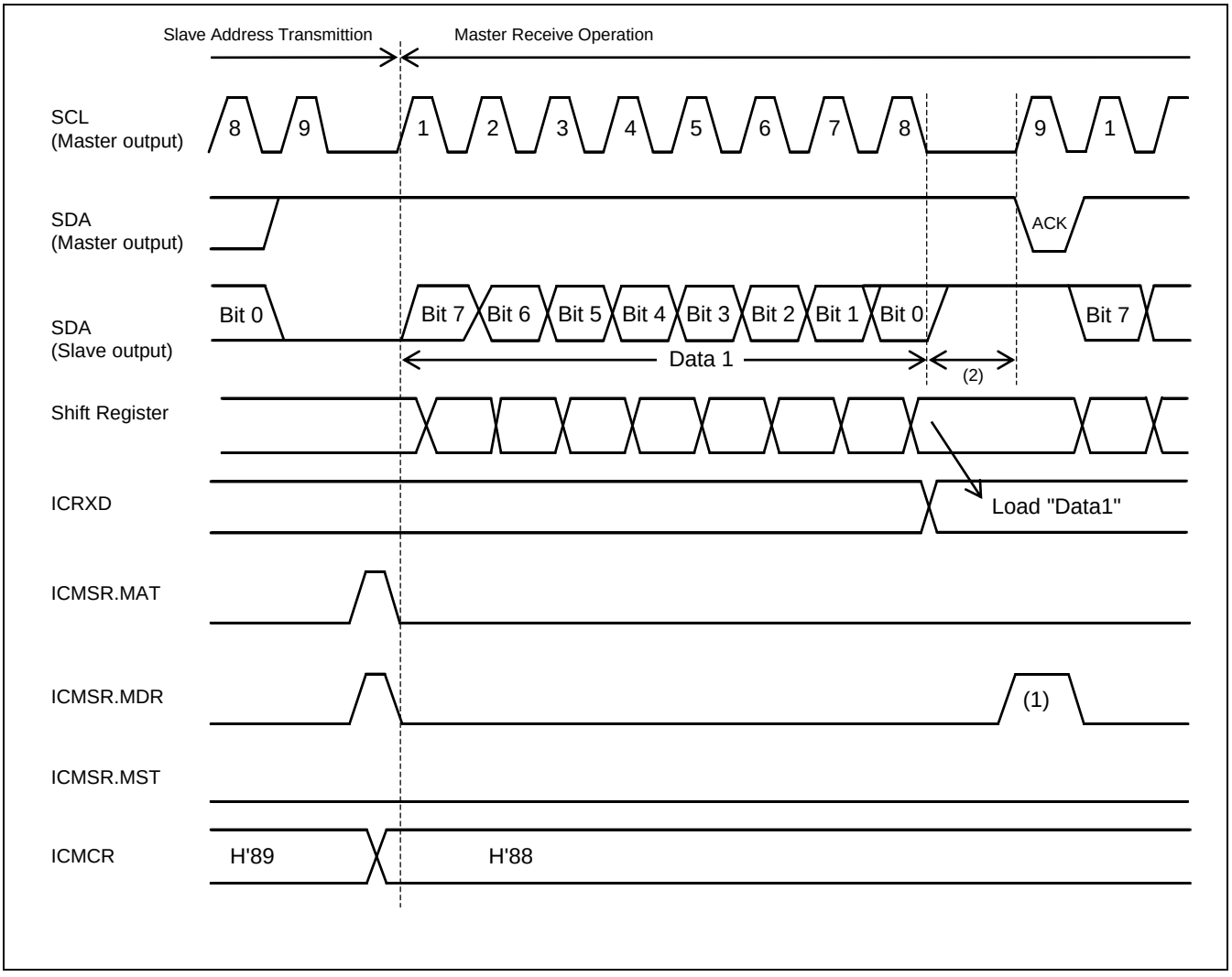



**Figure 107.13 Data Receive Mode Operation Timing**

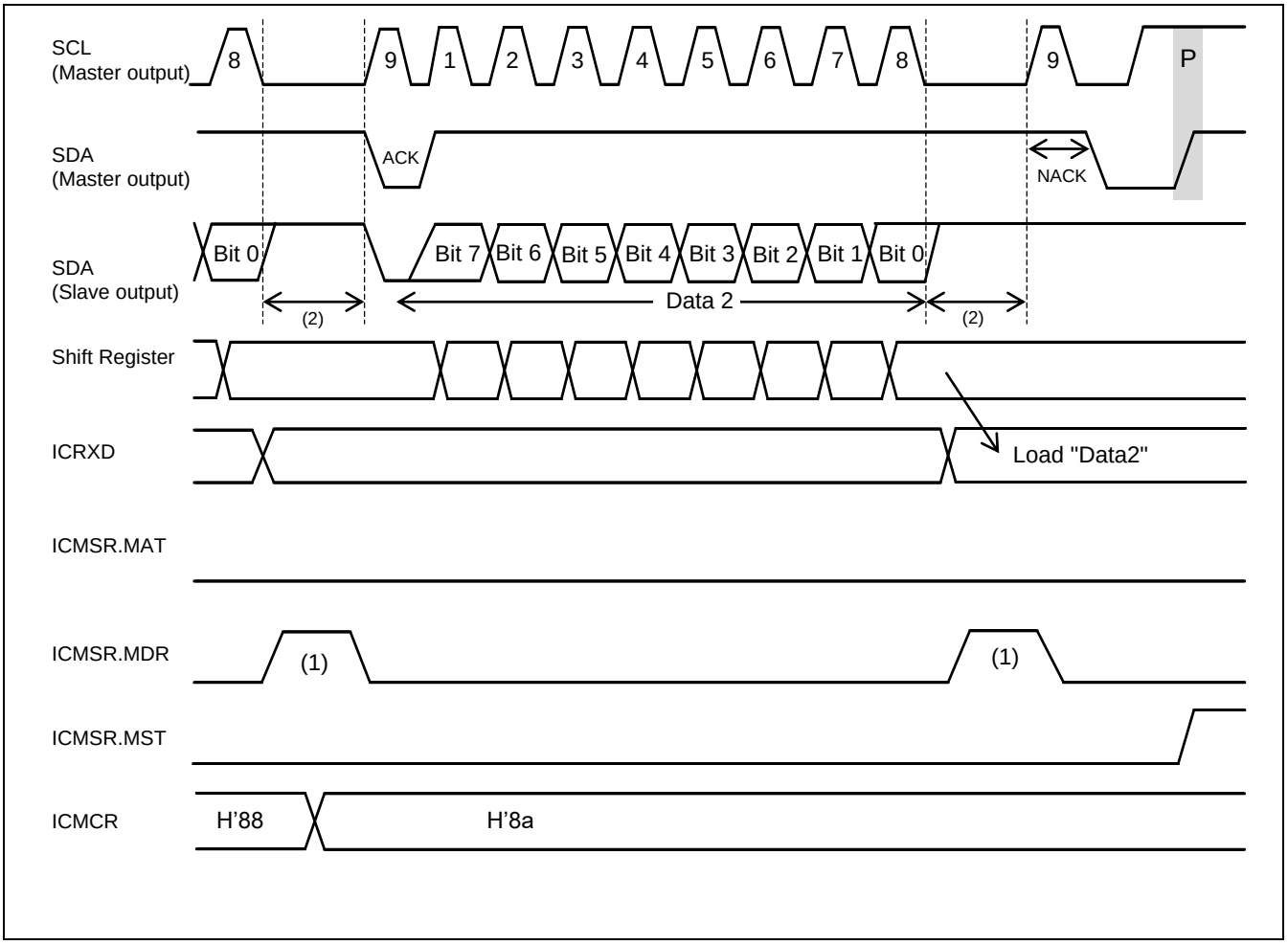



**Figure 107.14 Data Receive Mode Operation Timing**

### 107.3.10 Programming Example

#### 107.3.10.1 Master Transmitter

In order to set up the master interface to transmit a data packet on the I2C bus, follow the procedure below (an example for I2C0):

**(a) Set value for the clock control register**

1. Reset I2C with SRCR and SRSTCLR.
2. Set the Clock Control register 2 (ICCCR2) = H’07 (CDFD = 1, HLSE = 1, SME = 1).
3. Set the Clock Control register (ICCCR) = H’06 (SCGD = 0, CDF = 6, I2C internal clock I2Cck: 19 MHz).
   - Setting ICCCR, ICCCR2 for using Duty ratio fixed or Duty ratio variable. The case for using Variable Duty ratio, have to setting ICCCR2 too. The case for using Fixed Duty ratio, have to setting ICCCR only.
4. Set the SCL Mask Control register (ICMPR) = H’1C (Variable Duty ratio only).
5. Set the SCL High Control register (ICHPR) = H’73 (Variable Duty ratio only).
6. Set the SCL Low Control register (ICLPR) = H’85 (Variable Duty ratio only).
7. Set the First Bit Setup Cycle register (ICFBSCR) = H’0F ($1^{st}$ bit setup cycle = 17 × Tcyc).

**(b) Set value for the master control register, first data byte, and address**

1. Clear the Master Status register (ICMSR) = H’00 (all 0 clear).
2. Set the Master Interrupt Enable register (ICMIER) = H’09 (MDEE = 1, MATE = 1).
3. Set the Master Address register (ICMAR) = Slave device address + H’00 (Write mode).
4. Set the Transmit Data register (ICTXD) = $1^{st}$ transmission data.
5. Set the Master Control register (ICMCR) = H’89 (MDBS = 1, MIE = 1, ESG = 1).

**(c) Wait for outputting address**

1. Wait for master event (an interrupt of the ICMSR.MAT and ICMSR.MDE).
2. Set ICMCR to H'88 (to suspend the data transmission, the master device will hold the SCL low until the MDE bit is cleared).
   — If only one byte of data is transmitted, set the ICMCR to H'8A, meaning that the stop generation is enabled. This generates a stop on the bus as soon as one byte has been transmitted.
3. Clear the ICMSR.MAT & ICMSR.MDE bits.

**(d) Monitor transmission of data**

1. Wait for master event, ICMSR.MDE.
2. ICTXD = subsequent data.
3. Clear the ICMSR.MDE bit.

   Clear ICMSR.MDE after setting the last byte to be transmitted. After the last byte data is loaded into the shift register, ICMSR.MDE is generated. Before clearing ICMSR.MDE, you must set ICMSR to H'8A (set the force stop control bit).

**(e) Wait for end of transmission**

1. Wait for the master event, ICMSR.MST.
2. Clear the ICMSR.MST bit.

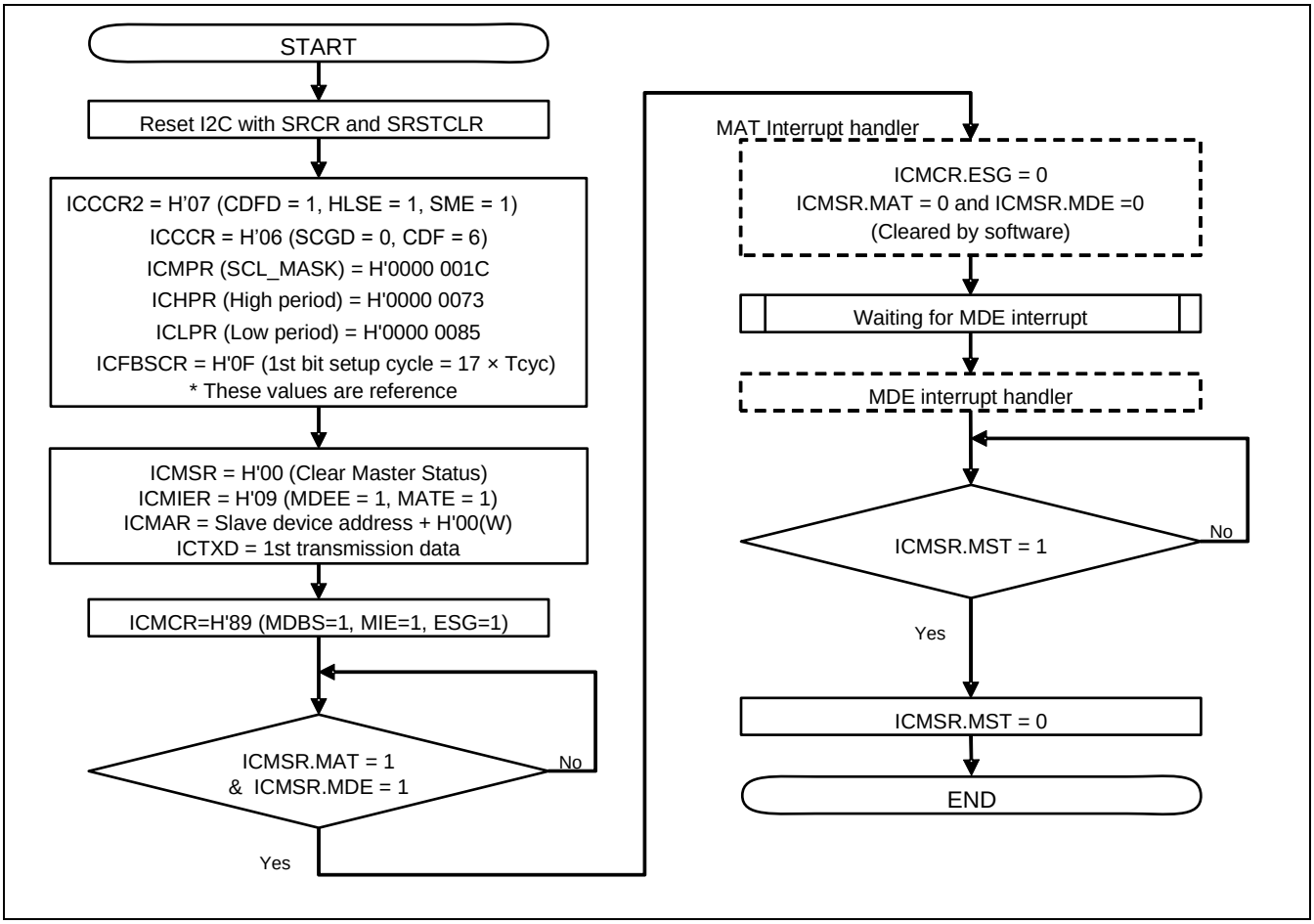



**Figure 107.15 Master Transmitter Operation Setting**

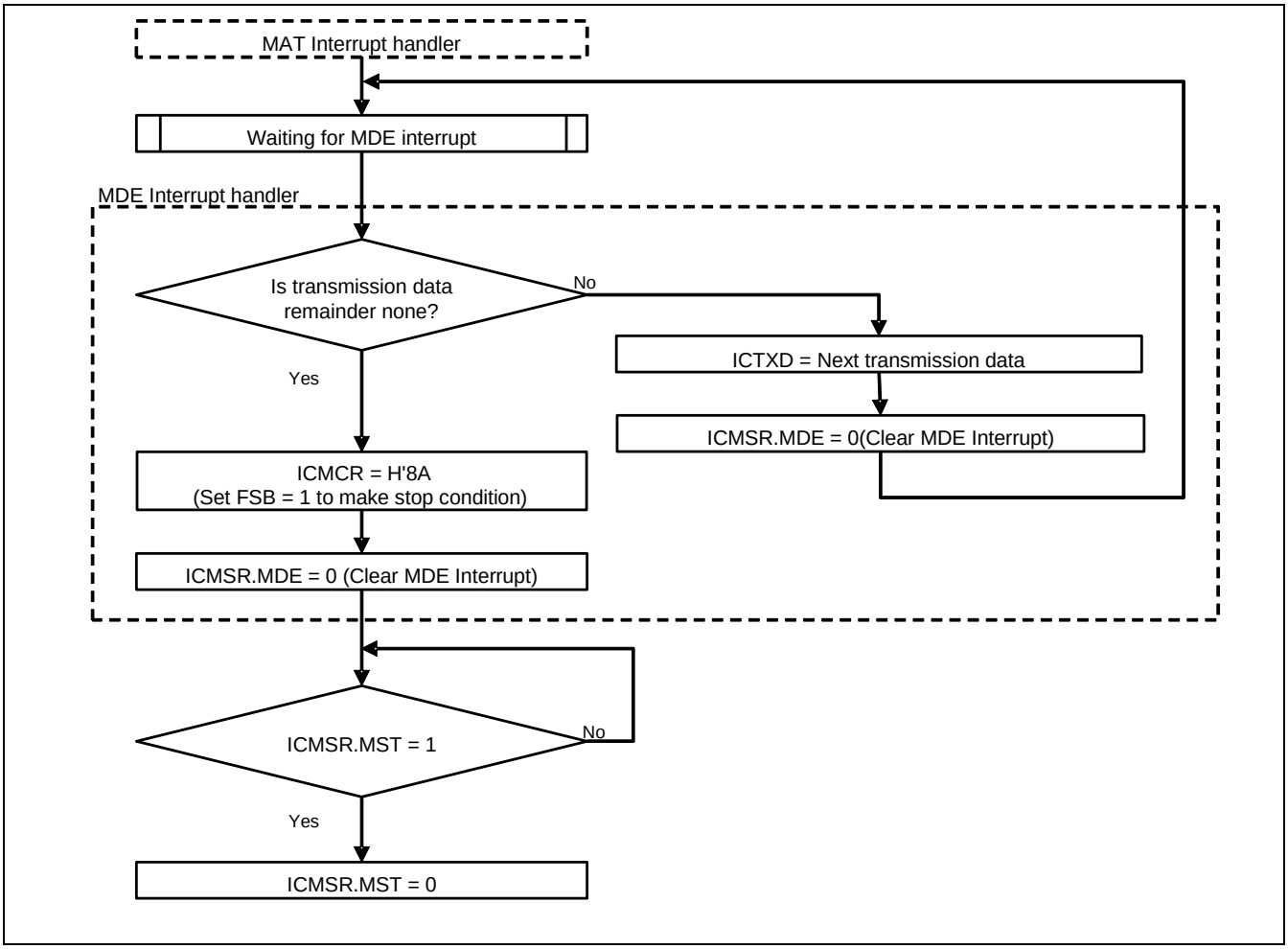



**Figure 107.16 MDE Interrupt Handler for Master Transmitter Operation Setting**

### 107.3.10.2 Master Receiver

In order to set up the master interface to receive a data packet on the I2C bus, follow the procedure below:

**(a) Set value for the clock control register**

The same procedure as the master transmitter is applied. See the program example for the master transmitter.

**(b) Set value for the master control register and address**

1. Clear the Master Status register (ICMSR) = H’00 (all 0 clear)
2. Set the Master Interrupt Enable register (ICMIER) = H’03 (MDRE = 1, MATE = 1)
3. Set the Master Address Register (ICMAR) = Slave device address + H’01 (Read mode)
4. Set the Master Control Register (ICMCR) = H'89 (MDBS = 1, MIE = 1, ESG = 1)

**(c) Wait for outputting address**

1. Wait for master event (an interrupt of the ICMSR.MAT and ICMSR.MDR).
2. Set the ICMCR to H'88 (to suspend the data transmission, the master device will hold the SCL low until the MDR bit is cleared).
   — If only one byte of data is transmitted, set the ICMCR to H'8A, meaning that the stop generation is enabled. This generates a stop on the bus as soon as one byte has been received.
3. Clear the ICMSR.MAT & ICMSR.MDR bits.

**(d) Monitor reception of data**

1. Wait for master event, ICMSR.MDR.
2. Set the ICMCR to H'8A (set the ICMCR.FSB) before the last byte data transfer is started.
3. Clear the MDR bit after reading data from the ICRXD.

**(e) Wait for end of reception**

1. Handle the receive interrupt (ICMSR.MDR) in the last byte: that is, read the data and clear the ICMSR.MDR.
2. Wait for master event, ICMSR.MST.
3. Clear the ICMSR.MST bit.

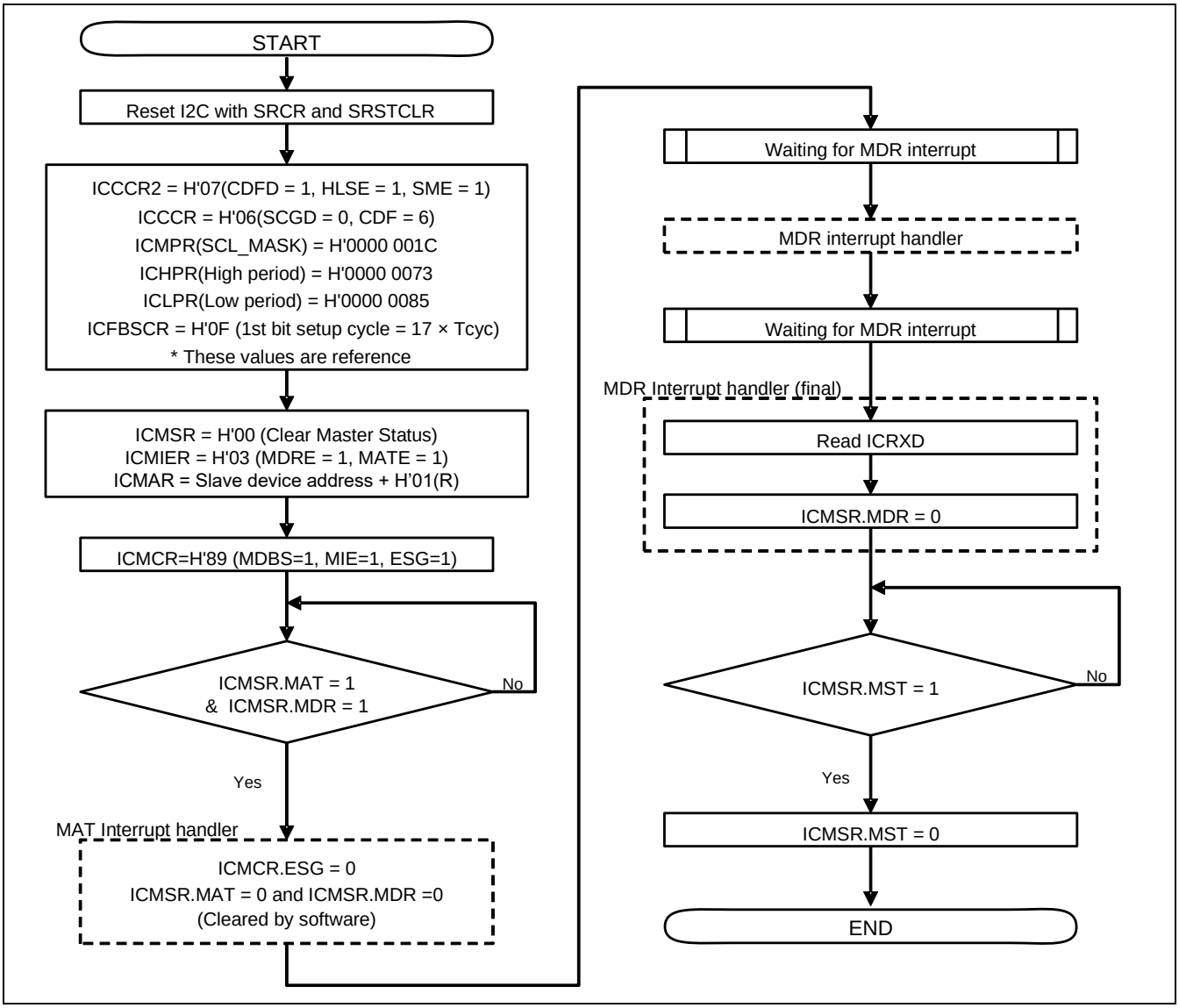



**Figure 107.17 Master Receiver Operation Setting**

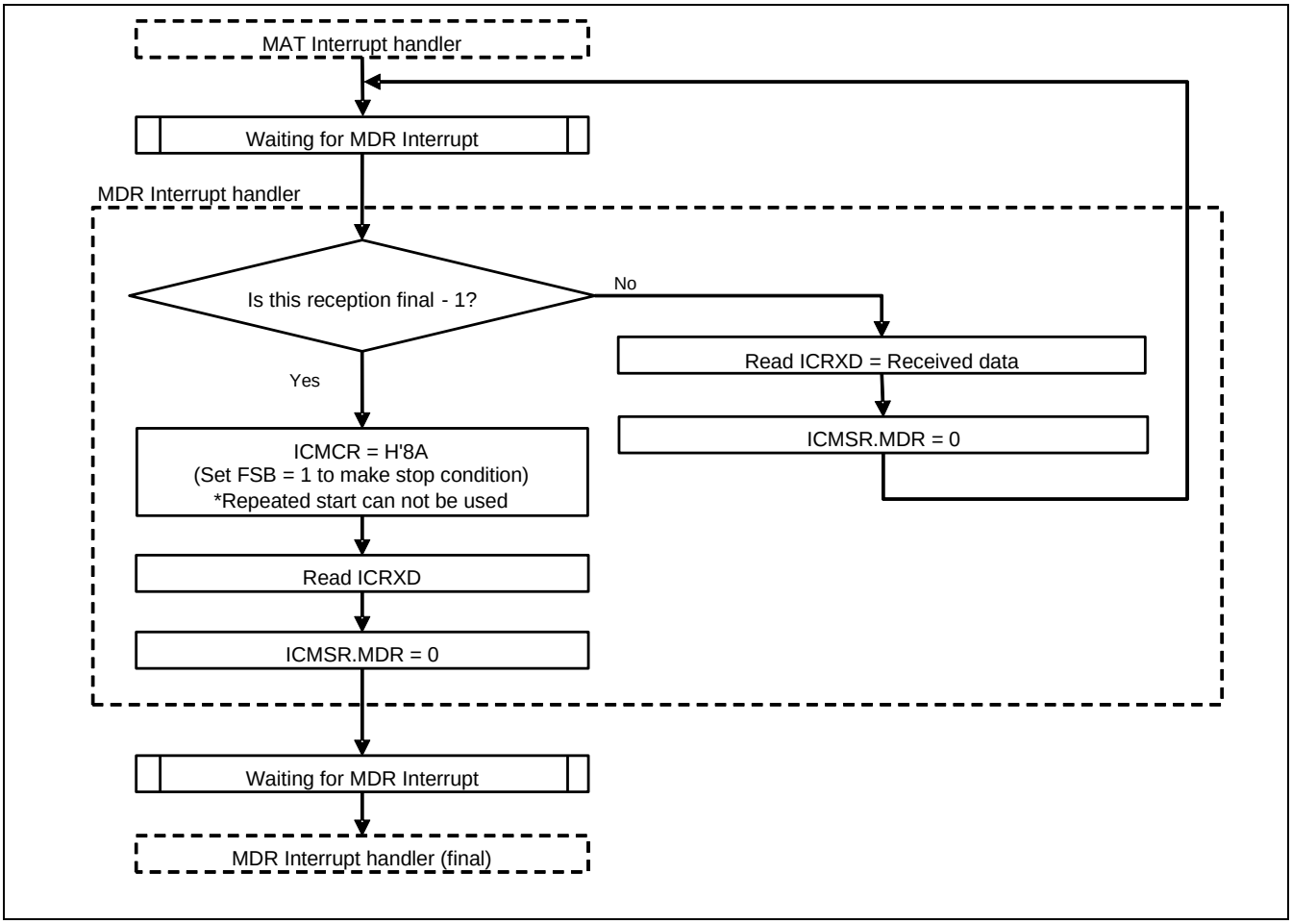



**Figure 107.18 MDR Interrupt Handler for Master Receiver Operation Setting**

#### 107.3.10.3 Master Transmitter—Repeated START—Master Receiver

In order to set up the master interface to transmit a data packet on the I2C bus, issue a repeated START condition, then read byte data back from the slave, follow the procedure below:

**(a) Set value for the clock control register**

The same procedure as the master transmitter is applied. See the program example for the master transmitter.

**(b) Set value for the master control register and address**

1. Clear the Master Status register (ICMSR) = H’00 (all 0 clear).
2. Set the Master Interrupt Enable register (ICMIER) = H’0B (MDEE = 1, MDRE = 1, MATE = 1).
3. Set the Master Address register (ICMAR) = Slave device address + H’00 (Write mode).
4. Set the Transmit Data register (ICTXD) = 1st transmission data.
5. Set the Master Control register (ICMCR) = H’89 (MDBS = 1, MIE = 1, ESG = 1).

**(c) Wait for outputting slave-address for master transmission**

1. Wait for master event (an interrupt of the ICMSR.MAT and ICMSR.MDE bits).
2. Clear ICMCR.ESG bit..
3. Clear the ICMSR.MAT and ICMSR.MDE bits.

**(d) Monitor transmission of data**

1. Wait for master event (an interrupt of the ICMSR.MDE bit).
2. Set ICMAR = Slave device address + H’01 (Read mode).
3. Set ICMCR = H'89 (MDBS = 1, MIE = 1, ESG = 1).
4. Clear the ICMSR.MDE bit.

**(e) Wait for outputting slave-address for master reception**

1. Wait for master event (an interrupt of ICMSR.MAT and ICMSR.MDR bits).
2. Clear ICMCR.ESG bit. (To suspend stop the data transmission, the master device will hold the SCL low until the ICMSR.MDR bit is cleared.)
3. Clear the ICMSR.MAT and ICMSR.MDR bits.

**(f) Monitor reception of data**

1. Wait for master event (the ICMSR.MDR bit).
2. Clear the ICMSR.MDR bit after reading data from the receive data register.
3. Set ICMCR.FSB to 1 (set the force stop control bit) before the last byte data transfer is started.

**(g) Wait for end of reception**

1. Handle the receive interrupt (ICMSR.MDR) in the last byte: that is, read the data and clear the ICMSR.MDR.
2. Wait for master event, ICMSR.MST.
3. Clear the ICMSR.MST bit.

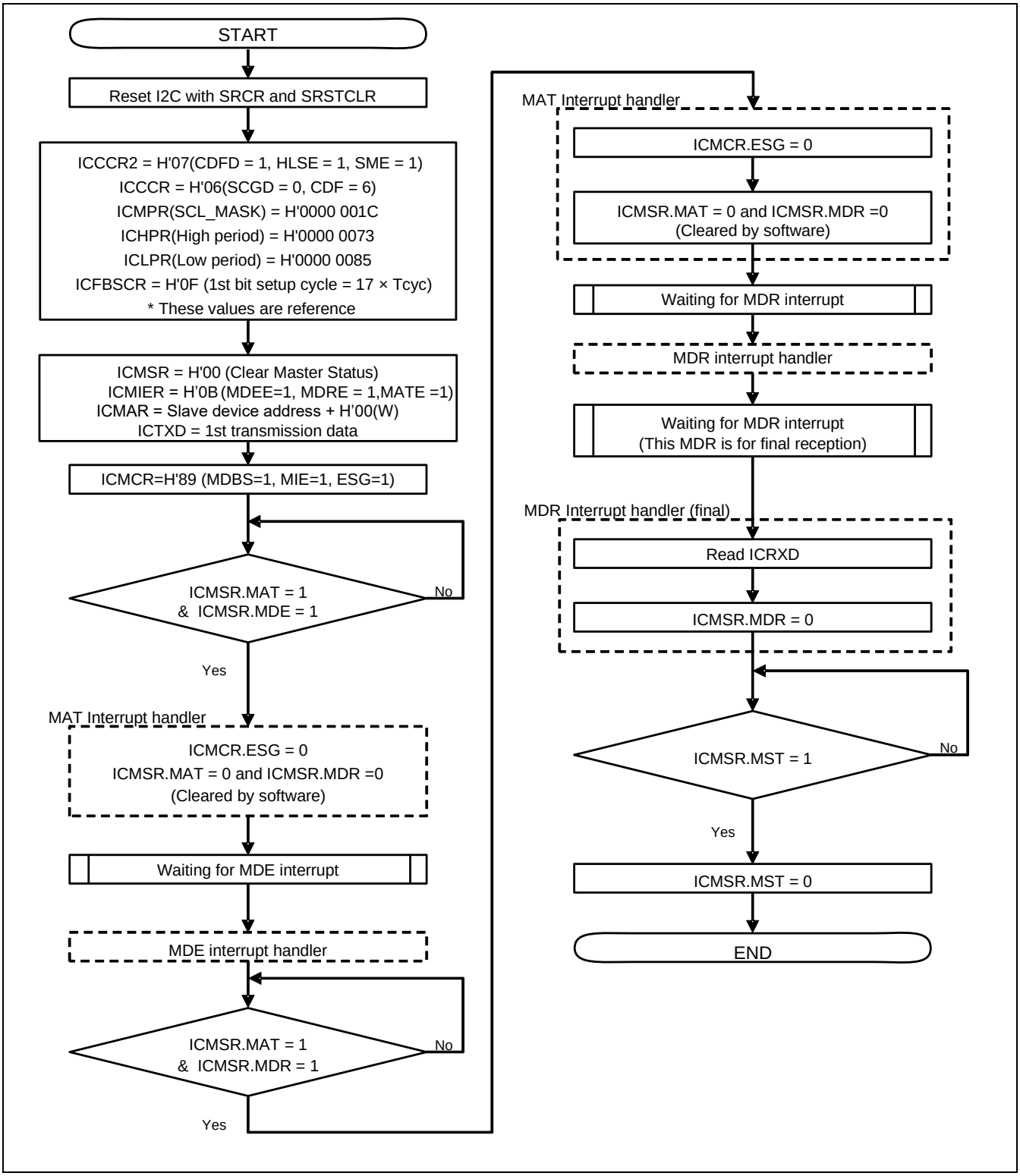



**Figure 107.19 Master Transmitter – Repeated START – Receiver Operation Setting**

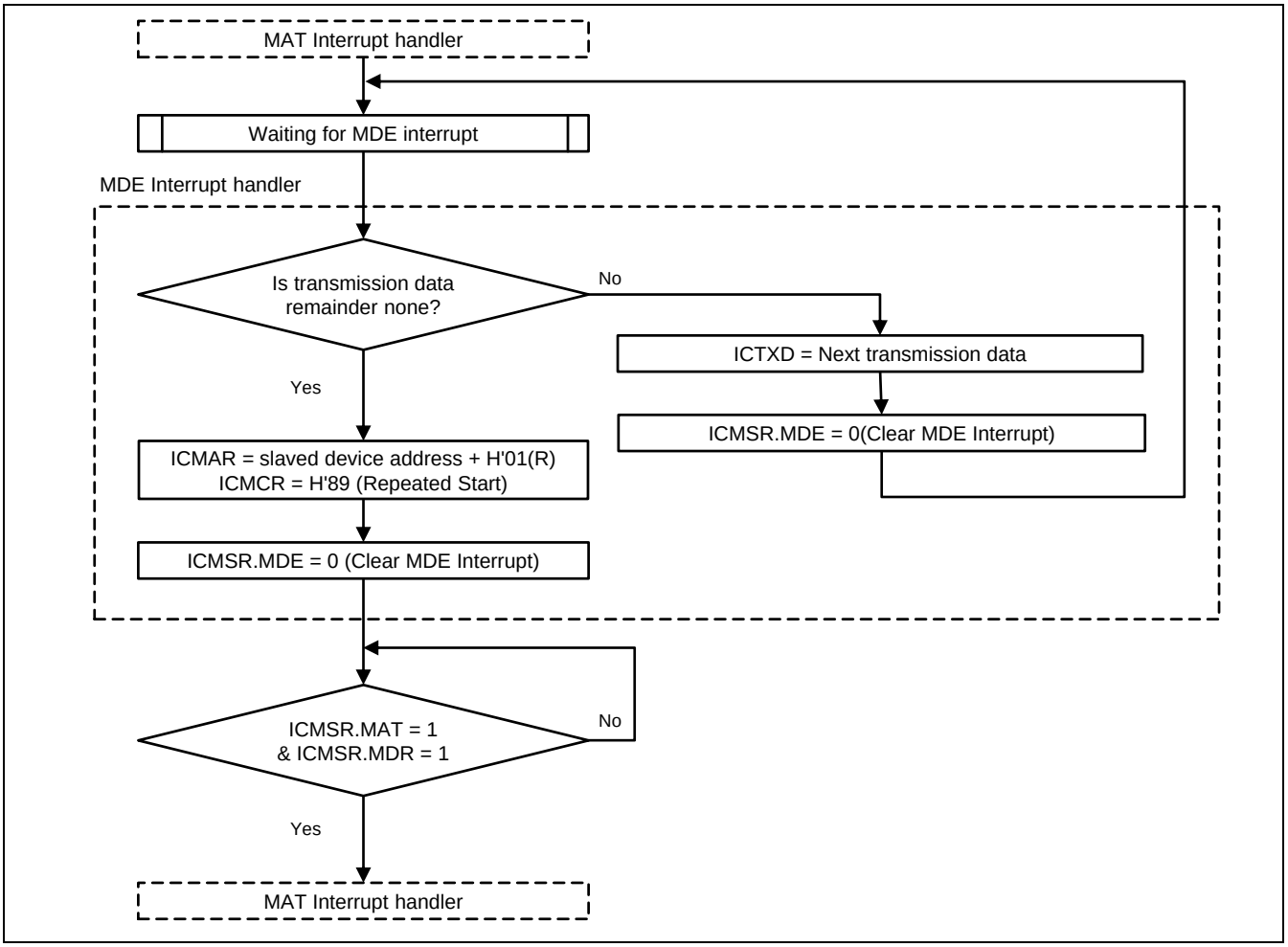



**Figure 107.20 MDE Interrupt Handler for Master Transmitter – Repeated START – Receiver Operation Setting**

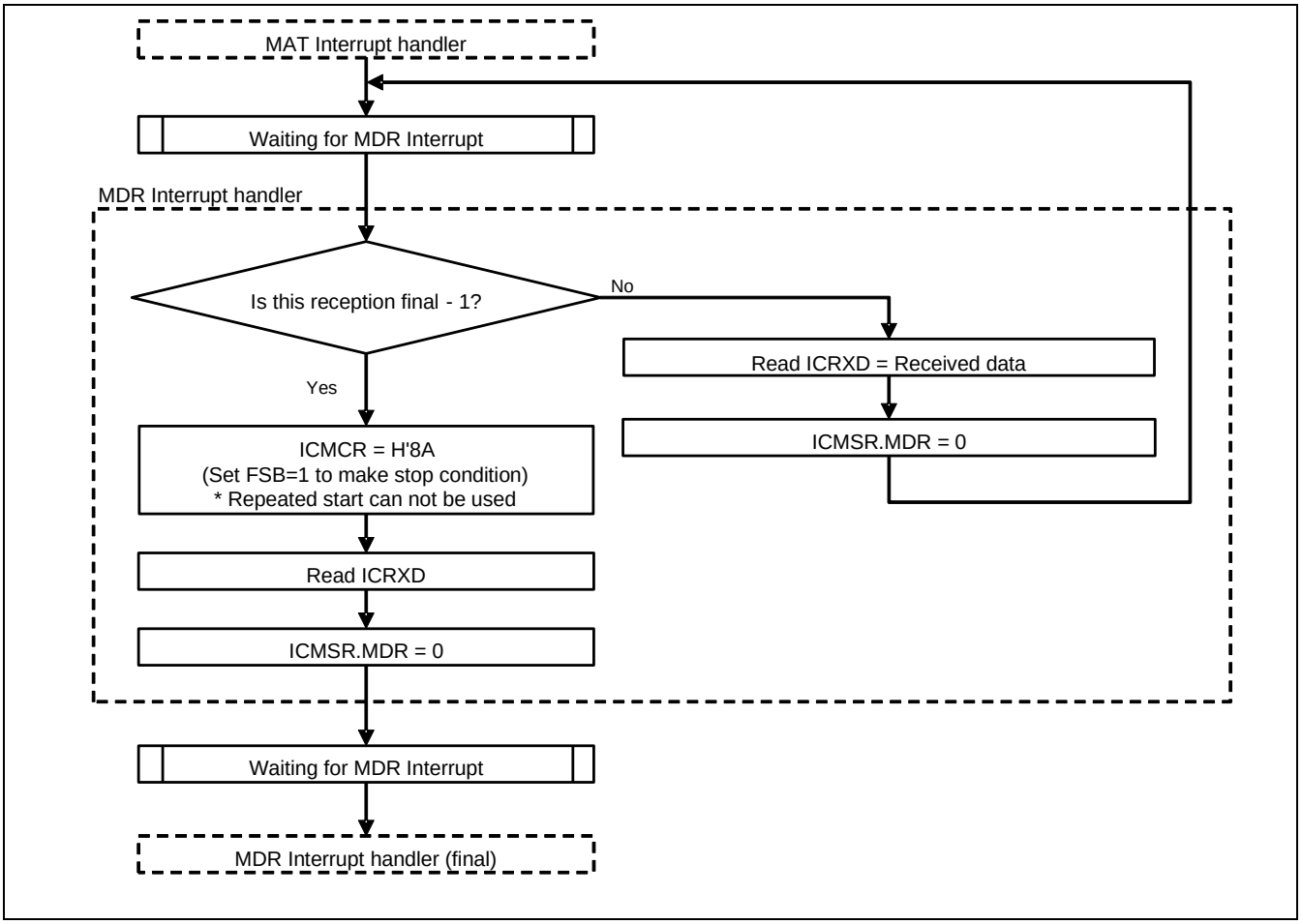



**Figure 107.21 MDR Interrupt Handler for Master Transmitter – Repeated START – Receiver Operation Setting**

#### 107.3.10.4 Slave Transmitter

In order to set up the master interface to transmit a data packet on the I2C bus, follow the procedure below (an example for I2C0):

**(a) Set value for the slave control register, first data byte and address**

1. Reset I2C with SRCR and SRSTCLR.
2. Clear the Slave Status register (ICSSR) = H'00 (all 0 clear).
3. Set the slave Interrupt register (ICSIER) = H'09 (SDEE = 1, SARE = 1).
4. Set Slave Address register (ICSAR) = Slave device address (unique value from other slave devices).
5. Set the Transmit Data register (ICTXD) = 1st transmission data.
6. Set the Slave control register (ICSCR) = H'0E (SDBS = 1, SIE = 1, GCAE = 1).

**(b) Wait for receiving address**

1. Wait for slave event (an interrupt of the ICSSR.SAR).
2. Clear the ICSSR.SAR bit.

**(c) Monitor transmission of data**

1. Wait for master event, ICSSR.SDE.
2. ICTXD = subsequent data.
3. Clear the ICSSR.SDE bit.

   Clear ICSSR.SDE after setting the last byte to be transmitted. After the last byte data is loaded into the shift register, ICSSR.SDE is generated. So, skip writing to ICTXD at ICSSR.SDE after last byte data is loaded into the shift register.

**(d) Wait for end of transmission**

1. Wait for slave event, ICSSR.SSR.
2. Clear the ICSSR.SSR bit.

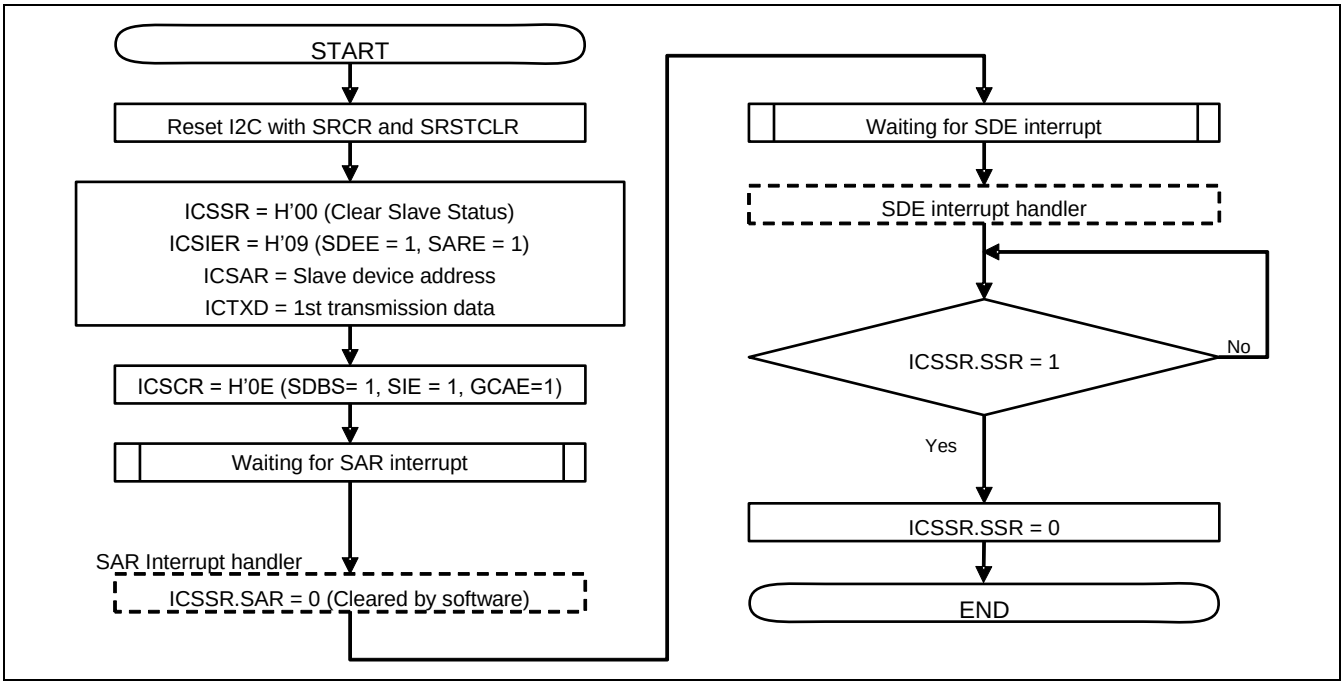



**Figure 107.22 Slave Transmitter Operation Setting**

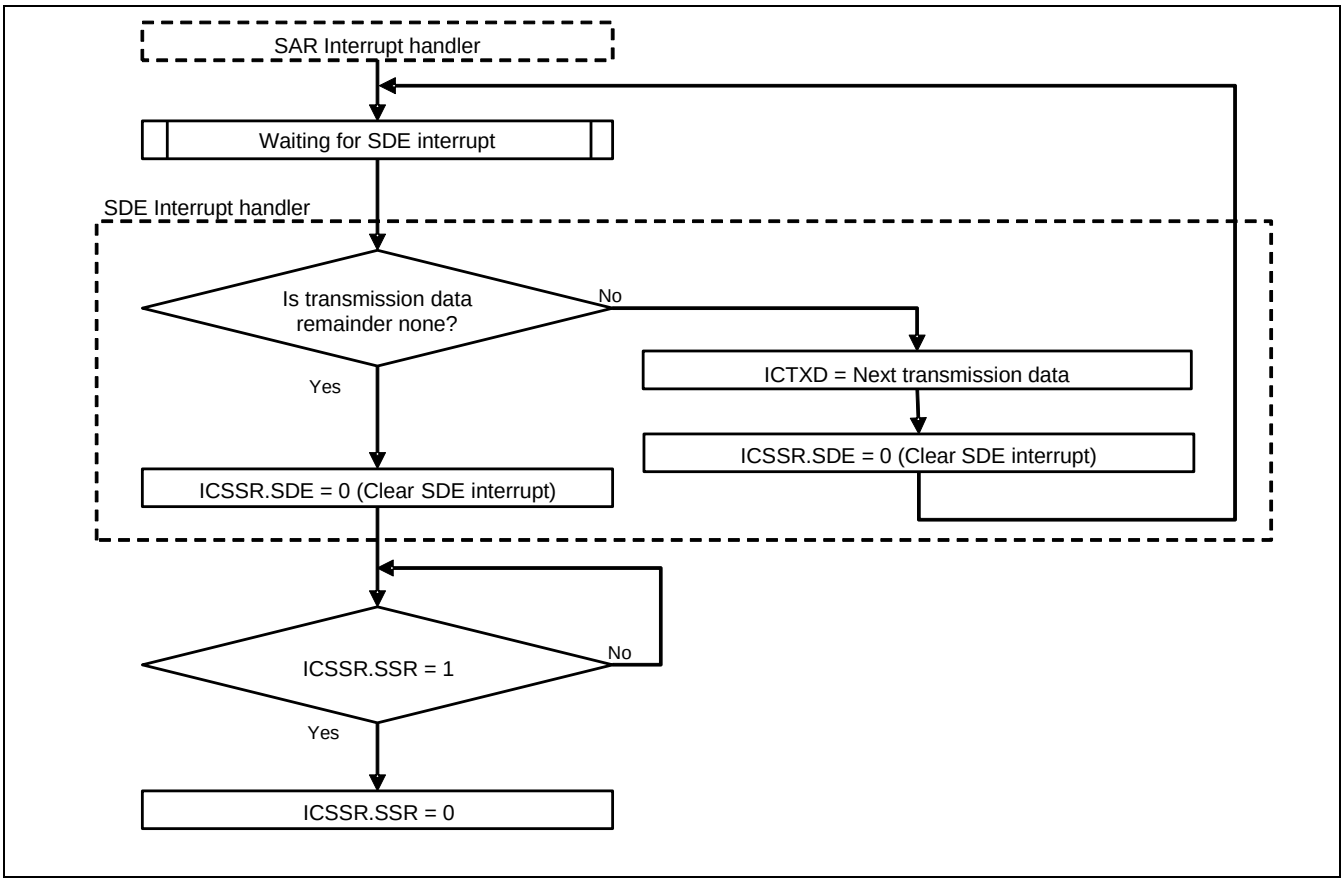



**Figure 107.23 SDE Interrupt Handler for Slave transmitter Operation Setting**

#### 107.3.10.5 Slave Receiver

In order to set up the master interface to receive a data packet on the I2C bus, follow the procedure below:

**(a) Set value for the master control register and address**

1. Reset I2C with SRCR and SRSTCLR.
2. Clear the Slave Status register (ICSSR) = H’00 (all 0 clear).
3. Set the slave Interrupt register (ICSIER) = H’03 (SDRE = 1, SARE = 1).
4. Set Slave Address register (ICSAR) = Slave device address (unique value from other slave devices).
5. Set the Slave control register (ICSCR) = H’0E (SDBS = 1, SIE = 1, GCAE = 1).

**(b) Wait for receiving address**

1. Wait for slave event (an interrupt of the ICSSR.SAR).
2. Clear the ICSSR.SAR bit.

**(c) Monitor reception of data**

1. Wait for slave event, ICSSR.SDR bit.
2. Clear the ICSSR.SDR bit after reading data from ICRXD register.

**(d) Wait for end of reception**

1. Handle the ICSSR.SDR in the last byte: that is, read the data and clear the ICSSR.SDR.
2. Wait for master event, ICSSR.SSR.
3. Clear the ICSSR.SSR bit.

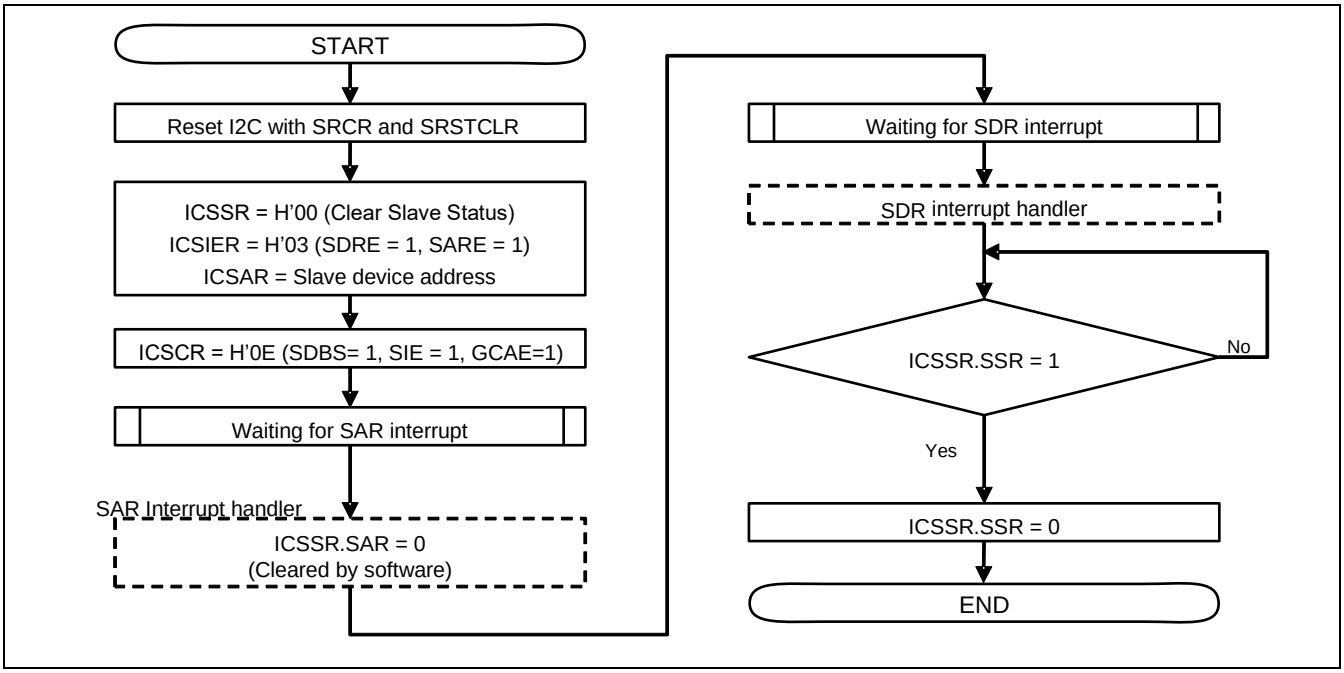



**Figure 107.24 Slave Receiver Operation Setting**

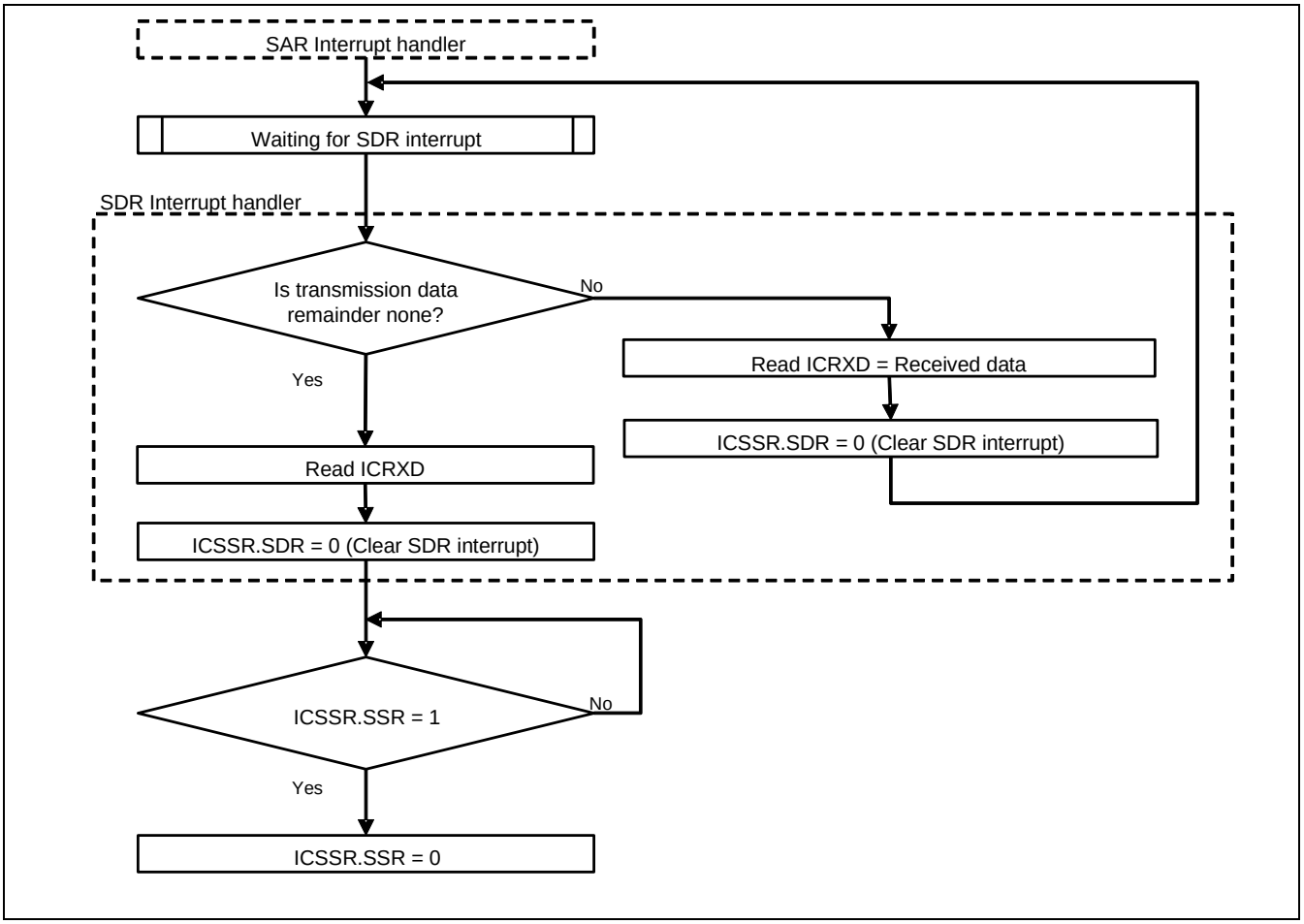



**Figure 107.25 SDR Interrupt Handler for Slave Receiver Operation Setting**

### 107.3.10.6 DMA Mode - Master Transmit Operation

The 1st byte is transferred by PIO, after that enter to DMA mode to start transfer the following bytes.

In DMA transmit mode, I2C module asserts a DMA request to the DMAC. After receiving the request, DMAC write data into I2C data register, and return DMARACK to I2C module. For instance, in order to transfer N0, N1, … , N10 byte. Byte N0 will be transferred by PIO, bytes N1 to N10 will be transferred by using DMAC.

The transmission procedure and operations are described as below:

**(a) Set value for the clock control register**

1. Reset I2C with SRCR and SRSTCLR.
2. Set the Clock Control register 2 (ICCCR2) = H'07 (CDFD = 1, HLSE = 1, SME = 1).
3. Set the Clock Control register (ICCCR) = H'06 (SCGD = 0, CDF = 6, I2C internal clock I2Cck: 19 MHz).
   — Setting ICCCR, ICCCR2 for using Duty ratio fixed or Duty ratio variable. The case for using Variable Duty ratio, have to setting ICCCR2 too. The case for using Fixed Duty ratio, have to setting ICCCR only.
4. Set the SCL Mask Control register (ICMPR) = H'1C (Variable Duty ratio only).
5. Set the SCL High Control register (ICHPR) = H'73 (Variable Duty ratio only).
6. Set the SCL Low Control register (ICLPR) = H'85 (Variable Duty ratio only).
7. Set the First Bit Setup Cycle register (ICFBSCR) = H'0F (1st bit setup cycle = 17 × Tcyc).

**(b) Setting of DMAC**

1. Set the number of DMA times ("Total transmit times – 1").
   - In this phase, DMACHCR.DE = 0.

**(c) Set value for the master control register, first data byte, and address**

1. Clear the Master Status register (ICMSR) = H'00 (all 0 clear).
2. Set the Master Interrupt Enable register (ICMIER) = H'09 (MDEE = 1, MATE = 1).
3. Set the Master Address register (ICMAR) = Slave device address + H'00 (Write mode).
4. Set the Transmit Data register (ICTXD) = 1st transmission data.
5. Set the Master Control register (ICMCR) = H'89 (MDBS = 1, MIE = 1, ESG = 1).

**(d) Wait for outputting address**

1. Wait for master event, ICMSR.MAT and ICMSR.MDE.
2. Set ICMCR to H'88 (to suspend the data transmission, the master device will hold the SCL low until the ICMSR.MDE bit is cleared).
   - If only one byte of data is transmitted, set the ICMCR to H'8A, meaning that the stop generation is enabled. This generates a stop on the bus as soon as one byte has been transmitted.
3. Clear the ICMSR.MAT & ICMSR.MDE bits.

**(e) Enable DMA**

1. Wait for master event, ICMSR.MDE.
2. Set the DMA Enable register (ICDMAER) = H'01 (TMDMAE = 1).
3. Set the DMACHCR.DE = 1 to enable DMA.

**(f) DMA interrupt handler**

1. Wait for DEI interrupt.
2. Set ICDMAER = H'00 (TMDMAE = 0) to disable DMA transmission.

**(g) MDE interrupt handler**

1. Wait for MDE interrupt.
2. Set ICMCR = H’8A (FSB = 1) to make stop condition.
3. Clear MDE interrupt (ICMSR.MDE = 0).

**(h) Wait for end of transmission**

1. Wait for ICMSR.MST = 1.
2. Clear ICMSR.MST = 0.

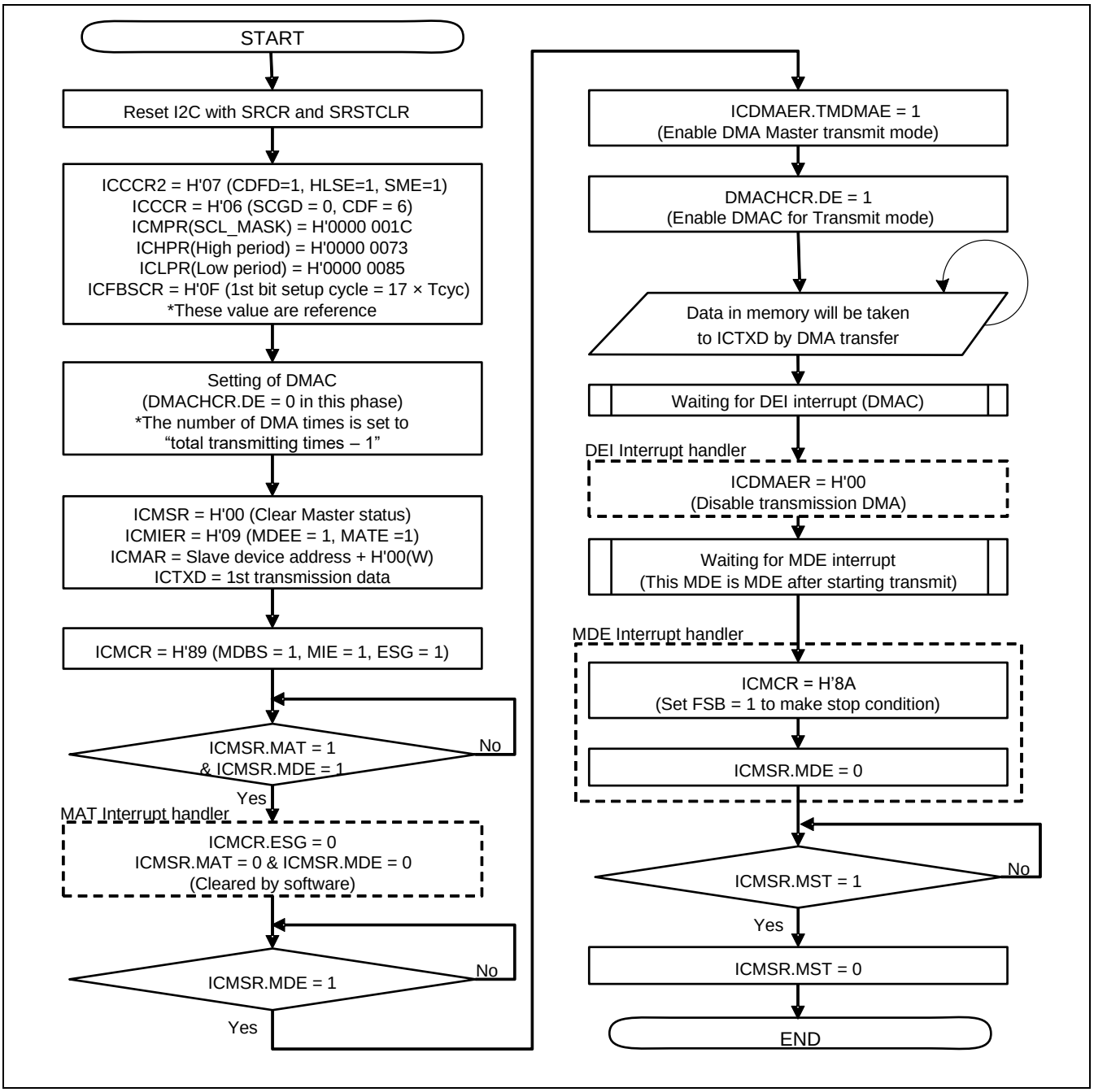



**Figure 107.26 Example DMA mode – Master Transmitted Operation Setting**

### 107.3.10.7 DMA Mode – Master Receive Operation

In DMA receive mode, I2C module asserts a DMA request to the DMAC. After receiving the request, DMAC reads the received data from I2C data register and returns DMARACK to I2C module. All received data bytes will be transferred by using DMAC.

The reception procedure and operations are described as below:

**(a) Set value for the clock control register**

The same procedure as the master transmitter is applied. See the program example for the master transmitter.

**(b) Set value for the master control register, first data byte, and address**

1. Clear the Master Status register (ICMSR) = H’00 (all 0 clear).
2. Set the Master Interrupt Enable register (ICMIER) = H’0B (MDEE = 1, MDRE = 1, MATE = 1).
3. Set the Master Address register (ICMAR) = Slave device address + H’00 (Write mode).
4. Set the Transmit Data register (ICTXD) = 1st transmission data.
5. Set the Master Control register (ICMCR) = H’89 (MDBS = 1, MIE = 1, ESG = 1).

**(c) Wait for outputting address**

1. Wait for master event (an interrupt of the ICMSR.MAT and ICMSR.MDE).
2. Set the ICMCR to H'88 (to suspend the data transmission, the master device will hold the SCL low until the MDE bit is cleared).
   — This generates a stop on the bus as soon as one byte has been transmitted.
3. Clear the ICMSR.MAT & ICMSR.MDE bits.

**(d) MDE interrupt handler**

1. Wait for master event, ICMSR.MDE.
2. Set the ICMAR = slave device address + H’01(Read mode).
3. Set the ICMCR = H’89 (MDBS = 1, MIE = 1, ESG = 1).
4. Clear the ICMSR.MDE = 0.

**(e) Enable DMA of I2C**

1. Wait for master event, ICMSR.MAT and ICMSR.MDR.
2. Clear the ICMCR.ESG = 0.
3. Set the DMA Enable register (ICDMAER) = H’02 (RMDMAE = 1).
4. Clear the ICMSR.MAT and ICMSR.MDR bits.

**(f) Setting of DMAC**

1. Set the number of DMA times (“Total receiving times - 2”).
   — In this phase, DMACHCR.DE = 0.

**(g) Enable DMA of DMAC**

1. Set the DMACHCR.DE = 1 to enable DMA.

**(h) DMA interrupt handler**

1. Wait for DEI interrupt.
2. Set ICDMAER = H’00 (RMDMAE = 0) to disable DMA transmission.

**(i) MDR interrupt handler**

1. Wait for master event, ICMSR.MDR.
2. Set the ICMCR = H’8A (MDBS = 1, MIE = 1, FSB = 1).
3. Read ICRXD.
4. Clear the ICMSR.MDR = 0.

**(j) Final MDR interrupt handler**

1. Wait for master event, ICMSR.MDR.
2. Read ICRXD.
3. Clear the ICMSR.MDR = 0.

**(k) Wait for end of transmission**

1. Wait for the master event, ICMSR.MST.
2. Clear the ICMSR.MST = 0.

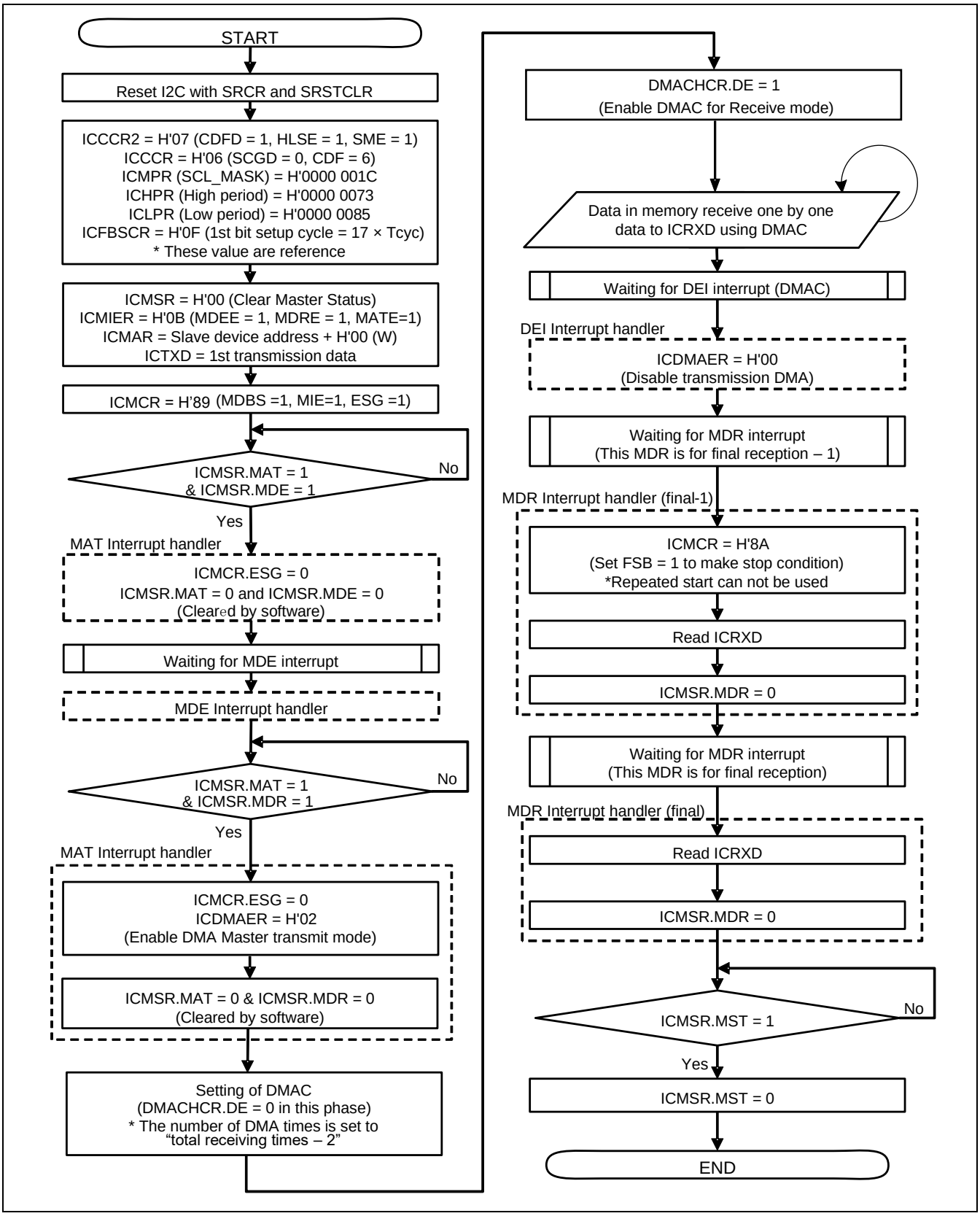



**Figure 107.27 Example DMA mode – Master Received Operation Setting**

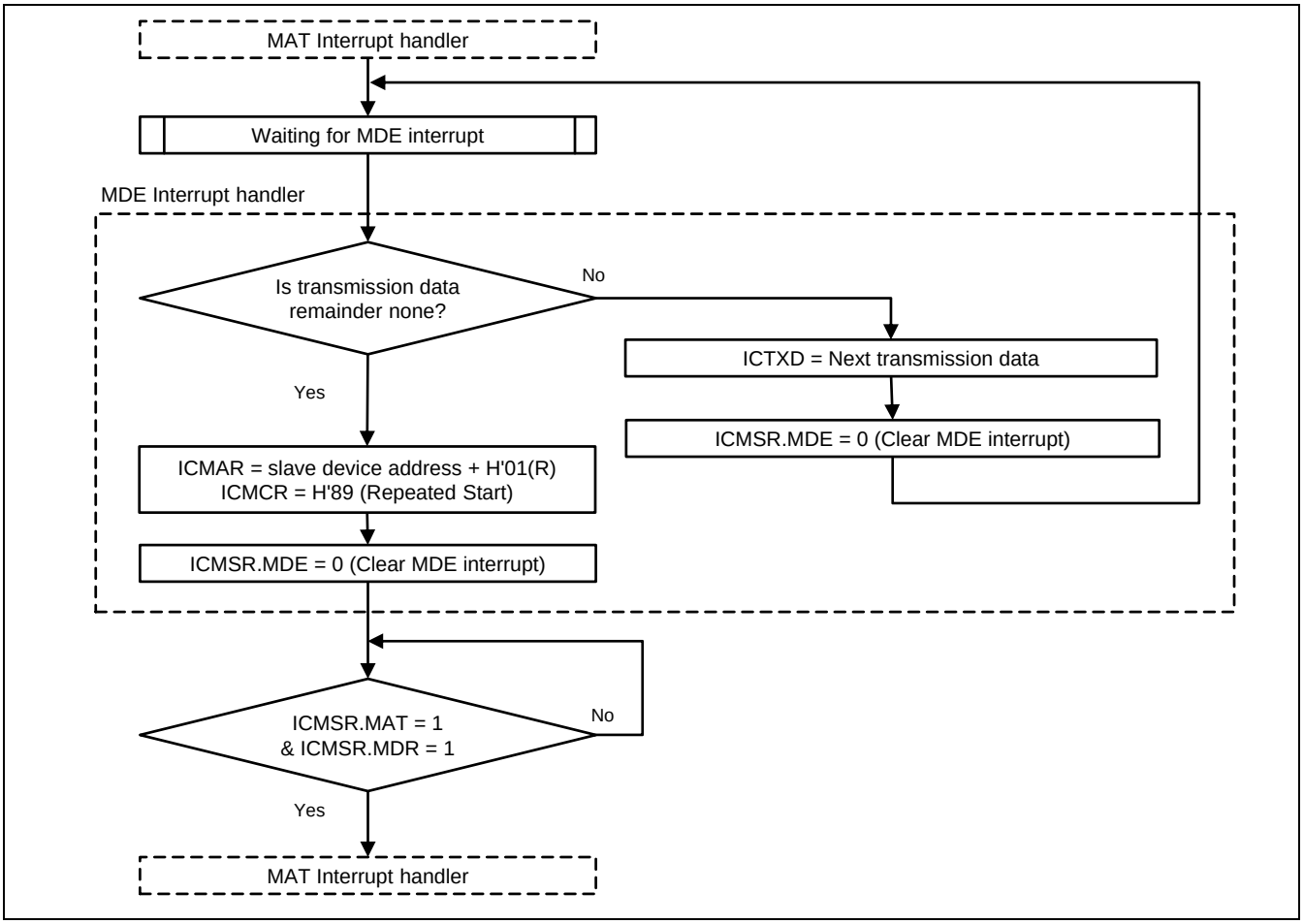



**Figure 107.28 MDE Interrupt Handler for Example DMA mode – Master Received Operation Setting**

### 107.3.10.8 DMA Mode – Slave Transmit Operation

The 1st byte is transferred by PIO, after that enter to DMA mode to start transfer the following bytes.

In DMA transmit mode, I2C module asserts a DMA request to the DMAC. After receiving the request, DMAC write data into I2C data register, and return DMARACK to I2C module. For instance, in order to transfer N0, N1, … , N10 byte. Byte N0 will be transferred by PIO, bytes N1 to N10 will be transferred by using DMAC.

The transmission procedure and operations are described as below:

**(a) Setting of DMAC**

1. Set the number of DMA times (“Total transmit times – 1”).
   — In this phase, DMACHCR.DE = 0.

**(b) Set value for the slave control register, first data byte, and address**

1. Reset I2C with SRCR and SRSTCLR.
2. Clear the Slave Status register (ICSSR) = H’00 (all 0 clear).
3. Set the Slave Interrupt Enable register (ICSIER) = H’09 (SDEE = 1, SARE = 1).
4. Set the Slave Address register (ICSAR) = Slave device address.
5. Set the Transmit Data register (ICTXD) = 1st transmission data.
6. Set the DMA Enable register (ICDMAER) = H’04 (TSDMAE = 1).
7. Set the Slave Control register (ICSCR) = H’0E (SDBS = 1, SIE = 1, GCAE = 1).

**(c) Wait for receiving address**

1. Wait for slave event, ICSSR.SAR.
2. Clear the ICSSR.SAR = 0

**(d) Enable DMA**

1. Wait for slave event, ICSSR.SDE.
2. Set the DMACHCR.DE = 1 to enable DMA.

**(e) DMA interrupt handler**

1. Wait for DEI interrupt.
2. Set ICDMAER = H’00 (TSDMAE = 0) to disable DMA transmission.

**(f) SDE interrupt handler**

1. Wait for slave event, ICSSR.SDE.
2. Clear SDE interrupt (ICSSR.SDE = 0).

**(g) Wait for end of transmission**

1. Wait for ICSSR.SSR = 1.
2. Set ICSSR.SSR = 0.

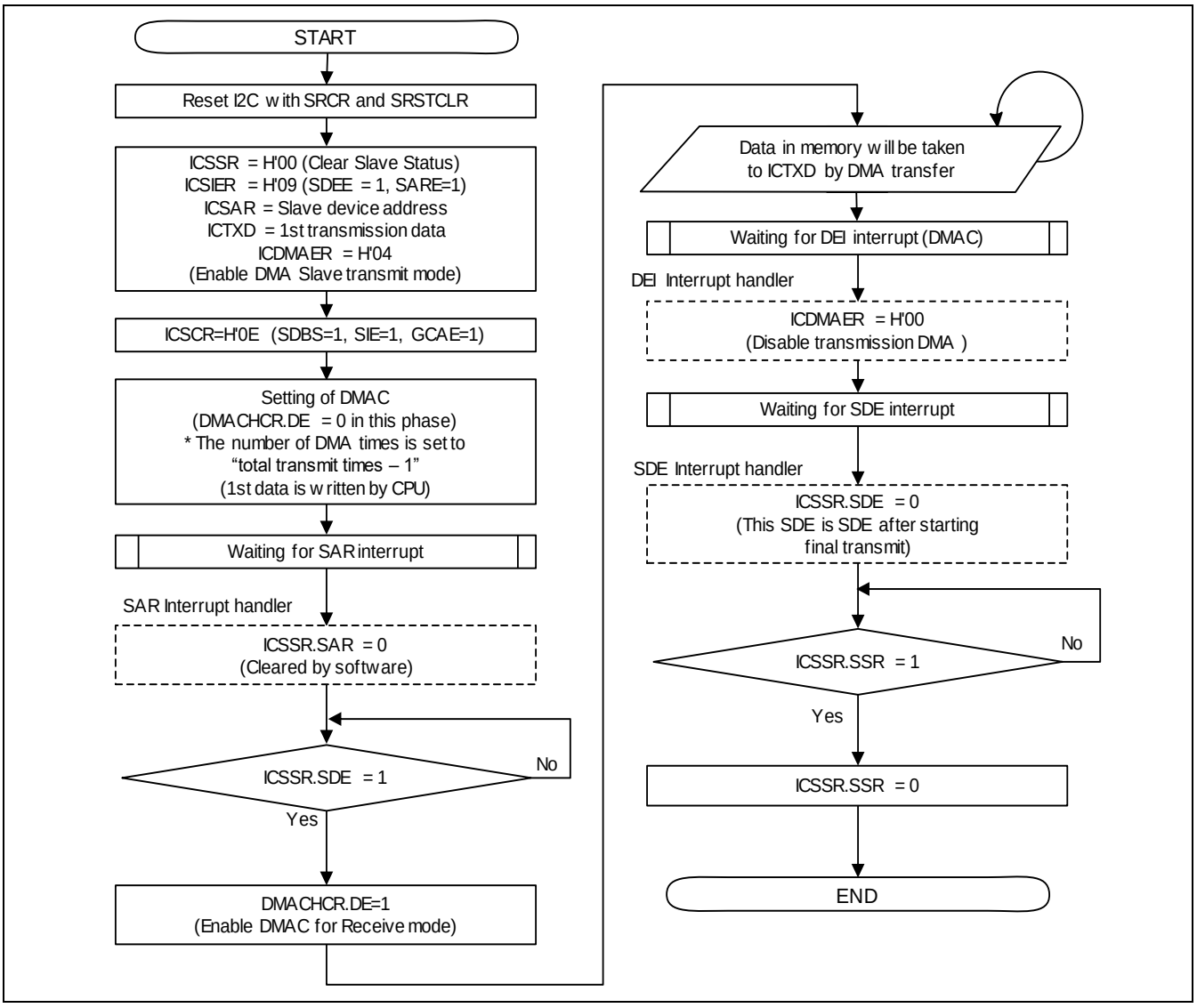



**Figure 107.29 Example DMA mode – Slave Transmitted Operation Setting**

### 107.3.10.9 DMA Mode – Slave Receive Operation

In DMA receive mode, I2C module asserts a DMA request to the DMAC. After receiving the request, DMAC reads the received data from I2C data register and returns DMARACK to I2C module. All received data bytes will be transferred by using DMAC.

The reception procedure and operations are described as below:

**(a) Setting of DMAC**

1. Set the number of DMA times (“Total receiving times”).
   — In this phase, DMACHCR.DE = 0.

**(b) Set value for the slave control register, first data byte, and address**

1. Reset I2C with SRCR and SRSTCLR.
2. Clear the Slave Status register (ICSSR) = H’00 (all 0 clear).
3. Set the Slave Interrupt Enable register (ICSIER) = H’03 (SDRE = 1, SARE = 1).
4. Set the Slave Address register (ICSAR) = Slave device address.
5. Set the DMA Enable register (ICDMAER) = H’08 (RSDMAE = 1).
6. Set the Slave Control register (ICSCR) = H’0E (SDBS = 1, SIE = 1, GCAE = 1).

**(c) Wait for receiving address**

1. Wait for slave event, ICSSR.SAR.
2. Clear the ICSSR.SAR = 0.

**(d) Enable DMA**

1. Wait for slave event, ICSSR.SDR.
2. Set the DMACHCR.DE = 1 to enable DMA.

**(e) DMA interrupt handler**

1. Wait for DEI interrupt.
2. Set ICDMAER = H’00 (RSDMAE = 0) to disable DMA transmission.

**(f) Wait for end of transmission**

1. Wait for ICSSR.SSR = 1.
2. Set ICSSR.SSR = 0.

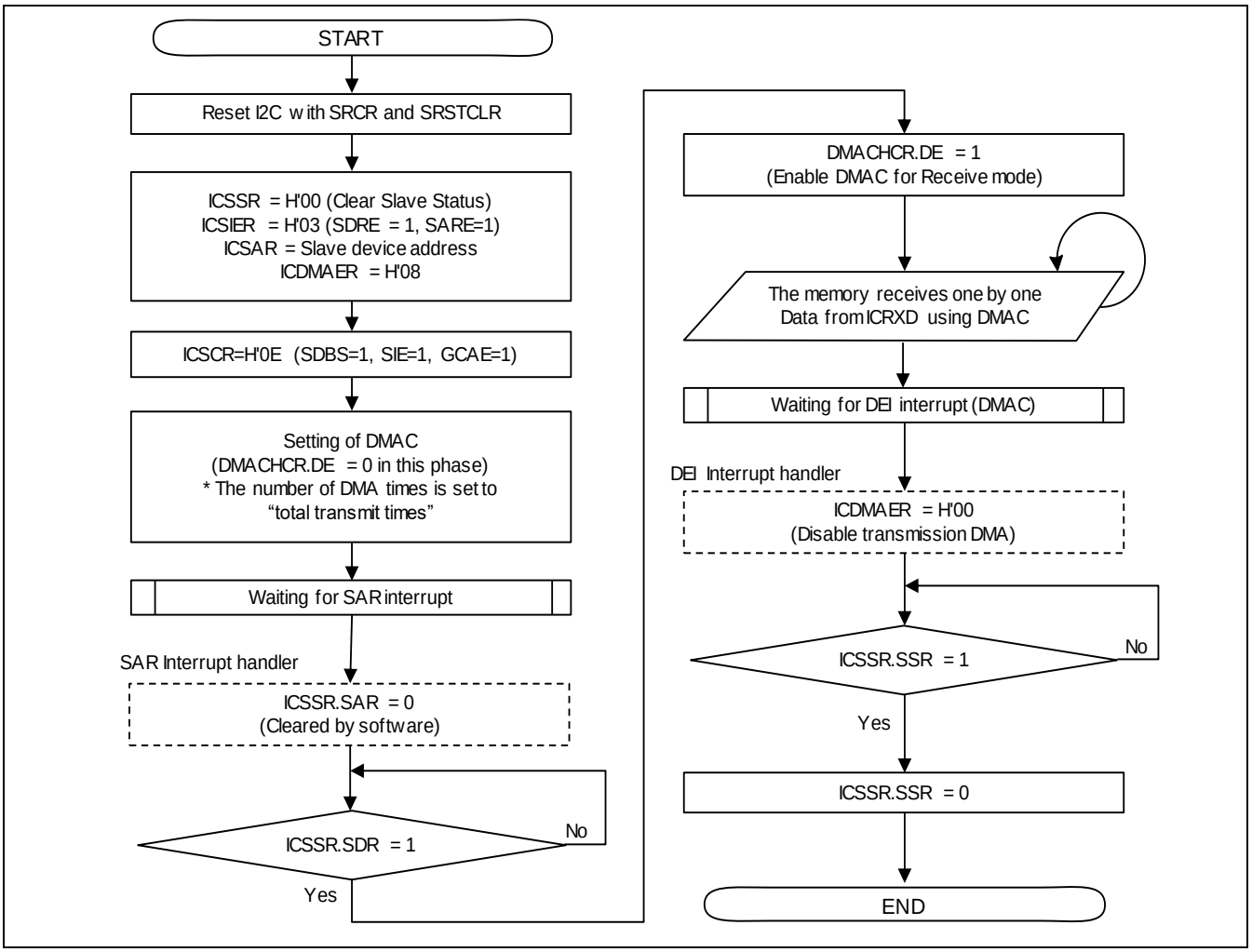



**Figure 107.30 Example DMA mode – Slave Received Operation Setting**

### 107.3.10.10 DMA Mode – Master Transmit continuous mode Operation

The Start trigger (ESG) is set by PIO, 1st Byte transmission is started and completed automatically.
After that enter to DMA mode to start transfer the following bytes.

In DMA transmit continuous mode, I2C module asserts a DMA request to the DMAC. After receiving the request, DMAC write data into I2C data register, and return DMARACK to I2C module.
Each Byte is written to I2C data register by the DMAC, I2C transmission is started automatically and assert DMA request to the DMAC again.
This transmission continues the number of (MDMACTSZ + 1) × TMDMATSZ. After this number of transfer, I2C stops this transmission by de-asserting TMDMACE.

The transmission procedure and operations are described as below:

1. Setting ICCCR, ICCCR2 for using Normal mode or Duty variable mode.
2. Setting for DMAC source, destination address, transfer count, data.
3. Set ICMIER = H'09 to enable MDE, MAT interrupt (condition to issue DMA request).
4. Set the 1st data byte which will be transferred by asserting ESG.
5. Set 1st bit setup delay time between SCL and SDA, default is H'04 = Tcyc × 6
6. Set ICDMAER = H'XX00YY81 to enable DMA Master Transmit mode and DMA Master Continuous Transmitted mode. (MDMACTSZ = H'XX, TMDMACSZ = H'YY. This means “XX times of YY bytes transmission”)
7. Set ICMAR.STM1 = 0 (Transmit direction); set ICMAR.SADD to select the Slave-address.
8. Monitor the FSDA bit in the master control register. Confirm that this bit is low, meaning that other I2C devices are not using the bus. After confirmation, set the MIE (bit 3), ESG (bit 0) and MDBS (bit 7) bits in the master control register ICMCR to 1 to start master transmission. This setting will transfer the 1st byte by using PIO.
9. Wait for until the bits flag MDE = 1 and MAT= 0. This means complete 1stByte transmission.
   (ESG and MAT is cleared automatically)
10. Set DMAOR = 1 to start DMA transfer. DMAC writes the transmit data into ICTXD register from memory space specified by DMAC register until transmission finishes.
11. After the number of (MDMACTSZ + 1) × TMDMATSZ transfer, I2C will stops this mode by clearing TMDMACE.
12. Wait for the clearing TMDMACE.
    Reset the MST bit after confirming MNR (master NACK received).

Figure 107.31 shows Example Flow for DMA mode – Master Continuous Transmission

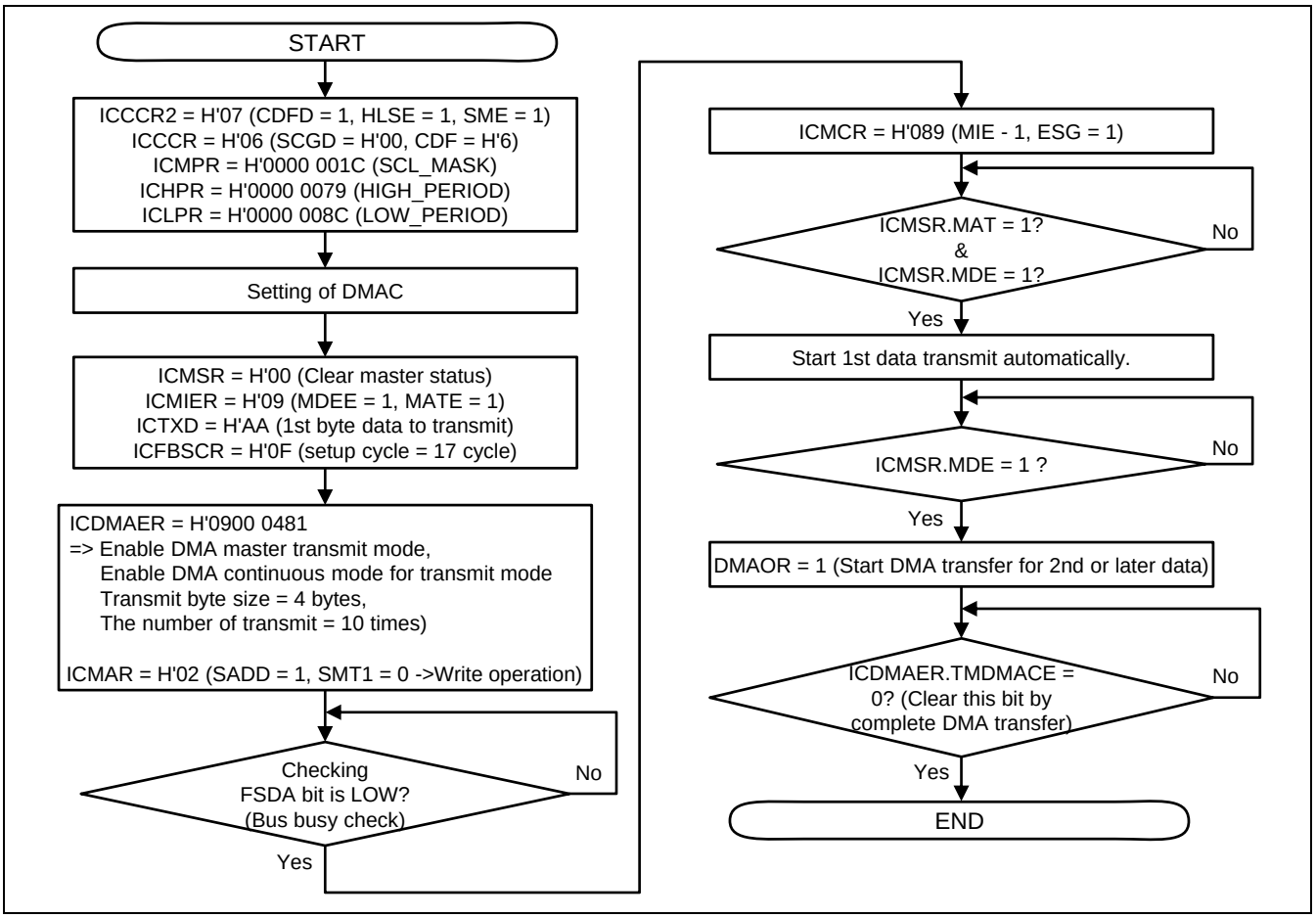



**Figure 107.31 Example DMA mode – Master Transmitted Continuous Mode Operation Setting**

### 107.3.10.11 DMA Mode – Master Receive continuous mode Operation

The Start trigger (ESG) is set by PIO, 1st Byte reception is started and completed automatically.
After that enter to DMA mode to start transfer the following bytes.

In DMA receive continuous mode, I2C module asserts a DMA request to the DMAC. After receiving the request, DMAC read data from I2C data register, and return DMARACK to I2C module.
Each Byte is read from I2C data register by the DMAC, I2C reception is started automatically and assert DMA request to the DMAC again.
This reception continues the number of (MDMACTSZ + 1) × RMDMATSZ. After this number of transfer, I2C stops this reception by de-asserting RMDMACE.

The reception procedure and operations are described as below:

1. Setting ICCCR, ICCCR2 for using Normal mode or Duty variable mode.
2. Setting for DMAC source, destination address, transfer count, data.
3. Set ICMIER = H'03 to enable MDR, MAT interrupt (condition to issue DMA request).
4. Set 1st bit setup delay time between SCL and SDA, default is H'04 = Tcyc × 6
5. Set ICDMAER = H'XXYY0042 to enable DMA Master Receive mode and DMA Master Continuous Received mode. (MDMACTSZ = H'XX, RMDMACSZ = H'YY. This means “XX times of YY bytes reception”)
6. Set ICMAR.STM1 = 1 (Received direction); set ICMAR.SADD to select the Slave-address.
7. Monitor the FSDA bit in the master control register. Confirm that this bit is low, meaning that other I2C devices are not using the bus. After confirmation, set the MIE (bit 3), ESG (bit 0) and MDBS (bit 7) bits in the master control register ICMCR to 1 to start master reception. This setting will transfer the 1st byte by using PIO.
8. Wait for until the bits flag MDR = 1 and MAT = 0. This means complete 1stByte transmission. (ESG and MAT is cleared automatically)
9. Set DMAOR = 1 to start DMA transfer. DMAC reads the received data from ICRXD register to memory space specified by DMAC register until reception finishes.
10. After the number of (MDMACTSZ + 1) × RMDMATSZ transfer, I2C will stops this mode by clearing RMDMACE.
11. Wait for the clearing RMDMACE.
    Reset the MST bit after confirming MNR (master NACK received).

Figure 107.32 shows Example Flow for DMA mode – Master Continuous Reception

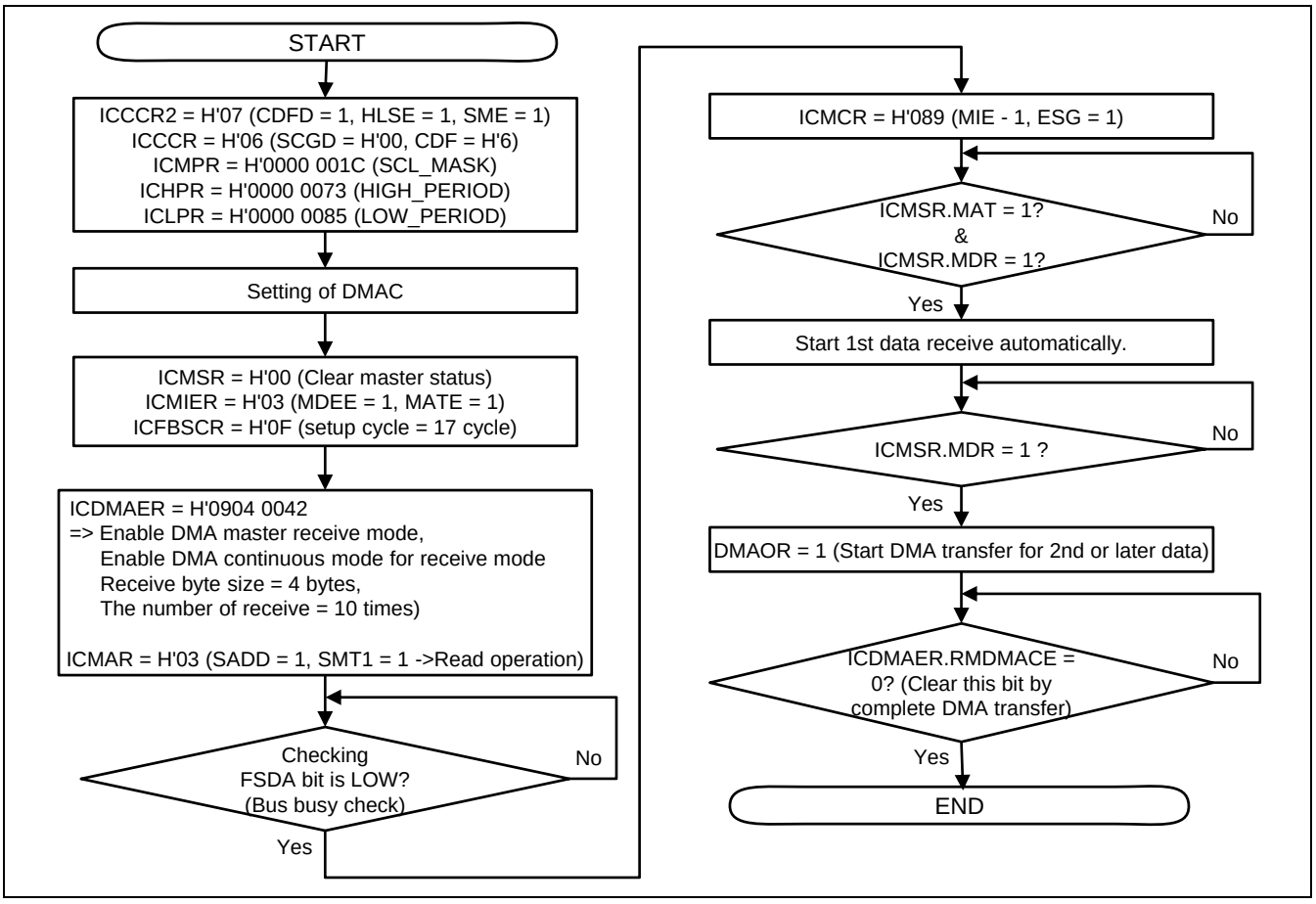



**Figure 107.32 Example DMA mode – Master Received Continuous Mode Operation Setting**

## 107.4 Usage Notes

### 107.4.1 Usage note for Multi-Master

Do not stop clock distribution to this module, even if this module is not directly used for data transfer, while this module is used in multi-master environment. This module is monitoring SCL and SDA bus line for detecting START and STOP condition for arbitration. If clock is stopped, arbitration problem may occur.

### 107.4.2 Usage note for SCL frequency calculation equation

In section 107.2.9, Clock Control Register (ICCCR), in the description of SCGD and CDF, following equation is described.

SCLfreq = I2Cck/(20 + SCGD × 8 + F[(tICF + tr + IntDelay) × I2Cck])

I2Cck: I2C internal clock frequency

tICF: I2C SCL falling time (depending on external load)

tr: I2C SCL rising time (depending on external load)

IntDelay: LSI internal delay corresponds to output buffer type.
Open drain buffer: 50 ns (typ.), 110 ns (max.)

F[n]: n rounded down to an integer

I2Cck = MOD_CLK/(1 + CDF)

MOD_CLK: module clock frequency

Based on parameter described above, this equation approximately estimates the SCL frequency, but includes some error in the resulting frequency. Before finalizing register setting to be used, measure actual SCL frequency on the board, and confirm that observed frequency satisfy the constraint. The reason why calculated result includes some error is parameters used in this equation varies depending on the characteristics of each sample. For example, tICF and tr depend on logical threshold of IO-cell, and intDelay depends on variation of fabrication process.

For adjustment of SCL frequency, modifying the setting of ICCCR [2:0]. CDF and ICCCR [8:3]. SCGD will be effective. Increasing the value of CDF or SCGD value will make SCL frequency lower. However, how they decrease SCL frequency is different. So, by increasing one register value and decreasing the other, SCL frequency can be adjusted.

### 107.4.3 Usage note for Pull-up resistance

Do not use internal pull up of PUEN and PUD registers in Pin Function Controller (PFC) section for I2C function. External pull up resistance on board is mandatory for SCL and SDA pins.

### 107.4.4 Usage note for the transmission and reception procedure

Be sure to reset I2C with Software Reset Register SRCR5 at the beginning of transmission and reception procedure.
A software reset can be set by controlling the Software Reset Register SRCR5 and Software Reset Clearing Register SRSTCLR5 of the “9. Module Standby, Software Reset”

Reset Sequence:

Set the target I2C bit of SRCRn to 1. (**) → Wait 1µsec → Set the target I2C bit of SRSTCLRn to 1. (**)
→ Wait for the target bit of SRCRn to 0. → End sequence

**: Other bit should be written 0.

### 107.4.5 Usage note for DMA mode of Receive Operation

If use DMA mode of Receive Operation more than once with repeated START,
issue "STOP condition" and start from beginning of transmission and reception procedure instead of repeated START.

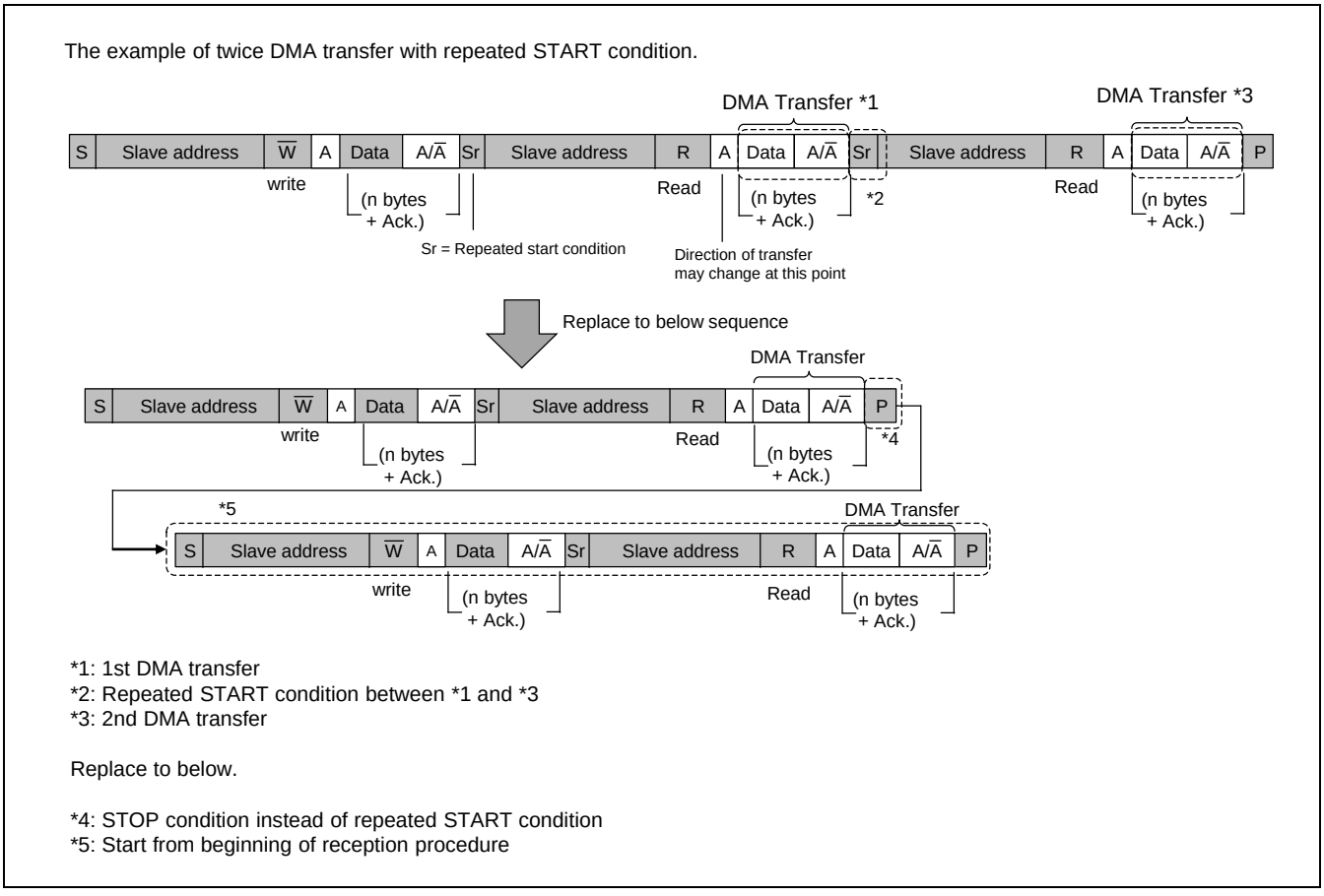



**Figure 107.33 Example of Usage Note for DMA mode of Receive Operation**

## 107.5 Safety Mechanisms

I2C module has no Mechanism of FuSa.

# 109. Clock-Synchronized Serial Interface with FIFO (MSIOF)

## 109.1 Overview

This LSI includes clock-synchronized serial I/O module with FIFO (MSIOF). In this module, MSOφ is used as a module clock.

### 109.1.1 Features

- Number of channels: Six channels
- FIFO capacity: 32 bits × 256 stages for transmission and 32 bits × 256 stages for reception
- MSB first or LSB first selectable for data transmission and reception
- Synchronization by frame synchronization pulse, level, or left/right channel switch
- Supports master and slave modes
- Interrupts: One interrupt line, with 12 internal factors per module.
- Serial clock
- DMA transfer
- Serial format

  Supports serial formats: IIS, SPI (master and slave modes).
- CLK/SYNC (SPI) in common transmit/receive mode

### 109.1.2 Block diagram

Figure 109.1 shows a block diagram of the MSIOF.

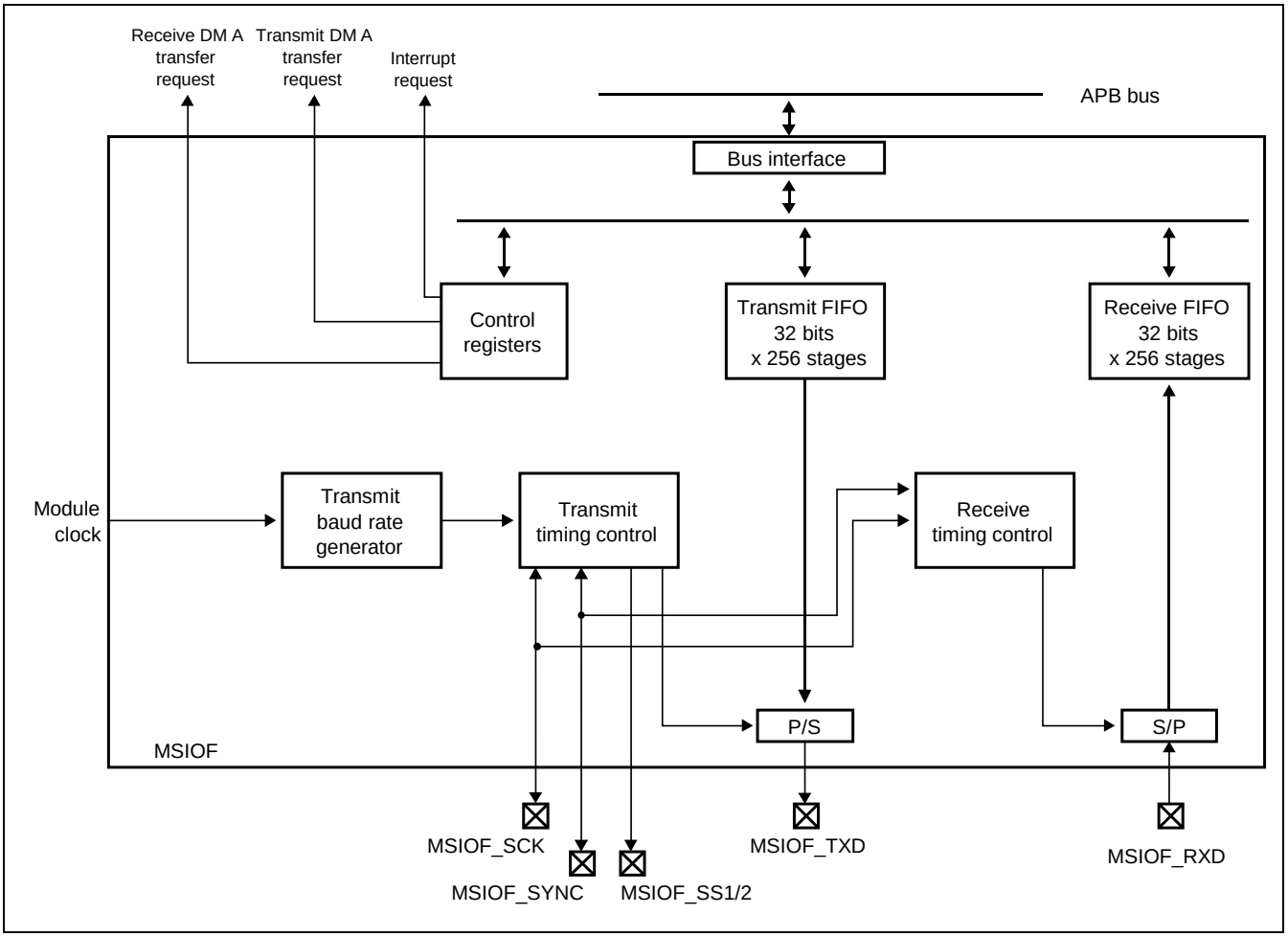



**Figure 109.1 Block Diagram of MSIOF**

### 109.1.3 External Pins

The pin configuration of this module is shown in Table 109.1 and Table 109.2.

**Table 109.1 Pin Configuration**

| Name | Abbreviation | I/O | Function | R-Car V4M |
|---|---|---|---|---|
| MSIOF_SCK | SCK | I/O | MSIOF serial clock for transmission | √ |
| MSIOF_SYNC | SYNC | I/O | MSIOF frame synchronization signal for transmission | √ |
| MSIOF_SS1 | SS1 | Output | MSIOF frame synchronization signal 1 for transmission<br>This signal can work only master mode. | √ |
| MSIOF_SS2 | SS2 | Output | MSIOF frame synchronization signal 2 for transmission<br>This signal can work only master mode. | √ |
| MSIOF_TXD | TXD (MOSI/MISO) | Output | MSIOF transmit data | √ |
| MSIOF_RXD | RXD (MISO/MOSI) | Input | MSIOF receive data | √ |

**Table 109.2 Pin Group**

| Module Name | Group Suffix | Pin Name<br>XXX is Abbreviation of Table 109.1 |
|---|---|---|
| MSIOF0 | — | MSIOF0_XXX |
| MSIOF1 | — | MSIOF1_XXX |
| MSIOF2 | — | MSIOF2_XXX |
| MSIOF3 | — | MSIOF3_XXX |
| MSIOF4 | — | MSIOF4_XXX |
| MSIOF5 | — | MSIOF5_XXX |

### 109.1.4 Register Configuration

Table 109.3 and Table 109.4 show the MSIOF register configuration and Table 109.5 shows the register state in each processing mode. Do not write to any address other than those listed in the tables. Otherwise, the operation is not guaranteed. Reading a non-listed address returns an undefined value. Access the registers should be accessed with the "Access Size" listed below. Otherwise, the operation is not guaranteed.

**Table 109.3 Module Base Address Summary**

| Module Name | Base Address | R-Car V4M |
|---|---|---|
| MSIOF0 | H'E6E9 0000 | √ |
| MSIOF1 | H'E6EA 0000 | √ |
| MSIOF2 | H'E6C0 0000 | √ |
| MSIOF3 | H'E6C1 0000 | √ |
| MSIOF4 | H'E6C2 0000 | √ |
| MSIOF5 | H'E6C2 8000 | √ |

**Table 109.4 Register Configuration**

| Name | Abbreviation | R/W | Address Offset | Access Size | R-Car V4M |
|---|---|---|---|---|---|
| MSIOF transmit mode register 1 | SITMDR1 | R/W | H'0000 | 32 | √ |
| MSIOF transmit mode register 2 | SITMDR2 | R/W | H'0004 | 32 | √ |
| MSIOF transmit mode register 3 | SITMDR3 | R/W | H'0008 | 32 | √ |
| MSIOF receive mode register 1 | SIRMDR1 | R/W | H'0010 | 32 | √ |
| MSIOF receive mode register 2 | SIRMDR2 | R/W | H'0014 | 32 | √ |
| MSIOF receive mode register 3 | SIRMDR3 | R/W | H'0018 | 32 | √ |
| MSIOF transmit clock select register | SITSCR | R/W | H'0020 | 16 | √ |
| MSIOF control register | SICTR | R/W | H'0028 | 32 | √ |
| MSIOF FIFO control register | SIFCTR | R/W | H'0030 | 32 | √ |
| MSIOF status register | SISTR | R/WC1*2 | H'0040 | 32 | √ |
| MSIOF interrupt enable register | SIIER | R/W | H'0044 | 32 | √ |
| MSIOF transmit FIFO data register | SITFDR | W | H'0050 | 32, 16*1, 8*1 | √ |
| MSIOF receive FIFO data register | SIRFDR | R | H'0060 | 32, 16*1, 8*1 | √ |

Notes: 1. 8-bit access and 16-bit access to SIRFDR and SITFDR are capable .
2. For this register, bits 28 and 12 are read-only bits. For details, refer to Section 109.2.11 MSIOF Status Register (SISTR)

**Table 109.5 Register States in Each Operating Mode**

| Abbreviation | Reset | Module Standby |
|---|---|---|
| SITMDR1 | Initialized | Retained |
| SITMDR2 | Initialized | Retained |
| SITMDR3 | Initialized | Retained |
| SIRMDR1 | Initialized | Retained |
| SIRMDR2 | Initialized | Retained |
| SIRMDR3 | Initialized | Retained |
| SITSCR | Initialized | Retained |
| SICTR | Initialized | Retained |
| SIFCTR | Initialized | Retained |
| SISTR | Initialized | Retained |
| SIIER | Initialized | Retained |
| SITFDR | Initialized | Retained |
| SIRFDR | Retained | Retained |

### 109.1.5 Connected Module

Table 109.6 shows the connected modules to the MSIOF.

**Table 109.6 Connected modules**

| Module name | Connected module name | Function of connected module |
|---|---|---|
| MSIOF | AP-System Core | Access the Register |
| | CPG | Output Clocks |
| | PFC | Select External pins |
| | Module Standby | Control to stop clocks |
| | Software Reset | Execute software reset |
| | INTC-AP, INTC-RT | Control to interrupt |
| | SYS-DMAC | Control Direct Memory Access |

## 109.2 Register Description

[Legend for Register Descriptions]

Initial value: Register value after a reset

—: Undefined value

R/W: Readable/writable. The written value can be read.

R/WC1: Readable/writable. Writing 1 initializes the bit, and writing 0 is ignored.

R: Read-only. The write value should always be initial value.

—/WB: Write-only. The read value is undefined.

### 109.2.1 MSIOF Transmit Mode Register 1 (SITMDR1)

SITMDR1 is a 32-bit readable/writable register that specifies the MSIOF transmit mode.
The settings of this register should be not changed when SICTR.RXE or SICTR.TXE are B'1. Otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRMD | PCON | SYNCMD[1:0] | | SYNCCH[1:0] | | SYNCAC | BITLSB | — | DTDL[2:0] | | | — | SYNCDL[2:0] | | |
| Initial value: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | FLD[1:0] | | — | TXSTP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 | TRMD | 1 | R/W | Transfer Mode<br>Selects the transfer mode.<br>0: Slave mode<br>1: Master mode |
| 30 | PCON | 0 | R/W | Transfer Signal Connection<br>0: Setting prohibited<br>1: MSIOF_SCK and MSIOF_SYNC are used as common signals for transmission and reception (MSIOF_SCK and MSIOF_SYNC). |
| 29, 28 | SYNCMD[1:0] | 00 | R/W | SYNC Mode<br>These bits specify the mode for the MSIOF_SYNC signal.<br>00: Frame start synchronization pulse<br>01: Reserved<br>10: Level mode/SPI<br>11: L/R mode |
| 27, 26 | SYNCCH[1:0] | 00 | R/W | Synchronization Signal Channel Select<br>These bits are valid only in master mode.<br>00: The frame synchronization signal output at MSIOF_SYNC.<br>01: The frame synchronization signal output at MSIOF_SS1.<br>10: The frame synchronization signal output at MSIOF_SS2.<br>11: Setting prohibited |
| 25 | SYNCAC | 0 | R/W | MSIOF_SYNC Polarity<br>0: Active-high signal in synchronization pulse or level mode, or driven high then low in L/R mode<br>1: Active-low signal in synchronization pulse or level mode, or driven low then high in L/R mode |
| 24 | BITLSB | 0 | R/W | MSB/LSB First<br>0: MSB first<br>1: LSB first |
| 23 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 to 20 | DTDL[2:0] | 001 | R/W | Data Pin Bit Delay for MSIOF_SYNC Pin<br>The value specified with TXDIZ in SICTR is output during transmission.<br>B'1xx is valid only in SPI mode.<br>In SPI mode, these bits should be specified so that the sum of the delays specified through DTDL and SYNCDL becomes an integer value.<br>000: No bit delay<br>001: 1-clock-cycle delay<br>010: 2-clock-cycle delay<br>101: 0.5-clock-cycle delay<br>110: 1.5-clock-cycle delay<br>Other than above: Setting prohibited |
| 19 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 18 to 16 | SYNCDL[2:0] | 000 | R/W | Frame Synchronization Signal Timing Delay<br>These bits extend the transmit frame synchronization signal. The value specified with TXDIZ in SICTR is output during transmission.<br>These bits are invalid when SYNCMD [1:0] = B'01. B'1xx is valid only in SPI mode. In SPI mode, these bits should be specified so that the sum of the delays specified through DTDL and SYNCDL becomes an integer value.<br>In L/R mode, these bits should always be set to B’000.<br>000: No bit delay<br>001: 1-clock-cycle delay<br>010: 2-clock-cycle delay<br>011: 3-clock-cycle delay<br>101: 0.5-clock-cycle delay<br>110: 1.5-clock-cycle delay<br>Other than above: Setting prohibited |
| 15 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 3, 2 | FLD [1:0] | 00 | R/W | Frame Synchronization Signal Interval<br>Specify the minimum idle time between frames in the number of serial clock cycles. This setting is valid only in master mode.<br>In L/R mode, these bits should always be set to B’00.<br>In SPI mode, these bits shall not be set to B’00.<br>00: 0-clock-cycle delay<br>01: 1-clock-cycle delay<br>10: 2-clock-cycle delay<br>11: 3-clock-cycle delay |
| 1 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 0 | TXSTP | 1 | R/W | Transmission Stop<br>0: Setting prohibited<br>1: Stop a frame from starting to transmit until either the number of FIFO stages equals the frame size or the FIFO is full and simultaneously a frame started cannot be interrupted in between. |

### 109.2.2 MSIOF Transmit Mode Register 2 (SITMDR2)

SITMDR2 is a 32-bit readable/writable register that specifies the MSIOF transmit mode.
The settings of this register should be not changed when SICTR.RXE or SICTR.TXE are B'1. Otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | GRP | — | BITLEN1[4:0] | | | | | WDLEN1[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 30 | GRP | 0 | R/W | Group Count<br>0: Group count 1<br>1: Group count 2 |
| 29 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 28 to 24 | BITLEN1[4:0] | 01111 | R/W | Data Size (8 to 32 bits)<br>The word size (bits) of Group 1 is set to the value specified in these bits + 1. Either 8, 16, 24, or 32 bits can be specified as word size. |
| 23 to 16 | WDLEN1[7:0] | H'00 | R/W | Word Count (1 to 256 words)<br>The word count of Group 1 is set to WDLEN1 + 1. |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 109.2.3 MSIOF Transmit Mode Register 3 (SITMDR3)

SITMDR3 is a 32-bit readable/writable register that specifies the MSIOF transmit mode.
The settings of this register should be not changed when SICTR.RXE or SICTR.TXE are B'1. Otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | BITLEN2[4:0] | | | | | WDLEN2[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 24 | BITLEN2[4:0] | 01111 | R/W | Word Size (8 to 32 bits)<br>The word size (bits) of Group 2 is set to the value specified in these bits + 1. Either 8, 16, 24, or 32 bits can be specified as word size. |
| 23 to 16 | WDLEN2[7:0] | H'00 | R/W | Word Count (1 to 256 words)<br>The word count of Group 2 is set to WDLEN2 + 1. |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 109.2.4 MSIOF Receive Mode Register 1 (SIRMDR1)

SIRMDR1 is a 32-bit readable/writable register that specifies the MSIOF receive mode.
The settings of this register should be not changed when SICTR.RXE or SICTR.TXE are B'1. Otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRMD | — | SYNCMD[1:0] | | — | — | SYNCAC | BITLSB | — | DTDL[2:0] | | | — | SYNCDL[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R/W | R/W | R | R | R/W | R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 | TRMD | 0 | R/W | Transfer Mode<br>Selects the transfer mode.<br>This bit should be set to 0. |
| 30 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 29, 28 | SYNCMD [1:0] | 00 | R/W | SYNC Mode<br>The mode setting in these bits should be the same as that in SITMDR1.SYNCMD.<br>00: Frame start synchronization pulse<br>01: Reserved<br>10: Level mode/SPI<br>11: L/R mode |
| 27, 26 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 25 | SYNCAC | 0 | R/W | SYNC Polarity<br>0: Active-high signal in synchronization pulse or level mode, or driven high then low in L/R mode<br>1: Active-low signal in synchronization pulse or level mode, or driven low then high in L/R mode<br>This bit must have the same setting as the SYNCAC bit of SITMDR1. |
| 24 | BITLSB | 0 | R/W | MSB/LSB First<br>0: MSB first<br>1: LSB first |
| 23 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 22 to 20 | DTDL[2:0] | 001 | R/W | Data Pin Bit Delay for MSIOF_SYNC Pin<br>000: No bit delay<br>001: 1-clock-cycle delay<br>010: 2-clock-cycle delay<br>Other than above: Setting prohibited<br>In case of SPI mode, only B’000 is allowed to set to this field. |
| 19 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 18 to 16 | SYNCDL [2:0] | 000 | R/W | MSIOF_SYNC Timing Delay<br>These bits should always be set to B’000. |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 109.2.5 MSIOF Receive Mode Register 2 (SIRMDR2)

SIRMDR2 is a 32-bit readable/writable register that specifies the MSIOF receive mode.
The settings of this register should be not changed when SICTR.RXE or SICTR.TXE are B'1. Otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | GRP | — | BITLEN1[4:0] | | | | | WDLEN1[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 30 | GRP | 0 | R/W | Group Count<br>0: Group count 1<br>1: Group count 2<br>When using reception in master mode, set the same value as the value specified by the GRP bit in SITMDR2. |
| 29 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 28 to 24 | BITLEN1[4:0] | 01111 | R/W | Word Size (8 to 32 bits)<br>The word size (bits) of Group 1 is set to the value specified in these bits + 1. Either 8, 16, 24, or 32 bits can be specified as word size.<br>When using reception in master mode, set the same value as the value specified by the BITLEN1 bits in SITMDR2. |
| 23 to 16 | WDLEN1[7:0] | H'00 | R/W | Word Count (1 to 256 words)<br>The word count of Group 1 is set to WDLEN1 + 1.<br>When using reception in master mode, set the same value as the value specified by the WDLEN1 bits in SITMDR2. |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 109.2.6 MSIOF Receive Mode Register 3 (SIRMDR3)

SIRMDR3 is a 32-bit readable/writable register that specifies the MSIOF receive mode.
The settings of this register should be not changed when SICTR.RXE or SICTR.TXE are B'1. Otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | BITLEN2[4:0] | | | | | WDLEN2[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 28 to 24 | BITLEN2[4:0] | 01111 | R/W | Word Size (8 to 32 bits)<br>The word size (bits) of Group 2 is set to the value specified in these bits + 1. Either 8, 16, 24, or 32 bits can be specified as word size.<br>When using reception in master mode, set the same value as the value specified by the BITLEN2 bits in SITMDR3. |
| 23 to 16 | WDLEN2[7:0] | H'00 | R/W | Word Count (1 to 256 words)<br>The word count of Group 2 is set to WDLEN2 + 1.<br>When using reception in master mode, set the same value as the value specified by the WDLEN2 bits in SITMDR3. |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 109.2.7 MSIOF Control Register (SICTR)

SICTR is a 32-bit readable/writable register that specifies the MSIOF operating state.

The values written to the TXE, RXE, TFSE, RFSE, TSCKE, and RSCKE bits become valid (can be read from the bits) several cycles after writing. These bits should not be modified at the same time. Modify one bit at a time, and check that the new value can be read from the bit, and then modify another bit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSCKIZ[1:0] | | RSCKIZ[1:0] | | TEDG | REDG | — | — | TXDIZ[1:0] | | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSCKE | TFSE | RSCKE | RFSE | — | — | TXE | RXE | — | — | — | — | — | — | TXRST | RXRST |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R/W | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31, 30 | TSCKIZ[1:0] | 00 | R/W | Transmit Clock Input/output Polarity Select in SPI Mode When Transmission is Disabled<br>When SPI mode is not used, these bits must always be set to B'00.<br>These bits must always be set B'10 or B'11 when SPI mode.<br>[Master mode]<br>00: Outputs MSIOF_SCK when transmission is disabled.<br>01: Setting prohibited<br>10: Outputs 0.<br>11: Outputs 1.<br>[Slave mode]<br>00: Inputs MSIOF_SCK when transmission is disabled.<br>01: Setting prohibited<br>10: Inputs 0 through MSIOF_SCK when transmission is disabled.<br>11: Inputs 1 through MSIOF_SCK when transmission is disabled. |
| 29, 28 | RSCKIZ[1:0] | 00 | R/W | Receive Clock Polarity Select in SPI Mode<br>Set the same value as the value specified by the TSCKIZ bits. Other setting is prohibited. |
| 27 | TEDG | 0 | R/W | Transmit Timing<br>0: Outputs transmit data at the rising edge of the clock.<br>1: Outputs transmit data at the falling edge of the clock. |
| 26 | REDG | 0 | R/W | Receive Timing<br>0: Samples receive data at the falling edge of the clock.<br>1: Samples receive data at the rising edge of the clock. |
| 25, 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23, 22 | TXDIZ[1:0] | 00 | R/W | Pin Output When Transmission is Disabled<br>These bits specify the MSIOF_TXD pin output state when transmission is disabled.<br>00: Outputs 0.<br>01: Outputs 1.<br>10: Setting prohibited<br>11: Setting prohibited |
| 21 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 15 | TSCKE | 0 | R/W | Transmit Serial Clock Output Enable<br>This bit is valid in master mode.<br>When this bit is set to 1, the MSIOF initializes the baud rate generator and initiates the operation. At the same time, the MSIOF outputs the clock generated by the baud rate generator to the MSIOF_SCK pin. After the clock is output, 1 can be read.<br>When transmitting data, set this bit to 1 before setting the TFSE and TXE bits. After data transmission, clear the TFSE and TXE bits and then clear this bit to 0.<br>[Write data]<br>0: Disables the MSIOF_SCK output.<br>(Outputs the value specified with TSCKIZ.)<br>1: Enables the MSIOF_SCK output.<br>[Read data]<br>0: Does not output MSIOF_SCK.<br>(Outputs the value specified with TSCKIZ.)<br>1: Outputs MSIOF_SCK. |
| 14 | TFSE | 0 | R/W | Transmit Frame Synchronization Signal Output Enable<br>This bit is valid in master mode.<br>When this bit is set to 1, the MSIOF initializes the frame counter and initiates the operation. After the transmit frame synchronization signal is output, 1 can be read. When 0 is written to this bit, 0 is set after transmission of the frame.<br>[Write data]<br>0: Disables the MSIOF_SYNC output.<br>(Outputs the value specified with SYNCAC.)<br>1: Enables the MSIOF_SYNC output.<br>[Read data]<br>0: Does not output MSIOF_SYNC.<br>(Outputs the value specified with SYNCAC.)<br>1: Outputs MSIOF_SYNC. |
| 13 | RSCKE | 0 | R/W | Receive Serial Clock Output Enable<br>The write value should always be 0. |
| 12 | RFSE | 0 | R/W | Receive Frame Synchronization Signal Output Enable<br>The write value should always be 0. |
| 11, 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 9 | TXE | 0 | R/W | Transmit Enable<br>When this bit is set to 1, the MSIOF starts data transmission from the beginning of the next frame (at the rising edge of the frame synchronization signal). After the valid data is output, 1 can be read.<br>When value 1 set to this bit becomes valid, the MSIOF issues a transmit data transfer request according to the setting of the TFWM bits in SIFCTR. When transmit data is stored in the transmit FIFO, transmission of data from the MSIOF_TXD pin begins. When 0 is written to this bit, 0 is set after transmission of the frame.<br>This bit is initialized upon a transmit reset.<br>[Write data]<br>0: Disables the MSIOF_TXD output.<br>(Outputs the value specified with TXDIZ.)<br>1: Enables the MSIOF_TXD output.<br>[Read data]<br>0: Does not output MSIOF_TXD.<br>(Outputs the value specified with TXDIZ.)<br>1: Outputs MSIOF_TXD. |
| 8 | RXE | 0 | R/W | Receive Enable<br>When this bit is set to 1, the MSIOF starts data reception from the beginning of the next frame (at the rising edge of the frame synchronization signal).<br>When value 1 set to this bit becomes valid, the MSIOF starts receiving data through the MSIOF_RXD pin. When receive data is stored in the receive FIFO, the MSIOF issues a receive data transfer request according to the setting of the RFWM bits in SIFCTR.<br>This bit is initialized upon a receive reset.<br>[Write data]<br>0: Disables data reception through MSIOF_RXD.<br>1: Enables data reception through MSIOF_RXD.<br>[Read data]<br>0: Data is not received through MSIOF_RXD.<br>1: Data can be received through MSIOF_RXD. |
| 7 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 1 | TXRST | 0 | R/W | Transmit Reset<br>When value 1 set to this bit becomes valid, the MSIOF immediately sets transmit data through the MSIOF_TXD pin to 0, and initializes the transmit data registers and transmit-related status. When value 1 set to this bit becomes valid when PCON = 1, the RXRST bit is also becomes valid. The following registers and bits are initialized.<br>SITFDR (Transmit FIFO write pointer)<br>TDREQ bit in SISTR<br>TXE bit<br>This bit is read as 1 until the reset operation is completed for about 20 cycles of module clock. No data should be written to SICTR or the transmit FIFO during this time period.<br>[Write data]<br>0: Does not reset transmit operation.<br>1: Resets transmit operation.<br>[Read data]<br>0: Transmit operation reset is completed.<br>1: Transmit operation is being reset. |
| 0 | RXRST | 0 | R/W | Receive Reset<br>When value 1 set to this bit becomes valid, the MSIOF immediately disables reception through the MSIOF_RXD pin, and initializes the receive data registers and receive-related status. The following registers and bits are initialized.<br>SIRFDR (Receive FIFO read pointer)<br>RDREQ bit in SISTR<br>RXE bit<br>This bit is read as 1 until the reset operation is completed for about 20 cycles of module clock. No data should be written to SICTR or read the receive FIFO during this time period.<br>[Write data]<br>0: Does not reset receive operation.<br>1: Resets receive operation.<br>[Read data]<br>0: Receive operation reset is completed.<br>1: Receive operation is being reset. |

### 109.2.8 MSIOF Transmit Clock Select Register (SITSCR)

SITSCR is a 16-bit readable/writable register that specifies the conditions for generating transmit serial clock in master mode. SITSCR can be specified when the TRMD bit in SITMDR1 is set to B'1.
The settings of this register should be not changed when SICTR.TSCKE is B’1. Otherwise, the operation cannot be guaranteed.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSSEL[1:0] | | MSIMM | BRPS[4:0] | | | | | — | — | — | — | — | BRDV[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 15, 14 | MSSEL[1:0] | 00 | R/W | Master Clock Source Select<br>The master clock is the clock input to the baud rate generator.<br>00: Selects module clock as the source of master clock.<br>Others: Setting prohibited |
| 13 | MSIMM | 0 | R/W | Master Clock Direct Select<br>0: Selects the clock output from the baud rate generator as the serial clock.<br>1: Setting prohibited |
| 12 to 8 | BRPS[4:0] | 00000 | R/W | Prescaler Setting<br>These bits specify the master clock (MSOφ) division ratio in the count value of the prescaler in the baud rate generator.<br>The specifiable value is from B’00000 (× 1/1) to B’11111 (× 1/32). |
| 7 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | BRDV[2:0] | 000 | R/W | Baud Rate Generator's Division Ratio<br>These bits specify the frequency division ratio for the output stage of the baud rate generator.<br>The final frequency division ratio of the baud rate generator is determined as BRPS × BRDV (1/1024 max.).<br>000: Prescaler output × 1/2<br>001: Prescaler output × 1/4<br>010: Prescaler output × 1/8<br>011: Prescaler output × 1/16<br>100: Prescaler output × 1/32<br>101: Setting prohibited<br>110: Setting prohibited<br>111: Prescaler output × 1/1<br>Note:<br>- B'111 is valid only when the BRPS [4:0] bits are set to B'00001.<br>- BRPS × BRDV = 1/1 is invalid（BRPS × BRDV = 1/2 min). |

### 109.2.9 MSIOF Transmit FIFO Data Register (SITFDR)

SITFDR is a 32/16/8-bit write-only register that specifies the transmit FIFO data of the MSIOF.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SITFD1[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SITFD2[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 to 16 | SITFD1[15:0] | H'0000 | —/WB | These bits specify the upper 16 bits of the FIFO data to be output through MSIOF_TXD. |
| 15 to 0 | SITFD2[15:0] | H'0000 | —/WB | These bits specify the lower 16 bits of the FIFO data to be output through MSIOF_TXD. |

### 109.2.10 MSIOF Receive FIFO Data Register (SIRFDR)

SIRFDR is a 32/16/8-bit read-only register that stores the receive FIFO data of the MSIOF.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIRFD1[15:0] | | | | | | | | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIRFD2[15:0] | | | | | | | | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 to 16 | SIRFD1[15:0] | Undefined | R | Store the upper 16 bits of the FIFO data received through MSIOF_RXD. |
| 15 to 0 | SIRFD2[15:0] | Undefined | R | Store the lower 16 bits of the FIFO data received through MSIOF_RXD. |

### 109.2.11 MSIOF Status Register (SISTR)

Each bit in SISTR becomes an MSIOF interrupt source when the corresponding bit in SIIER is set to 1.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TFEMP | TDREQ | — | — | — | — | TEOF | — | TFSERR | TFOVF | TFUDF | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/WC | R | R | R | R | R | R/WC | R | R/WC | R/WC | R/WC | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | RFFUL | RDREQ | — | — | — | — | REOF | — | RFSERR | RFUDF | RFOVF | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/WC | R | R | R | R | R | R/WC | R | R/WC | R/WC | R/WC | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 29 | TFEMP | 0 | R/WC1 | Transmit FIFO Empty<br>This bit is set to 1, when the TXE bit in SICTR is 1, and transmit FIFO is empty.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: Transmit FIFO is not empty.<br>1: Transmit FIFO is empty. |
| 28 | TDREQ | 0 | R | Transmit Data Transfer Request<br>This bit is set, when the empty space in the transmit FIFO exceeds the size specified by the TFWM bits in SIFCTR.<br>When transferring transmit data through the DMAC, this bit is always cleared after one DMAC access. After DMAC access, when the conditions for setting this bit are satisfied, the MSIOF again sets this bit to 1.<br>This bit is valid when the TXE bit in SICTR is 1.<br>This bit indicates a state; if the size of empty space in the transmit FIFO becomes less than the size specified by the TFWM bits in SIFCTR, the MSIOF clears this bit.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued. However, when the TDMAE bit is 1, only a DMAC transfer request is issued.<br>0: The size of empty space in the transmit FIFO has not exceeded the size specified by the TFWM bits in SIFCTR.<br>1: The size of empty space in the transmit FIFO has exceeded the size specified by the TFWM bits in SIFCTR. |
| 27 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 | TEOF | 0 | R/WC1 | Frame Transmission End<br>This bit is set, when one-frame data transmission is completed.<br>This bit is valid when the TXE bit in SICTR is 1.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: One-frame transmission end is not detected.<br>1: One-frame transmission end is detected. |
| 22 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 21 | TFSERR | 0 | R/WC1 | Transmit Frame Synchronization Error<br>This bit is set, when the next transmit frame synchronization timing arrives before the previous data transmission has been completed.<br>If a transmit frame synchronization error occurs, the MSIOF performs transmission only for the slots that can be transferred.<br>This bit is valid when the TXE bit in SICTR is 1.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: No transmit frame synchronization error has occurred.<br>1: A transmit frame synchronization error has occurred. |
| 20 | TFOVF | 0 | R/WC1 | Transmit FIFO Overflow<br>A transmit FIFO overflow means that there has been an attempt to write to SITFDR when the transmit FIFO is full.<br>If a transmit FIFO overflow occurs, the MSIOF ignores the write operation causing the overflow.<br>This bit is valid when the TXE bit in SICTR is 1.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: No transmit FIFO overflow has occurred.<br>1: A transmit FIFO overflow has occurred. |
| 19 | TFUDF | 0 | R/WC1 | Transmit FIFO Underflow<br>A transmit FIFO underflow means that loading for transmission has occurred when the transmit FIFO is empty. Output data is unknown when under flow is occurred.<br>This bit is valid when the TXE bit in SICTR is 1.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: No transmit FIFO underflow has occurred.<br>1: A transmit FIFO underflow has occurred.<br>Note: If transmission size is more than 256-words per frame, do not use this bit. See 109.4.5 for details. |
| 18 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 | RFFUL | 0 | R/WC1 | Receive FIFO Full<br>This bit is valid when the RXE bit in SICTR is 1.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: Receive FIFO is not full.<br>1: Receive FIFO is full. |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 12 | RDREQ | 0 | R | Receive Data Transfer Request<br>This bit is set, when the valid data space in the receive FIFO exceeds the size specified by the RFWM bits in SIFCTR.<br>When transferring receive data through the DMAC, this bit is always cleared by one DMAC access. After DMAC access, when the conditions for setting this bit are satisfied, the MSIOF again sets this bit to 1.<br>This bit is valid when the RXE bit in SICTR is 1.<br>This bit indicates a state; if the size of valid data space in the receive FIFO becomes less than the size specified by the RFWM bits in SIFCTR, the MSIOF clears this bit.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued. However, when the RDMAE bit is 1, only a DMAC transfer request is issued.<br>0: The size of valid data space in the receive FIFO has not exceeded the size specified by the RFWM bits in SIFCTR.<br>1: The size of valid data space in the receive FIFO has exceeded the size specified by the RFWM bits in SIFCTR. |
| 11 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | REOF | 0 | R/WC1 | Frame Reception End<br>The MSIOF issues the frame reception end flag upon completion of one-frame data reception.<br>This bit is valid when the RXE bit in SICTR is 1.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: One-frame reception end is not detected.<br>1: One-frame reception end is detected. |
| 6 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5 | RFSERR | 0 | R/WC1 | Receive Frame Synchronization Error<br>A receive frame synchronization error occurs when the next receive frame synchronization timing arrives before the previous data reception has been completed.<br>If a receive frame synchronization error occurs, the MSIOF performs reception only for the slots that can be transferred.<br>This bit is valid when the RXE bit in SICTR is 1.<br>Writing 1 to this bit will clear this bit.<br>Writing 0 to this bit will be ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: No receive frame synchronization error has occurred<br>1: A receive frame synchronization error has occurred |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 4 | RFUDF | 0 | R/WC1 | Receive FIFO Underflow<br>A receive FIFO underflow means that reading of SIRFDR has occurred when the receive FIFO is empty.<br>If a receive FIFO underflow occurs, the value read from SIRFDR is not guaranteed.<br>This bit is valid when the RXE bit in SICTR is 1.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: No receive FIFO underflow has occurred.<br>1: A receive FIFO underflow has occurred. |
| 3 | RFOVF | 0 | R/WC1 | Receive FIFO Overflow<br>A receive FIFO overflow means that writing has been caused by receiving operation when the receive FIFO is full.<br>If a receive FIFO overflow occurs, the receive data causing the overflow is lost.<br>This bit is valid when the RXE bit in SICTR is 1.<br>Writing 1 to this bit clears this bit.<br>Writing 0 to this bit is ignored.<br>If the interrupt request from this bit is enabled, an MSIOF interrupt is issued.<br>0: No receive FIFO overflow has occurred.<br>1: A receive FIFO overflow has occurred. |
| 2 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 109.2.12 MSIOF Interrupt Enable Register (SIIER)

SIIER is a 32-bit readable/writable register that enables the issuance of MSIOF interrupts. When each bit in this register is set to 1 and the corresponding bit in SISTR is set to 1, the MSIOF issues an interrupt.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TDMAE | — | TFEMPE | TDREQE | — | — | — | — | TEOFE | — | TFSERRE | TFOVFE | TFUDFE | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R/W | R/W | R | R | R | R | R/W | R | R/W | R/W | R/W | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDMAE | — | RFFULE | RDREQE | — | — | — | — | REOFE | — | RFSERRE | RFUDFE | RFOVFE | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R/W | R/W | R | R | R | R | R/W | R | R/W | R/W | R/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 | TDMAE | 0 | R/W | Transmit Data DMA Transfer Request Enable<br>Specifies whether to send an interrupt as an interrupt request to the CPU or a transfer request to the DMAC. The TDREQE bit can be set as the request source.<br>0: Sends an interrupt request to the CPU.<br>1: Sends a DMA transfer request to the DMAC. |
| 30 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 29 | TFEMPE | 0 | R/W | Transmit FIFO Empty Enable<br>0: Disables interrupts due to transmit FIFO empty.<br>1: Enables interrupts due to transmit FIFO empty. |
| 28 | TDREQE | 0 | R/W | Transmit Data Transfer Request Enable<br>0: Disables interrupts due to transmit data transfer requests.<br>1: Enables interrupts due to transmit data transfer requests. |
| 27 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 23 | TEOFE | 0 | R/W | Frame Transmission End Enable<br>0: Disables a frame transmission end interrupt.<br>1: Enables a frame transmission end interrupt. |
| 22 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 21 | TFSERRE | 0 | R/W | Transmit Frame Synchronization Error Enable<br>0: Disables interrupts due to transmit frame synchronization errors.<br>1: Enables interrupts due to transmit frame synchronization errors. |
| 20 | TFOVFE | 0 | R/W | Transmit FIFO Overflow Enable<br>0: Disables interrupts due to transmit FIFO overflow.<br>1: Enables interrupts due to transmit FIFO overflow. |
| 19 | TFUDFE | 0 | R/W | Transmit FIFO Underflow Enable<br>0: Disables interrupts due to transmit FIFO underflow.<br>1: Enables interrupts due to transmit FIFO underflow. |
| 18 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | RDMAE | 0 | R/W | Receive Data DMA Transfer Request Enable<br>Specifies whether to send an interrupt as an interrupt request to the CPU or a transfer request to the DMAC. The RDREQE bit can be set as the request source.<br>0: Sends an interrupt request to the CPU.<br>1: Sends a DMA transfer request to the DMAC. |
| 14 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 13 | RFFULE | 0 | R/W | Receive FIFO Full Enable<br>0: Disables interrupts due to receive FIFO full.<br>1: Enables interrupts due to receive FIFO full. |
| 12 | RDREQE | 0 | R/W | Receive Data Transfer Request Enable<br>0: Disables interrupts due to receive data transfer requests.<br>1: Enables interrupts due to receive data transfer requests. |
| 11 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | REOFE | 0 | R/W | Frame Reception End Enable<br>0: Disables a frame reception end interrupt.<br>1: Enables a frame reception end interrupt. |
| 6 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5 | RFSERRE | 0 | R/W | Receive Frame Synchronization Error Enable<br>0: Disables interrupts due to receive frame synchronization errors.<br>1: Enables interrupts due to receive frame synchronization errors. |
| 4 | RFUDFE | 0 | R/W | Receive FIFO Underflow Enable<br>0: Disables interrupts due to receive FIFO underflow.<br>1: Enables interrupts due to receive FIFO underflow. |
| 3 | RFOVFE | 0 | R/W | Receive FIFO Overflow Enable<br>0: Disables interrupts due to receive FIFO overflow.<br>1: Enables interrupts due to receive FIFO overflow. |
| 2 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 109.2.13 MSIOF FIFO Control Register (SIFCTR)

SIFCTR is a 32-bit readable/writable register that indicates the area available for the transmit/receive FIFO transfer.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TFWM[2:0] | | | TFUA[8:0] | | | | | | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RFWM[2:0] | | | RFUA[8:0] | | | | | | | | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 31 to 29 | TFWM[2:0] | 000 | R/W | Transmit FIFO Watermark<br>A transfer request of the transmit FIFO is issued by the TDREQE bit in SIIER.<br>The transmit FIFO always operates as a 256-stage FIFO regardless of the setting of these bits.<br>000: Issues a transfer request when 64 stages of the transmit FIFO are empty.<br>001: Issues a transfer request when 32 more stages of the transmit FIFO are empty.<br>010: Issues a transfer request when 24 or more stages of the transmit FIFO are empty.<br>011: Issues a transfer request when 16 or more stages of the transmit FIFO are empty.<br>100: Issues a transfer request when 12 or more stages of the transmit FIFO are empty.<br>101: Issues a transfer request when 8 or more stages of the transmit FIFO are empty.<br>110: Issues a transfer request when 4 or more stages of the transmit FIFO are empty.<br>111: Issues a transfer request when 1 or more stages of transmit FIFO are empty. |
| 28 to 20 | TFUA[8:0] | H'100 | R | Transmit FIFO Usable Area<br>Indicate the number of words that can be transferred by the CPU or DMAC as B'0 0000 0000 (full) to B'1 0000 0000 (empty). |
| 19 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 13 | RFWM[2:0] | 000 | R/W | Receive FIFO Watermark<br>A transfer request of the receive FIFO is issued by the RDREQE bit in SIIER.<br>The receive FIFO always operates as a 256-stage FIFO regardless of the setting of these bits.<br>000: Issues a transfer request when 1 stage or more of the receive FIFO are valid.<br>001: Issues a transfer request when 4 or more stages of the receive FIFO are valid.<br>010: Issues a transfer request when 8 or more stages of the receive FIFO are valid.<br>011: Issues a transfer request when 16 or more stages of the receive FIFO are valid.<br>100: Issues a transfer request when 32 or more stages of the receive FIFO are valid.<br>101: Issues a transfer request when 64 or more stages of the receive FIFO are valid.<br>110: Issues a transfer request when 128 or more stages of the receive FIFO are valid.<br>111: Issues a transfer request when 256 stages of the receive FIFO are valid. |

| Bit | Bit Name | Initial Value | R/W | Descriptions |
|---|---|---|---|---|
| 12 to 4 | RFUA[8:0] | All 0 | R | Receive FIFO Usable Area<br>Indicate the number of words that can be transferred by the CPU or DMAC as B'0 0000 0000 (empty) to B'1 0000 0000 (full). |
| 3 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

## 109.3 Operation

### 109.3.1 Operating Mode

#### (1) Common transmit/receive mode

Common clock and frame synchronization signals are used for transmission and reception.

Master mode: MSIOF_SCK and MSIOF_SYNC are output.
Slave mode: MSIOF_SCK and MSIOF_SYNC are input.

### 109.3.2 Serial Clocks

#### (1) Clock output in master mode

In master mode, the baud rate generator is used to generate the serial clock. The division ratio is from 1/2 to 1/1024.

#### (2) Clock input in slave mode

In slave mode, the clock input for transmission and reception is used as the serial clock.

#### (3) Multiple channel function

This module provides the multiple channel function as shown in Figure 109.2.

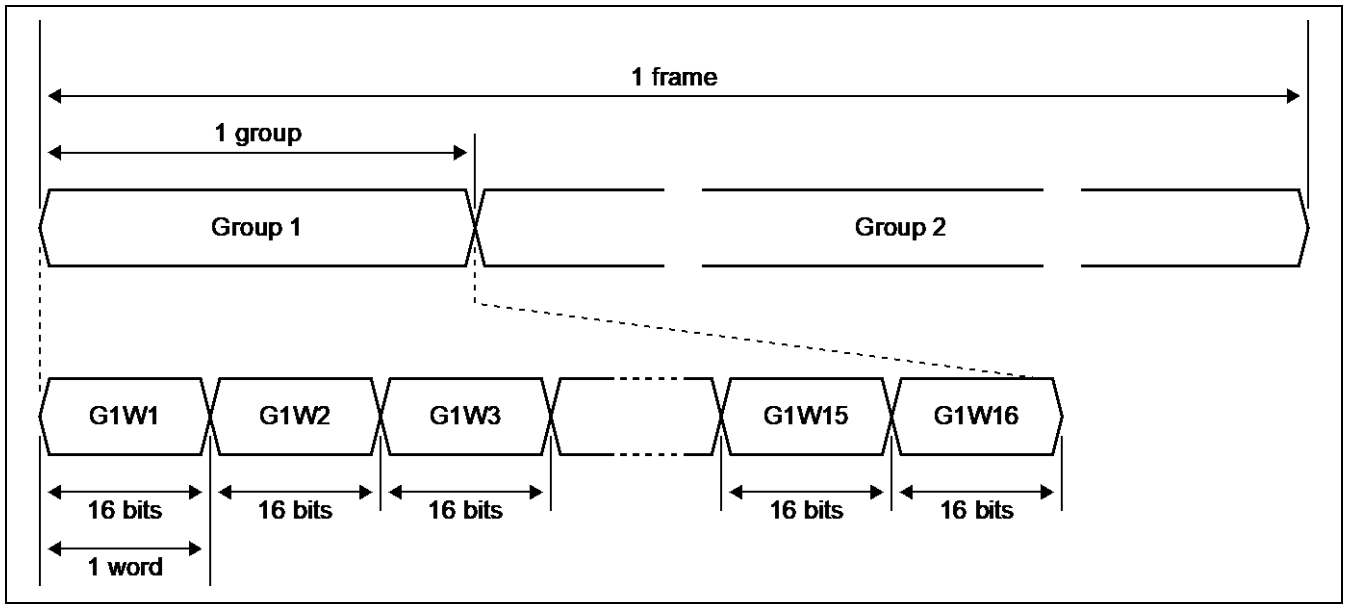



**Figure 109.2 Multiple Channel Structure**

The following conditions can be specified for the multiple channel function.

- Up to 2 groups in one frame
- Up to 256 words in one group
- 8, 16, 24, or 32 bits in one word

### 109.3.3 Serial Timing

#### (1) MSIOF_SYNC

The MSIOF_SYNC is a frame synchronization signal. The following three modes are available.

- Frame start synchronization pulse: 1-bit-width pulse indicating the start of a frame
- Level/SPI: Level signal asserted during frame transmission
- L/R: 1/2-frame-width pulse indicating the first-half groups in a high level and the last-half groups in a low level

Figure 109.3 to Figure 109.5 show the synchronization timing in these modes using the MSIOF_SYNC signal.

#### (a) Frame start synchronization pulse

The rising edge of the synchronization pulse indicates the start of a frame.

The delay between the rising edge of the synchronization signal and the start of data transmission or reception can be specified with the DTDL bits. The width of the synchronization pulse can be extended through the SYNCDL bit setting.

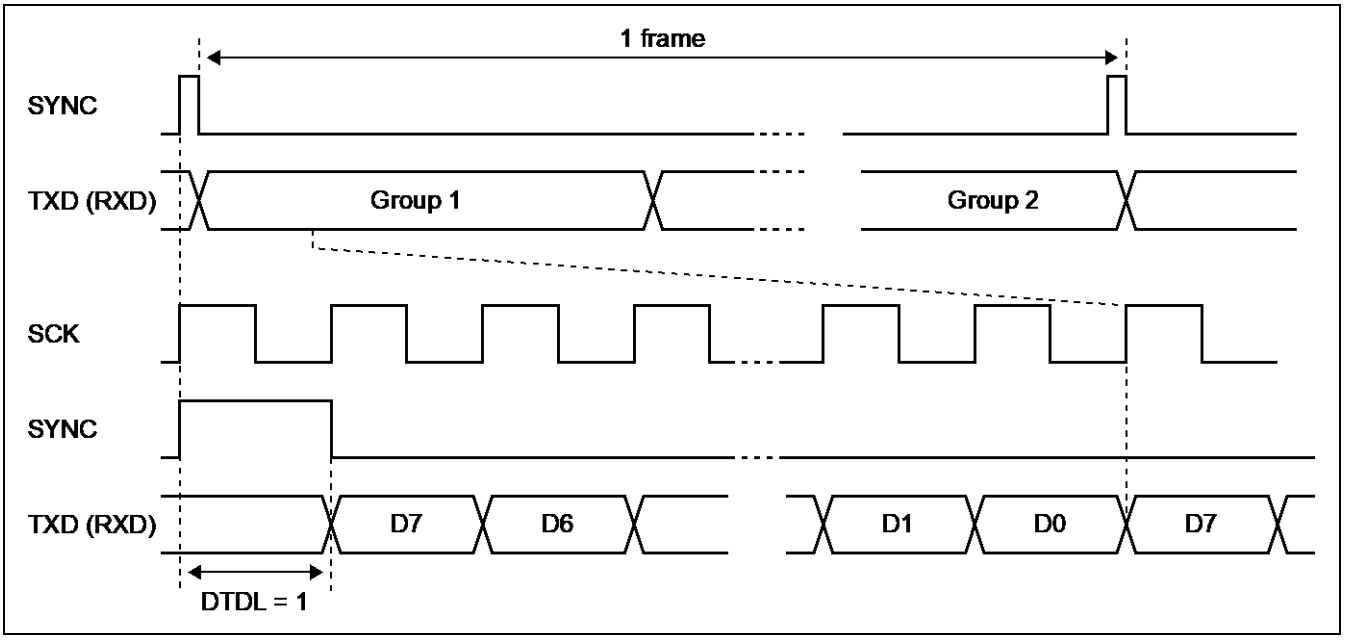



**Figure 109.3 Synchronization Timing in Start Frame Synchronization Pulse Mode**

**(b) Level synchronization**

The synchronization signal is driven high for the frame length.

The delay between the rising edge of the synchronization signal and the start of transmission or reception can be specified with the DTDL bits and the delay between the end of transmission or reception and the falling edge of the synchronization signal can be specified with the SYNCDL bits (DTDL = 0 to 2 and SYNCDL = 0 to 3). These settings are valid only in master mode. This mode is available for the SPI master or slave operation.

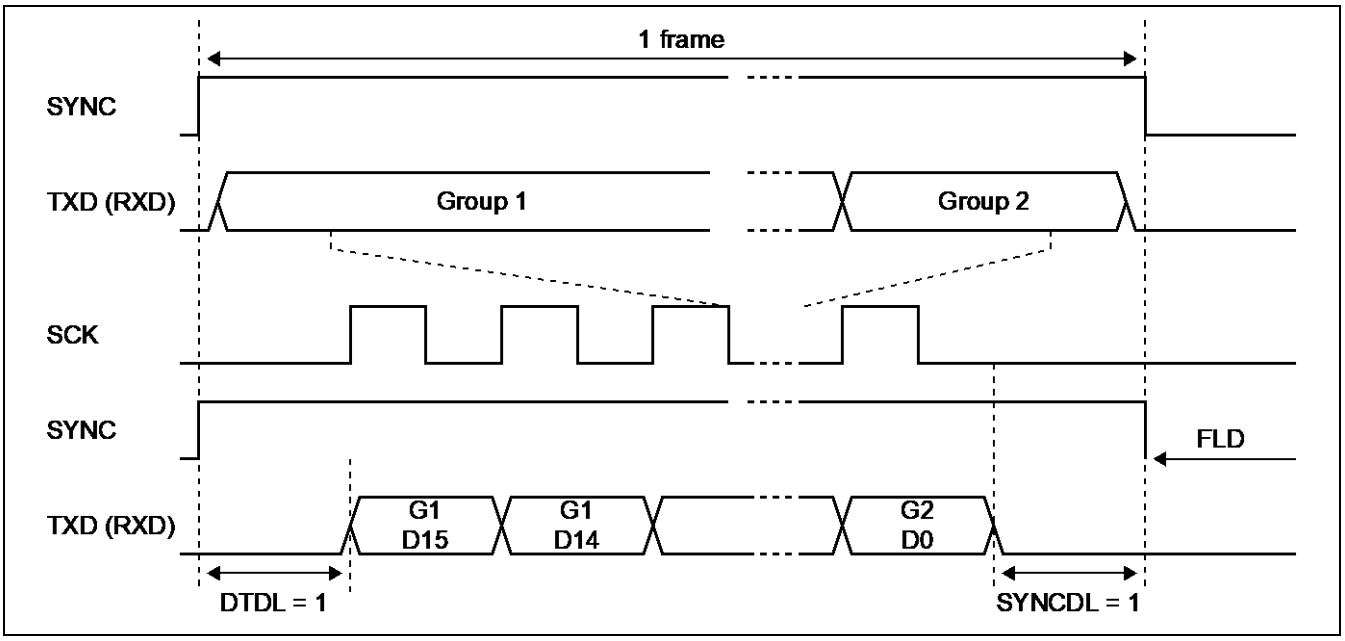



**Figure 109.4 Synchronization Timing in Level Synchronization Mode**

**(c) L/R synchronization**

A high level in the synchronization signal indicates the first-half groups and a low level indicates the last-half groups.

The delay between the rising or falling edge of the synchronization signal and the start of transmission or reception can be specified with the DTDL bits.

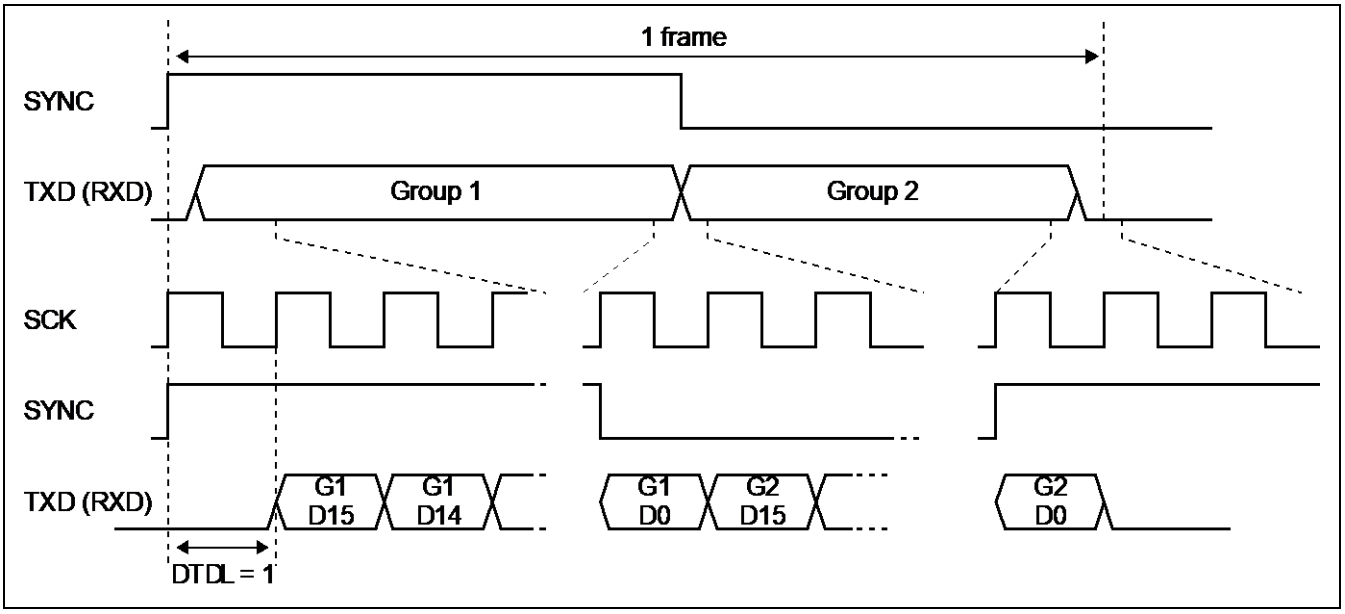



**Figure 109.5 Synchronization Timing in L/R Synchronization Mode**

**(2) Transmit/receive timing**

The timing of MSIOF_TXD transmission relative to MSIOF_SCK and of MSIOF_RXD can be specified as sampling on either edge, as listed below. The respective settings are made in TEDG and REDG bits of SICTR.

- Falling-edge sampling
- Rising-edge sampling

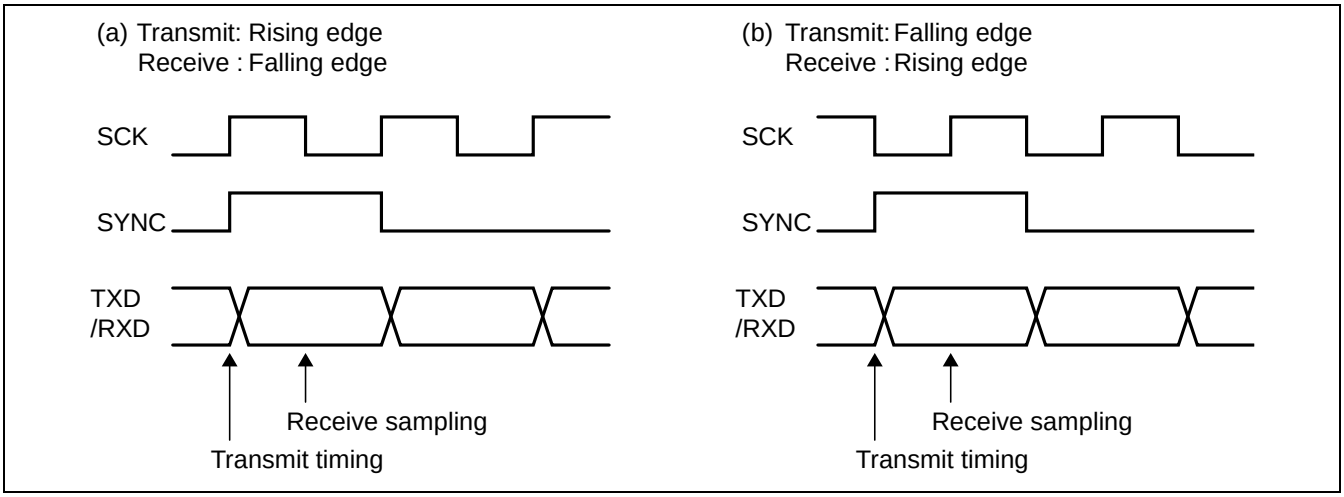



**Figure 109.6 Transmit/Receive Timing.**

### 109.3.4 Transfer Data Allocation to Registers

**(1) Transmit/receive data**

Transmit/receive data (FIFO data) should be written to or read from the following registers.

- Transmit FIFO data writing: SITFDR (32/16/8-bit access)
- Receive FIFO data reading: SIRFDR (32/16/8-bit access)

Figure 109.7 shows the bit alignment of these registers.

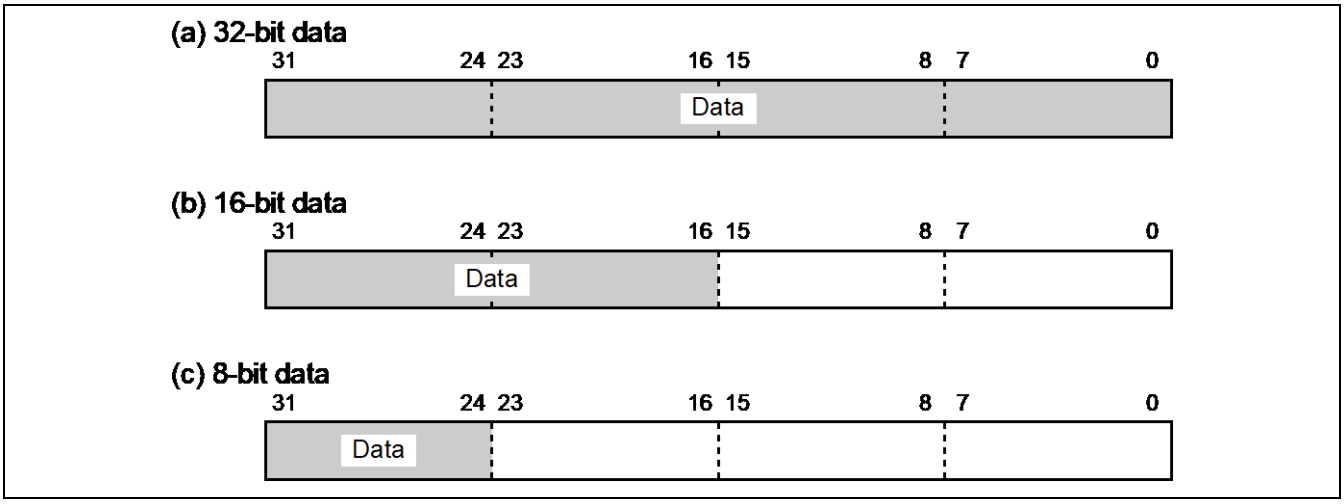



**Figure 109.7 Transmit/Receive Data Bit Alignment**

### 109.3.5 FIFO

#### (1) Overview

The transmission and receive FIFOs of the MSIOF have the following features.

- 32 bits × 256 stages for transmission and 32 bits × 256 stages for reception
- The FIFO pointer is updated in one read or write cycle regardless of the access size of the CPU or DMAC. (One access cannot be divided into multiple accesses.)

#### (2) Transfer request

A request for FIFO data transfer can be issued to the CPU or DMAC as the following interrupt sources.

- FIFO transmit request: TDREQ (interrupt source for transmission)
- FIFO receive request: RDREQ (interrupt source for reception)

The conditions for requesting FIFO data transfer can be specified separately for transmission and reception. The conditions for the transmit FIFO and receive FIFO are specified in the TFWM[2:0] bits and RFWM[2:0] bits in SIFCTR, respectively.

**Table 109.7 Condition for Issuing Transmit Request**

| TFWM[2:0] | Number of Requested Stages | Condition for Transmit Request | Areas Used |
|---|---|---|---|
| 000 | 192 | 64 stages of empty area | Smallest |
| 001 | 224 | 32 or more stages of empty area | ↑ |
| 010 | 232 | 24 or more stages of empty area | ↑ |
| 011 | 240 | 16 or more stages of empty area | ↑ |
| 100 | 244 | 12 or more stages of empty area | ↑ |
| 101 | 248 | 8 or more stages of empty area | ↑ |
| 110 | 252 | 4 or more stages of empty area | ↑ |
| 111 | 256 | 1 or more stages of empty area | Largest |

**Table 109.8 Condition for Issuing Receive Request**

| RFWM[2:0] | Number of Requested Stages | Condition for Receive Request | Areas Used |
|---|---|---|---|
| 000 | 1 | 1 or more stages of valid data | Smallest |
| 001 | 4 | 4 or more stages of valid data | ↑ |
| 010 | 8 | 8 or more stages of valid data | ↑ |
| 011 | 16 | 16 or more stages of valid data | ↑ |
| 100 | 32 | 32 or more stages of valid data | ↑ |
| 101 | 64 | 64 or more stages of valid data | ↑ |
| 110 | 128 | 128 or more stages of valid data | ↑ |
| 111 | 256 | 256 stages of valid data | Largest |

The maximum FIFO stages are always available even if the data area or empty area exceeds the size specified for the transfer condition. Accordingly, an overflow error or an underflow error occurs if the data area or empty area exceeds the maximum number of FIFO stages.

The FIFO transfer request is canceled when the specified condition is not satisfied even if the FIFO does not become empty or full.

**(3) Number of FIFOs**

The number of transmit and receive FIFO stages used are indicated by the following registers.

- Transmit FIFO: The number of empty FIFO stages is indicated by the TFUA [8:0] bits in SIFCTR.
- Receive FIFO: The number of valid data stages is indicated by the RFUA [8:0] bits in SIFCTR.

These registers show the number of data stages that can be transferred by the CPU. DMA transfer should be set the following setting.

SIFCTR.TFWM[2:0] = B’111
SIFCTR.RFWM[2:0] = B’000

### 109.3.6 Transmit and Receive Procedures

#### (1) Transmission in master mode

Figure 109.8 and Figure 109.9 shows an example of settings and operation for transmission in master mode.

| No. | Operating flow | Description | Operation |
|---|---|---|---|
| 1 | SITMDR1/2/3<br>SITSCR<br>SIFCTR<br>SIIER setting | Operating mode<br>Clock setting<br>FIFO water mark<br>Setting interrupt/DMA enable | |
| 2 | SICTR<br>Set SICTR.TSCKE to 1 | Transmission rate setting<br>Baud rate generator starting | Output serial clock |
| 3 | TSCKE = 1 ? NO / YES | Read SICTR.TSCKE = 1 | |
| 4 | SITFDR | Write transmission data to FIFO | |
| 5 | Set SICTR.TXE to 1 | Transmission enable | |
| 6 | TXE = 1 ? NO / YES | Read SICTR.TXE = 1 | |
| 7 | Set SICTR.TFSE to 1 | Starting MSIOF_SYNC output | |
| 8 | TFSE = 1 ? NO / YES | Read SICTR.TFSE = 1 | |
| 9 | Starting transmission | | Data transmission |
| 10 | TDREQ = 1 ? NO / YES | | Generating interrupt/DMA request |
| 11 | Write transmission data to FIFO | | |
| 12 | Transmission end ? NO / YES | | |
| 13 | Set SICTR.TFSE to 0 | Frame sync disable | |
| 14 | TFSE = 0 ? NO / YES | Read SICTR.TFSE = 0 | |
| 15 | Set SICTR.TXE to 0 | Transmission disable | |
| 16 | TXE = 0 ? NO / YES<br>Finish | Read SICTR.TXE = 0 | |

**Figure 109.8 Example of Transmit Procedure in Master Mode (without DMAC)**

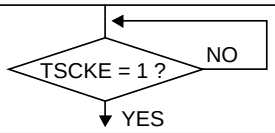
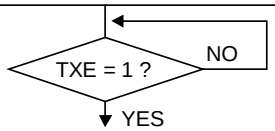
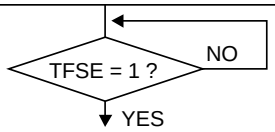
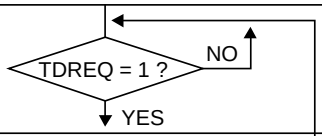
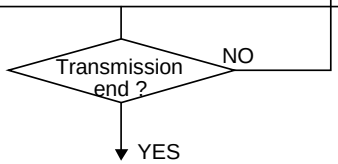
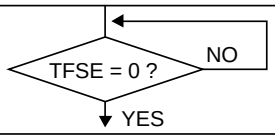
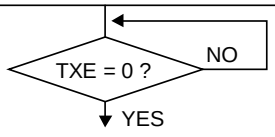
| No. | Operating flow | Description | Operation |
|---|---|---|---|
| 1 | • Setting interupt controller<br>• Setting source address of DMAC channel<br>• Setting Destination address of DMAC channel is SITFDR<br>• Setting number of Transfer in DMAC transfer count register<br>• Setting transfer as fixed destination address mode.<br>• Setting Resouse Select register of DMAC channel, please refer to Chapter describe DMAC module to get the value.<br>• DMAC start. | • DMAC setting, Refer To Chapter Describe DMAC module for More detail<br>• Setting to interrupt controller for MSIOF interrupt event, refer to chapter describe interrupt controller for detail. | |
| 2 | SITMDR1/2/3<br>SITSCR<br>SIFCTR<br><br>SIIER setting | • operating mode<br>• clock setting<br>• fifo water mark<br>• interrupt: Refer to Section 109.3.7 or SIIER register description (109.2.12) for MSIOF's Interrupt Factor.<br>• dma enable (SIIER.TDMAE) | |
| 3 | SICTR<br>Set SICTR.TSCKE to 1 | Transmission rate setting<br>Baud rate generator starting | Output serial clock |
| 4 |  | Read SICTR.TSCKE = 1 | |
| 5 | Set SICTR.TXE to 1 | Transmission enable | |
| 6 |  | Read SICTR.TXE = 1 | |
| 7 | Set SICTR.TFSE to 1 | Starting MSIOF_SYNC output | |
| 8 |  | Read SICTR.TFSE = 1 | |
| 9 | Starting transmission | | Data transmission |
| 10 |  | | Generating interrupt/DM A request |
| 11 | Write transmission data to FIFO | | |
| 12 |  | Wait bit TE in DMAC CHCR register set to 1, refer to chapter describe DMAC module for detail. Or use interrupt to avoid software polling, DMAC module support interrupt when the transfer end. | |
| 13 | Set SICTR.TFSE to 0 | Frame sync disable | |
| 14 |  | Read SICTR.TFSE = 0 | |
| 15 | Set SICTR.TXE to 0 | Transmission disable | |
| 16 |  <br>Finish | • Read SICTR.TXE = 0<br>• Disable DMAC channel, refer to chapter describe DMAC module for detail | |

**Figure 109.9 Example of Transmit Procedure in Master Mode (with DMAC)**

### (2) Reception in master mode (with SITMDR1.PCON = 1)

Figure 109.10 and Figure 109.11 show an example of settings and operation for reception in master mode with SITMDR1.PCON = 1.

| No. | Operating flow | Description | Operation |
|---|---|---|---|
| 1 | SIRMDR1/2/3<br>SITMDR1/2/3<br>SITSCR<br>SIFCTR<br>SIIER setting | Operating mode<br>Operating mode (for SPI dummy transmission)<br>Serial clock<br>FIFO water mark<br>Interrupt/DMA setting | |
| 2 | SICTR<br>Set SICTR.TSCKE to 1 | Baud rate setting<br>Baud rate generator starting | Output serial clock |
| 3 | TSCKE = 1 ?<br>NO<br>YES | Read SICTR.TSCKE = 1 | |
| 4 | SITFDR | Write dummy transmission data to SITFDR<br>The number of dummy data should equal<br>the number of reception data. | |
| 5 | Set SICTR.TXE to 1<br>Set SICTR.RXE to 1 | Transmission enable<br>Reception enable | |
| 6 | TXE = 1 ?<br>RXE = 1 ?<br>NO<br>YES | Read SICTR.TXE = 1<br>Read SICTR.RXE = 1 | |
| 7 | Set SICTR.TFSE to 1 | Frame sync enable | Output MSIOF_SYNC |
| 8 | TFSE = 1 ?<br>NO<br>YES | Read SICTR.TFSE = 1 | |
| 9 | Synchronizing with MSIOF_SYNC<br>(1) Transmission FIFO dummy data<br>(2) Reception data | | Transmission and reception |
| 10 | TDREQ=1?<br>RDREQ=1?<br>NO<br>YES | | Generating<br>interrupt/DMA request |
| 11 | Write dummy data to FIFO<br>Read fifo data from SIRFDR | Transmission dummy data<br>Reception data reading | |
| 12 | Transmission/<br>reception end?<br>NO<br>YES | | |
| 13 | Set SICTR.TFSE to 0 | Disable frame sync enable | |
| 14 | TFSE = 0 ?<br>NO<br>YES | Read SICTR.TFSE = 0 | |
| 15 | Set SICTR.TXE and SICTR.RXE to 0 | Disable transmission enable and reception enable | |
| 16 | TXE = 0 ?<br>RXE = 0 ?<br>NO<br>YES<br>Finish | Read SICTR.TXE = 0<br>Read SICTR.RXE = 0 | |

**Figure 109.10 Example of Receive Procedure in Master Mode (SITMDR1.PCON = 1) (without DMAC)**

| No. | Operating flow | Description | Operation |
|---|---|---|---|
| 1 | • Setting Interrupt controller<br>• In receive master mode, dummy data need to be written into SITFDR, so 2 DMAC channel must be used.<br>• Setting DMAC for transfer dummy data:<br>- Setting Source Address is memories address which locate dummy data.<br>- Setting Destination Address is SITFDR.<br>- Setting the number of transfer in DMAC transfer count register equal the number of received data.<br>- DMAC transfer as fixed destination address mode.<br>• Setting DMAC for received data:<br>- Setting Source address is SIRFDR register.<br>- Setting Destination address is memories address which locate the received data.<br>- Setting the number of transfer in DMAC transfer count register equal the number of received data.<br>- DMAC transfer as incremental destination address mode.<br>• DMAC start for 2 channel. | • Setting Interrupt Controller to enable MSIOF interrupt, refer to chapter describe Interrupt Controller for detail<br>• DMAC setting, refer to chapter describe DMAC module for detail. | |
| 2 | SIRMDR1/2/3<br>SITMDR1/2/3<br>SITSCR<br>SIFCTR<br><br>SIIER setting | • operating mode<br>• operating mode (for SPI dummy transmission)<br>• serial clock<br>• fifo water mark<br>• interrupt : Refer to section 109.3.7 or SIIER register description (109.2.12) for MSIOF's Interrupt Factor.<br>• dma enable (SIIER.RDMAE) | |
| 3 | SICTR<br>Set SICTR.TSCKE to 1 | baud rate setting<br>baud rate generater starting | output serial clock |
| 4 | TSCKE = 1? No / Yes | Read SICTR.TSCKE = 1 | |
| 5 | Set SICTR.TXE to 1<br>Set SICTR.RXE to 1 | Trasnmission enable<br>Reception enable | |
| 6 | TXE = 1? RXE = 1? No / Yes | Read SICTR.TXE = 1<br>Read SICTR.RXE = 1 | |
| 7 | Set SICTR.TFSE to 1 | Frame sync enable | Output MSIOF_SYNC |
| 8 | TFSE = 1? No / Yes | Read SICTR.TFSE = 1 | |
| 9 | Synchronizing with MSIOF_SYNC<br>(1) Transmission fifo dummy data<br>(2) Reception data | | transmission and reception |
| 10 | TDREQ = 1? RDREQ = 1? No / Yes | | Generating interrupt / DMA request |
| 11 | Write dummy data to fifo<br>Read fifo data from SIRFDR | Transmission dummy data<br>Reception data reading | |
| 12 | transmission/ reception end No / Yes | Wait bit TE in DMAC CHCR register set to 1, refer to chapter describe DMAC module for detail. Or use interrupt to avoid software polling, DMAC module support interrupt when the transfer end. | |
| 13 | Set SICTR.TFSE to 0 | Disable frame sync enable | |
| 14 | TFSE = 0? No / Yes | Read SICTR.TFSE = 0 | |
| 15 | Set SICTR.TXE and SICTR.RXE to 0 | Disable transmission enable and reception enable | |
| 16 | TXE = 0? RXE = 0? No / Yes<br>finish | Read SICTR.TXE = 0<br>Read SICTR.RXE = 0 | |

**Figure 109.11 Example of Receive Procedure in Master Mode (SITMDR1.PCON = 1) (with DMAC)**

### (3) Transmission in slave mode

Figure 109.12 shows an example of settings and operation for transmission in slave mode transmission size is less than or equal to 256-words per frame).

Figure 109.13 shows an example of settings and operation for transmission in slave mode (transmission size is more than 256-words per frame without DMAC).

Figure 109.14 shows an example of settings and operation for transmission in slave mode (transmission size is more than 256-words per frame with DMAC).

| No. | Operating flow | Description | Operation |
|---|---|---|---|
| 1 | Master device setting | Depend on master device | Signal polarity |
| 2 | SITMDR1/2/3<br>SIFCTR<br>SIIER setting | Operating mode<br>FIFO water mark<br>Interrupt/DMA enable (Don't use SIIER.TFUDFE) | |
| 3 | SITFDR | Transmission data setting | |
| 4 | Set SICTR.TXE to 1 | Setting transmission enable | |
| 5 | TXE = 1 ?<br>NO<br>YES | Read SICTR.TXE = 1 | |
| 6 | Data transmission synchronize with MSIOF_SYNC | | Data transmission |
| 7 | TDREQ = 1 ?<br>NO<br>YES | | Generating interrupt/DMA request |
| 8 | Write FIFO data to SITFDR | Data transmission | |
| 9 | Transmission end ?<br>NO<br>YES | | |
| 10 | Set SICTR.TXE to 0 | Transmission disable | |
| 11 | TXE = 0 ?<br>NO<br>YES<br>Finish | Read SICTR.TXE = 0 | |

**Figure 109.12 Example of Transmit Procedure in Slave Mode (transmission size is less than or equal to 256-words per frame)**

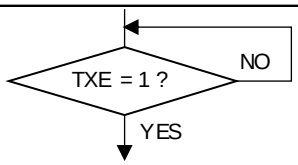
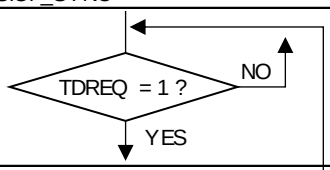
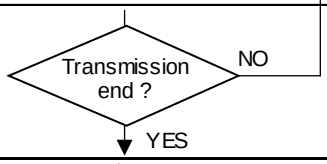
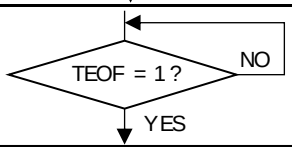
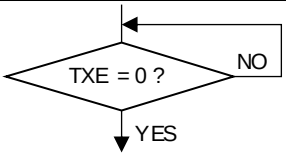
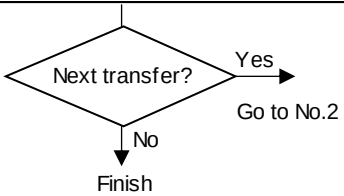
| No. | Operating flow | Description | Operation |
|---|---|---|---|
| 1 | Master device setting | Depend on master device | Signal polarity |
| 2 | SITMDR1/2/3<br>SIFCTR<br>SIIER setting | Operating mode<br>FIFO w ater mark<br>Interrupt/DMA enable (Don't use SIIER.TFUDFE) | Do not include dummy data w hen setting the transfer w ord count.(SITMDR2.WDLEN1, SITMDR3.WDLEN2) |
| 3 | PFC setting | Change MSIOF_TXD, MSIOF_RXD, MSIOF_SCK, MSIOF_SYNC to module side | |
| 4 | SITFDR | Transmission data setting | Need to transmit FIFO full |
| 5 | Set SICTR.TXE to 1 | Setting transmission enable | |
| 6 |  | Read SICTR.TXE = 1 | |
| 7 | Data transmission synchronize w ith MSIOF_SYNC | | Data transmission |
| 8 |  | | Generating interrupt/DMA request |
| 9 | Write FIFO data to SITFDR | Data transmission<br>After the last data is stored, store additional dummy data(1-w ord). | Do not need to transmit dummy data. |
| 10 |  | | |
| 11 |  | | |
| 12 | Set SICTR.TXE to 0 | Transmission disable | |
| 13 |  | Read SICTR.TXE = 0 | |
| 14 | PFC setting | Change MSIOF_SCK, MSIOF_SYNC to GPIO(In) side | |
| 15 | Assert SW reset for MSIOF | SRCRn.MSIOF (Refer to sec 9. Module Standby, Softw are Reset) | Clear dummy data<br>n = 6 |
| 16 | De-assert SW reset for MSIOF | SRSTCLRn.MSIOF (Refer to sec 9. Module Standby, Softw are Reset) | n = 6 |
| 17 |  | | |

**Figure 109.13 Example of Transmit Procedure in Slave Mode (transmission size is more than 256-words per frame without DMAC)**

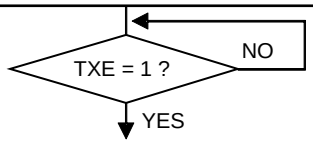
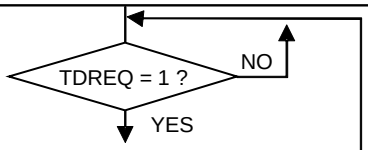
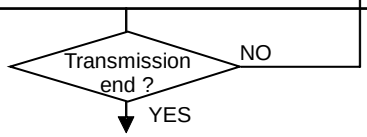
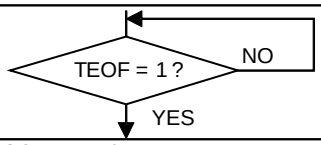
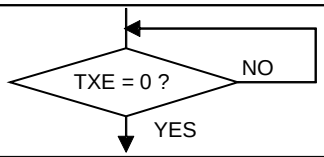
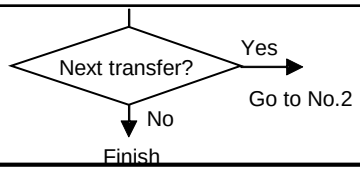
| No. | Operating flow | Description | Operation |
|---|---|---|---|
| 1 | Master device setting | Depend on master device | Signal polarity |
| 2 | • Setting interupt controller<br>• Setting source address of DMAC channel<br>• Setting Destination address of DMAC channel is SITFDR<br>• Setting number of Transfer in DMAC transfer count register(add 1 word data as dummy)<br>• Setting transfer as fixed destination address mode.<br>• Setting Resouse Select register of DMAC channel, please refer to Chapter describe DMAC module to get the value.<br>• DMAC start. | • DMAC setting, Refer To Chapter Describe DMAC module for More detail<br>• Setting to interrupt controller for MSIOF interrupt event, refer to chapter describe interrupt controller for detail. | |
| 3 | SITMDR1/2/3<br>SIFCTR<br>SIIER setting | • Operating mode<br>• FIFO water mark<br>• interrupt: Refer to Section 109.3.7 or SIIER register description (109.2.12) for MSIOF's Interrupt Factor(Do not use SIIER.TFUDFE).<br>• dma enable (SIIER.TDMAE) | Do not include dummy data when setting the transfer word count.(SITMDR2.WDLEN1, SITMDR3.WDLEN2) |
| 4 | PFC setting | Change MSIOF_TXD, MSIOF_RXD, MSIOF_SCK, MSIOF_SYNC to module side | |
| 5 | Set SICTR.TXE to 1 | Setting transmission enable | |
| 6 |  | Read SICTR.TXE = 1 | |
| 7 | Wait for MSIOF Tx FIFO full | SIFCTR.TFUA = 6'h00 | |
| 8 | Data transmission synchronize with MSIOF_SYNC | | Data transmission |
| 9 |  | | Generating interrupt/DMA request |
| 10 | Write transmission data to FIFO | | |
| 11 |  | Wait bit TE in DMAC CHCR register set to 1, refer to chapter describe DMAC module for detail. Or use interrupt to avoid software polling, DMAC module support interrupt when the transfer end. | Do not need to transmit dummy data. |
| 12 |  | | |
| 13 | Set SICTR.TXE to 0 | Transmission disable | |
| 14 |  | Read SICTR.TXE = 0 | |
| 15 | PFC setting | Change MSIOF_SCK, MSIOF_SYNC to GPIO(In) side | |
| 16 | Assert SW reset for MSIOF | SRCRn.MSIOF (Refer to sec 9. Module Standby, Software Reset) | Clear dummy data<br>n = 6 |
| 17 | De-assert SW reset for MSIOF | SRSTCLRn.MSIOF (Refer to sec 9. Module Standby, Software Reset) | n = 6 |
| 18 |  | | |

**Figure 109.14 Example of Transmit Procedure in Slave Mode (transmission size is more than 256-words per frame with DMAC)**

### (4) Reception in slave mode

Figure 109.15 shows an example of settings and operation for reception in slave mode.

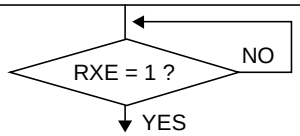
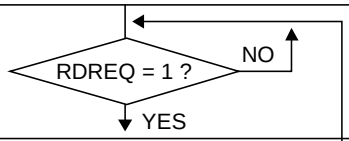
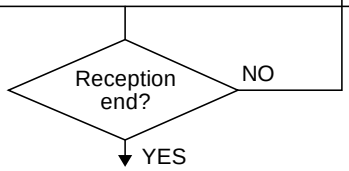
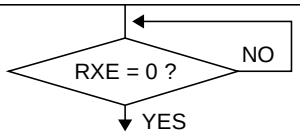
| No. | Operating flow | Description | Operation |
|---|---|---|---|
| 1 | Master device setting | Depend on master device | Signal polarity |
| 2 | SITMDR1<br>SIRMDR1/2/3<br>SIFCTR<br>SIIER setting | Operating mode (For slave setting)<br>Operating mode<br>FIFO water mark<br>Interrupt/DMA enable | TRMD = 0, PCON = 1 |
| 3 | Set SICTR.RXE to 1 | Setting receive enable | |
| 4 |  | Read SICTR.RXE = 1 | |
| 5 | Data reception | | Receive data |
| 6 |  | | Generating interrupt/DMA request |
| 7 | Read SIRFDR | Data read from FIFO | |
| 8 |  | | |
| 9 | Set SICTR.RXE to 0 | Reception disable | |
| 10 |  | Read SICTR.RXE = 0 | |

**Figure 109.15 Example of Receive Procedure in Slave Mode**

### (5) Reset

After a power-on reset is applied, module stop state is canceled, or a reset signal is asserted through the TXRST or RXRST setting in SICTR, it takes about 20 cycles of module clock to complete the internal reset of the module. During a period from the beginning of reset until the TXRST or RXRST value is read as 0, do not modify the control register or access the FIFOs.

The MSIOF can reset the transmit and receive units by setting the following bits to 1.

- Transmit reset: TXRST bit in SICTR
- Receive reset: RXRST bit in SICTR

**Table 109.9 Transmit and Receive Reset**

| Type | Objects Initialized |
|---|---|
| Transmit reset | SITFDR (Transmit FIFO write pointer)<br>TDREQ bit in SISTR<br>TXE bit in SICTR<br>(When SITMDR1.PCON = 1, Receive reset is also asserted.) |
| Receive reset | SIRFDR (Receive FIFO read pointer)<br>RDREQ bit in SISTR<br>RXE bit in SICTR |

### (6) Initial operating mode

After a power-on reset, both transmit and receive units are initialized to master mode (0 is output through MSIOF_SCK and MSIOF_SYNC). When using slave mode, keep the connected device outputting 0 until the operating mode setting is completed.

### 109.3.7 Interrupts

The MSIOF has one type of interrupt.

#### (1) Interrupt sources

Interrupts can be generated by several sources. Each source is shown as an MSIOF status in SISTR. Table 109.10 lists the MSIOF interrupt sources.

**Table 109.10 MSIOF Interrupt Sources**

| No. | Classification | Bit Name | Function Name | Description |
|---|---|---|---|---|
| 1 | Transmission | TDREQ | Transmit FIFO transfer request | The transmit FIFO has empty space of specified size or more. |
| 2 | | TFEMP | Transmit FIFO empty | The transmit FIFO is empty. |
| 3 | | TEOF | Frame transmission end | Transmission of data with a length of one frame is completed. |
| 4 | Reception | RDREQ | Receive FIFO transfer request | The receive FIFO stores data of specified size or more. |
| 5 | | RFFUL | Receive FIFO full | The receive FIFO is full. |
| 6 | | REOF | Frame reception end | Reception of data with a length of one frame is completed. |
| 7 | Error | TFUDF | Transmit FIFO underflow | Serial data transmit timing has arrived while the transmit FIFO is empty. |
| 8 | | TFOVF | Transmit FIFO overflow | The transmit FIFO is written to while it is full. |
| 9 | | RFOVF | Receive FIFO overflow | Serial data is received while the receive FIFO is full. |
| 10 | | RFUDF | Receive FIFO underflow | The receive FIFO is read while the receive FIFO is empty. |
| 11 | | TFSERR | Transmit FS error | A transmit frame synchronization signal is input before the specified number of bits has been reached (in slave mode). |
| 12 | | RFSERR | Receive FS error | A receive frame synchronization signal is input before the specified number of bits has been reached (in slave mode). |

Whether an interrupt is issued from an interrupt source is determined by the SIIER settings. If an interrupt source bit is set to 1 and the corresponding bit in SIIER is set to 1, an MSIOF interrupt is issued.

#### (2) TDREQ and RDREQ

The transmit FIFO transfer request (TDREQ) and receive FIFO transfer request (RDREQ) are signals indicating the state. If the state changes after TDREQ or RDREQ is set, it is automatically cleared by the MSIOF.

When the DMA transfer is used, a DMA transfer request signal is pulled low for one cycle at the end of DMA transfer.

**(3) Processing when errors occur**

- Transmit FIFO underflow (TFUDF)
  An undefined value is output.
- Transmit FIFO overflow (TFOVF)
  The contents of the transmit FIFO are protected, and the write operation causing the overflow is ignored.
- Receive FIFO overflow (RFOVF)
  Data causing the overflow is discarded and lost.
- Receive FIFO underflow (RFUDF)
  An undefined value is output on the bus.
- Transmit FS error (TFSERR)
  The internal counter is reset according to the synchronization signal in which an error occurs.
- Receive FS error (RFSERR)
  The internal counter is reset according to the synchronization signal in which an error occurs.

### 109.3.8 Transmit and Receive Timing

Examples of the MSIOF serial transmission and reception are shown in Figure 109.16 and Figure 109.17.

**(1) 16-bit synchronization pulse**

Synchronization pulse method, one group, one word, 16 bits in a word, no bit delay

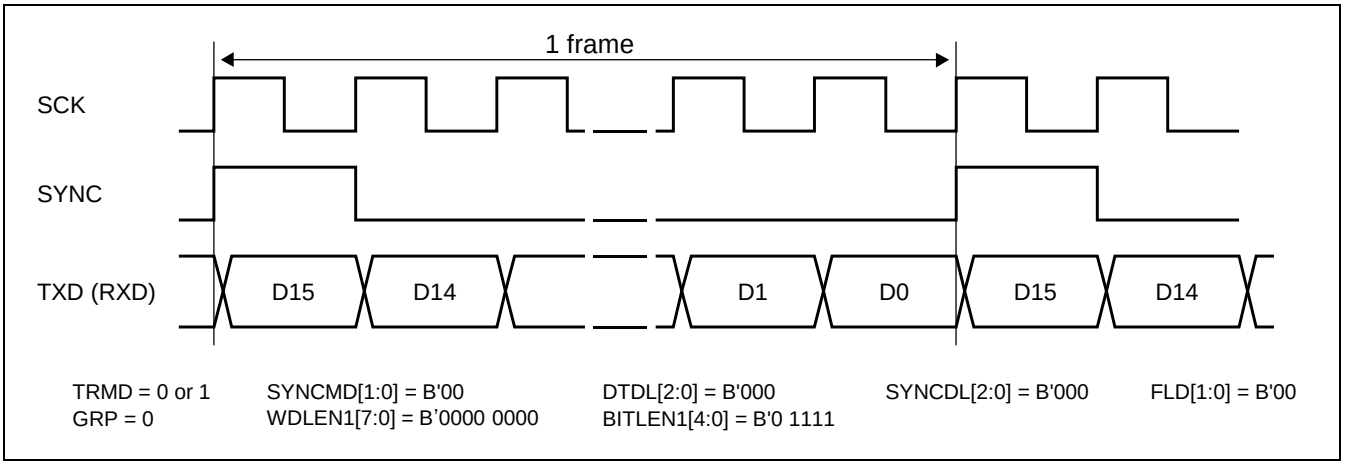



**Figure 109.16 Transmit and Receive Timing (16 Bits)**

**(2) 32-bit synchronization pulse**

Synchronization pulse method, one group, 32 words, 32 bits in a word, 1-bit delay

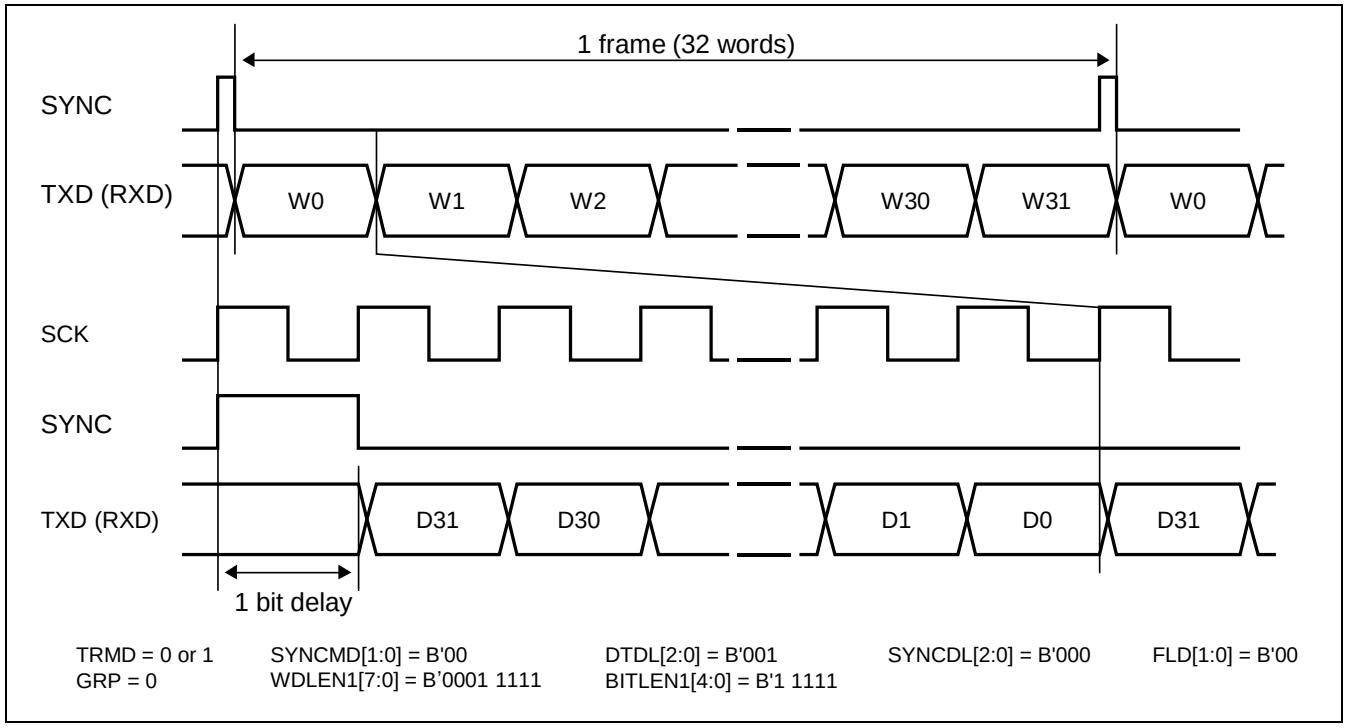



**Figure 109.17 Transmit and Receive Timing (32 Bits)**

**(3) 32-bit IIS transmission**

L/R mode, 2 groups, 1 word for each group, 32-bit word for each group, 1-bit delay

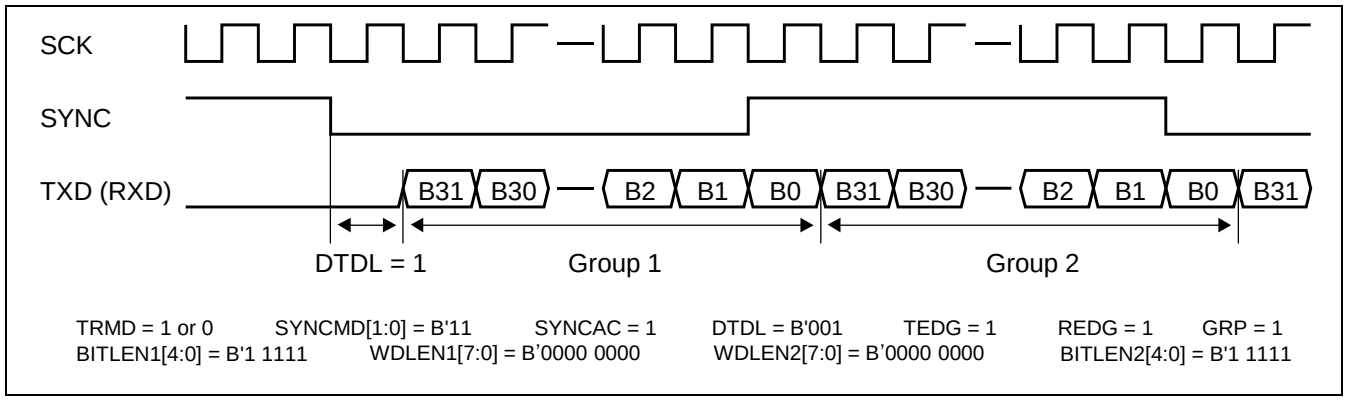



**Figure 109.18 Transmit Timing (32-Bit IIS)**

### 109.3.9 SPI

With the appropriate register settings, the MSIOF can be used as an SPI device.

**(1) Example of SPI device connection**

Figure 109.19 shows an example of connection with an SPI device.

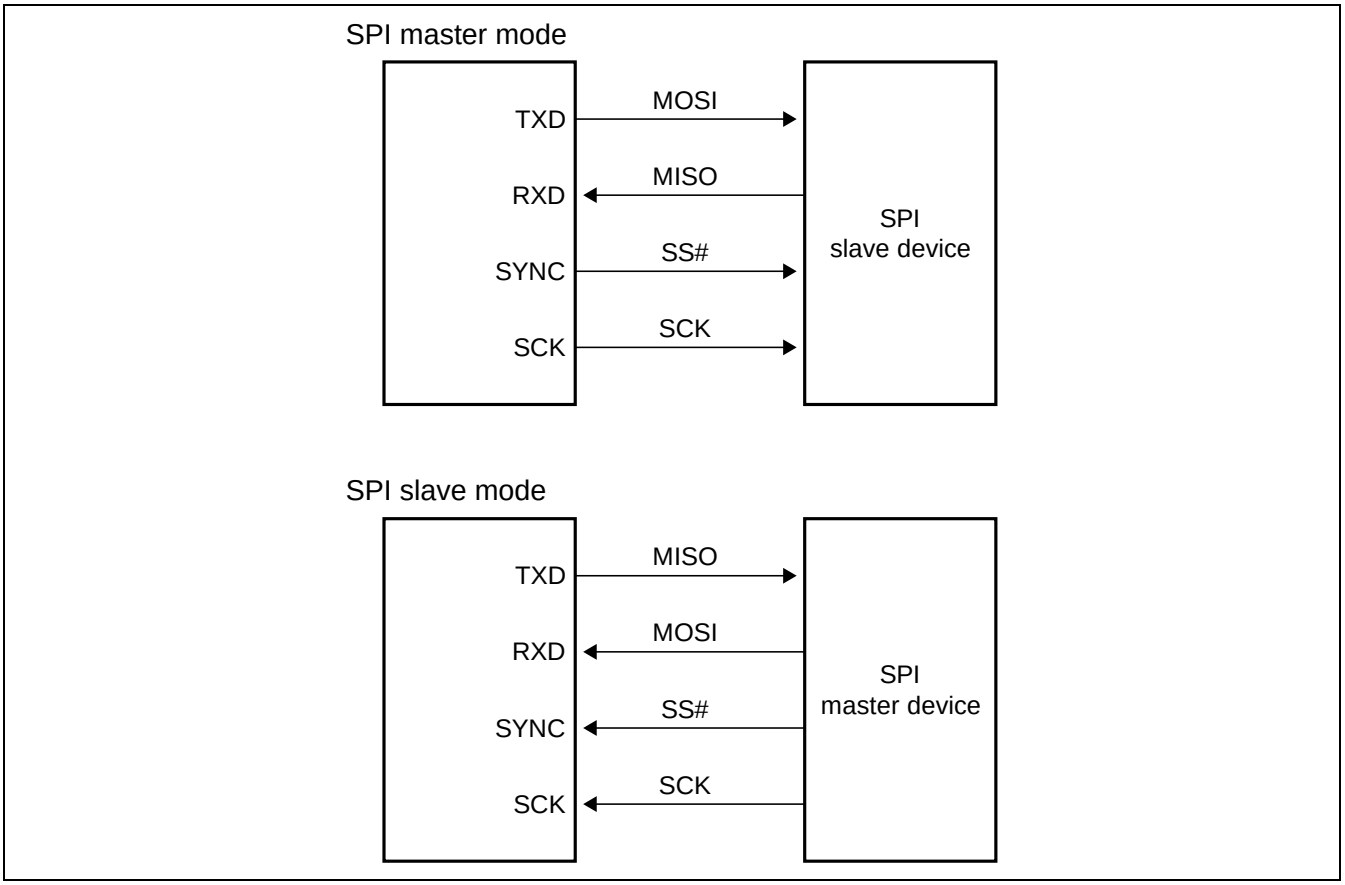



**Figure 109.19 Example of SPI Device Connection**

### (2) SPI serial clock timing

Figure 109.20 and Figure 109.21 show the data and clock timing in SPI mode. As shown in the figures, four types of serial transfer formats can be used.

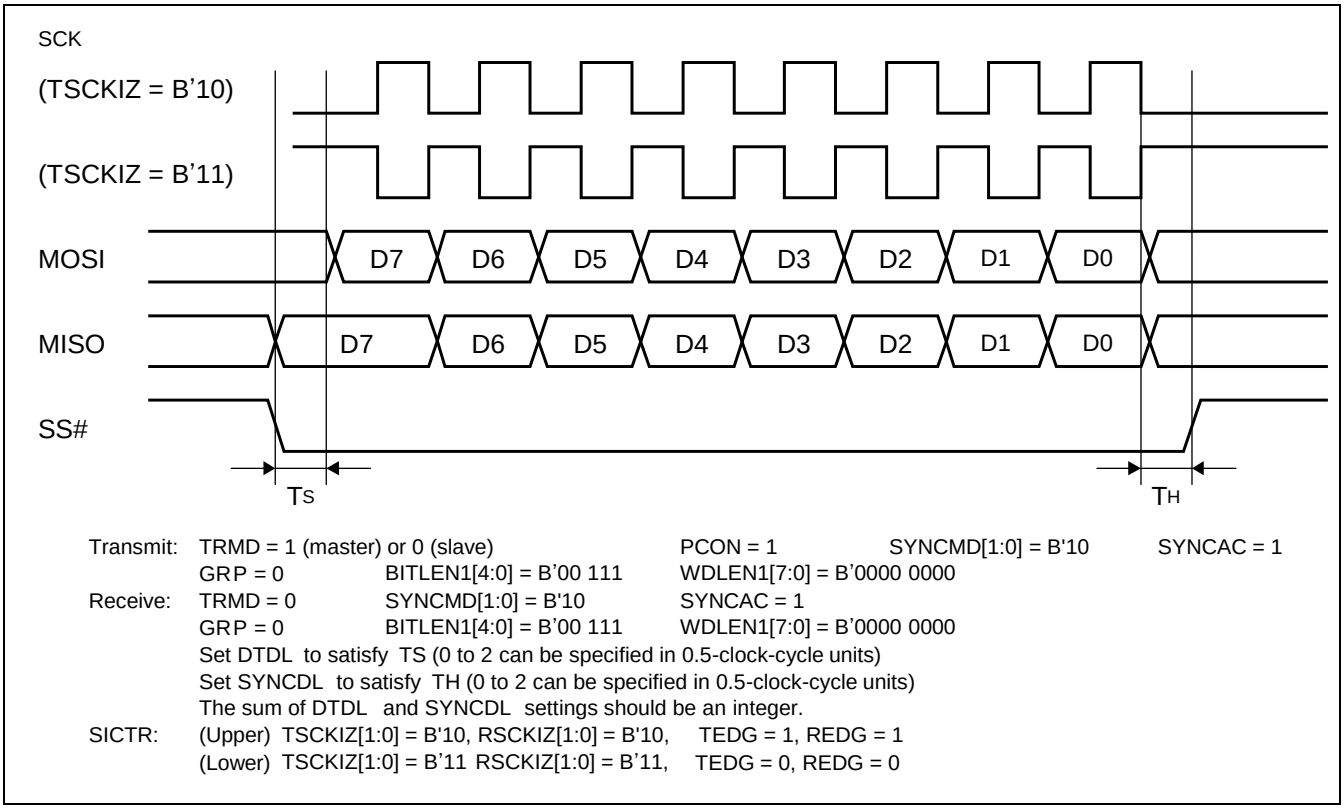



**Figure 109.20 SPI Clock and Data Timing 1**

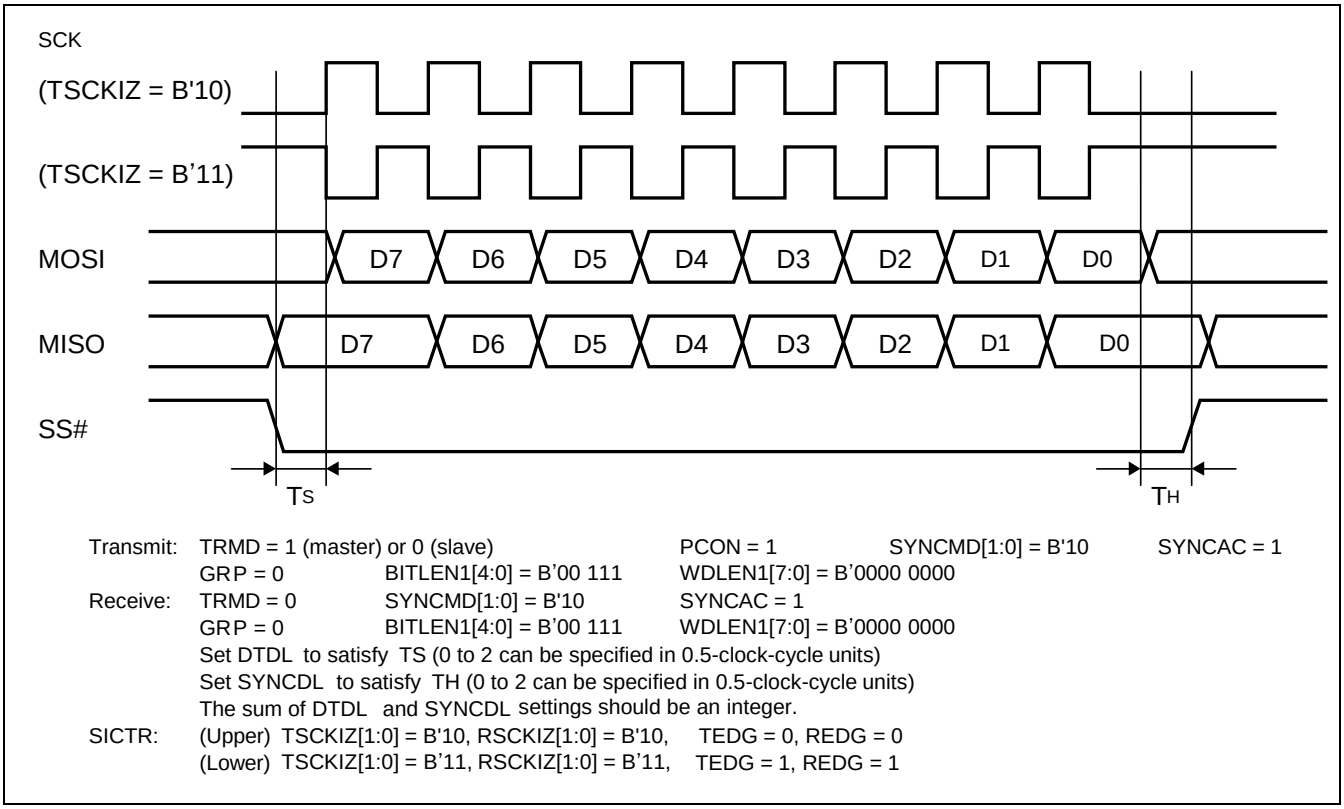



**Figure 109.21 SPI Clock and Data Timing 2**

### 109.3.10 IIS

This module can support IIS format with following setting which is shown in Figure 109.22 and Figure 109.23. BITLEN1 and BITLEN2 bits will change by usage.

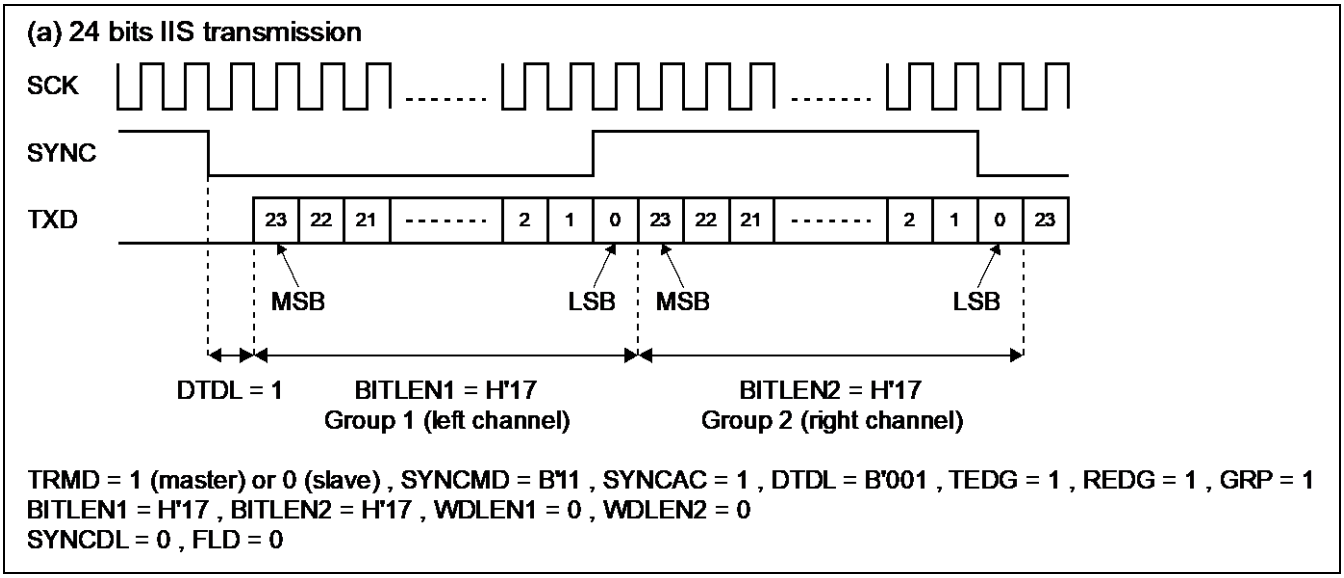



**Figure 109.22 24 Bits IIS Transmission**

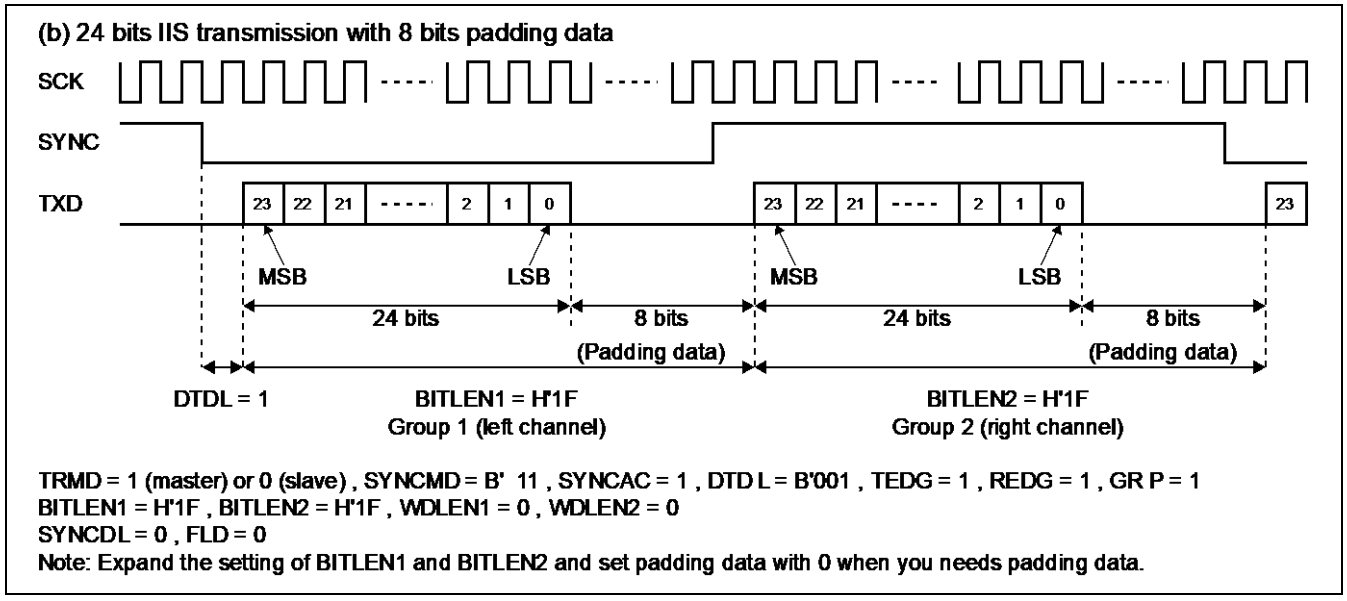



**Figure 109.23 24 Bits IIS Transmission with 8 Bits Padding Data**

## 109.4 Usage Notes

### 109.4.1 Notes for Using SPI Mode

- For SPI master mode operation, set the FLD bits in SITMDR1 for at least 3 cycles of module clock.
- For SPI slave mode operation, set the frame synchronization signal interval to at least 4 cycles of module clock.
- For SPI slave mode operation, the setting for data delay should be always set to 0.
- For SPI slave mode operation, TXDIZ bits work on SS signal invalid state.

### 109.4.2 Notes for Using SPI slave Mode

1. In case of SPI slave mode, this module can't receive any data from external device, before SICTR [8]. RXE is set to 1. Prepare scheme to guarantee data will arrive only after SICTR [8]. RXE is set to 1, by higher level arbitration.

2. In case of SPI slave mode, this module needs to receive assert of SYNC signal to start receiving data. For this purpose, SYNC signal from master device must be connected to SYNC port, even in the environment, in which only two devices (master and slave) exist. Also, SYNC signal must be asserted after SICTR [8]. RXE bit is set. SYNC signal assertion while SICTR [8]. RXE is 0, received data can't guarantee.

3. When the MSIOF operates with all the following condition 1 to 3, the MSIOF in SPI slave mode cannot start the next transmission. The MSIOF cannot support SPI mode under following condition.

- Condition

1) MSIOF is in SPI slave mode (SITMDR1.TRMD = 0, SITMDR1.SYNCMD[1:0] = 10)
2) SYNC (CS) signal is being asserted from master
3) TXE or RXE in SICTR is set to 1 during SYNC (CS) is asserted

This limitation means the MSIOF cannot respond to the SYNC (CS).
It is necessary to control the MSIOF enabling timing by the master and system.

[Workaround]
Use master mode for SPI.
If use slave mode for SPI, consider following alternative.

[Alternative]
If the master device can support handshaking, use the handshaking transmission.
For the MSIOF in SPI slave mode, use with the GPIO output for SPI handshaking; the system operation must be tested and evaluated thoroughly by user.

4. In case of SPI slave mode, this module must not set SICTR[1].TXRST or SICTR[0].RXRST during SYNC signal asserted.

### 109.4.3 Notes for Transmitting Audio data via MSIOF

If you want to transmit audio data via MSIOF, don't set the master clock (MSOφ) to SSCG.

### 109.4.4 Notes on Fixed Value related to MSIOF registers

It is necessary to change from the initial value to the specific fixed value for these bits in the MSIOF registers. Setting other value is prohibited.

**Table 109.11 Fixed Value for the MSIOF Registers without any condition**

| Section | | Bit Number | Fixed Value |
|---|---|---|---|
| 109.2.1 | MSIOF Transmit Mode Register 1 (SITMDR1) | 30 | B'1 |

### 109.4.5 Notes for Transmit FIFO Underflow (SISTR.TFUDF)

If transmission size is less than or equal to 256-words per frame, transmit FIFO Underflow (SISTR.TFUDF) needs to be ignored. If transmission size is more than 256-words per frame, use Transmit FIFO Empty (SISTR.TFEMP) instead of this bit. To skip Empty (SISTR.TFEMP) state after the transmission completion, write 1-word dummy data to the transmit FIFO and perform reset after the transferring. If the next transfer data can be prepared, it can be possible to store next data instead of dummy data. In that case, the reset process can be skipped. If Empty is detected during transfer, reset after the stop processing of each module completed, and redo initialization.

The following is an example flow when Transmit FIFO Empty (SISTR.TFEMP) error is detected.

1. SISTR.TFEMP interrupt detection
2. Clear interrupt enable of MSIOF
3. Wait for SISTR.TEOF/REOF=1 state of MSIOF
4. Stop processing of DMAC (After DMACHCR.DE=0 at DMAC, wait for DMACHCR.DE=0)
5. DMAC channel reset (DMACHCLR.CLR[m]=1 at DMAC, m=usage channel)
6. Change PFC setting(MSIOF_SCK, MSIOF_SYNC) to GPIO(In) side
7. Assert SW reset for activating MSIOF channels in use (SRCRx.MSIOF) (x= 6)
8. De-assert SW reset for using MSIOF channels in use (SRSTCLRx.MSIOF) (Regarding procedure of SW reset, please refer to the sec 9. Module Standby, Software Reset) (x=6)
9. Reconfigure MSIOF and DMAC

When using PIO access, the No4 and No5 and the reconfiguration of DMAC in No9 are unnecessary. (The reconfiguration of MSIOF in No9 is necessary.)

Note: DMAC is SYS-DMAC

### 109.4.6 Notes for FIFO Underflow/Overflow (SISTR.TFUDF/RFOVF)

Basically, big data size transfers (256 words or more), it recommend a BITLEN = B'1 1111(32 bit) setting. It is possible to set it in a way other than BITLEN = B'1 1111(32 bit), but since the transfer efficiency is reduced, overflow/underflow may occur. In that case, it needs to be suppressed by the system by adjusting the QoS. Refer to section 18 AXI-Bus for QoS settings.

## 109.5 Safety Mechanisms

This module has no Mechanism of FuSa.

# 110. PWM

## 110.1 Overview

### 110.1.1 Features

- PWM output cycle settable (10 bits)
- PWM output cycle settable within the range from 2 cycles to 2^24 × 1023 cycles of internal bus clock (i.e. from 30 ns to 257.5 seconds when internal bus clock = 66.66 MHz)
- High-level width of the PWM output signal settable (10 bits)
- Continuous pulse output or single pulse output selectable

### 110.1.2 Block Diagram

Figure 110.1 shows a block diagram of the PWM timer.

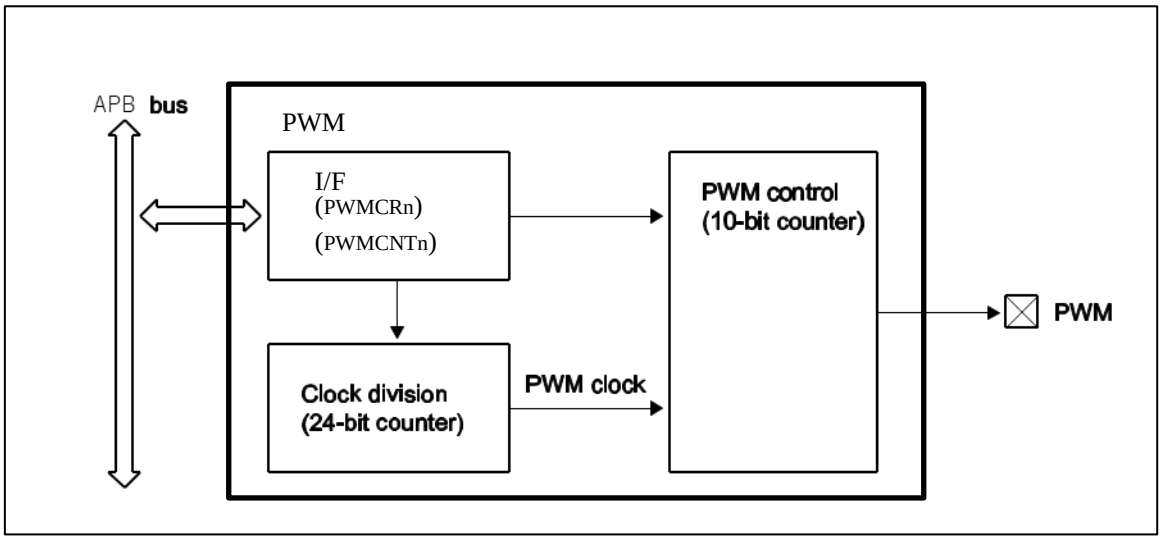



**Figure 110.1 Block Diagram of PWM Timer**

### 110.1.3 External Pins

Table 110.1 shows the pin configuration of the PWM timer.

Table 110.1 Pin Configuration

| Pin Name | I/O | Function |
|---|---|---|
| PWM0 | Output | PWM0 timer pulse output |
| PWM1 | Output | PWM1 timer pulse output |
| PWM2 | Output | PWM2 timer pulse output |
| PWM3 | Output | PWM3 timer pulse output |
| PWM4 | Output | PWM4 timer pulse output |

### 110.1.4 Register Configuration

Table 110.2 lists the registers of the PWM timer. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined. Access the register as a longword (32 bits). Operation cannot be guaranteed if the register is not accessed as a longword.

Table 110.2 List of Registers

| Channel | Name | Abbreviation | R/W | Address | Access Size |
|---|---|---|---|---|---|
| 0 | PWM control register 0 | PWMCR0 | R/W | H'E6E3 0000 | 32 |
| | PWM count register 0 | PWMCNT0 | R/W | H'E6E3 0004 | 32 |
| | PWM error injection register 0 | PWMEI0 | R/W | H'E6E3 000C | 32 |
| 1 | PWM control register 1 | PWMCR1 | R/W | H'E6E3 1000 | 32 |
| | PWM count register 1 | PWMCNT1 | R/W | H'E6E3 1004 | 32 |
| | PWM error injection register 1 | PWMEI1 | R/W | H'E6E3 100C | 32 |
| 2 | PWM control register 2 | PWMCR2 | R/W | H'E6E3 2000 | 32 |
| | PWM count register 2 | PWMCNT2 | R/W | H'E6E3 2004 | 32 |
| | PWM error injection register 2 | PWMEI2 | R/W | H'E6E3 200C | 32 |
| 3 | PWM control register 3 | PWMCR3 | R/W | H'E6E3 3000 | 32 |
| | PWM count register 3 | PWMCNT3 | R/W | H'E6E3 3004 | 32 |
| | PWM error injection register 3 | PWMEI3 | R/W | H'E6E3 300C | 32 |
| 4 | PWM control register 4 | PWMCR4 | R/W | H'E6E3 4000 | 32 |
| | PWM count register 4 | PWMCNT4 | R/W | H'E6E3 4004 | 32 |
| | PWM error injection register 4 | PWMEI4 | R/W | H'E6E3 400C | 32 |

Note: Do not access addresses other than those listed above. If access is attempted, a malfunction may occur.

Table 110.3 Register States in Each Operating Mode

| Name | Abbreviation | Power-on Reset/Software Reset | Module Standby |
|---|---|---|---|
| PWM control register n | PWMCRn | H'0000 0000 | Retained |
| PWM count register n | PWMCNTn | H'0000 0000 | Retained |
| PWM error injection register n | PWMEIn | H'0000 0000 | Retained |

Note: n = 0~4.

The registers of the PWM timer are mapped into the internal bus address space.

## 110.2 Register Description

Registers in the PWM timer are allocated to and arranged in the address space of the internal bus.

Legend for Register Description

Initial value: Register value after a reset

—: Undefined value

R/W: Readable/writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

R/WC1: Readable/writable. Writing 1 initializes the bit. Writing 0 is ignored.

R: Read-only. When the bit is reserved, the write value should always be 0. Writing 1 to these bits can cause a malfunction of PWM.

—/W: Write-only. The read value is undefined.

### 110.2.1 PWM Control Register n (PWMCRn) (n = 0 to 4)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CC0 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CCMD | — | — | — | SYNC | — | — | — | — | FS | | SS0 | ECEN | — | — | EN0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R/W | R | R | R | R | R/W | R/W | R/W | R/W | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>The read value is always 0. The write value should always be 0. |
| 19 to 16 | CC0 | 0000 | R/W | Clock Control<br>The clock frequency of the PWM counter is obtained by dividing the frequency of the clock signal of internal bus clock.<br>When the internal bus clock frequency is f MHz, the following PWM clock frequencies are obtained.<br>0000: f MHz<br>0001: f MHz/2^2<br>0010: f MHz/2^4<br>0011: f MHz/2^6<br>0100: f MHz/2^8<br>0101: f MHz/2^10<br>0110: f MHz/2^12<br>0111: f MHz/2^14<br>1000: f MHz/2^16<br>1001: f MHz/2^18<br>1010: f MHz/2^20<br>1011: f MHz/2^22<br>1100: f MHz/2^24<br>When 1101, 1110, or 1111 is set, the PWM clock frequency is f MHz/2^24. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | CCMD | 0 | R/W | CC0 Frequency Division Mode<br>Changes the PWM clock frequency set by the clock control (CC0) bits.<br>0: The PWM clock frequency set by CC0 is used.<br>1: The PWM clock frequency set by CC0 is changed as follows:<br>1 - 0000: f MHz/2<br>1 - 0001: f MHz/2^3<br>1 - 0010: f MHz/2^5<br>1 - 0011: f MHz/2^7<br>1 - 0100: f MHz/2^9<br>1 - 0101: f MHz/2^11<br>1 - 0110: f MHz/2^13<br>1 - 0111: f MHz/2^15<br>1 - 1000: f MHz/2^17<br>1 - 1001: f MHz/2^19<br>1 - 1010: f MHz/2^21<br>1 - 1011: f MHz/2^23<br>If 1100, 1101, 1110, or 1111 is set for CC0, the PWM clock frequency is f MHz/2^24 irrespective of the CCMD value. |
| 14 to 12 | — | All 0 | R | Reserved<br>The read value is always 0. The write value should always be 0. |
| 11 | SYNC | 0 | R/W | Specifies whether to allow the set values in the PWM count register (PWMCNTn) to be reflected in the timer operation synchronously with setting the PWM control register (PWMCRn).<br>0: Allows the PWMCNTn set values to be reflected in the timer operation irrespective of PWMCRn setting.<br>1: Allows the PWMCNTn set values to be reflected in the timer operation synchronously with PWMCRn setting.<br>For details, refer to section 110.3 Operation. |
| 10 to 7 | — | All 0 | R | Reserved<br>The read value is always 0. The write value should always be 0. |
| 6 to 5 | FS*1 | All 0 | R/W | Frequency Selection for Deglitcher function. *2<br>The frequency of Deglitcher function is obtained by dividing the frequency of the clock signal of internal bus clock*3.<br>When the internal bus clock*3 frequency is f MHz, the following PWM clock frequencies are obtained.<br>00: f MHz/333<br>01: f MHz/166<br>10: f MHz/3330<br>11: f MHz/1665 |
| 4 | SS0 | 0 | R/W | Single Pulse Output<br>0: The timer operates in continuous pulse output mode.<br>1: The timer operates only for a single cycle and then stops. |
| 3 | ECEN*1 | 0 | R/W | Error Check Enable.*2<br>0: Disable deglitcher function<br>1: Enable deglitcher function |
| 2 to 1 | — | All 0 | R | Reserved<br>The read value is always 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | EN0 | 0 | R/W | Channel Enable<br>0: The channel is held in the idle state, and outputs a high level.<br>1: The channel outputs high and low levels in a predetermined cycle.<br>This bit is automatically cleared in single pulse output mode. |

Notes: *1. Functional safety use only. Keep initial value.
*2 This explanation is the explanation for PWMCRn of Channel 1.
*3: Refer to the information about internal bus clock in section 110.1.1 Features.

### 110.2.2 PWM Count Register n (PWMCNTn) (n = 0 to 4)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CYC0 | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | PH0 | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R | Reserved<br>The read value is always 0. The write value should always be 0. |
| 25 to 16 | CYC0 | H'000 | R/W | PWM Cycle<br>Sets the PWM output cycle. The cycle set by these bits is the sum of the high-level and low-level periods.<br>Setting H'000 is prohibited. |
| 15 to 10 | — | All 0 | R | Reserved<br>The read value is always 0. The write value should always be 0. |
| 9 to 0 | PH0 | H'000 | R/W | PWM High-Level Period<br>Sets the period for which the PWM timer outputs a high-level signal.<br>Setting H'000 is prohibited. |

Note: For the CYC0 and PH0 bits, set the number of cycles to be counted of the PWM clock signal whose frequency has been set by the CC0 and CCMD bits in the PWM control register.

### 110.2.3 PWM Error Injection Register n (PWMEIn) (n = 0 to 4)

Note: *1. This explanation is for PWMEIn of Channel 1.

PWMEIn is a 32-bit readable/writable register that controls Error Injection. This error injection is used in test of connection between PWM and MFIS.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PWMEI |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

This register is functional safety use only. Keep initial value.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | - | All 0 | R | Reserved<br>The read value is always 0. The write value should always be 0. |
| 0 | PWMEI | 0 | R/W | PWM Error Injection<br>0: Normal operation<br>1: PWM loopback function check detection error is fixed 1. |

## 110.3 Operation

The duty ratio of a PWM output pulse can be obtained by setting a high-level period and a cycle. The timer channel counts the PWM clock signal pulses using a 10-bit counter to generate the PWM output pulse having the specified period and cycle. When the channel function is enabled (EN0 = 1), the timer outputs a high level until the counter value reaches the value set in the PH0 bits of the PWM count register (PWMCNTn). The output goes low when the PH0 value is reached, and is held low until the counter value reaches the value set in the CYC0 bits of PWMCNTn. When the CYC0 value is reached, the output goes high and the counter is reset.

Unless the channel function is enabled, the output is held high, and the counter is held in the reset state.

Figure 110.2 shows the PWM state transition during the operation described above.

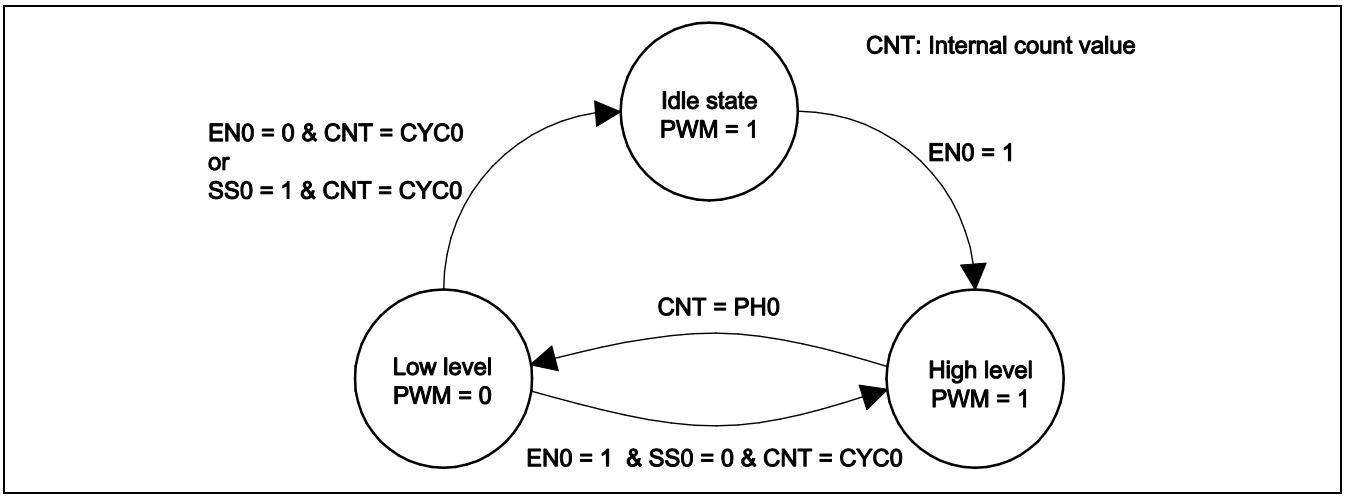



**Figure 110.2 PWM Timer State Transition**

Note that after a channel has been enabled by writing 1 to EN0, the number of clock pulses before output of the first falling edge from a channel is greater than the value set in the PWM control register and PWM count register by up to four cycles of the internal bus clock. After that, the PWM output rises and falls according to a set timing. Figure 110.3 schematically shows this output timing. This figure shows an internal operation.

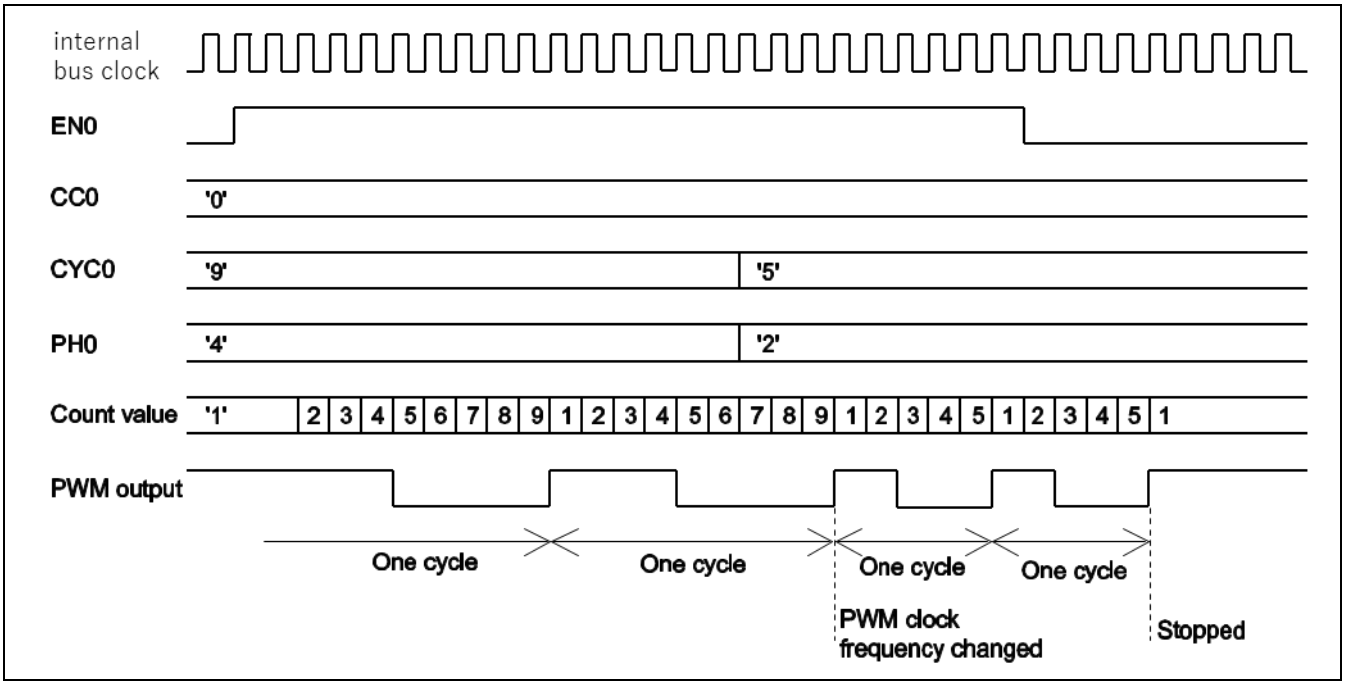



**Figure 110.3 PWM Output Timing**

1. The period from enabling the channel to the first falling edge output from the channel is the sum of a set value and up to four cycles of the internal bus clock.
2. After that, the PWM high and low widths are the same as the set values.
3. When the PWM count register is changed during operation (EN0 = 1), timing of the output will be in accord with the new settings after output of the current cycle has been completed.
4. When the CC0 and CCMD bit values in the PWM control register are changed during operation (EN0 = 1), timing of the output will be in accord with the new settings after output of the current cycle has been completed.
5. To stop the timer operation, write 0 to the EN0 bit. The timer stops operation after completing the output of the current cycle. Note that, however, if 1 is written to EN0 again before completing the output of the current cycle, the timer does not stop but continues to operate.
6. For the PWM timer output, continuous pulse output mode and single pulse output mode are supported. In continuous pulse output mode, the timer outputs pulses continuously while EN0 is 1. In single pulse output mode, the timer outputs a single pulse. In this mode, EN0 is automatically cleared.
7. To change the values of the clock control bits (CC0) and the CC0 frequency division mode bit (CCMD) in the PWMCRn register and of the PWM cycle bits (CYC0) and the PWM high-level period bits (PH0) in the PWMCNTn register at the same time during the PWM operation, the following procedure should be followed:
   1. Write 1 to the SYNC bit. (Write the original set values to the other bits in PWMCRn.)
   2. Change the PWMCNTn value.
   3. Change the CC0 and CCMD values in PWMCRn. (Write the original set values to the other bits in PWMCRn. The SYNC bit should be held at 1.)

   By following the above procedure, the new PWMCNTn value is reflected in the timer operation synchronously with PWMCRn setting.
   When changing only the PWMCNTn value while SYNC = 1, writing to PWMCRn is required after changing the PWMCNTn value. In this case, write the same values as the original set values to the PWMCRn bits. On the other hand, when changing only the PWMCRn value, writing to PWMCNTn is not required.
   If PWMCNTn and PWMCRn values are changed while SYNC = 0, the change to the values might not be completed within a single PWM output cycle but actually take two or more cycles. This may disturb the period or duty cycle of the PWM output.
   Note that, when the PWMCNTn value is changed while SYNC = 1, the new values are read from PWMCNTn even before PWMCRn is set.

## 110.4 Usage Notes

When using the PWM timer, note the following points.

When the CYC0 value is equal to or smaller than PH0 value in the PWM count register, the PWM output is held high. In this case, to start pulse output, clear EN0 to 0, set CYC0 to the value greater than the PH0 value, and then set EN0 to 1.

Note that, a maximum period equivalent to CYC0 = H'3FF at the set PWM clock frequency may be required before pulse output is started.

## 110.5 Safety Mechanisms

### 110.5.1 PWM feedback function using a deglitcher (pad check logic)

PWM module has supporting ECM connection Safety Mechanism.

This LSI chip supports a PWM loopback function to check the correctness of output values by monitoring a loopback signal from the IO cell. Detected errors are reported through the INTC or an external pin, via the MFIS module.

The PWM fault diagnosis mechanism is implemented as hardware inside the PWM module. Faults are detected inside the PWM module, and reported through the MFIS module, and then to the CPU in one of two ways, an interrupt or the signal on an external pin.

The deglitcher consists of three shift registers and an AND gate, in consideration of the delay of the loopback signal, detects errors with the signal that rejects glitches less than 3 clock cycles of timing signal (= capture_en) dividing internal bus clock by the division ratio set by PWMCR1.FS[1:0].

**a) Overview**

PWM feedback function is a safety mechanism for the digital I/O pins of PWM (only output pin). This safety mechanism checks whether the output of PWM is correctly transmitted to the output pin.

Channel 1 (Ch.1) is Safety-Related, and other channels are Non-Safety-Related.

**b) Related Registers**

Table 110.4 Related Registers

| Channe | Register Name | Abbreviation |
|---|---|---|
| 1 | PWM control register 1 | PWMCR1 |
| | PWM error injection register 1 | PWMEI1 |

**c) Hardware Description**

This LSI chip supports a PWM loopback function to check the correctness of output values by monitoring a loopback signal from the IO cell. Detected errors are reported through the INTC or an external pin, via the MFIS module.

The PWM fault diagnosis mechanism is implemented as hardware inside the PWM module. Faults are detected inside the PWM module, and reported through the MFIS module, and then to the CPU in one of two ways, an interrupt or the signal on an external pin.

The deglitcher consists of three shift registers and an AND gate, in consideration of the delay of the loopback signal, detects errors with the signal that rejects glitches less than 3 clock cycles of timing signal (= capture_en) dividing internal bus clock by the division ratio set by PWMCR1.FS[1:0].

Figure 110.4 shows a block diagram for the PWM functional safety mechanism.

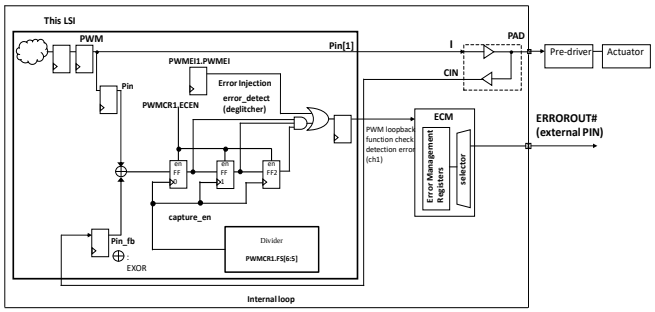



Figure 110.4 Block diagram for PWM Functional Safety Mechanism

Figure 110.5 timing chart is a timing chart for the deglitcher, an error is detected when the signal on pin [1] and the CIN signal are out of synchronization by three or more cycles of timing signal (= capture_en) dividing CLK (=internal bus clock) by the division ratio set by PWMCR1.FS[1:0] (gray-shaded portions). The signals shown in the timing chart correspond to the signals in Figure 110.4.

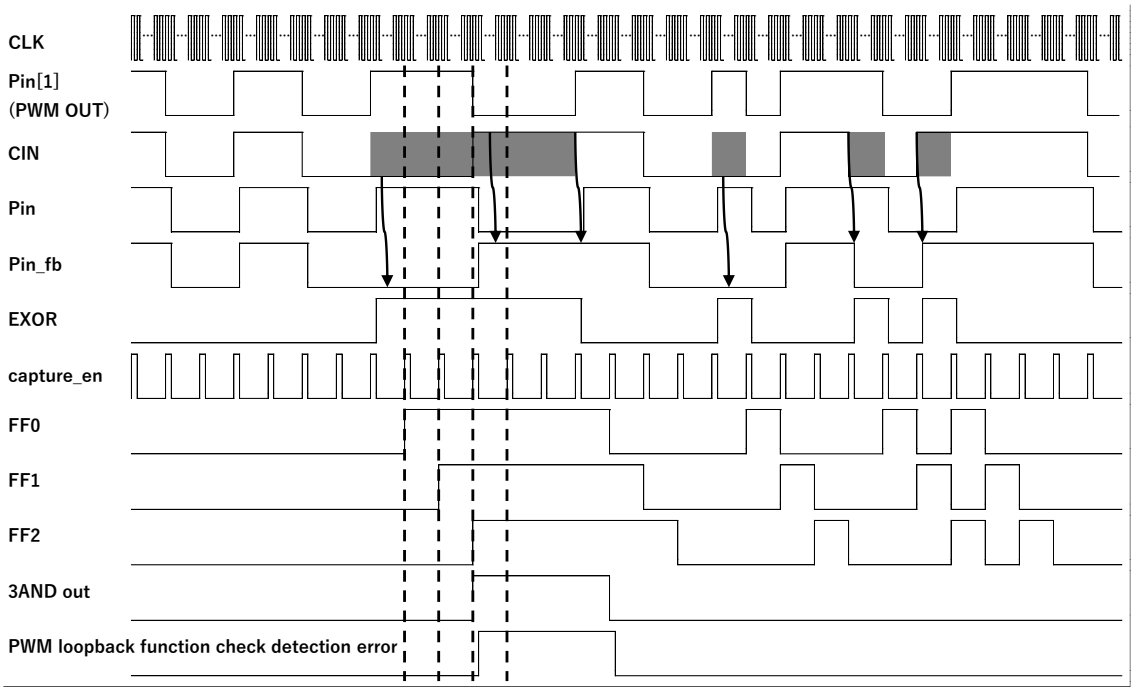



Figure 110.5 Deglitcher Timing Chart

# 112. SPI Multi I/O Bus Controller (RPC-IF)

## 112.1 Overview

The SPI multi I/O bus controller enables direct connection of the serial flash memory, HyperFlash, or Octal-SPI Flash memory to this LSI.

### 112.1.1 Features

This module allows the serial flash memory, HyperFlash, or Octal-SPI Flash memory connected to this LSI to be accessed by the external address space read mode or manual mode.

The external address space read mode is able to read from Flash memory only. The manual mode is able to write to and read from Flash memory.

**(1) Supported memory interface**

**(a) Serial Flash Memory Interface**

- Up to two serial flash memory devices can be connected. Note that two serial flash memory devices are operated in parallel interface and each 1xQSPI operation is not supported.
- When 1xQSPI configuration, QSPI0 pins in this LSI are connected to the serial flash memory.
- A data bus size of 1 bit or 4 bits can be selected for a serial flash memory device.
- Support single data rate (SDR) and Double data rate (DDR) transfer
- Only the master operation is supported (the slave operation is not supported).

**(b) Octal-SPI flash interface**

- One Octal-SPI flash can be connected.
  “Octal-SPI flash” indicates 8-bit SPI flash with one ChipSelect, one Clock source and one DataStrobe configuration.
- The data bus size is 1bit or 8bit can be selected for Octal-SPI device.
- Serial flash memory specification without “8-bit” description specifies general SPI flash interface (including SPI compatible mode) in this manual. This means when use without “8-bit” description, register settings are same as “Serial Flash Memory Interface”.

**(c) HyperFlash Interface**

- One HyperFlash can be connected.
- The data bus size is fixed to 8 bits.
- Wrapped Burst is not supported.

**(2) Transfer Mode**

**(a) External Address Space Read Mode**

- A read access from the bus master to the SPI multi I/O bus space is automatically converted into a read command to the device and the read data are returned to the bus master.
- Normal read and burst read operation
- Efficient data reception is possible by utilizing the on-chip read cache memory (64-bit line × 32 entries) during burst read operation.

**(b) Manual Mode**

- Desired command, read, and write accesses to the serial flash memory or HyperFlash memory are possible.
- Efficient data writing is possible by using write buffer.
- Not support the on-chip read cache memory during read operation.

### 112.1.2 Block Diagram

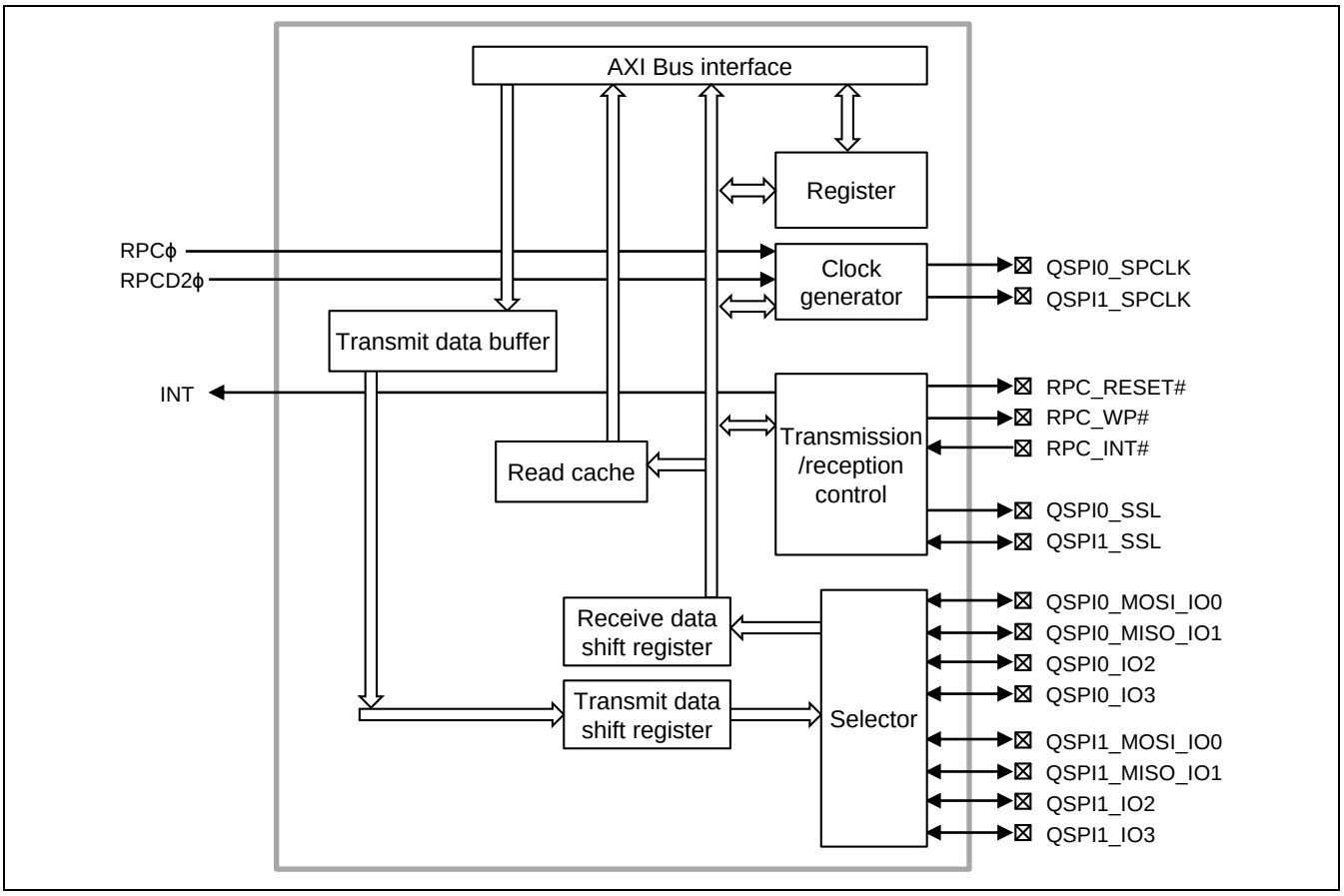



**Figure 112.1 RPC-IF Block Diagram**

### 112.1.3 External Pins

Table 112.1 shows the pin configuration of the RPC-IF.

**Table 112.1 Pin Name**

| Pin Name | Symbol | I/O | Function | | | R-Car V4M |
|---|---|---|---|---|---|---|
| | | | When serial flash is selected. | When Octal-SPI flash is selected | When HyperFlash is selected. | |
| Channel 0 clock pin*2 | QSPI0_SPCLK | Output | Clock output | Clock output | Differential clock output + | √ |
| Channel 0 master transmit data/data 0 | QSPI0_MOSI_IO0 | I/O | Master transmit data/data 0 | Data 0 | Data 0 | √ |
| Channel 0 slave transmit data/data 1 | QSPI0_MISO_IO1 | I/O | Slave transmit data/data 1 | Data 1 | Data 1 | √ |
| Channel 0 data 2 pin | QSPI0_IO2 | I/O | Data 2 | Data 2 | Data 2 | √ |
| Channel 0 data 3 pin | QSPI0_IO3 | I/O | Data 3 | Data 3 | Data 3 | √ |
| Channel 0 slave select pin | QSPI0_SSL | Output | Low active Slave selection | Low active Chip selection | Low active Chip selection | √ |
| Channel 1 clock pin*1, *2,*3 | QSPI1_SPCLK | Output | Clock output | Clock output | Differential clock output - | √ |
| Channel 1 master transmit data/data 0 pin | QSPI1_MOSI_IO0 | I/O | Master transmit data/data 0 | Data 4 | Data 4 | √ |
| Channel 1 slave transmit data/data 1 pin | QSPI1_MISO_IO1 | I/O | Slave transmit data/data 1 | Data 5 | Data 5 | √ |
| Channel 1 data 2 pin | QSPI1_IO2 | I/O | Data 2 | Data 6 | Data 6 | √ |
| Channel 1 data 3 pin | QSPI1_IO3 | I/O | Data 3 | Data 7 | Data 7 | √ |
| Channel 1 slave select pin*1*3 | QSPI1_SSL | I/O | Low active Slave selection | Read data strobe (RDS) | Read data strobe (RDS) | √ |
| Flash-RESET pin*1 | RPC_RESET# | Output | Reset signal output | Reset signal output | Reset signal output | √ |
| Write protect pin*1 | RPC_WP# | Output | Write protect (WP) | Write protect (WP) | Write protect (WP) | √ |
| INT pin*1 | RPC_INT# | Input | Interrupt input | Interrupt input | Interrupt input | √ |

Notes: When a serial flash memory is to be connected with BSZ[1:0] being set to 00, the serial flash device should be connected to the QSPI0_* pins.

In the text, the channel number is omitted (e.g. QSPIn_SSL).

These pins are 1.8V operation only

1. These pins should be connected depending on the serial flash specification.
2. The frequency of QSPIn_SPCLK is same as RPCD2ϕ.
3. When MD[4:1]=0010/0011(dma HyperFlash), 1010/1011(xip HyperFlash),
   -During RPC_RESET# is low level, QSPI1_SPCLK is low output and QSPI1_SSL is high output.
   -During RPC_RESET# is high level, QSPI_SPCLK is high output and QSPI1_SSL is Hi-Z.

### 112.1.4 Register Configuration

Refer to R-CarV4M_UM_112_RPC_r0p70.xlsx

Table 112.3 shows the register initial value depending on the boot mode.

**Table 112.3 Register initial value**

| | Initial value | | | |
|---|---|---|---|---|
| **Register Name** | **MD4-1 = 0100 (dma qspi single)** | **MD4-1 = 001* (dma HyperFlash)** | **MD4-1 = 101* (xip HyperFlash)** | **MD4-1 = other** |
| CMNCR | H'0155 7300 | H'0155 7301 | H'01AA 7301 | H'0155 7301 |
| DRCR | H'001F 0100 | H'001F 0100 | H'001F 0100 | H'001F 0101 |
| DRCMR | H'0003 0000 | H'00A0 0000 | H'00A0 0000 | H'0003 0000 |
| DRENR | H'0000 4700 | H'A222 D400 | H'A222 D400 | H'0000 4700 |
| DRDMCR | H'0000 0000 | H'0000 000E | H'0000 000E | H'0000 0000 |
| DRDRENR | H'0000 0000 | H'0000 5101 | H'0000 5101 | H'0000 0000 |
| PHYCNT | H'0807 8260 | H'0807 8263 | H'0803 8263 | H'0807 8260 |
| PHYOFFSET1 | H’3151 1144 | H’2151 1144 | H’2151 1144 | H’3151 1144 |
| PHYINT | H'0606 0002 | H'0707 0002 | H'0707 0002 | H'0606 0002 |

## 112.2 Register description

Refer to R-CarV4M_UM_112_RPC_r0p70.xlsx

## 112.3 Operation

### 112.3.1 System Configuration

With this module, one or two serial flash memory devices (data size of 1, 2, and 4 bits) or one HyperFlash (data size of 8 bits) can be connected. The connected devices can be selected using the BSZ[1:0] bits in CMNCR.

Examples of system configuration are shown in figures 112.2, 112.3, 112.4 and 112.5.

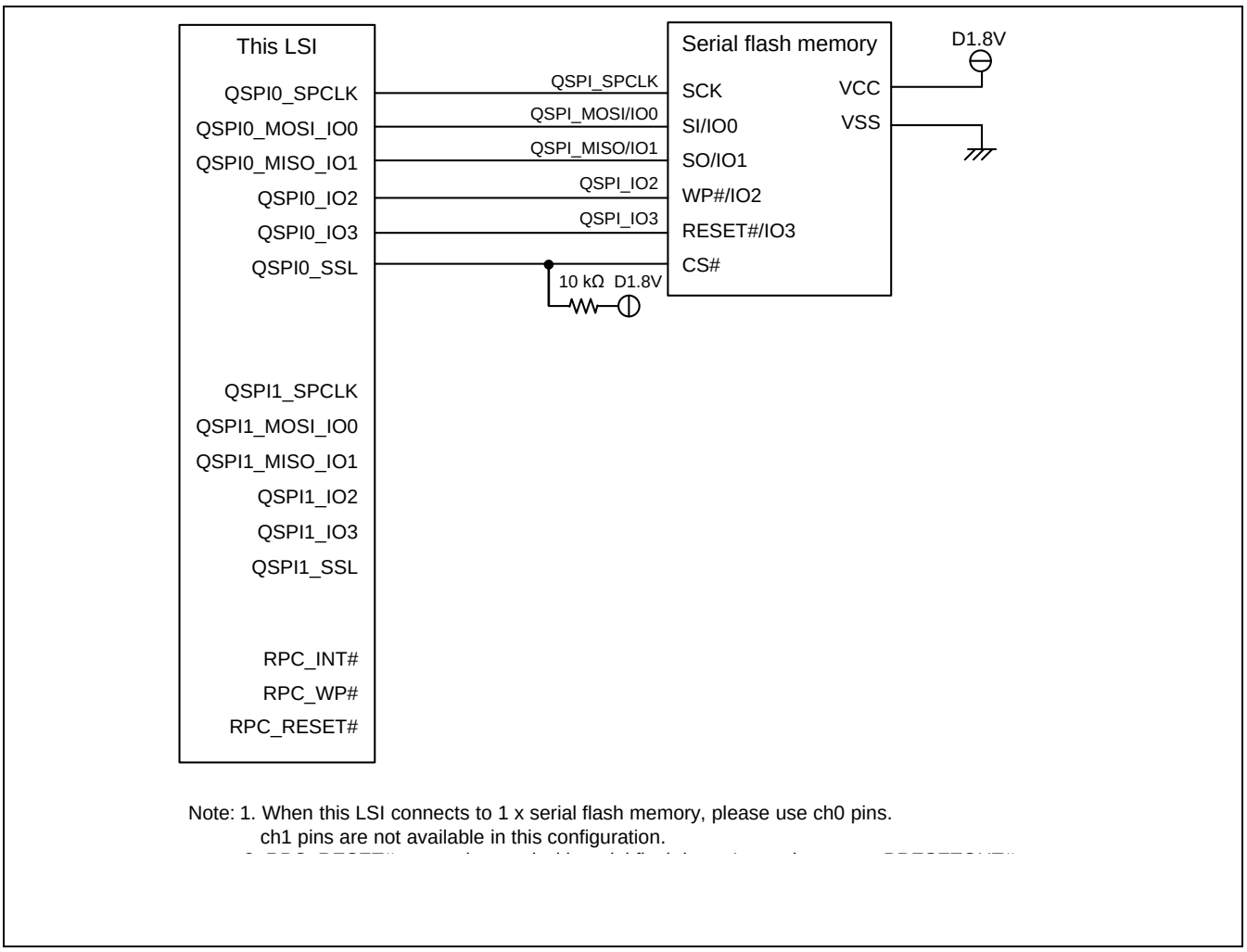



**Figure 112.2 Example of System Configuration with One 4-Bit Serial Flash Memory Device Connected**

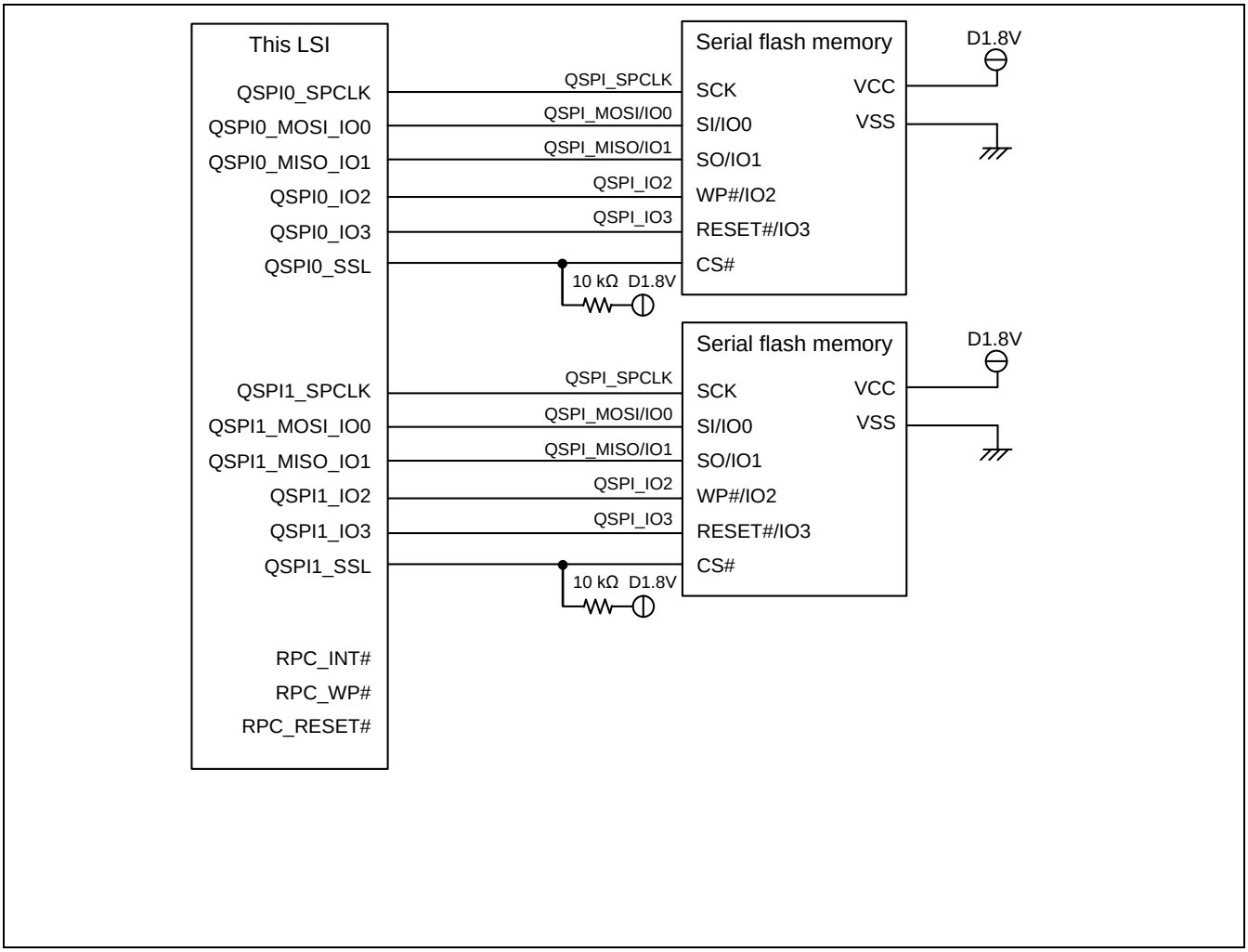



**Figure 112.3 Example of System Configuration with Two 4-Bit Serial Flash Memory Devices Connected (BSZ[1:0] Bits in CMNCR = 01)**

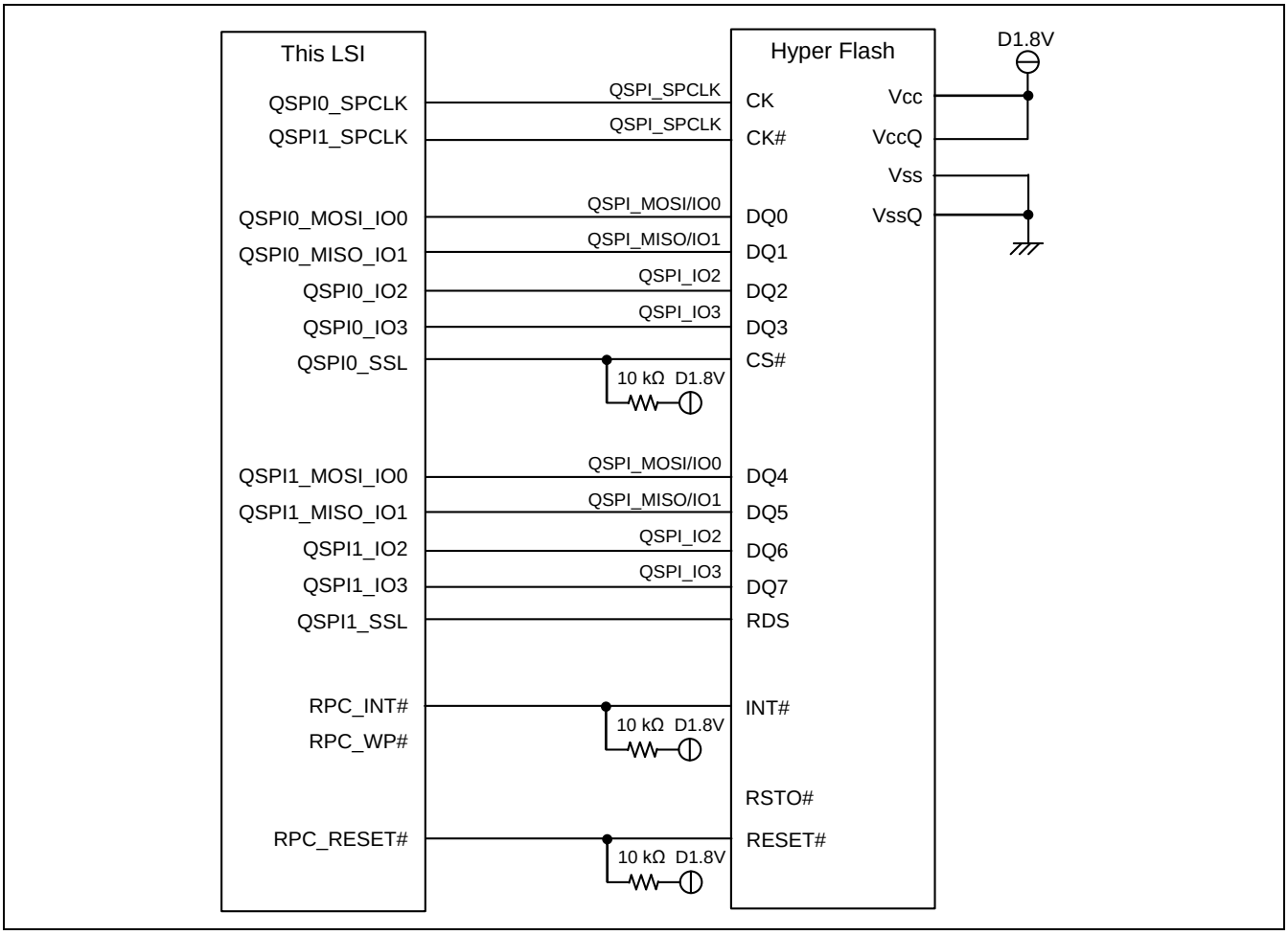



**Figure 112.4 Example of System Configuration with HyperFlash Connected**

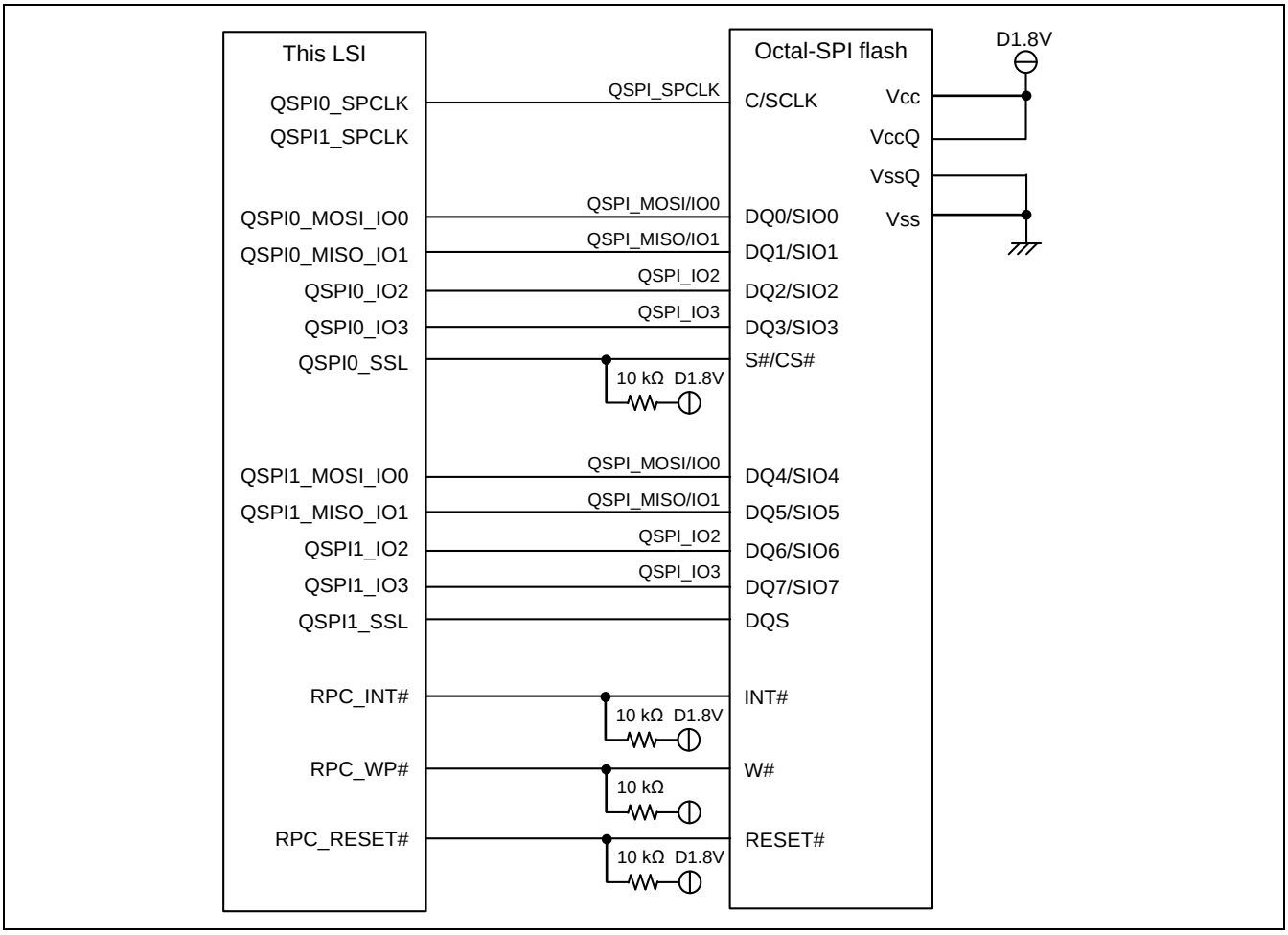



**Figure 112.5 Example of System Configuration with Octal-SPI Flash Memory Device Connected**

### 112.3.2 Address Map

In external address space read mode, the serial flash memory and HyperFlash to be connected are assigned in the SPI multi I/O bus space. The internal address space is 64MB, but with the DREAR register to specify the offset, a maximum of 4 Gbytes can be accessed when one serial flash memory is connected, and a maximum of 8 Gbytes can be accessed when two memory devices or a HyperFlash or an Octal-SPI flash memory are connected.

**Table 112.4 Address Map**

| Internal Address | Maximum Access Area |
|---|---|
| H'0800 0000 to H'0BFF FFFF | Serial flash × 1: 4 Gbytes |
| | Serial flash × 2 or HyperFlash × 1 or Octal-SPI flash × 1: 8 Gbytes |

Note: By setting access protection in LifeC, not only this address space but register area is also protected.

### 112.3.3 32-Bit Address Area in the Serial Flash Memory

Since the SPI multi I/O bus space is 64 Mbytes (with 26-bit address width), only a part of the 32-bit serial flash address area can be directly accessed. The fixed value set in the DREAR register is used as the upper-order bit value of a 32-bit address.

To output the address of the serial flash memory in 32 bits, set the ADE[3] bit in DRENR to 1, set the range of the external addresses used as the serial flash addresses to the EAC[2:0] bits in DREAR, and set the upper bit value of the 32-bit address as the fixed value to the EAV[7:0] bits in DREAR.

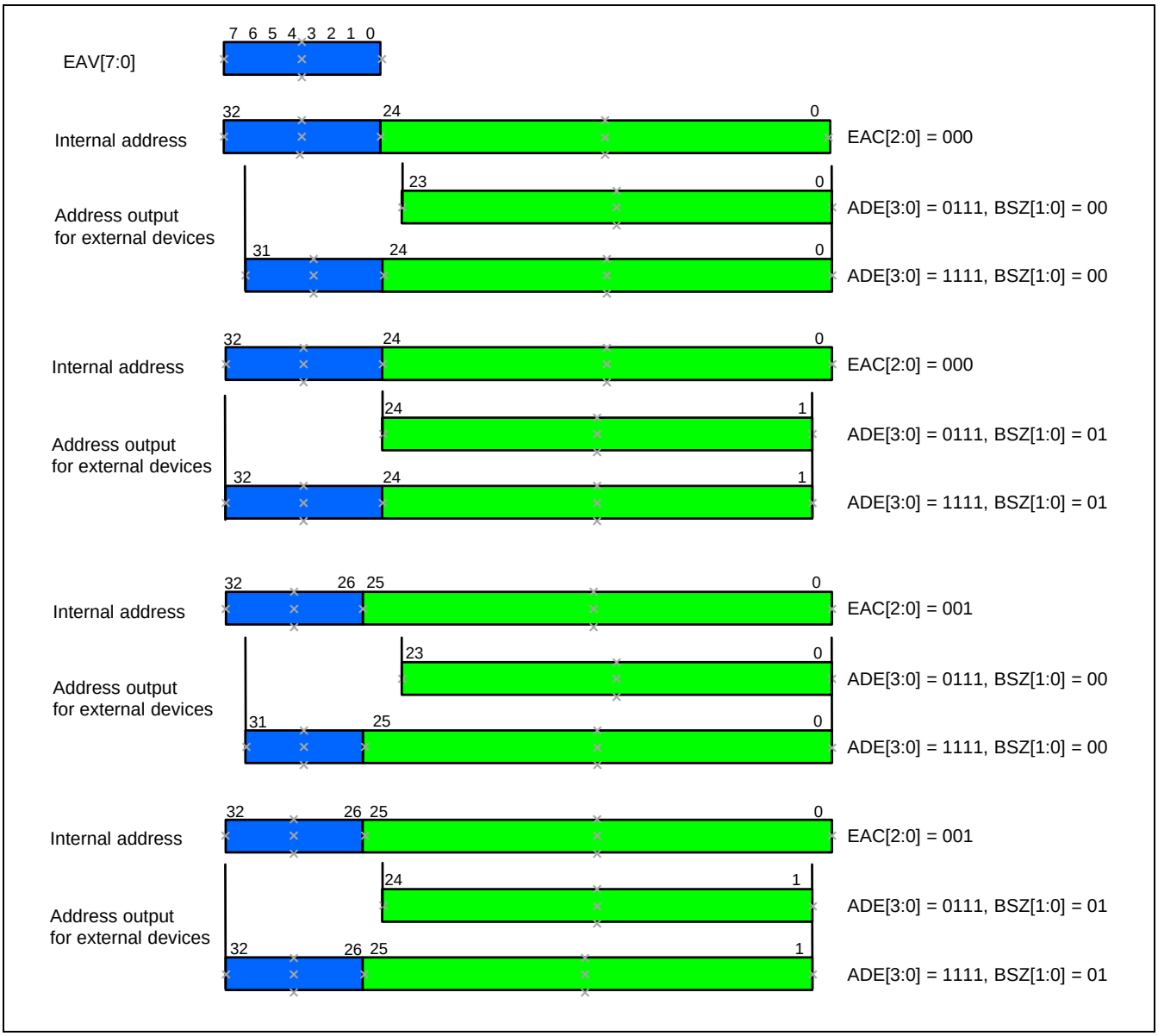



**Figure 112.6 32-Bit Address Setting**

Setting the ADE[3] bit in DRENR to 1 allows the serial flash address to be output using [31:0] bits. When EAC[2:0] = 000, external address bits [24:0] are valid. Set the value for [32:25] bits to EAV[7:0]. When EAC[2:0] = 001, external address bits [25:0] are valid. Set the value for [32:26] bits to EAV[7:1]. The address bits actually used for the external device are shown below.

When one serial flash memory is connected, internal address bits [31:0] are used.

When two serial flash memory devices are connected, internal address bits [32:1] are used.

When HyperFlash or an Octal SPI flash is connected, this address extension with DRENR.ADE[3] is not available, and internal address bits [25:0] (64 Mbytes area) are used. Use manual mode when access to the bigger area than 64 Mbytes area.

### 112.3.4 Operating Modes

This module has two operating modes: external address space read mode and manual mode.

In external address space read mode, a read access to the SPI multi I/O bus space is converted into the data read protocol and data is received from the external device. After data acquisition from the external device, data is returned to the bus master that is the issuing source of read access. For details, see section 112.3.5, External Address Space Read Mode.

The manual mode carries out an arbitrary protocol by using the register settings. For details, see section 112.3.7, Manual Mode.

### 112.3.5 External Address Space Read Mode

By external address space read mode, a read access to the SPI multi I/O bus space is converted into the data read protocol. Further, the commands, optional commands, option data, and dummy cycle issued for reading external device can be modified using registers.

In external address space read mode, either normal read operation or burst read operation can be selected. The transfer format is determined based on the common control register (CMNCR), SSL delay register (SSLDR), data read control register (DRCR), data read command setting register (DRCMR), data read extended address setting register (DREAR), data read option setting register (DROPR), data read enable setting register (DRENR), data read dummy cycle setting register (DRDMCR), and data read DDR enable register (DRDRENR).

#### (1) Normal Read Operation

When the RBE bit in DRCR is set to 0, normal read operation is performed.

In the normal read operation, the data is read from external device by reading access from bus master. After reading from external device, the QSPIn_SSL pin is negated.

The normal read operation timing with serial flash is shown in figure 112.7, Normal Read Operation Timing.

t1 is the time period from QSPIn_SSL pin assertion to QSPIn_SPCLK oscillation (clock delay), t2 is the time period from transmission of the last QSPIn_SPCLK edge of a transfer to QSPIn_SSL pin negation (QSPIn_SSL negation delay), and t3 is the time period from one transfer end to the next transfer start (next access).

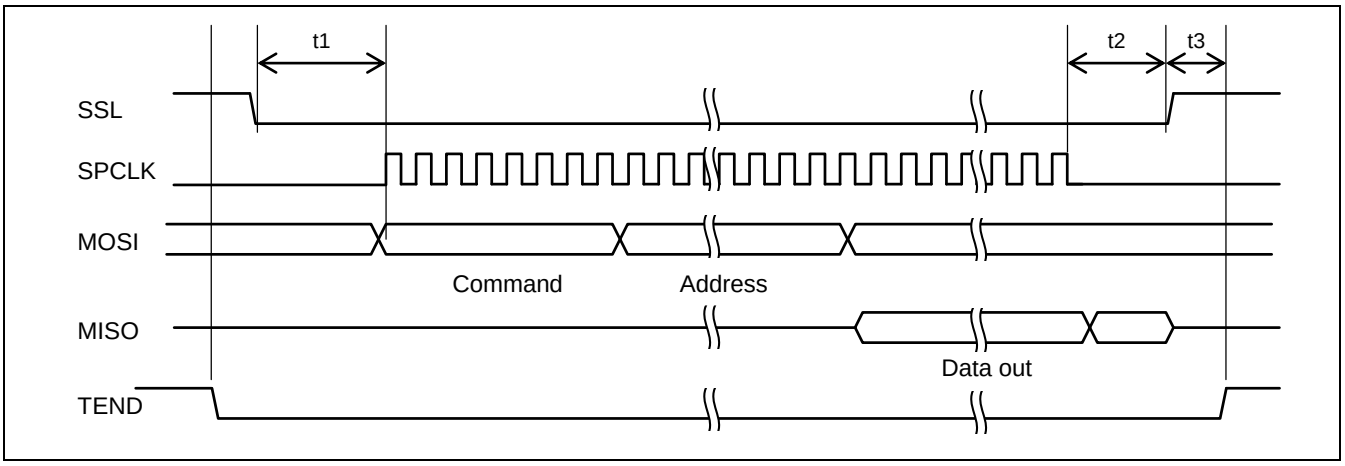



**Figure 112.7 Normal Read Operation Timing**

Note: t1 to t3 are specified in SSLDR.SCKDL, SSLDR.SLNDL and SSLDRSPNDL bits respectively.

### (2) Burst Read Operation

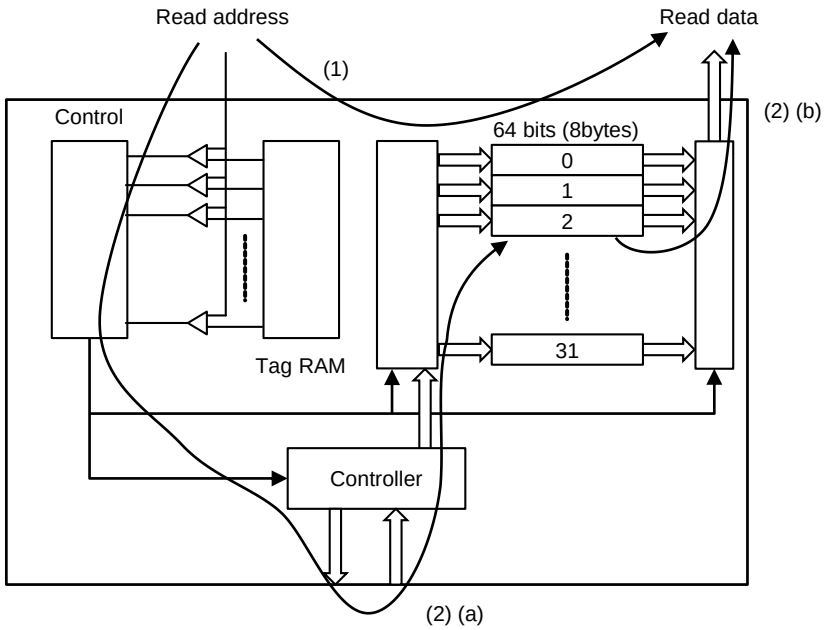
When the RBE bit in DRCR is set to 1, burst read operation is performed.

Read cache is enabled in the burst read operation. For read cache operation, see section 112.3.6, Read Cache.

For reading access from bus master, the read cache is first referred to for checking whether the data of read address of bus master is contained in the read cache memory. When the read cache memory contains the data, the data contained in the cache memory is returned to the bus master without accessing the external device. When the read cache memory does not contain the data, burst read is performed in the external device, the read data is stored in the read cache memory and the data is returned to the bus master. The data transfer length at that time is 64 bits x RBURST[4:0] bits and the data is always read from the 64-bit boundary.

The QSPIn_SSL pin status after data transfer can be selected by using the SSLE bit in DRCR. When the SSLE bit is set to 0, the QSPIn_SSL pin is always negated after data transfer. For an operation performed when the SSLE bit is set to 1, see (3) Burst Read Operation with Automatic QSPIn_SSL Negation, just below.

A pattern diagram of this operation and a burst read operation timing diagram with serial flash when SSLE bit is set to 0 are shown in figures 112.8 and 112.9.




(1) When the read cache contains data;
Data is read from the read cache, and returned to the bus master.
The external device is not accessed.
(2) When the read cache does not contain data;
(a) Data is read from the external device, and stored in the read cache.
(b) Data is read from the read cache, and returned to the bus master.

**Figure 112.8 Read Operation in Burst Read Operation**

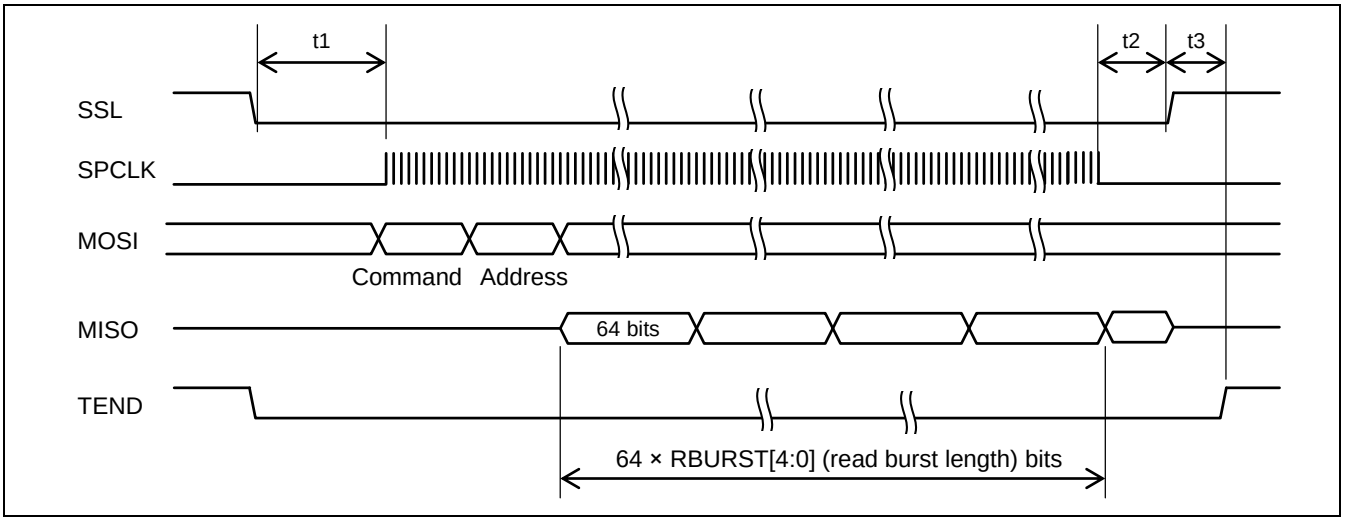



**Figure 112.9 Read Timing in Burst Read Operation (SSLE Bit = 0)**

Note: t1 to t3 are specified in SSLDR.SCKDL, SSLDR.SLNDL and SSLDRSPNDL bits respectively.

### (3) Burst Read Operation with Automatic QSPIn_SSL Negation

When SSLE bit in DRCR is set to 1, this module does not negate the QSPIn_SSL pin after the burst read transfer on serial flash connection. When accessing the next time, if the address is continuous with the previous read address, the burst read operation is performed without issuing the command, optional command, address, option data, or dummy cycle. If the address is not continuous with the previous read address, the QSPIn_SSL pin is once negated and the burst read operation is performed after issuing the command, optional command, address, option data, or dummy cycle.

Burst read timing diagrams for continuous address and non-continuous address with serial flash are shown in figures 112.10 and 112.11.

For the next access after negation of the QSPIn_SSL with the SSLN bit in DRCR with this operation, read SSLF = 0 in CMNSR to confirm that the QSPIn_SSL has been negated. This function is only supported to connect to serial flash memory.

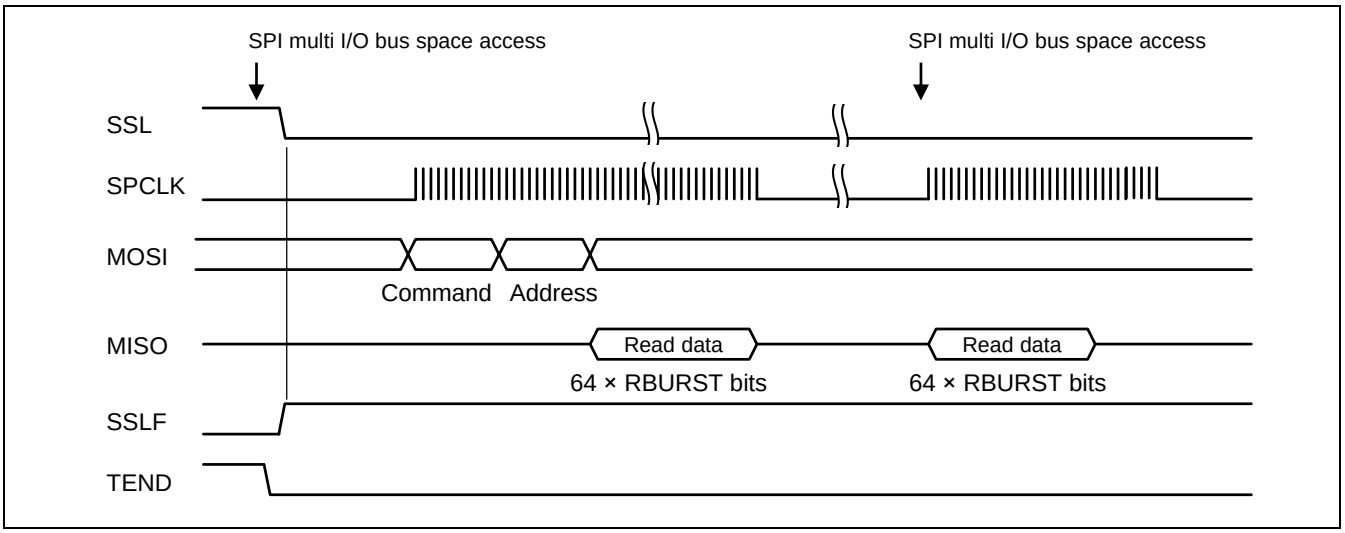



**Figure 112.10 Burst Read Timing for Continuous Address (SSLE Bit = 1)**

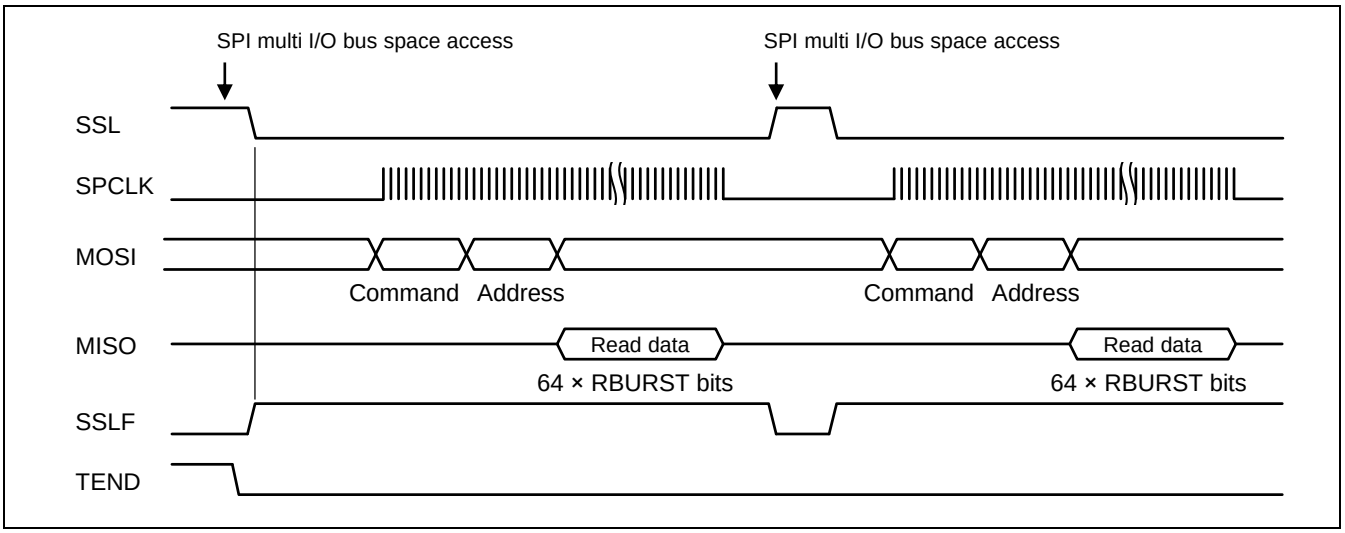



**Figure 112.11 Burst Read Timing for Non-Continuous Address (SSLE Bit = 1)**

### (4) Initial Setting Flow

An example of an initial setting flow in external address space read mode is shown in figure 112.12.

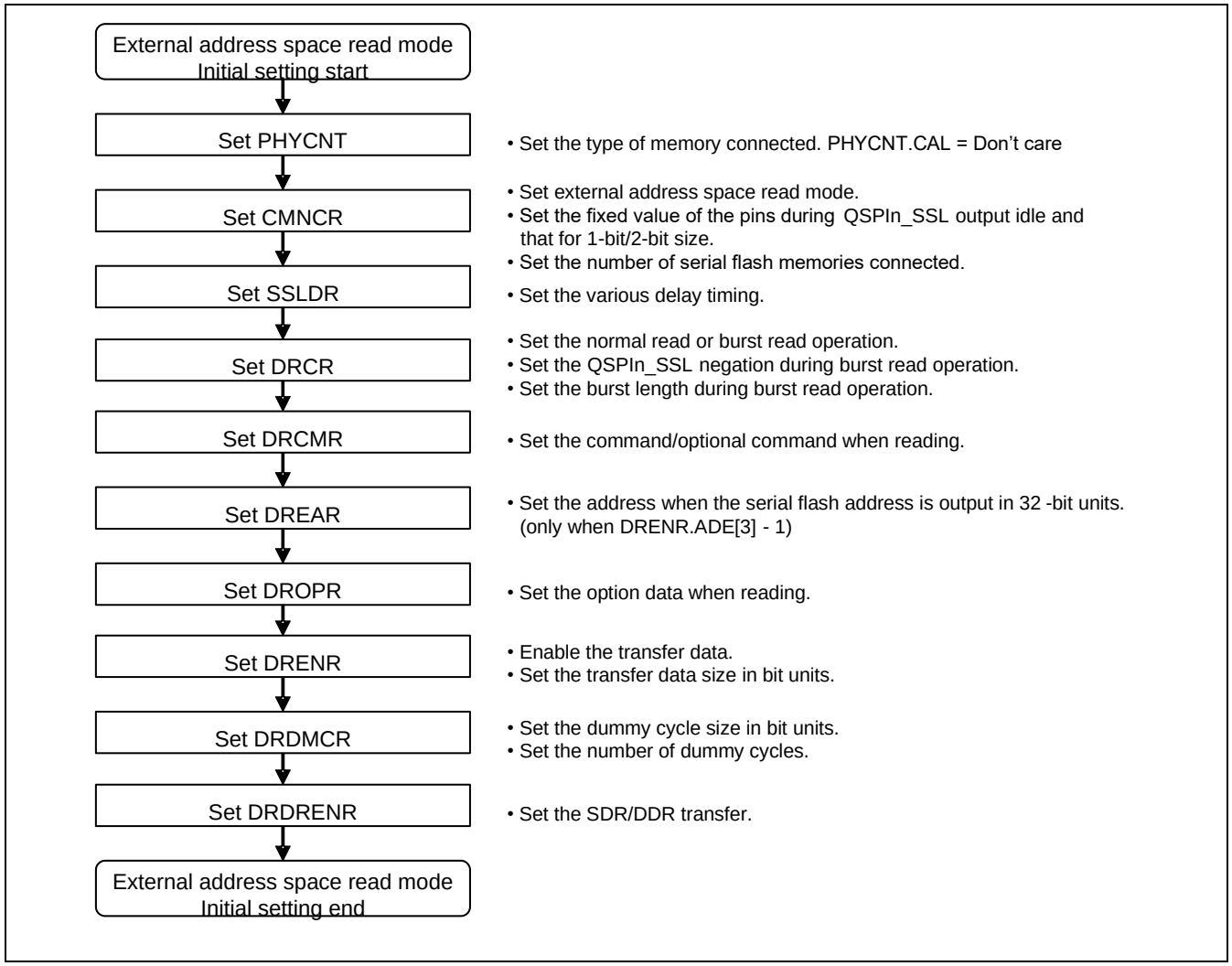



**Figure 112.12 Example of Initial Setting Flow in External Address Space Read Mode**

### 112.3.6 Read Cache

This module has a simple built-in read cache. The read cache can be used during external address space read mode and burst read operation. The read cache is configured with a line size of 64 bits and 32 entries.

Read cache configuration is shown in figure 112.13.

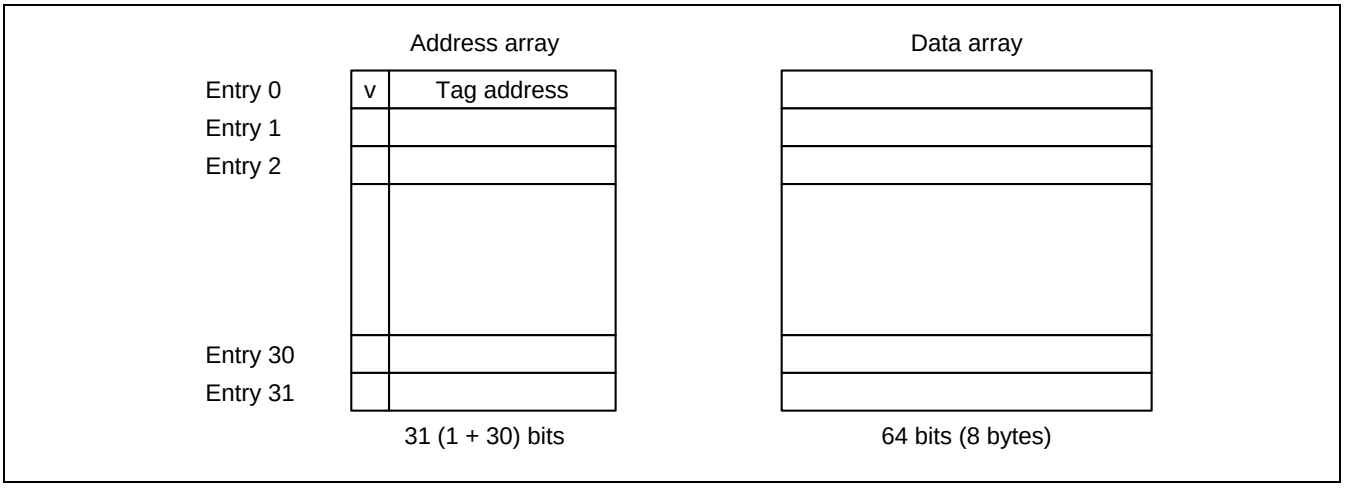



**Figure 112.13 Read Cache Configuration**

#### (1) Address Array

The V bit indicates whether the entry data is valid. When the V bit is 1, the data is valid and when V bit is 0, the data is invalid.

The tag address bits hold the address used for the serial flash memory. Address bits 32 to 3 are used for the purpose. Address bits 23 to 3 are enabled when address output is 24 bits and one serial flash memory is connected; and address bits 24 to 3 are enabled when two serial flash memory devices are connected.

Address bits 31 to 3 are enabled when address output is 32 bits and one serial flash memory is connected; and address bits 32 to 3 are enabled when two serial flash memory devices are connected.

#### (2) Data Array

It retains the 64-bit read data. Registration in the read cache is performed in line units.

#### (3) Read Operation

In case of read-hit, data is read from the read cache. In case of read-miss, after the 64 × RBURST (read burst length) data is read from the serial flash memory and the read cache is updated, the data is returned to the bus master.

#### (4) Data Replacement

The write pointer is used to update data. In case of read-miss, the RBURST (read burst length) portion data is replaced starting at the entry specified by the write pointer. In other words, the data is replaced in the storage order of the data. Whether data is referred to or not will not affect the replacement order of data.

### 112.3.7 Manual Mode

This module can carry out an arbitrary serial transfer operation by using the register settings.

The transfer format is determined based on the common control register (CMNCR), SSL delay register (SSLDR), manual mode control register (SMCR), manual mode command setting register (SMCMR), manual mode address setting register (SMADR), manual mode option setting register (SMOPR), and manual mode enable setting register (SMENR), manual mode read data register (SMRDR), manual mode write data register (SMWDR), manual mode dummy cycle setting register (SMDMCR), and manual mode DDR enable register (SMDRENR).

Manual mode can be used for reading the status of the serial flash memory and writing to the serial flash memory.

In this mode, one transfer refers to the operation from when the SPIE bit in SMCR is set to 1 to when the TEND bit is set to 1.

#### (1) Transfer Start

The transfer of data is started in the set transfer format by setting the SPIE bit in SMCR to 1. When write operation is enabled, the manual mode write data register is transmitted to the serial flash memory. When read operation is enabled, data read from the serial flash memory is stored into the manual mode read data register.

The operation timing of serial transfer is shown in figure 112.14.

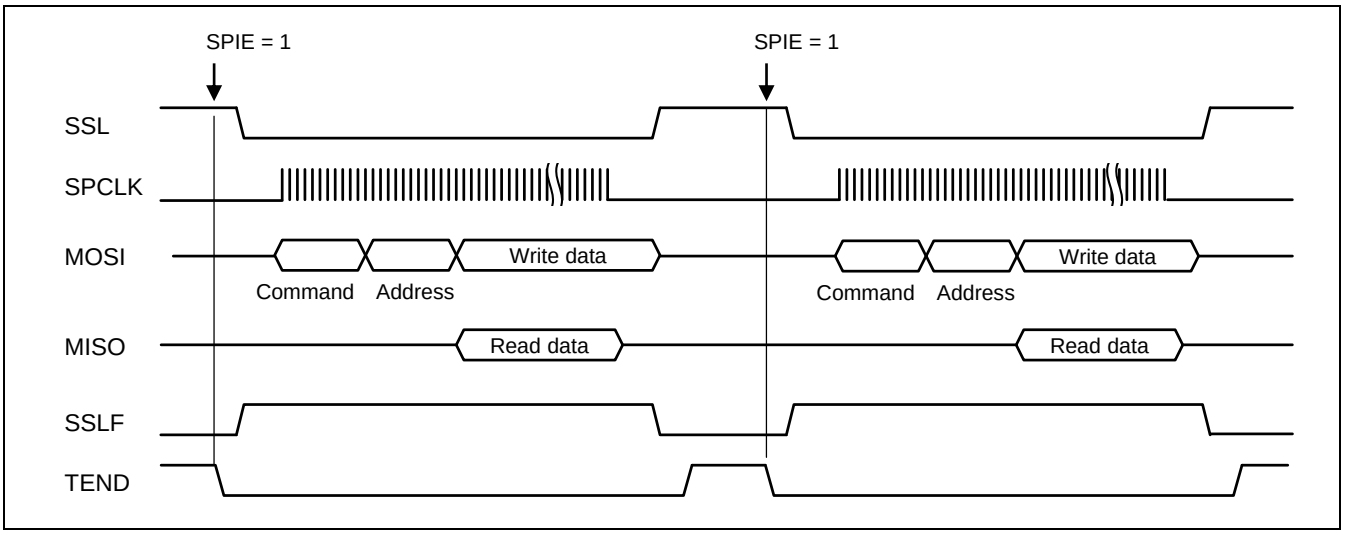



**Figure 112.14 Operation Timing of Serial Transfer**

#### (2) Read/Write Enable

Read operation: Data can be read by setting the SPIRE bit in SMCR to 1. The read data is stored into SMRDR.

Write operation: Data can be written by setting the SPIWE bit in SMCR to 1. The data stored in SMWDR is output.

When the data size is set to 1 bit using the SPIDB[1:0] bits in SMENR, data can be transmitted and received by setting the SPIRE and SPIWE bits to 1. However, when the data size is set to 4 bits by using the SPIDB[1:0] bits, only one of the SPIRE and SPIWE bits should be enabled. The operation is not guaranteed if both the bits are enabled.

### (3) Retention of QSPIn_SSL Pin Assertion

By setting the SSLKP bit in SMCR to 1, assertion of the QSPIn_SSL pin can be continued till the next transfer. With this function, the transfer can be carried out continuously with the QSPIn_SSL kept in the asserted state.

The data transfer timing using the SSLKP bit is shown in figure 112.15.

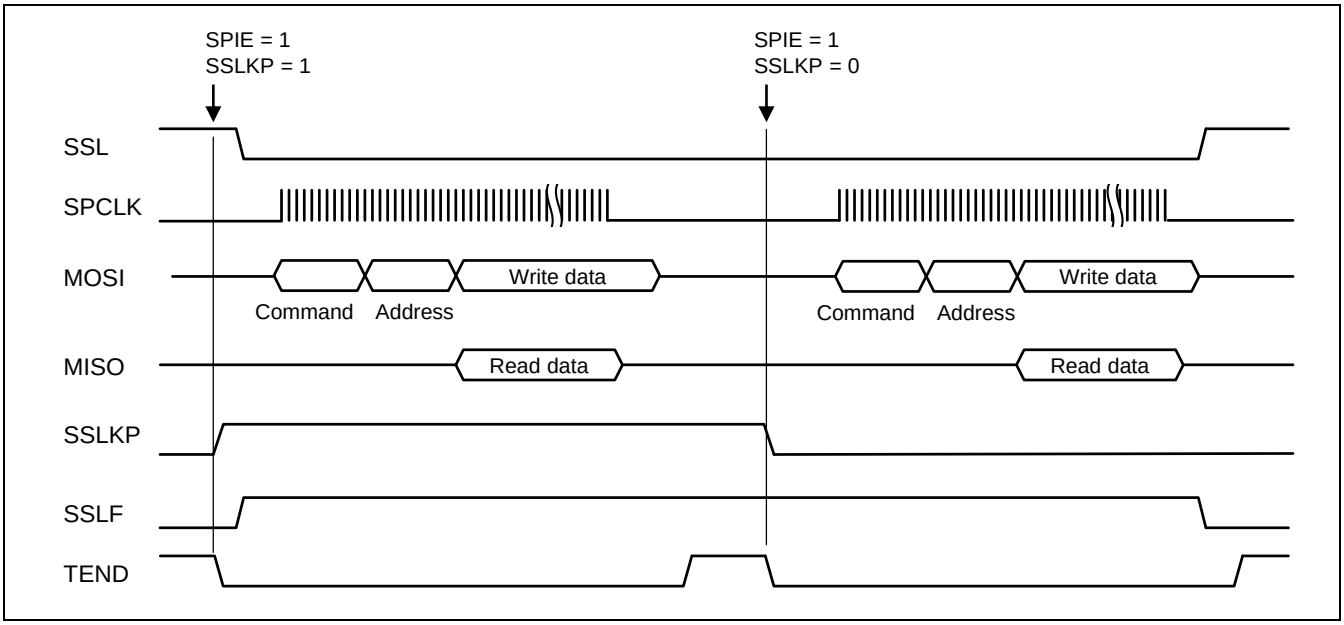



**Figure 112.15 Data Transfer Timing Chart Using the SSLKP Bit**

### (4) Initial Setting Flow

An example of an initial setting flow in manual mode is shown in figure 112.16.

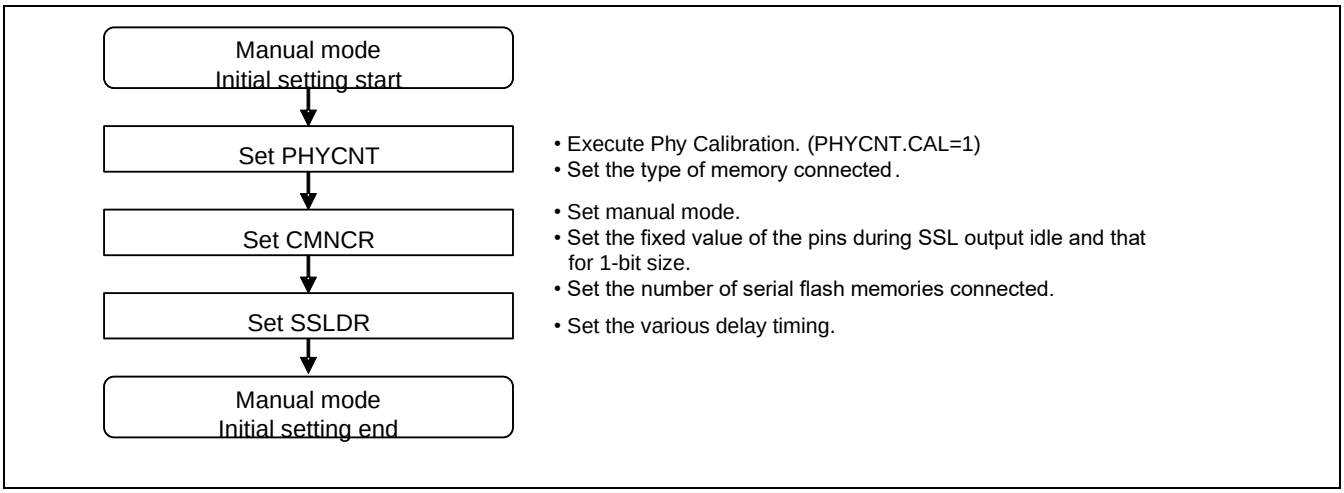



**Figure 112.16 Example of Initial Setting Flow in Manual Operating Mode**

### (5) Data Transfer Setting Flow

An example of a data transfer setting flow in manual mode is shown in figure 112.17.

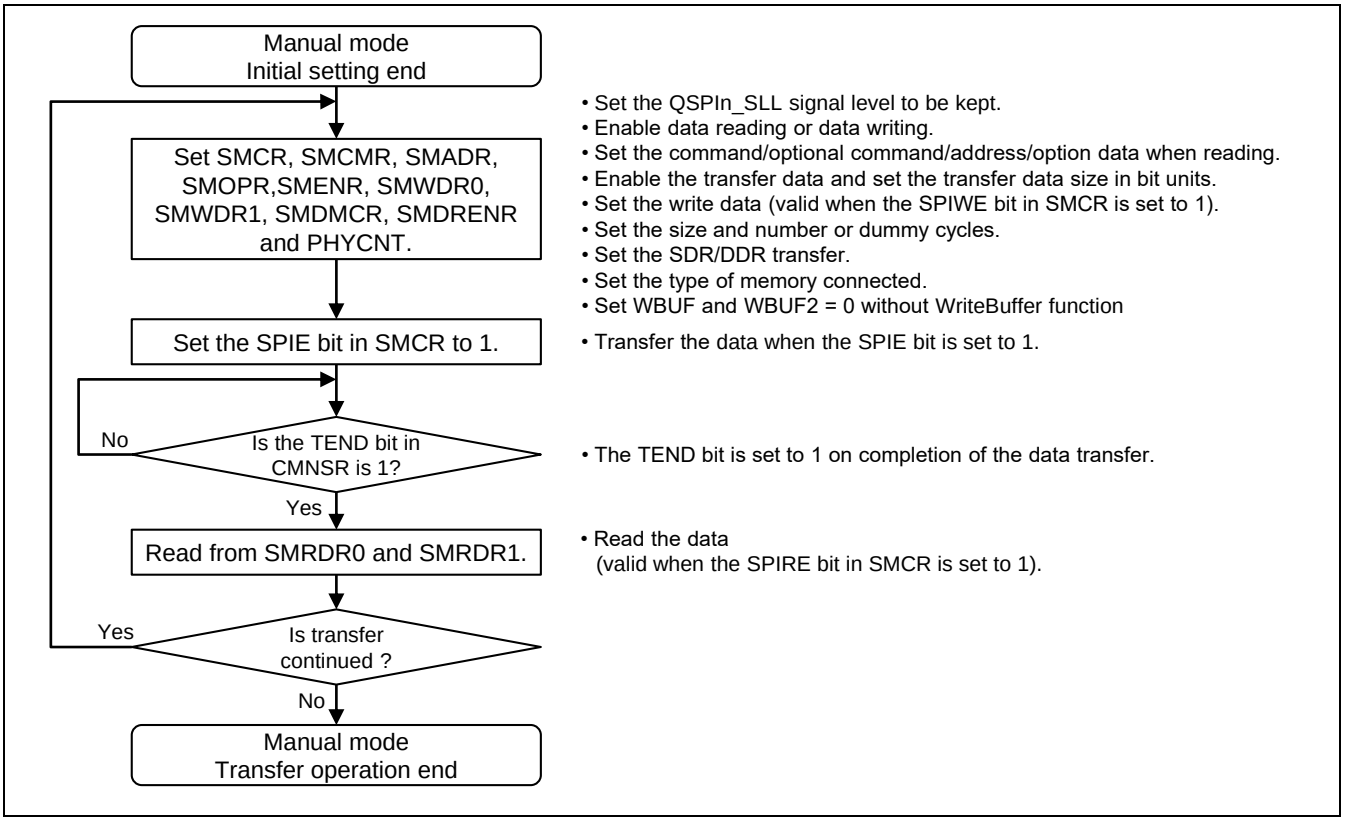



**Figure 112.17 Example of Data Transfer Setting Flow in Manual Operating Mode**

### 112.3.8 Command Sequence

This module handles input and output data in the order of command, optional command, address, option data, dummy cycle, and data.

#### (1) Data Registers

Table 112.5 and Table 112.6 shows the input and output data.

**Table 112.5 Data Registers (serial flash)**

| Data | | External Address Space Read Operation | Manual Operation |
|---|---|---|---|
| Command (8 bits) | | CMD[7:0] bits in DRCMR | CMD[7:0] bits in SMCMR |
| Optional Command (8 bits) | | OCMD[7:0] bits in DRCMR | OCMD[7:0] bits in SMCMR |
| Address (32/24 bits) | BSZ[1:0] = 00 (one serial flash memory connected) | 32 bits: DREAR.EAV[7:1 to 0] bits + lower [25 to 24:0] bits of the read address.<br>24 bits: Lower [23:0] bits of the read address | 32 bits: ADR[31:0] bits in SMADR.<br>24 bits: ADR[23:0] bits in SMADR |
| | BSZ[1:0] = 01 (two serial flash memory devices connected) | 32bits: DREAR.EAV[7:1 to 0] bits + lower [25 to 24:1] bits of the read address<br>24bits: Lower [24:1] bits of the read address | |
| Option data (8 bits × 4) | | DROPR | SMOPR |
| Dummy cycle (1 to 20 cycles) | | DRDMC | SMDMC register (only when read) |
| Transfer Data | | Normal read: 8, 16, and 32 bits,<br>Burst read: 64 x RBURST bits | Read: SMRDR0 and SMRDR1<br>Write: SMWDR0 and SMRDR1 |

**Table 112.6 Data Registers (HyperFlash)**

| Data | External Address Space Read Operation | Manual Operation |
|---|---|---|
| CA0 (47-40) | CMD[7:5] bits in DRCMR + [31:27] bits of the read address | CMD[7:5] bits in SMCMR + ADR[31:27] bits in SMADR |
| CA0 (39-32) | [26:19] bits of the read address | ADR[26:19] bits in SMADR |
| CA0 (31-24) | [18:11] bits of the read address | ADR[18:11] bits in SMADR |
| CA0 (23-16) | [10:3] bits of the read address | ADR[10:3] bits in SMADR |
| CA0 (15-8) | DROPR[15:8] | SMOPR[15:8] |
| CA0 (7-0) | DROPR[7:3] + [2:0] bits of the read address | SMOPR[7:3] + ADR[2:0] bits in SMADR |
| Dummy cycle (1 to 20 cycles) | DRDMC | SMDMC register |
| Transfer Data | Normal read: 8, 16, and 32 bits,<br>Burst read: 64 x RBURST bits | Read: SMRDR0 and SMRDR1<br>Write: SMWDR0 and SMRDR1 |

**Table 112.6a Data Registers (Octal-SPI flash with 8-8-8 protocol)**

| Data | External Address Space Read Operation | Manual Operation |
|---|---|---|
| Command (16 bits) | CMD[7:0] bits in DRCMR<br>(Rise edge of command phase) | CMD[7:0] bits in SMCMR<br>(Rise edge of command phase) |
| | OCMD[7:0] bits in DRCMR | OCMD[7:0] bits in SMCMR |
| | (Fall edge of command phase) | (Fall edge of command phase) |
| Address<br>(32/24 bits) | 32 bits: DREAR.EAV[7:1 to 0] bits + lower [25 to 24:0] bits of the read address. | 32 bits: ADR[31:0] bits in SMADR. |
| | 24 bits: Lower [23:0] bits of the read address | 24 bits: ADR[23:0] bits in SMADR |
| | Note:External address mode is available at PHYCNT.OCTA=2'b10 or 11 only.<br>(PHYCNT.OCTA=2'b01 setting is prohibited) | |
| Option data (8 bits × 4) | DROPR | SMOPR |
| Dummy cycle (1 to 20 cycles) | DRDMC | SMDMC register (only when read) |
| Transfer Data | Normal read: 8, 16, and 32 bits, | Read: SMRDR0 and SMRDR1 |
| | Burst read: 64 x RBURST bits | Write: SMWDR0 and SMRDR1 |

**(2) Data Enable**

In external address space read mode, transfer enable or disable of the command, optional command, address, option data, and dummy cycle can be controlled with the CDE, OCDE, ADE[3:0], OPDE[3:0], and DME bits in DRENR, respectively. The size and number of dummy cycles can be controlled with the data read dummy cycle setting register (DRDMCR).

Similarly, in manual mode, enable or disable of the command, optional command, address, option data, dummy cycle, and transfer data can be controlled with the CDE, OCDE, ADE[3:0], OPDE[3:0], DME, and SPIDE[3:0] bits in SMENR, respectively. However, disabling all the above parameters is prohibited in manual mode. At least one of them except dummy cycle must be enabled. The size and number of dummy cycles can be controlled with the manual mode dummy cycle setting register (SMDMCR).

For the address and option data in external address space read mode; and the address, option data, and transfer data in manual mode, the enable bit setting allowed is determined according to the transfer data size. For the allowed setting combinations of the enable bits and transfer data size, refer to the description of the pertinent register.

If data is disabled, that data is skipped, and input and output of the next data is carried out. The command, optional command, address, and option data are always output. During dummy cycles, the state of the used pins is Hi-Z. In external address space read mode, data is always input; and in manual mode, input and output of data is determined based on the settings of the SPIRE and SPIWE bits in SMCR.

There are some restrictions on dummy cycle insertion. Refer to the description of the DME bits in DRENR and SMENR for details.

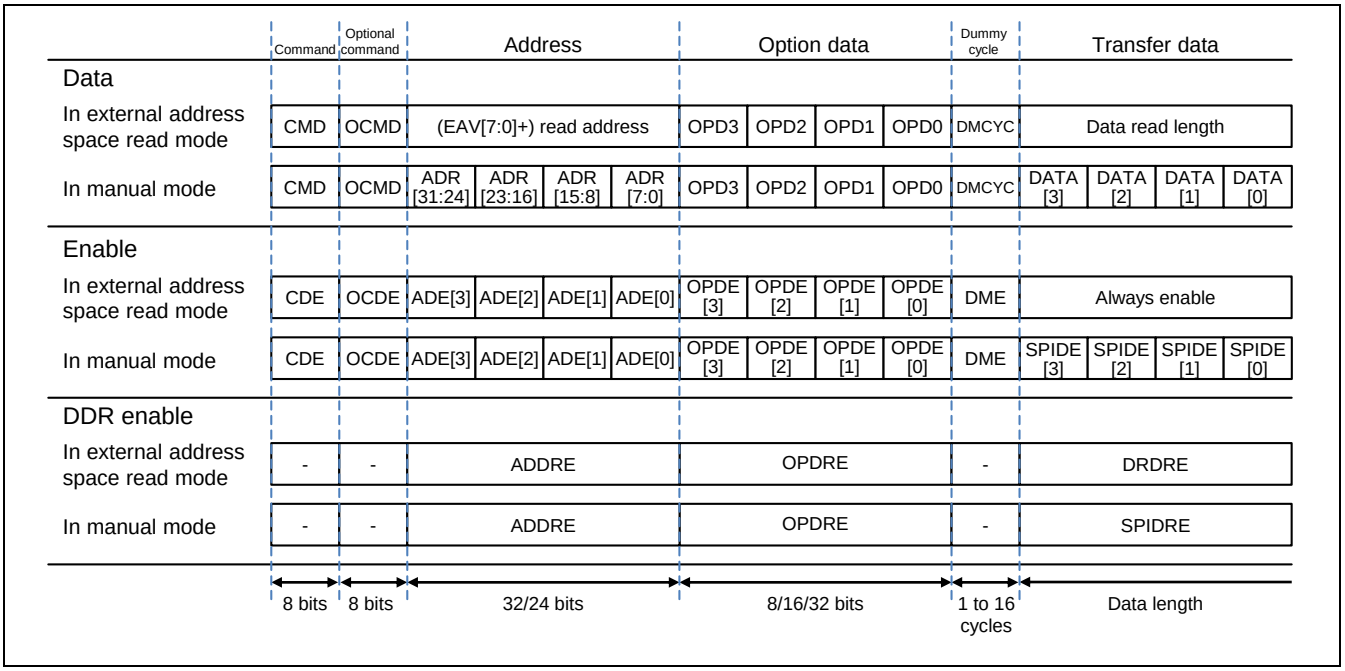



**Figure 112.18 The relationship between Data and Enable**

When HyperFlash is connected to this LSI, specify enable bits in CDE, OCDE, ADE[3:0], OPDE[3:0], and DME bits in DRENR and HYPE[2:0], ADDRE, OPDRE and DRDRE in DRDRENR in external address space read mode, and CDE, OCDE, ADE[3:0], OPDE[3:0],DME and SPIDE[3:0] in SMENR and HYPE[2:0], ADDRE, OPDRE and SPIDRE in SMDRENR manual mode.

**Table 112.7 Enable Registers (HyperFlash)**

| **External Address Space Read Operation** | | | **Manual Operation** | | |
|---|---|---|---|---|---|
| Register | Bits | Setting Value | Register | Bits | Setting Value |
| DRENR | CDE | 1 | SMENR | CDE | 1 |
| | OCDE | 1 | | OCDE | 1 |
| | ADE[3:0] | 0100 | | ADE[3:0] | 0100 |
| | OPDE[3:0] | 0000 | | OPDE[3:0] | 0000 |
| | DME | 1 | | DME | Read: 1<br>Write: 0 |
| DRDRENR | HYPE[2:0] | 101 | | SPIDE[3:0] | 16bit: 1000<br>32bit: 1100<br>64bit: 1111 |
| | ADDRE | 1 | SMDRENR | HYPE[2:0] | 101 |
| | OPDRE | 0 | | ADDRE | 1 |
| | DRDRE | 1 | | OPDRE | 0 |
| | | | | SPIDRE | 1 |

**Table 112.7a Enable Registers (Octal-SPI flash with 8-8-8 protocol)**

| **External Address Space Read Operation** | | | **Manual Operation** | | |
|---|---|---|---|---|---|
| Register | Bits | Setting Value | Register | Bits | Setting Value |
| DRENR | CDE | 1 | SMENR | CDE | 1 |
| | OCDE | 0 | | OCDE | 0 |
| | ADE[3:0] | 1100 | | ADE[3:0] | 1100 |
| | OPDE[3:0] | 0000 | | OPDE[3:0] | 0000 |
| | DME | 1 | | DME | Read: 1<br>Write: 0 |
| DRDRENR | HYPE[2:0] | 101 | | SPIDE[3:0] | 16bit: 1000<br>32bit: 1100<br>64bit: 1111 |
| | ADDRE | 1 | SMDRENR | HYPE[2:0] | 101 |
| | OPDRE | 0 | | ADDRE | 1 |
| | DRDRE | 1 | | OPDRE | 0 |
| | | | | SPIDRE | 1 |

**(3) Bit Size**

In external address space read mode, the size of the command, optional command, address, option data, and the read data in bit units is respectively controlled with the CDB[1:0], OCDB[1:0], ADB[1:0], OPDB[1:0], and DRDB[1:0] bits in DRENR. The size of the dummy cycle in bit units is also controlled with the DMDB[1:0] bits in DRDMCR.

Similarly, in manual mode, the size of the command, optional command, address, option data, and read write data in bit units is controlled with the CDB[1:0], OCDB[1:0], ADB[1:0], OPDB[1:0], and SPIDB[1:0] bits in SMENR. The size of the dummy cycle in bit units is also controlled with the DMDB[1:0] bits in SMDMCR.

**(a) 1-bit Size**

When the size is set to 1 bit, QSPI0_MISO_IO1 and QSPI1_MISO_IO1 pins will be the input pins and QSPI0_MOSI_IO0 and QSPI1_MOSI_IO0 pins will be the output pins. QSPI0_IO2, QSPI1_IO2, QSPI0_IO3, and QSPI1_IO3 pins are not used.

Figures 112.19 and 112.20 show the examples of the transfer format.

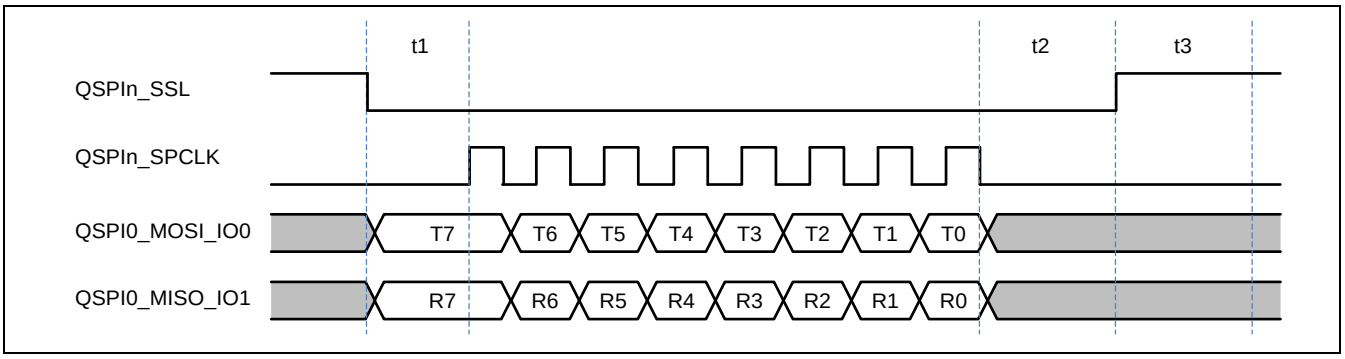



**Figure 112.19 Example of Transfer Format with 1-Bit Data Size and One Serial Flash Memory Connected**

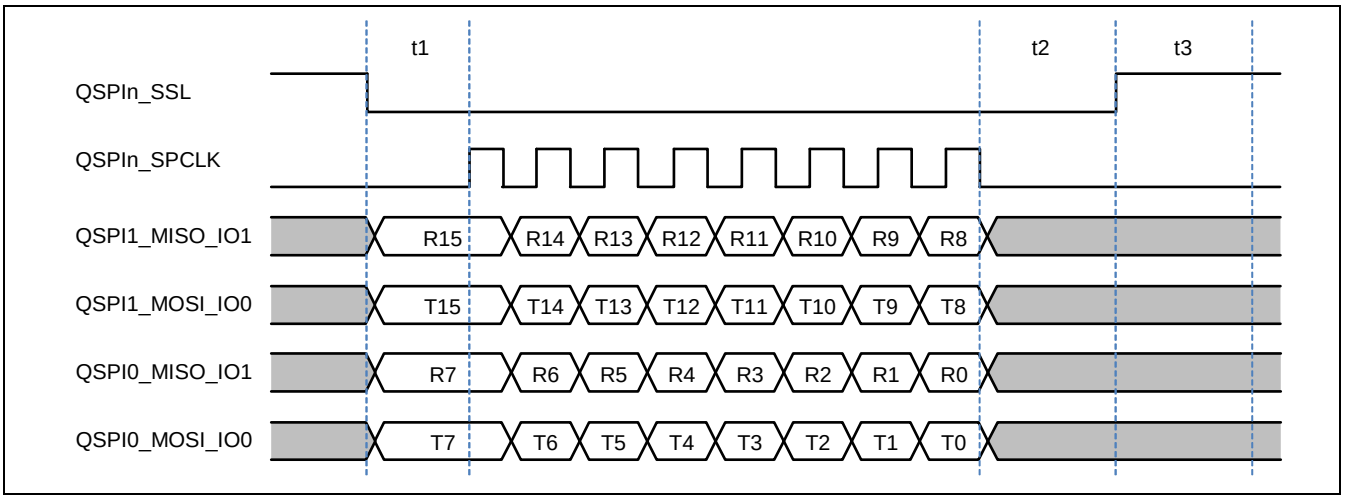



**Figure 112.20 Example of Transfer Format with 1-Bit Data Size and Two Serial Flash Memory Devices Connected**

**(b) 4-bit Size (serial flash)**

When the size is set to 4-bit, QSPI0_MOSI_IO0, QSPI1_MOSI_IO0, QSPI0_MISO_IO1, QSPI1_MISO_IO1, QSPI0_IO2, QSPI1_IO2, QSPI0_IO3, and QSPI1_IO3 pins will be either the input pins or the output pins. Figures 112.23 and 112.24 show the examples of the transfer format.

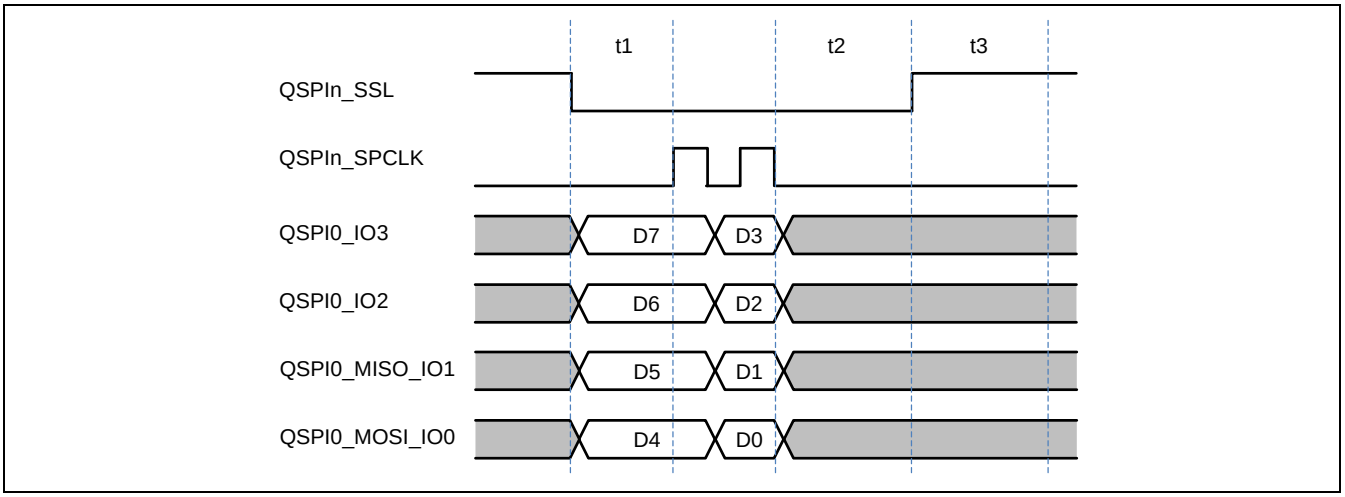



**Figure 112.23 Example of Transfer Format with 4-Bit Data Size and One Serial Flash Memory Connected**

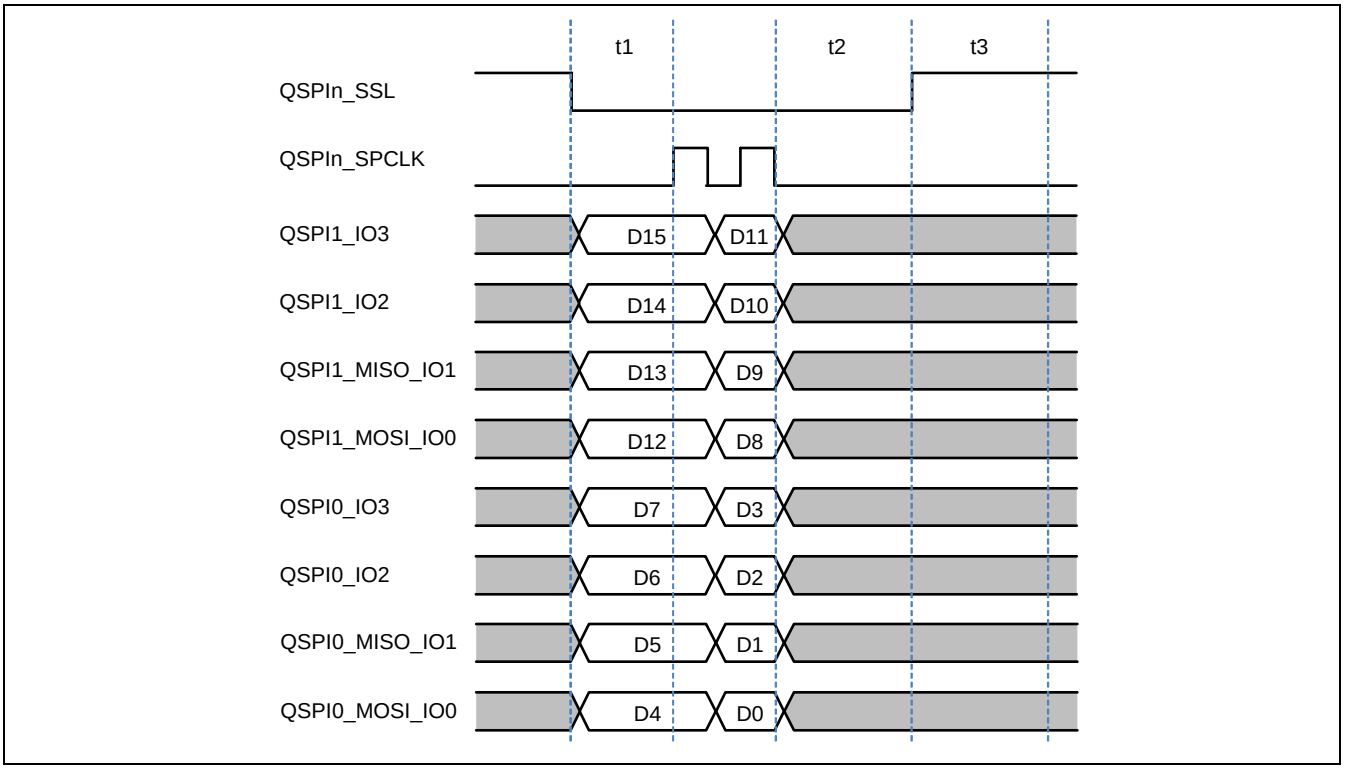



**Figure 112.24 Example of Transfer Format with 4-Bit Data Size and Two Serial Flash Memory Devices Connected**

**(c) 4-bit Size (Hyperflash)**

When HyperFlash is connected to this LSI, QSPI0_SPCLK and QSPI1_SPCLK will be differential clock, QSPI1_SSL will be input pin for RDS (Read Data Strobe) and QSPI0_MOSI_IO0, QSPI1_MOSI_IO0, QSPI0_MISO_IO1, QSPI1_MISO_IO1, QSPI0_IO2, QSPI1_IO2, QSPI0_IO3, and QSPI1_IO3 pins will be either the input pins or the output pins. Figure 112.25 shows the command/address sequence, and Figure 112.26 shows example of read data. In manual mode, the read data alignment is sequential, {DQ3[7:0], DQ2[7:0], DQ1[7:0], DQ0[7:0]} is stored in SMRDR1[31:0] and {DQ7[7:0], DQ6[7:0], DQ5[7:0], DQ4[7:0]} is stored in SMRDR0[31:0]. In external address space read mode, the read data is stored in little endian. For example, the 64-byte read data on RPC-IF area (0x08000000 ~ 0x0BFFFFFF) is {DQ7[31:0], DQ6[7:0], DQ5[7:0], DQ4[7:0], DQ3[7:0], DQ2[7:0], DQ1[7:0], DQ0[7:0]}.
Figure 112.27a shows example of write data(64bit). In manual mode, the write data alignment is sequential, the data in SMWDR1[7:0] is on DQ0, SMWDR1[15:8] is on DQ1, …, SMWDR0[31:24] is on DQ7. Figure 112.27b shows example of write data(16 bit). In manual mode, the write data alignment is sequential, the data in SMWDR0[23:16] is on DQ6, SMWDR0[31:24] is on DQ7. In Write Buffer mode, the write data alignment is little endian. The data stored in Write Buffer is output from H'EE20 8000 to H'EE20 80FF sequentially.

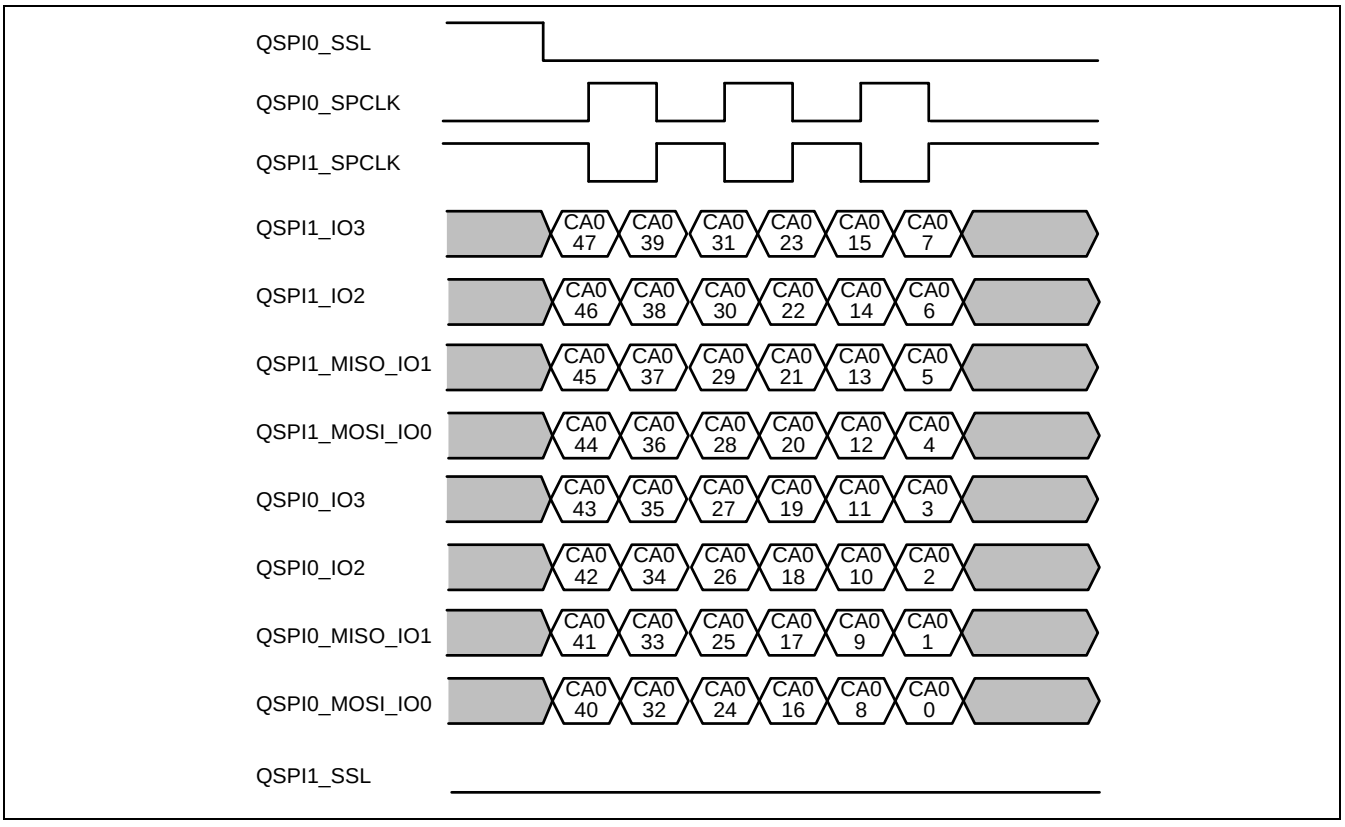



**Figure 112.25 Example of Transfer Format in Command/Address phase with HyperFlash**

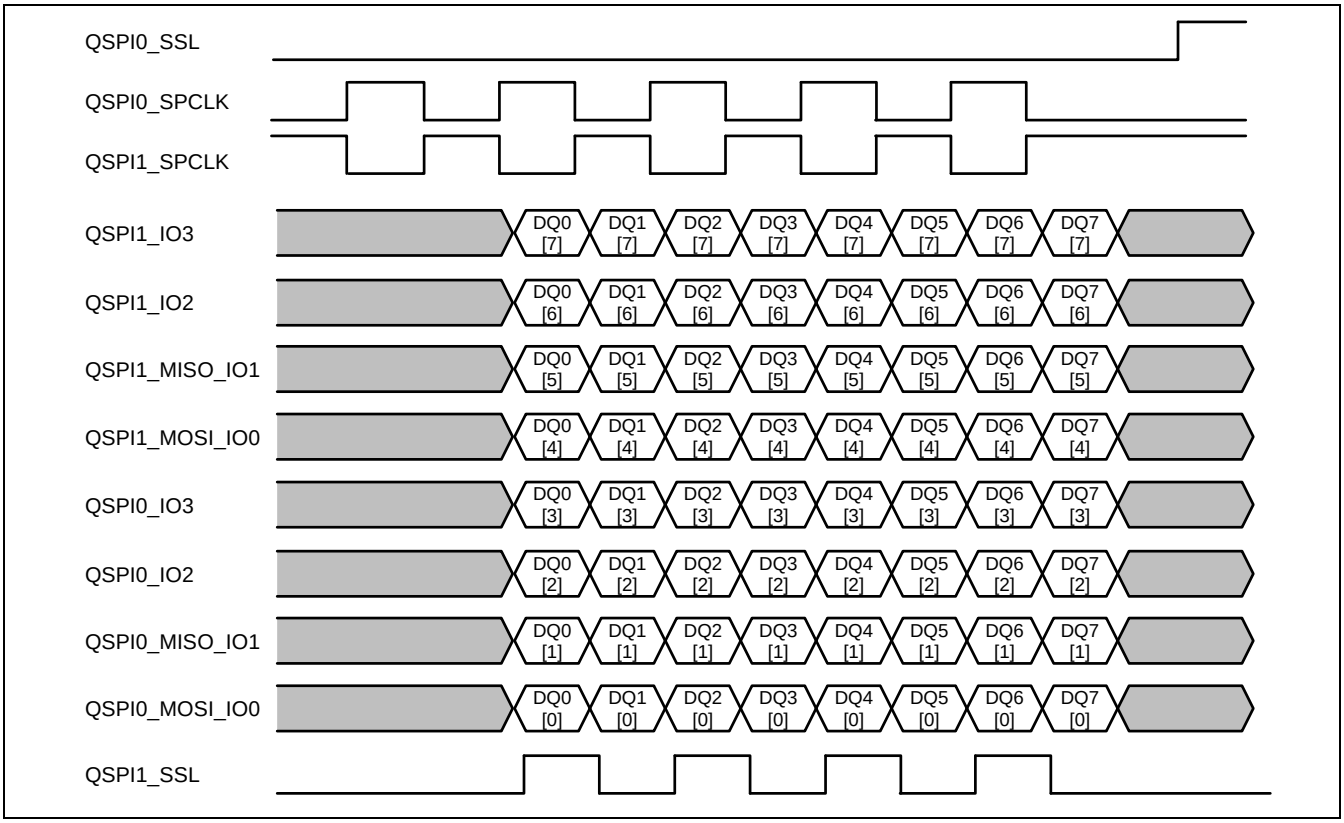



**Figure 112.26 Example of Transfer Format in Read phase with HyperFlash**

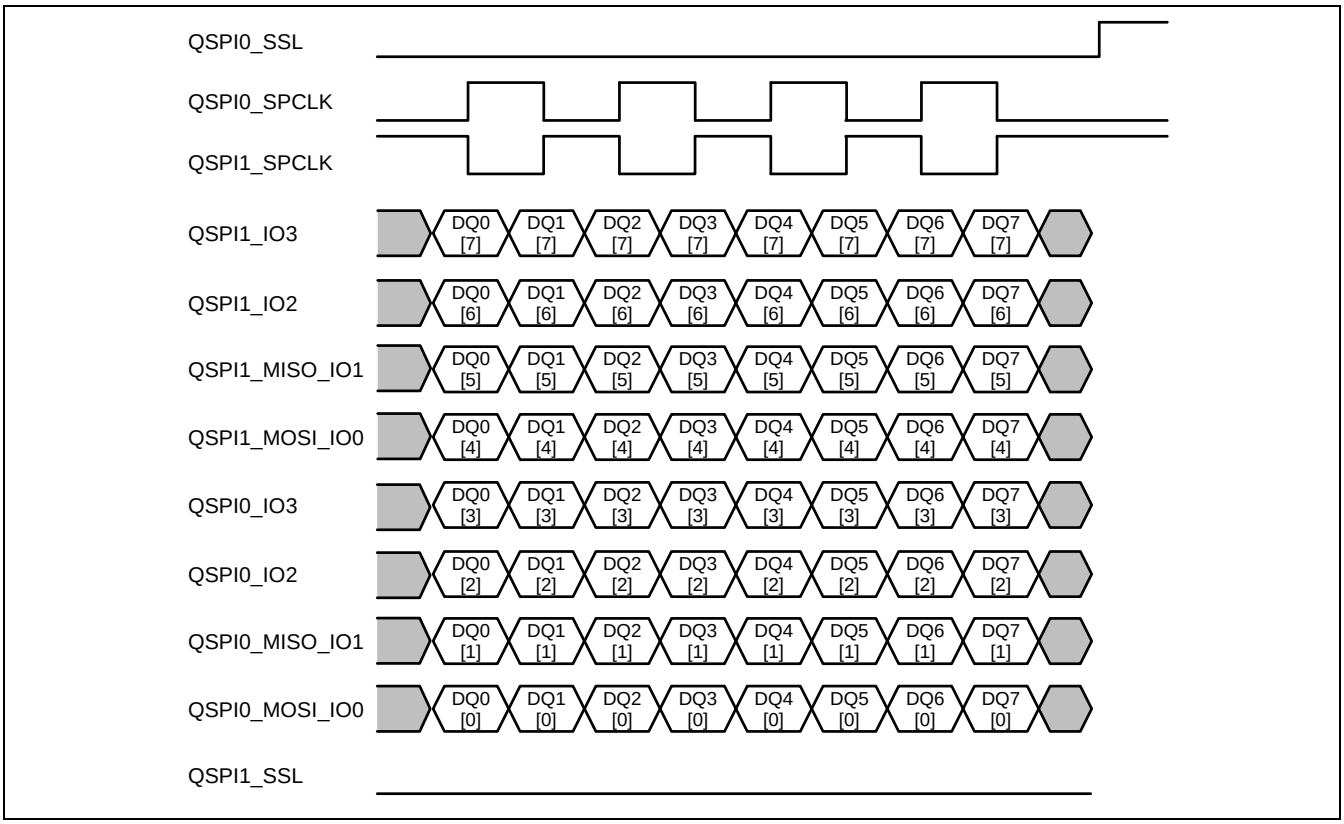



**Figure 112.27a Example of Transfer Format in Write phase with HyperFlash (64 bit)**

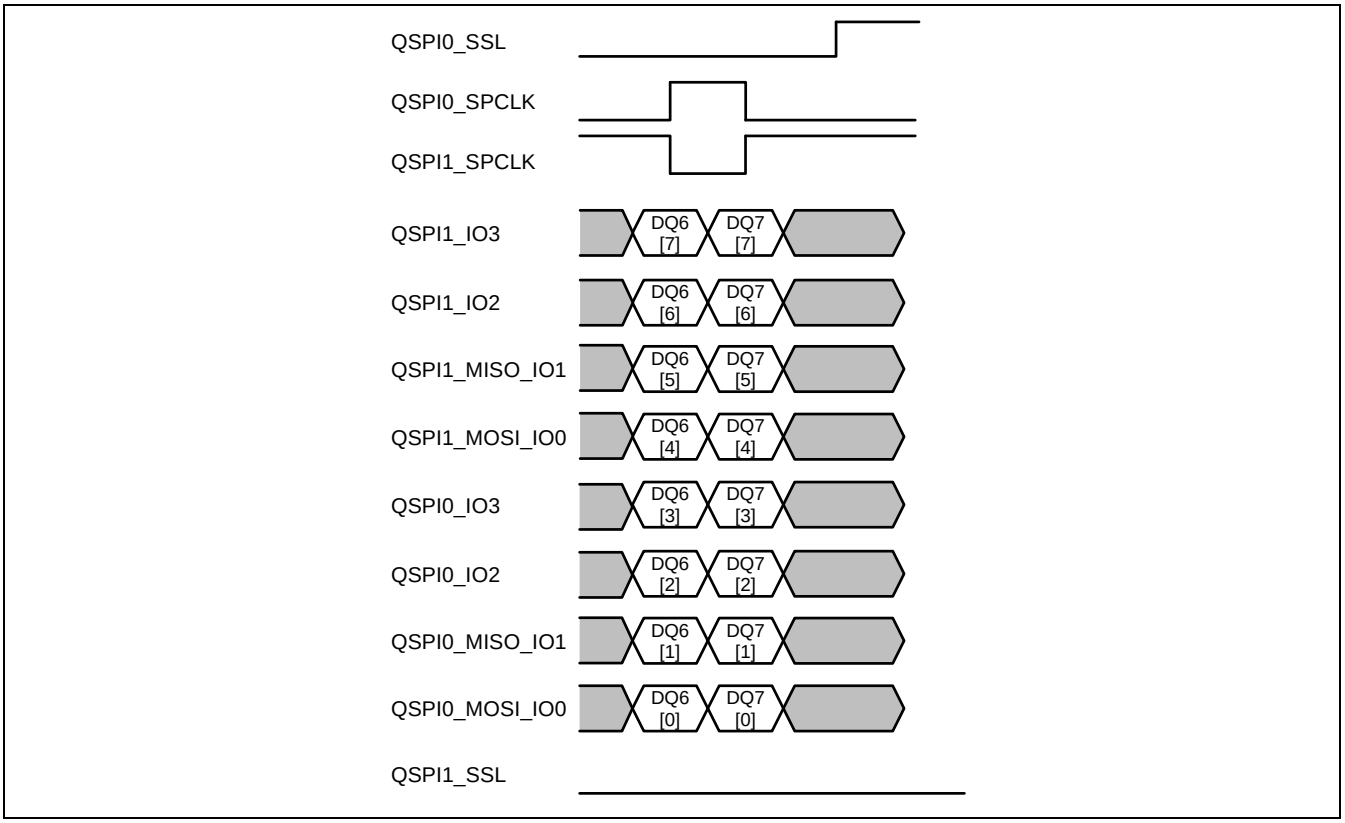



**Figure 112.27b Example of Transfer Format in Write phase with HyperFlash (16 bit)**

### 112.3.9 Data Pin Control

With this module, the status of pins can be automatically changed based on the data size to be used and the read/write settings. The pin status during the QSPIn_SSL negation can be set with the MOIIO3, MOIIO2, MOIIO1, and MOIIO0 bits in CMNCR.

The QSPI0_SSL and QSPIn_SPCLK pins are always output pins. The status of respective pins is specified in Table 112.8 to 112.11.

Note that connection with ch1 pins in 1 x serial flash configuration is prohibited.

**Table 112.8 Pin Status (1)**

| | | QSPIn_SSL Assertion | |
|---|---|---|---|
| | | Command, Optional Command, Address, Option Data | |
| Pin | QSPIn_SSL Negation | 1-bit Size | 4-bit Size *2 |
| QSPI0_MOSI_IO0, QSPI1_MOSI_IO0 | MOIIO0 bit value | Output | Output |
| QSPI0_MISO_IO1, QSPI1_MISO_IO1 | MOIIO1 bit value | Hi-Z | Output |
| QSPI0_IO2, QSPI1_IO2 | MOIIO2 bit value | specified in PFC *1 | Output |
| QSPI0_IO3, QSPI1_IO3 | MOIIO3 bit value | specified in PFC *1 | Output |

Notes: 1. QSPI0_IO2, QSPI1_IO2, QSPI0_IO3 and QSPI1_IO3 pin state should be controlled by PFC function. Refer to Section 7. Pin Function in User's Manual.
2. When CMNCR.BSZ[1:0]=01, Bit size becomes 8bits.

**Table 112.9 Pin Status (2)**

| | Transfer Data | | | |
|---|---|---|---|---|
| | External Address Space Read Operation | | Manual mode Operation | |
| | | | SPIRE Bit = 1, SPIWE Bit = 0 | |
| Pin | 1-bit Size | 4-bit Size *2 | 1-bit Size | 4-bit Size *2 |
| QSPI0_MOSI_IO0, QSPI1_MOSI_IO0 | IO0FV bit value | Input | IO0FV bit value | Input |
| QSPI0_MISO_IO1, QSPI1_MISO_IO1 | Input | Input | Input | Input |
| QSPI0_IO2, QSPI1_IO2 | specified in PFC *1 | Input | specified in PFC *1 | Input |
| QSPI0_IO3, QSPI1_IO3 | specified in PFC *1 | Input | specified in PFC *1 | Input |

Notes: 1. QSPI0_IO2, QSPI1_IO2, QSPI0_IO3 and QSPI1_IO3 pin state should be controlled by PFC function. Refer to Section 7. Pin Function in User's Manual.
2. When CMNCR.BSZ[1:0]=01, Bit size becomes 8bits.

**Table 112.10 Pin Status (3)**

| | Transfer Data | | | |
|---|---|---|---|---|
| | Manual mode Operation | | | |
| | SPIRE Bit = 0, SPIWE Bit = 1 | | SPIRE Bit = 1, SPIWE Bit = 1 | |
| Pin | 1-bit Size | 4-bit Size *2 | 1-bit Size | 4-bit Size *2 |
| QSPI0_MOSI_IO0, QSPI1_MOSI_IO0 | Output | Output | Output | Setting prohibited |
| QSPI0_MISO_IO1, QSPI1_MISO_IO1 | Hi-Z | Output | Input | Setting prohibited |
| QSPI0_IO2, QSPI1_IO2 | specified in PFC *1 | Output | specified in PFC *1 | Setting prohibited |
| QSPI0_IO3, QSPI1_IO3 | specified in PFC *1 | Output | specified in PFC *1 | Setting prohibited |

Notes: 1. QSPI0_IO2, QSPI1_IO2, QSPI0_IO3 and QSPI1_IO3 pin state should be controlled by PFC function. Refer to "7. Pin Function"
2. When CMNCR.BSZ[1:0]=01, Bit size becomes 8bits.

**Table 112.11 Pin Status (4)**

| | Dummy Cycle | |
|---|---|---|
| Pin | 1-bit Size | 4-bit Size *2 |
| QSPI0_MOSI_IO0, QSPI1_MOSI_IO0 | IO0FV bit value | Hi-Z |
| QSPI0_MISO_IO1, QSPI1_MISO_IO1 | Hi-Z | Hi-Z |
| QSPI0_IO2, QSPI1_IO2 | specified in PFC *1 | Hi-Z |
| QSPI0_IO3, QSPI1_IO3 | specified in PFC *1 | Hi-Z |

Notes: 1. QSPI0_IO2, QSPI1_IO2, QSPI0_IO3 and QSPI1_IO3 pin state should be controlled by PFC function. Refer to "7. Pin Function"
2. When CMNCR.BSZ[1:0]=01, Bit size becomes 8bits.

### 112.3.10 QSPIn_SSL Pin Control

Negation conditions of the QSPIn_SSL pin are as follows.

**(1) External Address Space Read Operation**

**(a) Normal read operation (RBE bit in DRCR = 0)**

QSPIn_SSL negated after completing the data transfer and t2 cycle.

**(b) Burst read operation without automatic QSPIn_SSL negation (RBE bit in DRCR = 1, SSLE bit in DRCR = 0)**

QSPIn_SSL negated after completing the data transfer and t2 cycle.

**(c) Burst read operation with automatic QSPIn_SSL negation (RBE bit in DRCR = 1, SSLE bit in DRCR = 1)**

- QSPIn_SSL negated after t2 cycle when the read address is not continuous with the previously read address
- QSPIn_SSL negated after the SSLN bit in DRCR is set to 1.

Note: This function is available when connecting to serial flash.

**(2) Manual Mode**

**(a) QSPIn_SSL pin assertion not retained (SSLKP bit in SMCR = 0)**

QSPIn_SSL negated after completing the data transfer.

**(b) QSPIn_SSL pin assertion retained (SSLKP bit in SMCR = 1)**

QSPIn_SSL not negated.

When to be negated, data should be transferred after setting the SSLKP bit to 0.

Note: QSPI1_SSL is used as input pin for data strobe when connecting to HyperFlash or Octal-SPI flash.

### 112.3.11 QSPIn_SPCLK Pin Control

QSPI0_SPCLK and QSPI1_SPCLK are output depending on the following setting

**Table 112.12 QSPIn_SPCLK output**

| Pin | HyperFlash (CMNCR.BSZ = 01 PHYCNT.PHYMEM = 11) | Serial flash x 1ch (CMNCR.BSZ = 00 PHYCNT.PHYMEM = 00 or 01) | Serial flash x 2ch or Octal-SPI Flash with 8-8-8 protocol (CMNCR.BSZ = 01 PHYCNT.PHYMEM = 00 or 01) |
|---|---|---|---|
| QSPI0_SPCLK | Output | Output | Output |
| QSPI1_SPCLK | Output in differential relation with QSPI0_SPCLK | Hi-Z | Output |

### 112.3.12 Flags

This module has two flag bits SSLF and TEND in CMNSR. These bits are read-only bits.

**(1) SSLF Bit**

This bit indicates the QSPIn_SSL pin status. The status is 1 when the QSPIn_SSL is asserted, and the status is 0 when the QSPIn_SSL is negated.

**(2) TEND Bit**

This bit indicates whether transfer of data is in progress or the transfer of data has ended.

During t1 time period, data transfer, t2 time period, t3 time period, and waiting for read access by burst read and QSPIn_SSL automatic negation, the TEND bit is read as 0 to indicate that the transfer of data is in progress.

When other than the above, the TEND bit is read as 1 to indicate that transfer of data has ended.

**(3) Register Re-writing Timing**

The status of the TEND bit determines the rewritable registers.

The registers which can be written to, except the SSLN bit in DRCR, should be modified when TEND = 1.

The data should be read when the TEND flag in SMRDR0 or SMRDR1 is 1;

CMNSR can always be read.

### 112.3.13 Write Buffer Operation

This module uses read cache as write buffer in write operation. This write buffer improve the performance.

**Table 112.13 Write Buffer Area**

| Address | Access Size |
|---|---|
| H'EE20 8000 to H'EE20 80FF | 4, 8, 16, 32, 64 Byte |

Note: This area should be accessed sequentially from the start address and transfer size to the device is 256-byte unit.
All Write Buffer area should be filled with transfer data in every transfer.
When accessing non-sequentially or at random, the operation is not guaranteed.
After Write Buffer Operation, this cache area must be cleared by DRCR.RCF bit.
Write Buffer operation supports 24bit or 32bit address only. SMENR.ADE should be specified 0111 (ADR[23:0]) or 1111 (ADR[31:0]).

Figure 112.28 shows the sequence to use write buffer. About the initial setting of Manual mode, refer to Figure 112.16.

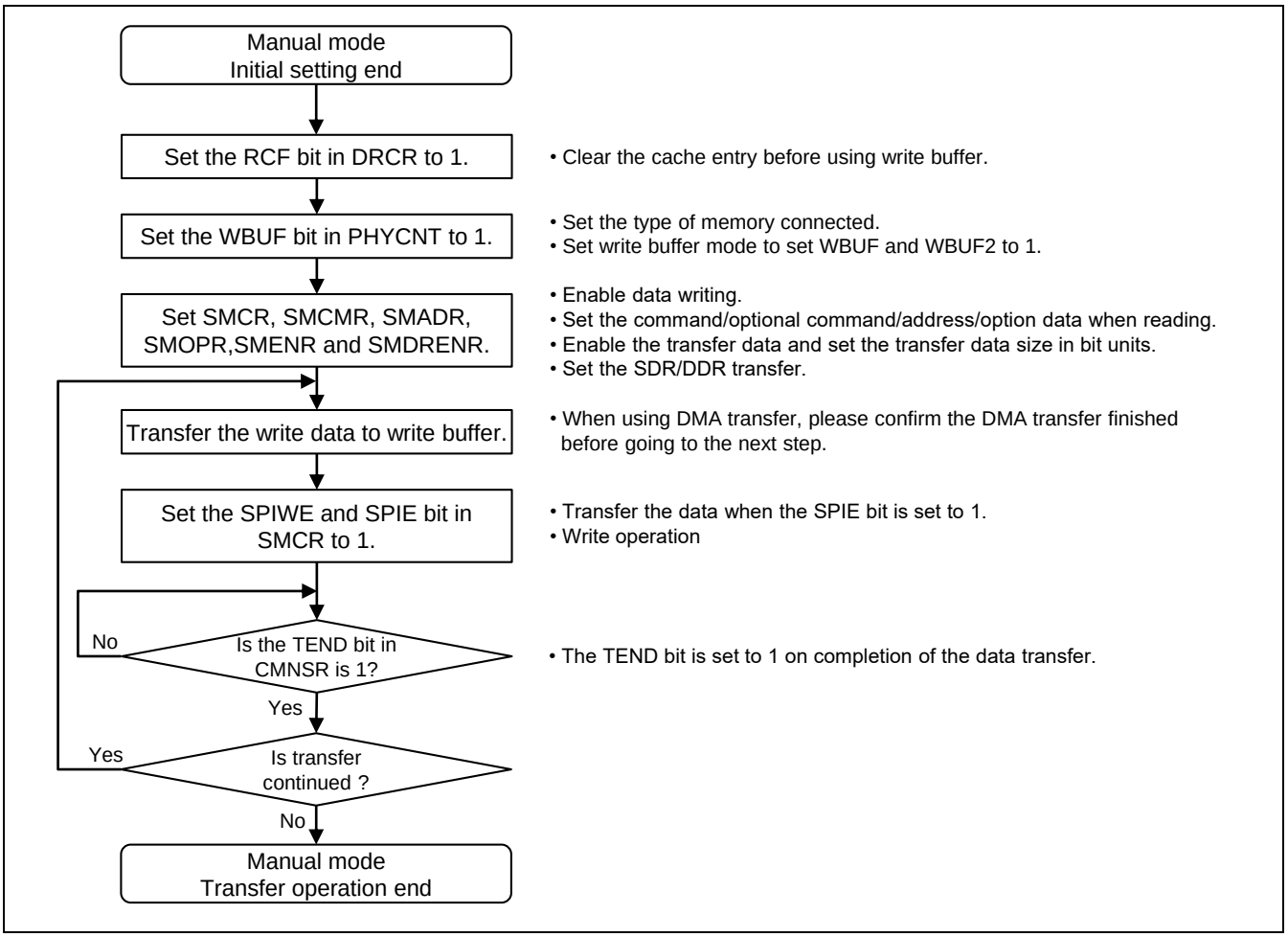



**Figure 112.28 Write Buffer Usage Sequence**

### 112.3.14 Data Alignment with Octal-SPI flash memory

When this LSI connects to Octal-SPI flash memory, this controller support two types of data alignment. Figure 112.29 and Figure 112.30 show sequential and alternate alignment respectively. Write data alignment in Write Buffer operation is same.

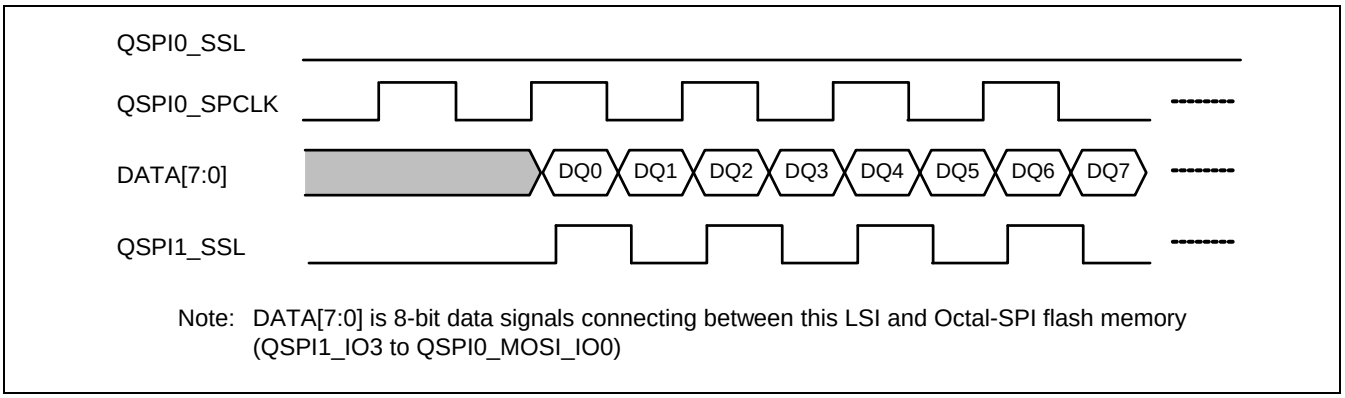



**Figure 112.29 Sequential Data Alignment with Octal-SPI flash (PHYCNT.OCTA = 10)**

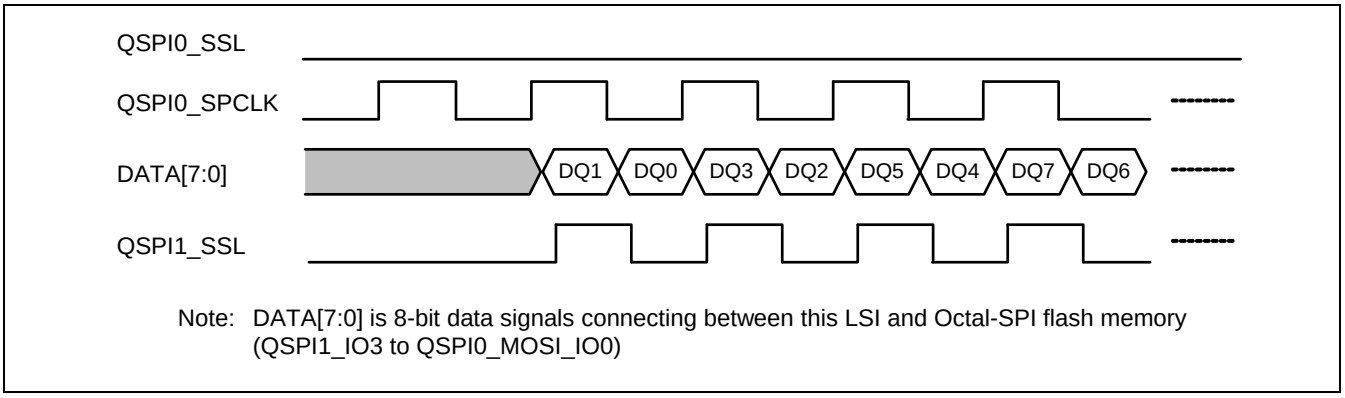
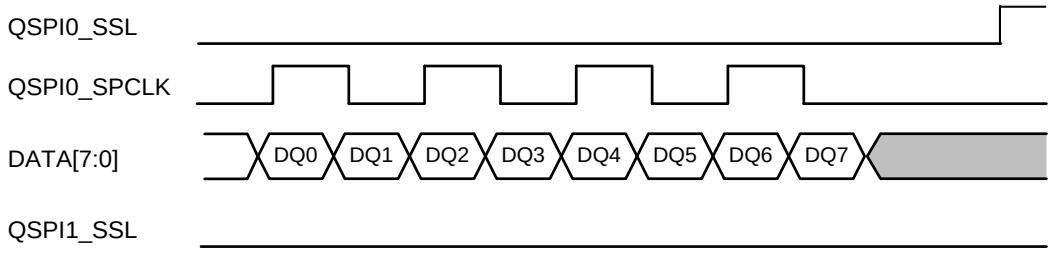



**Figure 112.30 Alternate Data Alignment with Octal-SPI flash (PHYCNT.OCTA = 01, 11)**

Figure 112.31 shows the write data alignment in 8-8-8 protocol in manual mode operation.




| | | | |
|---|---|---|---|
| DQ0[7:0] | SMWDR1[15:12], SMWDR1[7:4] | DQ4[7:0] | SMWDR0[15:12], SMWDR0[7:4] |
| DQ1[7:0] | SMWDR1[11:8], SMWDR1[3:0] | DQ5[7:0] | SMWDR0[11:8], SMWDR0[3:0] |
| DQ2[7:0] | SMWDR1[31:28], SMWDR1[23:20] | DQ6[7:0] | SMWDR0[31:28], SMWDR0[23:20] |
| DQ3[7:0] | SMWDR1[27:24], SMWDR1[19:16] | DQ7[7:0] | SMWDR0[27:24], SMWDR0[19:16] |

**Figure 112.31 Write Data Alignment with Octal-SPI flash in manual mode**

### 112.3.15 Supported Serial Flash Protocol

Table 112.14 shows the protocols of serial flash memory which are supported in this LSI.

**Table 112.14 Supported protocols of serial flash memory**

| Bit width (Command-Address-Data) | Command | Address | Data |
|---|---|---|---|
| 1-0-0 | SDR | NA | NA |
| 1-0-1 | SDR | NA | SDR |
| 1-1-1 | SDR | SDR | SDR |
| | SDR | DDR | DDR |
| 1-1-4 | SDR | SDR | SDR |
| 1-4-4 | SDR | SDR | SDR |
| | SDR | DDR | DDR |

### 112.3.16 Supported Octal-SPI Flash Protocol

Table 112.15 shows the protocols of Octal-SPI flash memory which are supported in this LSI.

**Table 112.15 Supported protocols of Octal-SPI flash memory**

| Bit width (Command-Address-Data) | Command | Address | Data |
|---|---|---|---|
| 1-0-0 | SDR | NA | NA |
| 1-0-1 | SDR | NA | SDR |
| 1-1-1 | SDR | SDR | SDR |
| 8-0-0 | SDR | NA | NA |
| | DDR | NA | NA |
| 8-0-8 | DDR | NA | DDR |
| 8-8-8 | DDR | DDR | DDR |

### 112.3.17 DDR operation

This controller supports hardware calibration to adjust the timing of data strobe to support DDR protocol. When this LSI connects to the device with data strobe like HyperFlash or Octal SPI flash, this controller adjusts the strobe point automatically. When this LSI connects to the device without data strobe like serial flash memory in DDR mode, more adjustment is necessary to support DDR operation.

To adjust the timing of DDR operation without data strobe flash device, dummy read operation is necessary by software to change the delay of the internal strobe. PHYCNT.STRTIM bits change the delay. The delay depends on the specified value, 0000(biggest) > 0001 > … > 1110 > 1111 (smallest), of PHYCNT.STRTIM bits. The maximum value of tVSD in Table 200.65 can be achieved among these setting values, so software can find the appropriate timing for changing these values one by one. Figure 112.32 shows the flow of this timing adjustment.

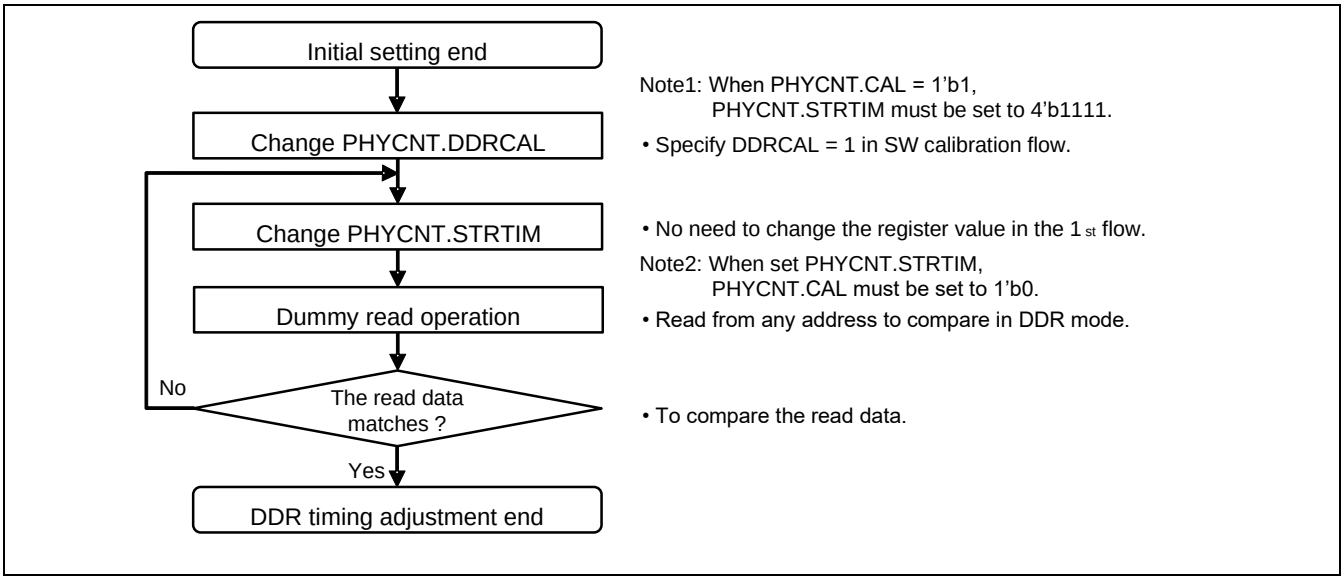



**Figure 112.32 DDR timing adjustment to connect to the device without data strobe**

This timing adjustment should be done in periodically because the timing changes with PVT condition in both this LSI and the connected device.

### 112.3.18 Security function

This controller supports security function to protect the data on the Flash device.

#### (1) SPI multi I/O bus space division

ADD_DIVn.ADD_DIVn (n = 1, 2, 3) bit specifies the address space division to separate the area into four. And SEC_CONF.AREAn (n = 0, 1, 2, 3) specifies the attribution of each area. The address specified ADD_DIVn.ADD_DIVn[19:0] bit are shifted and evaluate the bus address with 4K granularity. For example, when ADD_DIV1.ADD_DIV1[19:0] is specified to 20'h09FFF and SEC_CONF.AREA0 is specified to 2'b01 (non-secure) setting, Area0 becomes H'0800_0000 – H'09FF_FFFF and Non-Secure attribution, and Area1 starts from H'0A00_0000.

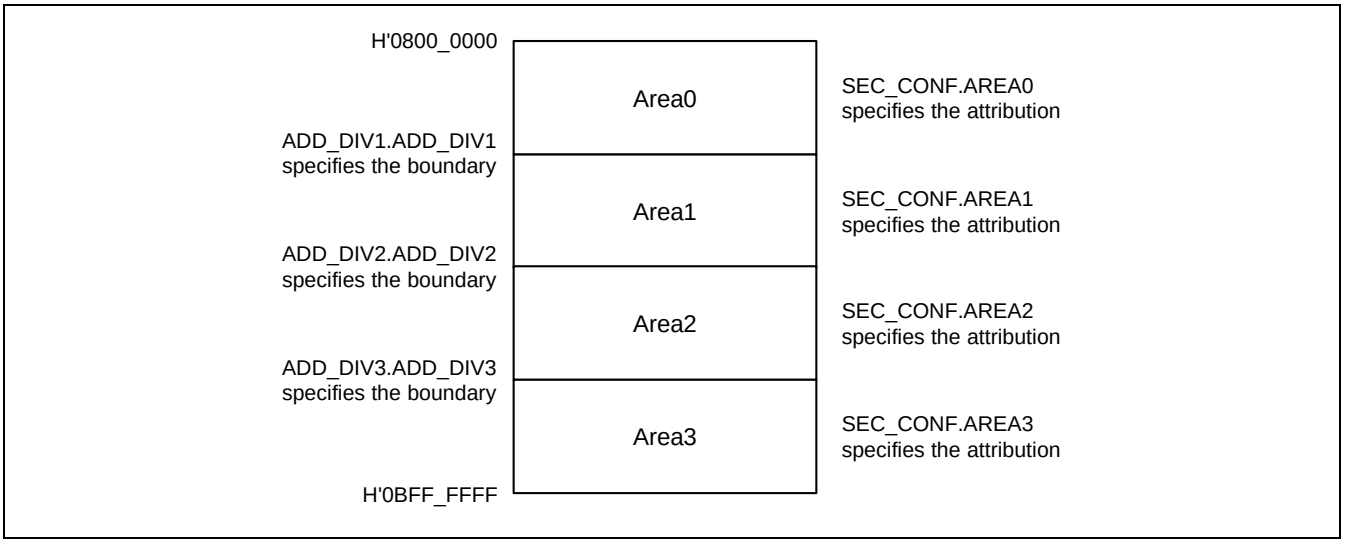



**Figure 112.33 Area protection to divide the address space**

Note that ADD_DIVn (n = 1 to 3) should be specified to keep the order of 0x0800_0000 < ADD_DIV1.ADD_DIV1 < ADD_DIV2.ADD_DIV2 < ADD_DIV3.ADD_DIV3 < H'0BFF_FFFF. If this order is not kept, the operation of this module is not guaranteed. Specifying ADD_DIV1.ADD_DIV1 = 20h'08000 to make H'0800_0000 ~ H'0800_0FFF in Area0 is not available.

**(2) Command restriction for serial flash**

This controller can limit the command to issue to the connected serial flash. SEC_CMDn (n = 0 to 15) and NON_SEC_CMDn (n = 0 to 15) specify the limited command. For example, when H'FF is specified in SEC_CMD0.LIMCMD[7:0] bit and H'FF command is specified in SMCMR.CMD bit, the operation by Secure CPU is not issued. LIMEN bit in each register enables this function.

**(3) Write/Erase protect function for serial flash**

This controller can protect the mistaken overwrite or erase on the other attribution to limit the Write/Erase command to the specific area on serial flash. ERASELISTn (n = 1, 2) and WRITELISTn (n = 1, 2) specify the prohibited command. When the target address belongs to secure attribution area specified in ADD_DIVn and SEC_CONF registers, the operation by non-secure attribution is not issued.

**(4) Access Right**

This controller has access right function for the register setting. There are two attribution, Secure and Non-Secure same as area protection. The master which specifies ARIGHT.RIGHT_EN bit has the right to access. The access right is reflected in each bit 3 to 1 according to the attribution of each master. To cancel the access right, the same attribution master should access to this register with RIGHT_EN = 0. The access is ignored when the attribution of access is different.
When the access is ignored, write result is not reflected to register in this function, then read result is all 0.

### 112.3.19 Hardware calibration

RPC has the hardware calibration function to adjust data receiving timing.
This function adjusts the delay of DelayLine circuit in following figure 112.34.
Target delay of this adjustment is 1/4 transfer period (If transfer period is 12.5ns at 80MHz, 3.25ns is target) and hardware calibration selects nearest delay tap from 32taps of DelayLine.

Trigger of hatdware calibration is following 2 cases.

1. Set PHYCNT.CAL=1
2. Before starting read action with external address space read mode(CMNCR.MD=0) and PHYCNT.DDRCAL=0.

This calibration result can be confirmed with register RDLMON and FDLMON of 112.3.20 PHY register interface

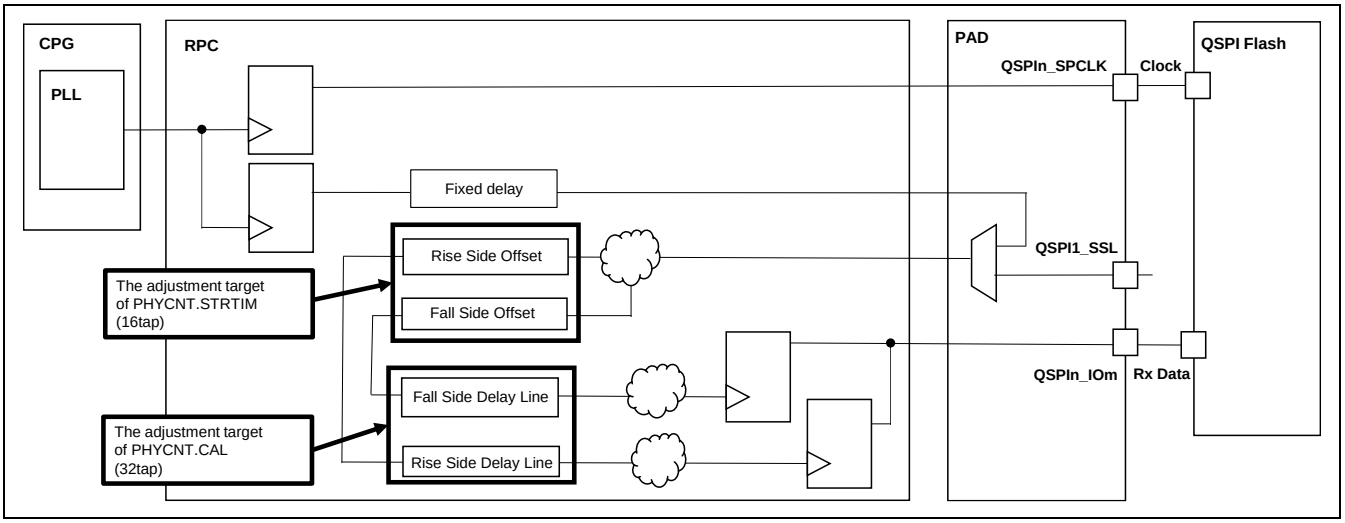



**Figure 112.34 The circuit of data receiving**

### 112.3.20 PHY register interface

This register interface is for check the result of hardware calibration.

#### (1) Register Configuration

Table 112.16 shows a list of the PHY register interface. For the details of PHY interface registers, refer to Section 112.1.4 and Section 112.2.

**Table 112.16 List of PHY register interface**

| Register Name | Abbreviation | R/W |
|---|---|---|
| PHY Address Register | PHYADD | R/W |
| PHY Write Data Register | PHYWR | R/W |
| PHY Read Data Register | PHYRD | R/W |

Note: Do not write to any addresses other than listed above. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed above are undefined.

Table 112.17 shows the register configuration of PHY. The registers in Table 112.17 can be accessed only by register in Table 112.16.

**Table 112.17 Register Configuration of PHY**

| Register Name | Abbreviation | R/W | Address of PHYADD.ADD | Initial value | Access Size | R-Car Series Products |
|---|---|---|---|---|---|---|
| | | | | | | R-Car V4M |
| Mode register | MD | R/W | H'00 | H'0000 0000 | 32 | √ |
| Rise edge Delay line tap monitor | RDLMON | R | H'26 | undefined | 32 | √ |
| Fall edge Delay line tap monitor | FDLMON | R | H'28 | undefined | 32 | √ |

Note: Do not write to any addresses other than listed above. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed above are undefined.

### (2) Register Description

The registers of the RPC-PHY are assigned to and located in the address space of the internal bus.

[Legend for Register Descriptions]

Initial value: Register value after a reset.

—: Undefined value

R/W: Readable/writable. The written value can be read.

R: Read-only. The write value should always be initial value.

W: Write-only.

#### (a) Mode register (MD)

MD is a 32bit register that specifies the mode of registers in PHY.

This register is used to enable to access other registers in Table 112.17 .

The settings of this register should be changed when the TEND flag in CMNSR is 1; otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PHYREGEN | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | PHYREGEN [15:0] | All 0 | R/W | PHY register enable<br>When this bit is set to H'A539, access to other PHY internal register is available. |
| 15 to 0 | - | All 0 | R | Reserved.<br>The write value should always be set to initial value. |

#### (b) Rise edge Delay line tap monitor (RDLMON)

RDLMON is a 32bit register to monitor rise edge delay line tap.

This register is used to check hardware calibration result of DelayLine.

The settings of this register should be changed when the TEND flag in CMNSR is 1; otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | QSPI1DLTAPMON | | | | | — | — | — | QSPI0DLTAPMON | | | | |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | - | - | R | Reserved. |
| 12 to 8 | QSPI1DLTAPMON [4:0] | - | R | Monitor selected delay line tap of QSPI1 side.<br>read value is calibration result of PHYCNT.CAL<br>0 0000 : The delay is smallest<br>・<br>・<br>1 1111 : The delay is longest |
| 7 to 5 | - | - | R | Reserved. |
| 4 to 0 | QSPI0DLTAPMON [4:0] | - | R | Monitor selected delay line tap of QSPI0 side.<br>read value is calibration result of PHYCNT.CAL<br>0 0000 : The delay is smallest<br>・<br>・<br>1 1111 : The delay is longest |

### (c) Fall edge Delay line tap monitor (FDLMON)

FDLMON is a 32bit register to monitor fall edge delay line tap .

This register is used to check hardware calibration result of DelayLine.

The settings of this register should be changed when the TEND flag in CMNSR is 1; otherwise, the operation cannot be guaranteed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | QSPI1DLTAPMON | | | | | — | — | — | QSPI0DLTAPMON | | | | |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | - | - | R | Reserved. |
| 12 to 8 | QSPI1DLTAPMON [4:0] | - | R | Monitor selected delay line tap of QSPI1 side.<br>read value is calibration result of PHYCNT.CAL<br>0 0000 : The delay is smallest<br>・<br>・<br>1 1111 : The delay is longest |
| 7 to 5 | - | - | R | Reserved. |
| 4 to 0 | QSPI0DLTAPMON [4:0] | - | R | Monitor selected delay line tap of QSPI0 side.<br>read value is calibration result of PHYCNT.CAL<br>0 0000 : The delay is smallest<br>・<br>・<br>1 1111 : The delay is longest |

**(3) The way of access to PHY internal register**

For both of write and read sequence, it is mandatory to set MD.PHYREGEN register to H'A539

**(a) Write sequence**

Execute write sequence according to the following procedure

1. Specify write data in PHYWR
2. Specify the PHY register address in PHYADD.ADD, PHYADD.RW=0 (write) and PHYADD.ACCEN=1 to start write sequence.

**(b) Read sequence**

Execute read sequence according to the following procedure

1. Specify the PHY register address in PHYADD.ADD, PHYADD.RW=1 (read) and PHYADD.ACCEN=1 to start read sequence.
2. Dummy Read to PHYRD to make time to finish reading from PHY internal register.
3. Read PHYRD register to get the read value

## 112.4 Usage Notes

### 112.4.1 Transfer to Read Data while the Signal on the QSPIn_SSL Pin is Negated

Set the SMENR.SPIDE[3:0] bits to B'1100 or B'1111 when transfer only for reading data is to proceed.

Transfer will not proceed normally if the setting of the SMENR.SPIDE[3:0] bits is B'1000.

### 112.4.2 Transfer to Read Data while the Signal on the QSPIn_SSL Pin is Asserted

When transfer only for reading data is to proceed, set the SMENR.SPIDE[3:0] bits to B'1100 or B'1111, or end the immediately preceding transfer with reading data.

When the immediately preceding transfer is of a command, optional command, address, or option data, or is transfer for writing data, the subsequent transfer only for reading data will not proceed normally if the setting of the SMENR.SPIDE[3:0] bits is B'1000.

### 112.4.3 Notes on Starting Transfer from the QSPIn_SSL Retained State in Manual Mode

Be sure to set the SPIWE bit in the SMCR register to 1 when the transfer of a command, optional command, address, or option data is started while the QSPIn_SSL pin is being asserted in manual mode.

### 112.4.4 RPC_RESET# operation

There are two ways to operate RPC_RESET# pins.

1. Use PHYINT.RSTEN, PHYINT.RSTVAL to operate.
   The RPC_RESET# state depends on PHYINT.RSTVAL value.
2. Activate reset to this module
   During reset period by Power-on-reset or Software Reset, RPC_RESET# pin state is L. After reset is deactivated, RPC_RESET# pin state is H.

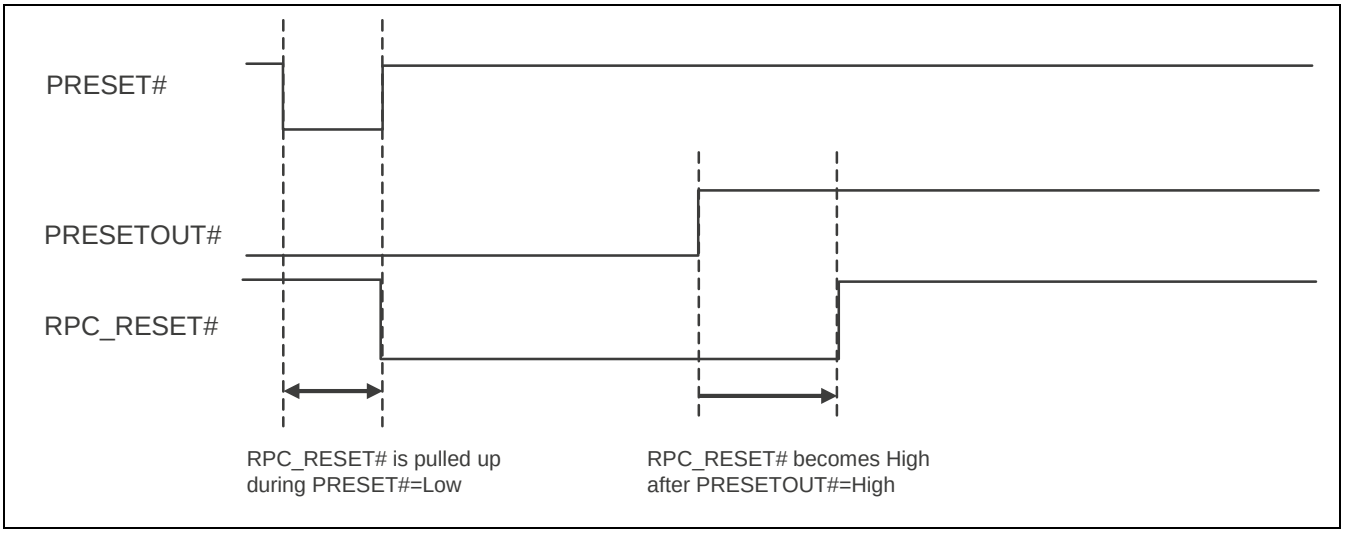



**Figure 112.35 Relationship of RPC_RESET# and PRESET# timing**

### 112.4.5 Software Reset after Module Stop operation

Be sure to supply software reset to this module after module stop is deactivated. Note that RPC_RESET# pin state is L while software reset is activated. Guarantee the reset period and accessible period after reset which are required in the connected devices.

### 112.4.6 DMA transfer

In read operation by DMA transfer, external address space read mode is available. The read address (source address in DMAC) is the memory area (H'08000000 – H'0BFFFFFF) by Auto-Request mode. In High Speed response mode (PHYCNT.HS = 1), the transfer size should be same as the cache size which is specified in DRCR.RBURST. Accessing to register area during DMA transfer is prohibited during High Speed response mode.
In write operation, refer to 112.3.13.

### 112.4.7 Frequency change

When the operation frequency changes by setting RPCCKCR in CPG, read after write operation on this register is necessary to guarantee the setting to be applied.
And before changing RPCCKCR setting, set CMNCR.MOIIOx(x=0,1,2,3) register to B'11 (Hi-Z).

### 112.4.8 Notes on RPC register area reading after read transfer with PHYCNT.HS=1 and read cache

When PHYCNT.HS=1, even if read access to RPC register area, read cache side address is also checked cache hit or not. And if Read cache is hit by this read access to RPC register area, read result becomes logical OR of register area data and read cache data. So, read data may be broken in this case.

For example, in the case of DRCR.EAV[7:0]=H'00, read cache treats register area address with PHYCNT.HS=1 as (2) and (3) in following Table 112.18.
And, when use PHYCNT.HS=1 setting, DMAC is used for read access to Memory area address. So, if last 256Byte address of DMAC transfer does not match with (4) and (5) in following Table 112.18, it is not applicable for this issue. Because, if that address is not matched, other address is stored in read cache after DMAC transfer.

In the case of last 256Byte address of DMAC transfer is matched, this issue can be avoided by PHYCNY.HS=0 or DRCR.RCF=1 before reading RPC register area.

**Table 112.18 Example of Address relationship with DRCR.EAV[7:0]=H'00**

| No. | Explanation of related address | Address |
|---|---|---|
| (1) | Register Area address | 0xEE20_0XXX |
| (2) | Read cache address with DRCR.EAC=B'000 at Register Area address | 0x0020_0XXX |
| (3) | Read cache address with DRCR.EAC=B'001 at Register Area address | 0x0220_0XXX |
| (4) | Memory area address to store 0x0020_0XXX address in Read cache | 0x0820_0XXX |
| (5) | Memory area address to store 0x0220_0XXX address in Read cache | 0x0A20_0XXX |

## 112.5 Safety Mechanisms

This module has no Mechanism of FuSa.

# 132. AES Accelerator (AES-ACC)

## 132.1 Overview

This module is to accelerate AES operation.

- AES Accelerator (AES-ACC) consists of access management unit, crypto engines, and seed generators. AES-ACC supports functions for confidentiality, integrity, and authentication. AES-ACC working with dedicated software provided by Renesas.
- This module is controlled by WCRC module, accessible both Cortex-R52 and Cortex-A76.
  The reference of WCRC is chapter 135 CRC.

### 132.1.1 Features

- 8 modules working independently.
- 2 modules of AES-ACC are connected to 1 WCRC, and they can operate independently at the same time.
- AES-ACC is connected to RT-DMAC through WCRC module, not directly.
- Operating clock is S0D2_RTφ.
- AES length 128 and 256 bits are supported.
- The supported chaining mode for AES are ECB, CBC, CTR, XTS, CMAC, GMAC, CCM and GCM.
- The hash engine supports GHASH.
- Key management function protects encryption keys from being exposed on memory or falsified by other functions.
- Self-test function.

### 132.1.2 Address Region of Each Channel

Table 132.1 Address region of each channel

| Domain name | Address region |
|---|---|
| CH0 | H'E780 0000 – H'E780 0FFF |
| CH1 | H'E780 1000 – H'E780 1FFF |
| CH2 | H'E780 2000 – H'E780 2FFF |
| CH3 | H'E780 3000 – H'E780 3FFF |
| CH4 | H'E780 4000 – H'E780 4FFF |
| CH5 | H'E780 5000 – H'E780 5FFF |
| CH6 | H'E780 6000 – H'E780 6FFF |
| CH7 | H'E780 7000 – H'E780 7FFF |

## 132.2 Register Description

No description on this section.

## 132.3 Operation

No description on this section.

## 132.4 Usage Notes

To use AES-ACC function, Module Stop for both AES-ACC Wrapper and each AES-ACC Channel shall be cleared.

Refer to chapter "9. Module Standby, Software Reset" for this setting.

## 132.5 Safety Mechanisms

No description on this section.

# 133. Life Cycle

## 133.1 Overview

Life Cycle (LifeC) has setting registers for data scramble/de-scramble at DBSC5. Host CPU can access to these registers via APB I/F.

### 133.1.1 Features

The data scramble/de-scramble setting register value are directly provided to DBSC5.

### 133.1.2 Register Configuration

Table 133-1 shows the LifeC register configurations. Accessing to the address not defined below is prohibited.

**Table 133-1 Register Configurations**

| Name | Abbreviation | R/W | Address | Size |
|---|---|---|---|---|
| SDRAM write/read data scramble/de-scramble function control | DBSCRM | R/W | H'E611 0238 | 32 |

## 133.2 Register Description

All registers are 32bit and access size are 32-bit.

Explanation of abbreviation of register

Initial value: Value of the register after power-on reset

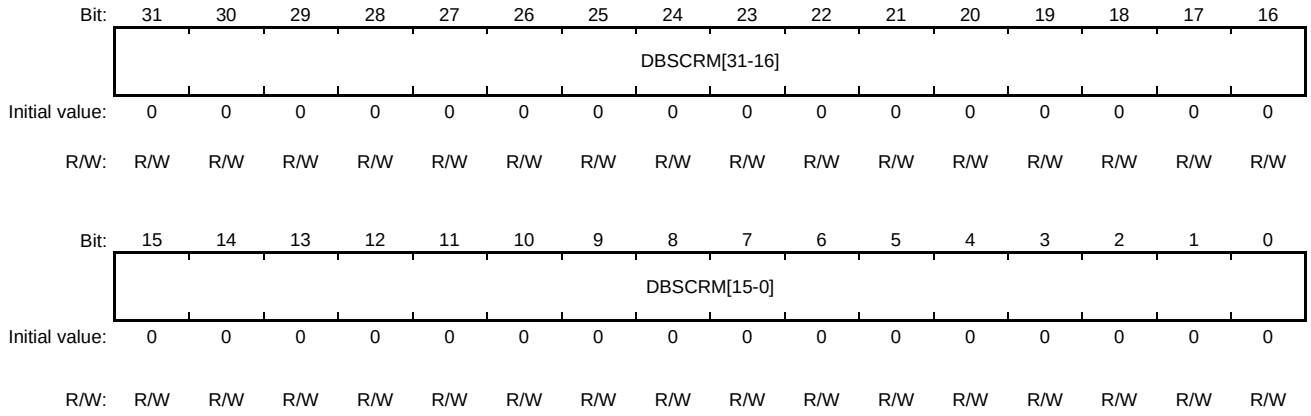
- : Undefined value

R/W : The bit or field is readable and writable.

R : The bit or field is readable only. When writing to the register, write 0 to it.

### 133.2.1 DBSCRM

DBSCRM controls SDRAM write/read data scramble/de-scramble function.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DBSCRM[31-0] | 0 | R/W | All 0: Disable scramble/descramble function.<br>Other: Use DBSCRM value as scramble/descramble seed. |

## 133.3 Operation

No description on this section.

## 133.4 Usage Notes

### 133.4.1 DBSCRM Setting

Setting to DBSCRM should be performed before DRAM access.

Dummy read to DBSCRM are necessary for wait time.

Resetting DBSCRM after DRAM access is prohibited because the data value written before resetting cannot be read correctly.

## 133.5 Safety Mechanisms

No description on this section.

# 134. ICUMX

## 134.1 Overview

The Intelligent Cryptographic Unit/Master Extension (ICUMX) is a hardware security module (HSM).

The features of the ICUMX are described below.

### 134.1.1 Features

- The ICUMX integrates an Intelligent Cryptographic Unit Processor (ICUP) that controls the ICUMX sub-system and runs the user-defined security services.
- The ICUMX includes an accelerator that supports a block cipher algorithm based on the AES (Advanced Encryption Standard, FIPS PUB 197). ICUMX supports DMA transfer for AES block data.
- The ICUMX integrates several public key cryptographic algorithms like RSA, ECC.
- The ICUMX integrates an accelerator that supports HASH function based on SHA-1, SHA-224, and SHA-256(Secure Hash Algorithm, FIPS-PUB 180) and SHA-3 (Secure Hash Algorithm, FIPS PUB 202).
- The ICUMX integrates a true random number generator.
- The ICUMX can be used as a slave peripheral to run security services upon request. The ICUMX handles processing of other user-defined security services (for example, checking contents of memory for completeness and integrity) and operates as a master handling procedures to strengthen measures for security.
- The ICUMX has exclusive read access and programming access to a specific area of memory which holds confidential data (keys and certificates) and highly confidential code for operating security services.
- The ICUMX zeroizes key materials in FA mode.

## 134.2 Register Description

No description on this section.

## 134.3 Operation

No description on this section.

## 134.4 Usage Notes

No description on this section.

## 134.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### 134.5.1 ECC for ICUMXB

#### 134.5.1.1 Overview

ECC protection is applied to dedicated SRAM in ICUMX core (ICUMXB). Local RAM, Instruction cache TAG RAM, Instruction cache DATA RAM, PKCC RAM and SDMAC DATA RAM are protected by ECC.

#### 134.5.1.2 Related Registers

For details, refer to section 14, ECC Error Detection and Correction of AXI and AXIAB, in ICUMXB User's Manual: Hardware.

#### 134.5.1.3 Hardware Description

Each RAM in ICUMXB is protected by SEC-DED error correction code (ECC) checking. When an error is detected, an interrupt is issued to ICUMX_INTC1 and an error notification is issued to error status register of ICUMXB. For details, refer to section 14, ECC Error Detection and Correction of AXI and AXIAB, in ICUMXB User's Manual: Hardware.

### 134.5.2 Hardware Redundancy for ICUMX

#### 134.5.2.1 Overview

The dual core lock step (DCLS) architecture is applied to ICUMX_AESD0 and ICUMX_DMACA0.

#### 134.5.2.2 Related Registers

Table 134.1 shows related registers for DCLS.

**Table 134.1 Related Registers for DCLS**

| Register Name | Abbreviation | R-Car V4M |
|---|---|---|
| Error injection lock control register | ICUMX_ERRLOCKCTL | √ |
| Clock monitor and DCLSERRCTL authority control setting register | ICUMX_CKMDCLSECA | √ |
| ICUMX_FUSA Error Status Register | ICUMX_FUSAERRST | √ |
| ICUMX_FUSA Error Status Clear Register | ICUMX_FUSAERRSC | √ |

For details, refer to section 11, ICUMXB Command Register (ICUMX_CMDREG), in ICUMXB User's Manual: Hardware.

#### 134.5.2.3 Hardware Description

When the DCLS error is detected, ICUMX_AESD0 or ICUMX_DMACA0 issues an error notification and an interrupt. An error notification is issued to ECM and an interrupt is issued to ICUMX_INTC1.

# 135. CRC

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents. However, this module can be also treated as non-safety-related function.

## 135.1 CRC_Wrapper (WCRC)

### 135.1.1 Overview

This module supports the transfer of large capacity data at high speed.

**(1) Features**

This module has an AXI slave interface. (128bit data bus 400MHz clock(S0D2_RTφ))
This module has 2 sets of interfaces for transferring to AES-ACC module.
This module contains one CRC and one KCRC module. (400MHz clock(S0D2_RTφ), Processing performance: 4 bytes / cycle)
Store the data captured from the AXI bus in FIFO, deliver it to CRC, KCRC or AES-ACC module and operate on each module.
The operated data is stored in FIFO and read via the AXI bus.
This module supports function of comparing generated CRC data with expected data that is written in FIFO.
This module supports function of read back (Data through mode) that can read the data captured into FIFO via AXI bus.

**Number of Units and Channels**

| | |
|---|---|
| Number of Units | 4 |
| Name | WCRCm (m = 0 to 3) |

### (2) Block Diagram

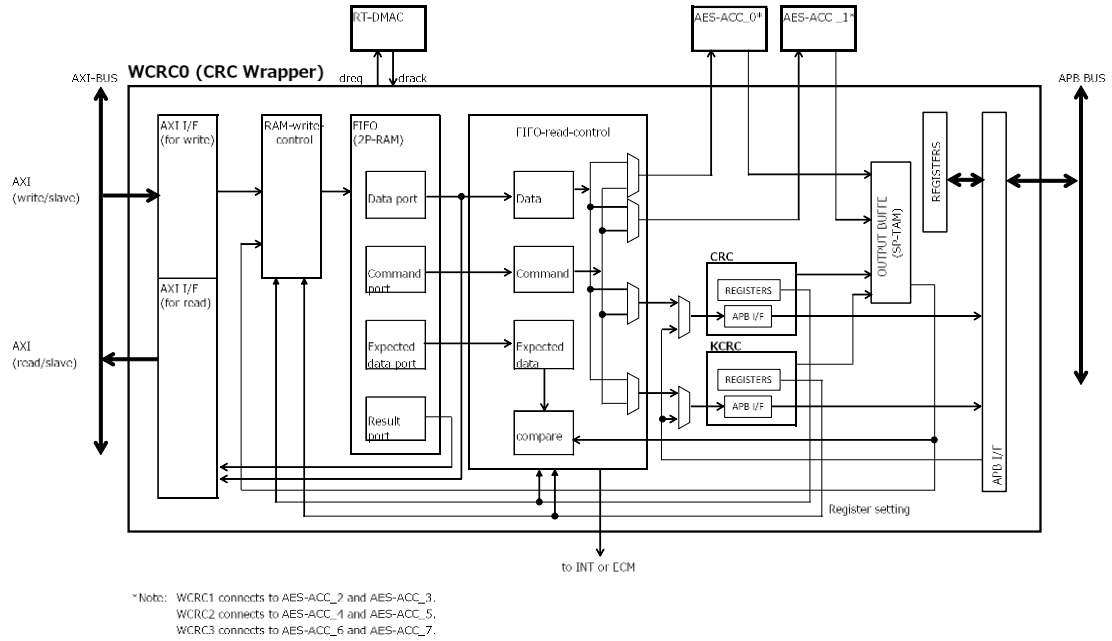



**Figure 135.1 CRC Wrapper Block Diagram**

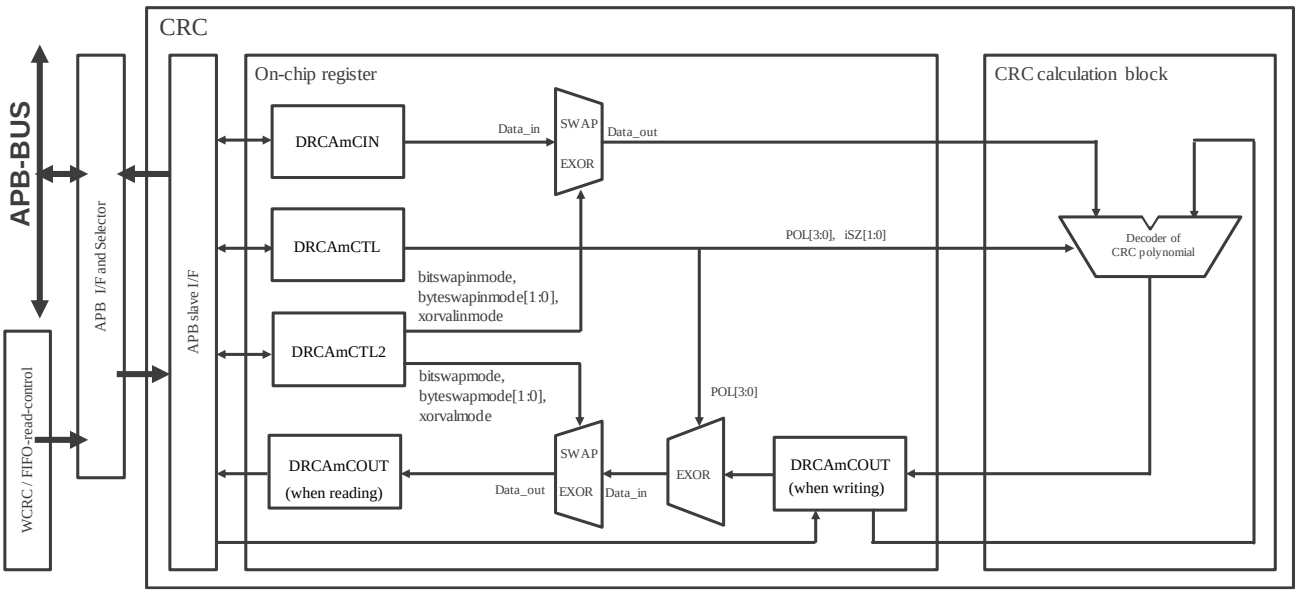



**Figure 135.2 CRC Wrapper Block Diagram (CRC module part)**

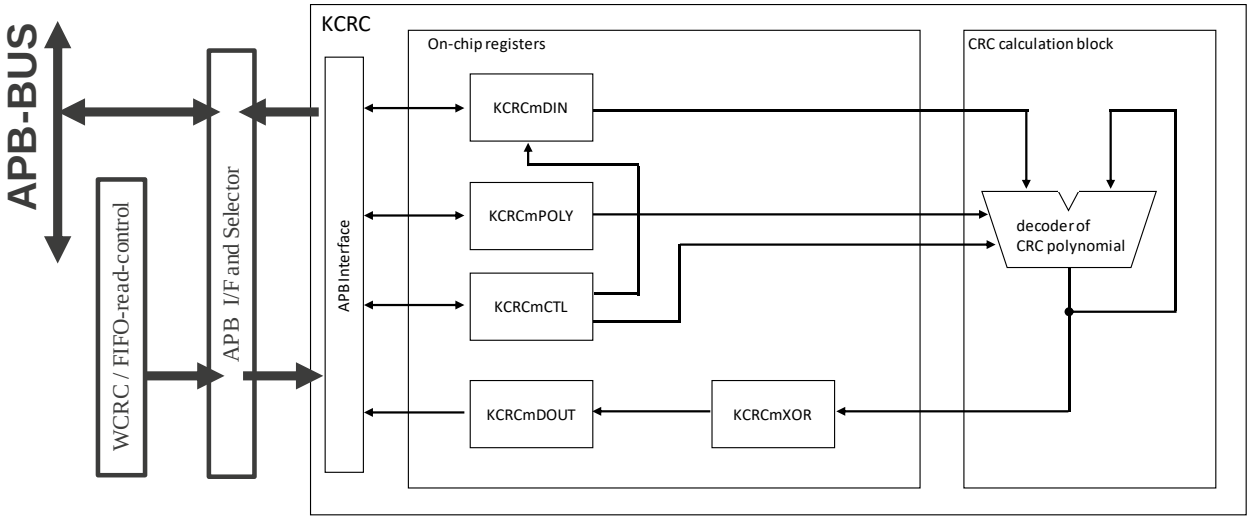



**Figure 135.3 CRC Wrapper Block Diagram (KCRC module part)**

### (3) Register Configurations

The base address for each channel of registers of the CRC Wrapper is as follows:

Table 135.1 offset address Configurations

| Module Name | Offset Base Address |
|---|---|
| WCRC0 | H'E708 0000 |
| WCRC1 | H'E709 0000 |
| WCRC2 | H'E70A 0000 |
| WCRC3 | H'E70B 0000 |

Each channel of the CRC Wrapper has the registers below.

Table 135.2 shows the CRC Wrapper register configurations and register value in each state.

Table 135.2 Register Configurations

| Register Name | Abbreviation | R/W | Offset from Base Address | Initial Value (Power On Reset by PRESET# Pin) | Initial Value (Software Reset) | Access Size |
|---|---|---|---|---|---|---|
| WCRCm CAIPn transfer enable register | WCRCm_CAIPn_EN | R/W | H'0000 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPn transfer stop register | WCRCm_CAIPn_STOP | —/W | H'0020 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPn transfer command enable register | WCRCm_CAIPn_CMDEN | R/W | H'0030 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPn wait register | WCRCm_CAIPn_WAIT | —/W | H'0080 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPn status register | WCRCm_CAIPn_STS | R/WC1 | H'0200 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPn interrupt enable register | WCRCm_CAIPn_INTEN | R/W | H'0240 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPn Buffer state read enable register | WCRCm_CAIPn_BUF_STS_RDEN | R/W | H'02A0 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPn Buffer state read register | WCRCm_CAIPn_BUF_STS | R/— | H'02A4 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPp transfer enable register | WCRCm_CAIPp_EN | R/W | H'0400 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPp transfer stop register | WCRCm_CAIPp_STOP | —/W | H'0420 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPp transfer command enable register | WCRCm_CAIPp_CMDEN | R/W | H'0430 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPp wait register | WCRCm_CAIPp_WAIT | —/W | H'0480 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPp status register | WCRCm_CAIPp_STS | R/WC1 | H'0600 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPp interrupt enable register | WCRCm_CAIPp_INTEN | R/W | H'0640 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPp Buffer state read enable register | WCRCm_CAIPp_BUF_STS_RDEN | R/W | H'06A0 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CAIPp Buffer state read register | WCRCm_CAIPp_BUF_STS | R/— | H'06A4 | H'0000 0000 | H'0000 0000 | 32 |

| Register Name | Abbreviation | R/W | Offset from Base Address | Initial Value (Power On Reset by PRESET# Pin) | Initial Value (Software Reset) | Access Size |
|---|---|---|---|---|---|---|
| WCRCm CRCm transfer enable register | WCRCm_CRCm_EN | R/W | H'0800 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm transfer stop register | WCRCm_CRCm_STOP | —/W | H'0820 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm transfer command enable register | WCRCm_CRCm_CMDEN | R/W | H'0830 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm compare setting register | WCRCm_CRCm_COMP | R/W | H'0840 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm compare result register | WCRCm_CRCm_COMP_RES | R/WC1 | H'0850 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm conversion setting register | WCRCm_CRCm_CONV | R/W | H'0870 | H'0000 0100 | H'0000 0100 | 32 |
| WCRCm CRCm wait register | WCRCm_CRCm_WAIT | —/W | H'0880 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm initial CRC code register | WCRCm_CRCm_INIT_CRC | R/W | H'0910 | H'FFFF FFFF | H'FFFF FFFF | 32 |
| WCRCm CRCm status register | WCRCm_CRCm_STS | R/WC1 | H'0A00 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm interrupt enable register | WCRCm_CRCm_INTEN | R/W | H'0A40 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm ECM output enable register | WCRCm_CRCm_ECMEN | R/W | H'0A80 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm Buffer state read enable register | WCRCm_CRCm_BUF_STS_RDEN | R/W | H'0AA0 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm CRCm Buffer state read register | WCRCm_CRCm_BUF_STS | R/— | H'0AA4 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm transfer enable register | WCRCm_KCRCm_EN | R/W | H'0C00 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm transfer stop register | WCRCm_KCRCm_STOP | —/W | H'0C20 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm transfer command enable register | WCRCm_KCRCm_CMDEN | R/W | H'0C30 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm compare setting register | WCRCm_KCRCm_COMP | R/W | H'0C40 | H'0000 0000 | H'0000 0000 | 32 |

| Register Name | Abbreviation | R/W | Offset from Base Address | Initial Value (Power On Reset by PRESET# Pin) | Initial Value (Software Reset) | Access Size |
|---|---|---|---|---|---|---|
| WCRCm KCRCm compare result register | WCRCm_KCRCm_COMP_RES | R/WC1 | H'0C50 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm conversion setting register | WCRCm_KCRCm_CONV | R/W | H'0C70 | H'0000 0100 | H'0000 0100 | 32 |
| WCRCm KCRCm wait register | WCRCm_KCRCm_WAIT | —/W | H'0C80 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm initial CRC code register | WCRCm_KCRCm_INIT_CRC | R/W | H'0D10 | H'FFFF FFFF | H'FFFF FFFF | 32 |
| WCRCm KCRCm status register | WCRCm_KCRCm_STS | R/WC1 | H'0E00 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm interrupt enable register | WCRCm_KCRCm_INTEN | R/W | H'0E40 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm ECM output enable register | WCRCm_KCRCm_ECMEN | R/W | H'0E80 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm Buffer state read enable register | WCRCm_KCRCm_BUF_STS_RDEN | R/W | H'0EA0 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm KCRCm Buffer state read register | WCRCm_KCRCm_BUF_STS | R/— | H'0EA4 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm common status register | WCRCm_COMMON_STS | R/WC1 | H'0F00 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm common interrupt enable register | WCRCm_COMMON_INTEN | R/W | H'0F40 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm common ECM output enable register | WCRCm_COMMON_ECMEN | R/W | H'0F80 | H'0000 0000 | H'0000 0000 | 32 |
| WCRCm error injection register | WCRCm_ERRINJ | R/W | H'0FC0 | H'0000 0000 | H'0000 0000 | 32 |

Note: Access to the addresses other than those listed above is prohibited. Otherwise, operation cannot be guaranteed. If read, an undefined value is read.

### 135.1.2 Register Descriptions

[Legend]

| | |
|---|---|
| Initial value: | Register value after a reset |
| —: | Undefined value |
| R/W: | Readable/writable. The written value can be read. |
| R: | Read-only. The write value should always be 0. |
| R/WC1: | Readable/writable. Writing 1 initializes the bit. Writing 0 is ignored. |
| —/W: | Write-only. The read value is undefined. |

All access to registers is made in longword units.

### (1) WCRCm CAIPn transfer enable register (WCRCm_CAIPn_EN)

WCRCm_CAIPn_EN is a register that sets enable the data transfer to/from each port for AES-ACC_n interface in FIFO.
Note: m = 0, 1, 2, 3
n = 0, 2, 4, 6

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | out_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | res_en | — | — | — | — | — | — | trans_en | in_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | out_en | 0 | R/W | Output data from Data port in FIFO enable<br>0: Disable output from Data port<br>1: Enable output from Data port |
| 15 to 9 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | res_en | 0 | R/W | Output result data from Result port in FIFO enable<br>0: Disable output from Result port<br>1: Enable output from Result port |
| 7 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | trans_en | 0 | R/W | Transferring data from Data port in FIFO to AES-ACC_n enable<br>0: Disable transferring data to AES-ACC_n<br>1: Enable transferring data to AES-ACC_n<br>When transfer to AES-ACC_n isn't performed, this bit should be 0. |
| 0 | in_en | 0 | R/W | Input data to Data/Command port in FIFO enable<br>0: Disable input to Data/Command port<br>1: Enable input to Data/Command port |

### (2) WCRCm CAIPn transfer stop register (WCRCm_CAIPn_STOP)

WCRCm_CAIPn_STOP is a register that stop transfer to AES-ACC_n interface.

Note: m = 0, 1, 2, 3
n = 0, 2, 4, 6

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | stop |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | stop | 0 | —/W | 1: Stop data transfer.<br>Writing 0 to this bit is ignored. |

### (3) WCRCm CAIPn transfer command enable register (WCRCm_CAIPn_CMDEN)

WCRCm_CAIPn_CMDEN is a register that sets enable command function for AES-ACC_n module.
For detail of command function, please refer section 135.1.3(6).
Note: m = 0, 1, 2, 3
n = 0, 2, 4, 6

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

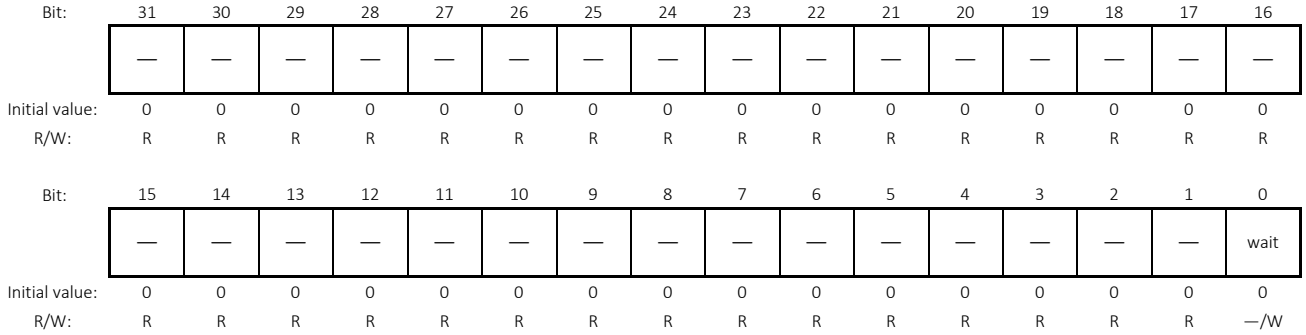
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | cmd_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | cmd_en | 0 | R/W | Command function enable<br>0: Disable command function<br>1: Enable command function |

### (4) WCRCm CAIPn wait register (WCRCm_CAIPn_WAIT)

WCRCm_CAIPn_WAIT is a register that wait subsequent command for AES-ACC_n interface.

Note: m = 0, 1, 2, 3
n = 0, 2, 4, 6




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | wait | 0 | —/W | 1: Wait subsequent command for 128 cycles @ S0D2_RTφ.<br>This bit should be accessed by command function via command port for AES-ACC_n in FIFO.<br>Otherwise, operation cannot be guaranteed.<br>Writing 0 to this bit is ignored. |

### (5) WCRCm CAIPn status register (WCRCm_CAIPn_STS)

WCRCm_CAIPn_STS is a register that indicates the state of operation related to AES-ACC_n module.
Note: m = 0, 1, 2, 3
n = 0, 2, 4, 6

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | stop_done | — | — | — | — | — | — | cmd_done | — | — | — | res_done | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/WC1 | R | R | R | R | R | R | R/WC1 | R | R | R | R/WC1 | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | trans_done |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | stop_done | 0 | R/WC1 | Indicates the state of stop operation by WCRCm_CAIPn_STOP register.<br>0: Stop operation in progress or not done.<br>1: Stop operation is done |
| 30 to 25 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 24 | cmd_done | 0 | R/WC1 | Indicates the state of command function.<br>0: Accessing to register by command in progress or not accessing<br>1: Accessing to register by command is done |
| 23 to 21 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 20 | res_done | 0 | R/WC1 | Indicates the state of result data transfer.<br>0: Transfer in progress or not transferring<br>1: Transfer is done |
| 19 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 0 | trans_done | 0 | R/WC1 | Indicates the state of input data transfer.<br>0: Transfer in progress or not transferring<br>1: Transfer is done |

### (6) WCRCm CAIPn interrupt enable register (WCRCm_CAIPn_INTEN)

WCRCm_CAIPn_INTEN is a register that sets enable for the interrupt of operation related to AES-ACC_n module.
Note: m = 0, 1, 2, 3
n = 0, 2, 4, 6

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | stop_done_ie | — | — | — | — | — | — | cmd_done_ie | — | — | — | res_done_ie | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R/W | R | R | R | R/W | R | R | R | R |

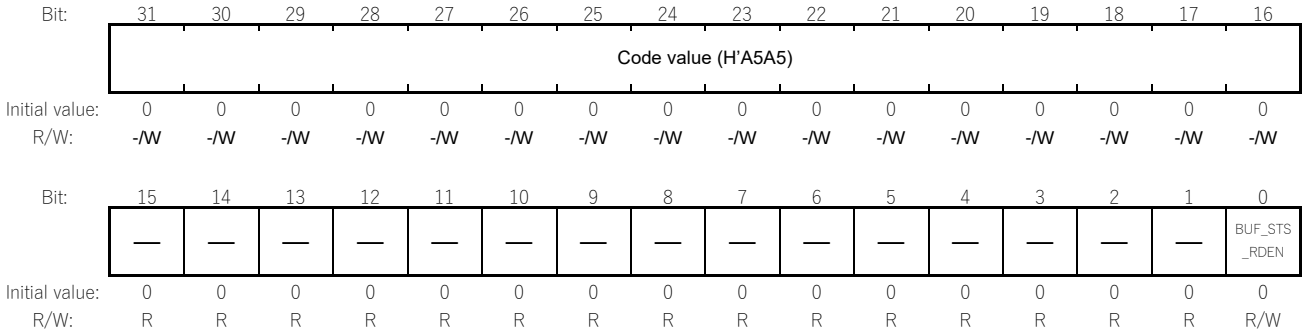
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | trans_done_ie |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | stop_done_ie | 0 | R/W | Interrupt for the complete of stop operation by WCRCm_CAIPn_STOP register.<br>0: Disable interrupt<br>1: Enable interrupt |
| 30 to 25 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 24 | cmd_done_ie | 0 | R/W | Interrupt for the complete of command function enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 23 to 21 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | res_done_ie | 0 | R/W | Interrupt for the complete of result data transfer enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 19 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | trans_done_ie | 0 | R/W | Interrupt for the complete of input data transfer enable.<br>0: Disable interrupt<br>1: Enable interrupt |

### (7) WCRCm CAIPn Buffer state read enable register (WCRCm_CAIPn_BUF_STS_RDEN)

WCRCm_CAIPn_BUF_STS_RDEN is a register that sets enable for the interrupt of operation related to AES-ACC_n module.

Note: m = 0, 1, 2, 3
n = 0, 2, 4, 6




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Code value | All 0 | -/W | Code value (H'A5A5)<br>Set "Code value" to these bits when writing to this register to enable writing to this register. When read, return 0. |
| 15 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | BUF_STS_RDEN | 0 | R/W | WCRCm_CAIPn_BUF_STS enable.<br>0: Disable WCRCm_CAIPn_BUF_STS<br>1: Enable WCRCm_CAIPn_BUF_STS |

### (8) WCRCm CAIPn Buffer state read register (WCRCm_CAIPn_BUF_STS)

WCRCm_CAIPn_BUF_STS is a register that sets enable for the interrupt of operation related to AES-ACC_n module.
Note: m = 0, 1, 2, 3
n = 0, 2, 4, 6

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | buf_empty | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 17 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 16 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 15 to 13 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 12 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 11 to 10 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 9 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 8 | buf_empty | 0 | R | Buffer empty flag.<br>0: Buffer is not empty<br>1: Buffer is empty |
| 7 to 6 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (9) WCRCm CAIPp transfer enable register (WCRCm_CAIPp_EN)

WCRCm_CAIPp_EN is a register that sets enable the data transfer to/from each ports for AES-ACC_p interface in FIFO.

Note: m = 0, 1, 2, 3
p = 1, 3, 5, 7

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | out_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | res_en | — | — | — | — | — | — | trans_en | in_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | out_en | 0 | R/W | Output data from Data port in FIFO enable<br>0: Disable output from Data port<br>1: Enable output from Data port |
| 15 to 9 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | res_en | 0 | R/W | Output result data from Result port in FIFO enable<br>0: Disable output from Result port<br>1: Enable output from Result port |
| 7 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | trans_en | 0 | R/W | Transferring data from Data port in FIFO to AES-ACC_p enable<br>0: Disable transferring data to AES-ACC_p<br>1: Enable transferring data to AES-ACC_p<br>When transfer to AES-ACC_p isn't performed, this bit should be 0. |
| 0 | in_en | 0 | R/W | Input data to Data/Command port in FIFO enable<br>0: Disable input to Data/Command port<br>1: Enable input to Data/Command port |

### (10) WCRCm CAIPp transfer stop register (WCRCm_CAIPp_STOP)

WCRCm_CAIPp_STOP is a register that stop transfer to AES-ACC_p interface.

Note: m = 0, 1, 2, 3
p = 1, 3, 5, 7

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | stop |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | stop | 0 | —/W | 1: Stop data transfer.<br>Writing 0 to this bit is ignored. |

### (11) WCRCm CAIPp transfer command enable register (WCRCm_CAIPp_CMDEN)

WCRCm_CAIPp_CMDEN is a register that sets enable command function for AES-ACC_p module.
For detail of command function, please refer section 135.1.3(6).
Note: m = 0, 1, 2, 3
p = 1, 3, 5, 7

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

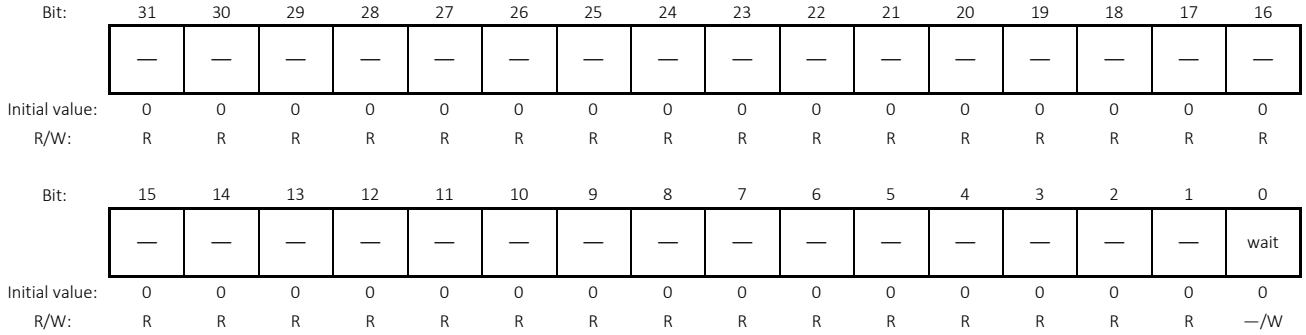
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | cmd_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | cmd_en | 0 | R/W | Command function enable<br>0: Disable command function<br>1: Enable command function |

### (12) WCRCm CAIPp wait register (WCRCm_CAIPp_WAIT)

WCRCm_CAIPp_WAIT is a register that wait subsequent command for AES-ACC_p interface.

Note: m = 0, 1, 2, 3
p = 1, 3, 5, 7



| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | wait |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | wait | 0 | —/W | 1: Wait subsequent command for 128 cycles @ S0D2_RTφ.<br>This bit should be accessed by command function via command port for AES-ACC_p in FIFO.<br>Otherwise, operation cannot be guaranteed.<br>Writing 0 to this bit is ignored. |

### (13) WCRCm CAIPp status register (WCRCm_CAIPp_STS)

WCRCm_CAIPp_STS is a register that indicates the state of operation related to AES-ACC_p module.
Note: m = 0, 1, 2, 3
p = 1, 3, 5, 7

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | stop_done | — | — | — | — | — | — | cmd_done | — | — | — | res_done | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/WC1 | R | R | R | R | R | R | R/WC1 | R | R | R | R/WC1 | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | trans_done |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | stop_done | 0 | R/WC1 | Indicates the state of stop operation by WCRCm_CAIPp_STOP register.<br>0: Stop operation in progress or not done.<br>1: Stop operation is done |
| 30 to 25 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 24 | cmd_done | 0 | R/WC1 | Indicates the state of command function.<br>0: Accessing to register by command in progress or not accessing<br>1: Accessing to register by command is done |
| 23 to 21 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 20 | res_done | 0 | R/WC1 | Indicates the state of result data transfer.<br>0: Transfer in progress or not transferring<br>1: Transfer is done |
| 19 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 0 | trans_done | 0 | R/WC1 | Indicates the state of input data transfer.<br>0: Transfer in progress or not transferring<br>1: Transfer is done |

### (14) WCRCm CAIPp interrupt enable register (WCRCm_CAIPp_INTEN)

WCRCm_CAIPp_INTEN is a register that sets enable for the interrupt of operation related to AES-ACC_p module.
Note: m = 0, 1, 2, 3
p = 1, 3, 5, 7

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | stop_done_ie | — | — | — | — | — | — | cmd_done_ie | — | — | — | res_done_ie | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R/W | R | R | R | R/W | R | R | R | R |

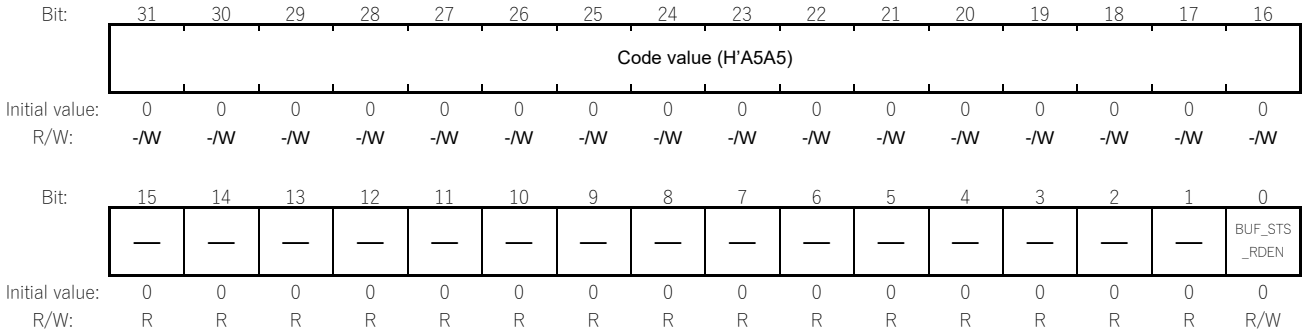
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | trans_done_ie |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | stop_done_ie | 0 | R/W | Interrupt for the complete of stop operation by WCRCm_CAIPp_STOP register.<br>0: Disable interrupt<br>1: Enable interrupt |
| 30 to 25 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 24 | cmd_done_ie | 0 | R/W | Interrupt for the complete of command function enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 23 to 21 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | res_done_ie | 0 | R/W | Interrupt for the complete of result data transfer enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 19 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | trans_done_ie | 0 | R/W | Interrupt for the complete of input data transfer enable.<br>0: Disable interrupt<br>1: Enable interrupt |

### (15) WCRCm CAIPp Buffer state read enable register (WCRCm_CAIPp_BUF_STS_RDEN)

WCRCm_CAIPp_BUF_STS_RDEN is a register that sets enable for the interrupt of operation related to AES-ACC_p module.

Note: m = 0, 1, 2, 3
p = 1, 3, 5, 7




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Code value | All 0 | -/W | Code value (H'A5A5)<br>Set "Code value" to these bits when writing to this register to enable writing to this register. When read, return 0. |
| 15 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | BUF_STS_RDEN | 0 | R/W | WCRCm_CAIPp_BUF_STS enable.<br>0: Disable WCRCm_CAIPp_BUF_STS<br>1: Enable WCRCm_CAIPp_BUF_STS |

### (16) WCRCm CAIPp Buffer state read register (WCRCm_CAIPp_BUF_STS)

WCRCm_CAIPp_BUF_STS is a register that sets enable for the interrupt of operation related to AES-ACC_p module.
Note: m = 0, 1, 2, 3
p = 1, 3, 5, 7

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | buf_empty | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 17 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 16 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 15 to 13 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 12 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 11 to 10 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 9 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 8 | buf_empty | 0 | R | Buffer empty flag.<br>0: Buffer is not empty<br>1: Buffer is empty |
| 7 to 6 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (17) WCRCm CRCm transfer enable register (WCRCm_CRCm_EN)

WCRCm_CRCm_EN is a register that sets enable the data transfer to/from each port for CRCm module in FIFO.

Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | out_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | res_en | — | — | — | — | — | — | trans_en | in_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | out_en | 0 | R/W | Output data from Data port in FIFO enable<br>0: Disable output from Data port<br>1: Enable output from Data port |
| 15 to 9 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | res_en | 0 | R/W | Output result data from Result port in FIFO enable<br>0: Disable output from Result port<br>1: Enable output from Result port |
| 7 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | trans_en | 0 | R/W | Transferring data from Data port in FIFO to CRCm enable<br>0: Disable transferring data to CRCm<br>1: Enable transferring data to CRCm<br>When transfer to CRCm isn't performed, this bit should be 0. |
| 0 | in_en | 0 | R/W | Input data to Data/Command/Expected data port in FIFO enable<br>0: Disable input to Data/Command/Expected data port<br>1: Enable input to Data/Command/Expected data port |

### (18) WCRCm CRCm transfer stop register (WCRCm_CRCm_STOP)

WCRCm_CRCm_STOP is a register that stop transfer to CRCm module.

Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | stop |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | stop | 0 | —/W | 1: Stop data transfer.<br>Writing 0 to this bit is ignored. |

### (19) WCRCm CRCm transfer command enable register (WCRCm_CRCm_CMDEN)

WCRCm_CRCm_CMDEN is a register that sets enable command function for CRCm module.
For detail of command function, please refer section 135.1.3(6).
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | cmd_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | cmd_en | 0 | R/W | Command function enable<br>0: Disable command function<br>1: Enable command function |

### (20) WCRCm CRCm compare setting register (WCRCm_CRCm_COMP)

WCRCm_CRCm_COMP is a register that sets enable comparing CRC result from CRCm module with expected data in FIFO.
For detail of comparing CRC result function, please refer section 135.1.3(7).
Note: m = 0, 1, 2, 3

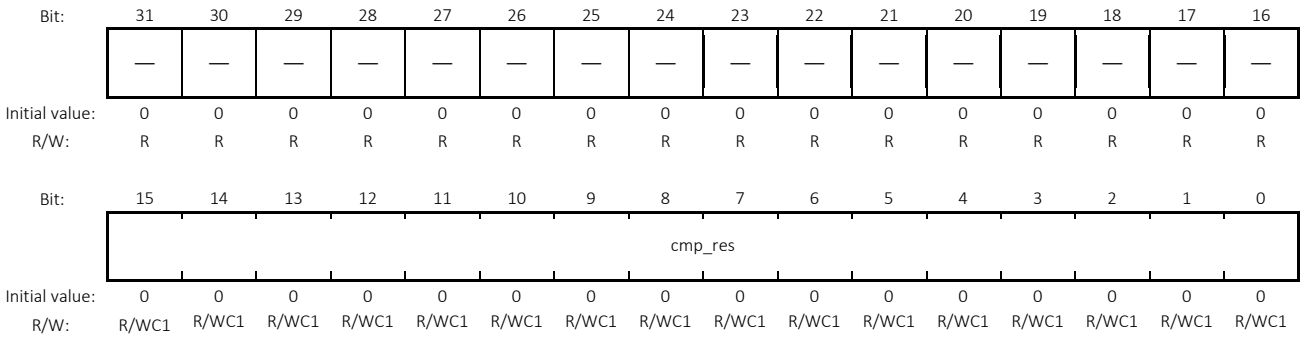
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | cmp_freq [1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | exp_reqsel | cmp_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 17 to 16 | cmp_freq | 00 | R/W | Set the frequency of comparing.<br>00：Compare every 16Byte CRC generated<br>01：Compare every 32Byte CRC generated<br>10：Setting prohibited<br>11：Compare every 64Byte CRC generated |
| 15 to 2 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 1 | exp_reqsel | 0 | R/W | Select DMA Transfer request source for transferring expected data.<br>0: dmareq_wcrc***m***_crc***m***_in<br>1: dmareq_wcrc***m***_crc***m***_res<br>***m***: 0 to 3 |
| 0 | cmp_en | 0 | R/W | Comparing CRC result with expected data enable<br>0: Disable comparing function<br>1: Enable comparing function |

### (21) WCRCm CRCm compare result register (WCRCm_CRCm_COMP_RES)

WCRCm_CRCm_COMP_RES is a register that indicates the result of comparing CRC for CRCm module.
Note: m = 0, 1, 2, 3

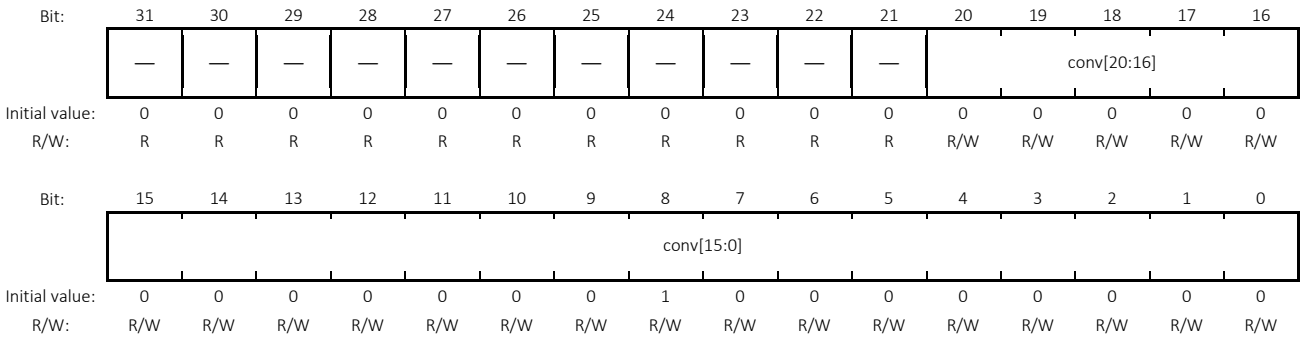



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 0 | cmp_res | H'0000 | R/WC1 | The result of comparing CRC<br>These bits indicate mismatched point of comparing data in 4-byte units.<br>0: No error<br>1: Mismatch<br>Figure 135.4 shows correspondence between comparing data and these bits. |

### (22) WCRCm CRCm conversion setting register (WCRCm_CRCm_CONV)

WCRCm_CRCm_CONV is a register that sets CRC conversion size to once for CRCm module.

Note: m = 0, 1, 2, 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 20 to 0 | conv | H'00 0100 | R/W | CRC conversion size specification<br>Specifies the CRC conversion size in 1-byte units.<br>Max:H'10 0000 |

### (23) WCRCm CRCm wait register (WCRCm_CRCm_WAIT)

WCRCm_CRCm_WAIT is a register that wait subsequent command for CRCm module.

Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

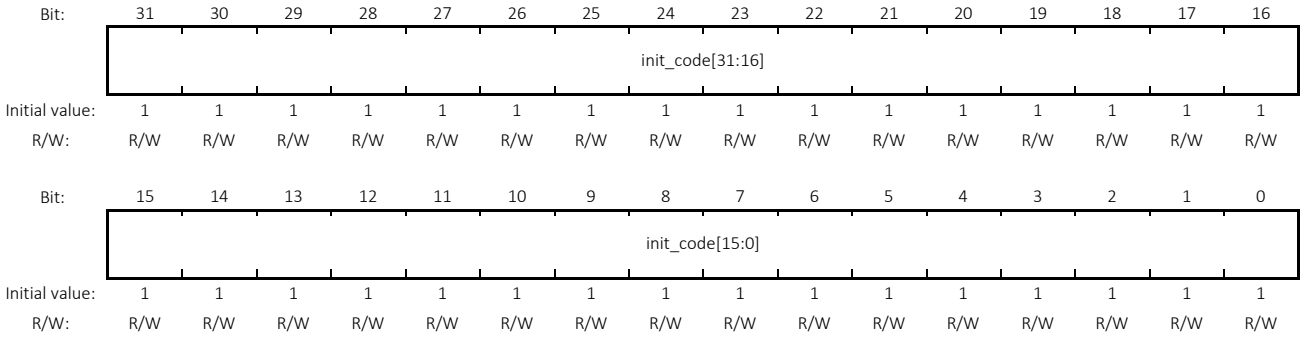
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | wait |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | wait | 0 | —/W | 1: Wait subsequent command for 128 cycles @ S0D2_RTφ.<br>This bit should be accessed by command function via command port for CRCm in FIFO.<br>Otherwise, operation cannot be guaranteed.<br>Writing 0 to this bit is ignored. |

### (24) WCRCm CRCm initial CRC code register (WCRCm_CRCm_INIT_CRC)

WCRCm_CRCm_INIT_CRC is a register that sets CRC code value to DCRAmCOUT register at auto clear.

Note: m = 0, 1, 2, 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | init_code | H'FFFF FFFF | R/W | CRC code value to DCRAmCOUT register at auto clear. |

### (25) WCRCm CRCm status register (WCRCm_CRCm_STS)

WCRCm_CRCm_STS is a register that indicates the state of operation related to CRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | stop_done | — | — | — | — | — | — | cmd_done | — | — | — | res_done | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/WC1 | R | R | R | R | R | R | R/WC1 | R | R | R | R/WC1 | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | comp_err | comp_done | — | — | — | — | — | — | — | — | — | — | — | trans_done |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/WC1 | R/WC1 | R | R | R | R | R | R | R | R | R | R | R | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | stop_done | 0 | R/WC1 | Indicates the state of stop operation by WCRCm_CRCm_STOP register.<br>0: Stop operation in progress or not done.<br>1: Stop operation is done |
| 30 to 25 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 24 | cmd_done | 0 | R/WC1 | Indicates the state of command function.<br>0: Accessing to register by command in progress or not accessing<br>1: Accessing to register by command is done |
| 23 to 21 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 20 | res_done | 0 | R/WC1 | Indicates the state of result data transfer.<br>0: Transfer in progress or not transferring<br>1: Transfer is done |
| 19 to 14 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 13 | comp_err | 0 | R/WC1 | Indicates the error of comparing CRC.<br>0: No comparing error or not comparing<br>1: Mismatch has occurred |
| 12 | comp_done | 0 | R/WC1 | Indicates the state of comparing CRC.<br>0: Comparison in progress or not comparing<br>1: Comparing is done |
| 11 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 0 | trans_done | 0 | R/WC1 | Indicates the state of input data transfer.<br>0: Transfer in progress or not transferring<br>1: Transfer is done |

### (26) WCRCm CRCm interrupt enable register (WCRCm_CRCm_INTEN)

WCRCm_CRCm_INTEN is a register that sets enable for the interrupt of operation related to CRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | stop_done_ie | — | — | — | — | — | — | cmd_done_ie | — | — | — | res_done_ie | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R/W | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | comp_err_ie | comp_done_ie | — | — | — | — | — | — | — | — | — | — | — | trans_done_ie |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | stop_done_ie | 0 | R/W | Interrupt for the complete of stop operation by WCRCm_CRCm_STOP register.<br>0: Disable interrupt<br>1: Enable interrupt |
| 30 to 25 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 24 | cmd_done_ie | 0 | R/W | Interrupt for the complete of command function enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 23 to 21 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | res_done_ie | 0 | R/W | Interrupt for the complete of result data transfer enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 19 to 14 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 13 | comp_err_ie | 0 | R/W | Interrupt for the error of comparing CRC enable<br>0: Disable interrupt<br>1: Enable interrupt |
| 12 | comp_done_ie | 0 | R/W | Interrupt for the complete of comparing CRC enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 11 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | trans_done_ie | 0 | R/W | Interrupt for the complete of input data transfer enable.<br>0: Disable interrupt<br>1: Enable interrupt |

### (27) WCRCm CRCm ECM output enable register (WCRCm_CRCm_ECMEN)

WCRCm_CRCm_ECMEN is a register that sets enable for error notification to ECM module on operation related to CRCm module.
Note: m = 0, 1, 2, 3

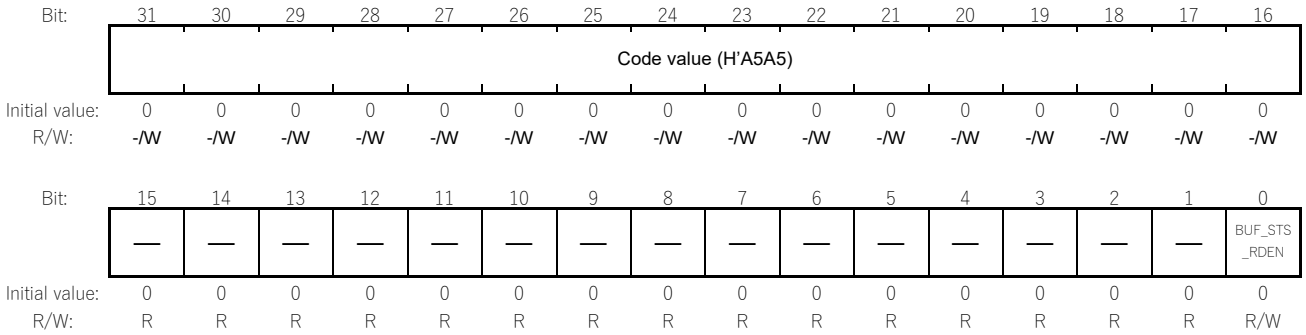
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | comp_ err_oe | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 13 | comp_err_oe | 0 | R/W | Error notification for the error of comparing CRC enable<br>0: Disable notification<br>1: Enable notification |
| 12 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (28) WCRCm CRCm Buffer state read enable register (WCRCm_CRCm_BUF_STS_RDEN)

WCRCm_CRCm_BUF_STS_RDEN is a register that sets enable for the interrupt of operation related to CRCm module.
Note: m = 0, 1, 2, 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Code value | All 0 | -/W | Code value (H'A5A5)<br>Set "Code value" to these bits when writing to this register to enable writing to this register. When read, return 0. |
| 15 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | BUF_STS_RDEN | 0 | R/W | WCRCm_CRCm_BUF_STS enable.<br>0: Disable WCRCm_CRCm_BUF_STS<br>1: Enable WCRCm_CRCm_BUF_STS |

### (29) WCRCm CRCm Buffer state read register (WCRCm_CRCm_BUF_STS)

WCRCm_CRCm_BUF_STS is a register that sets enable for the interrupt of operation related to CRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | res_comp_endflag | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | buf_empty | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 18 | res_comp_endflag | 0 | R | Comparing CRC result end flag.<br>0: Comparing CRC result is not end<br>1: Comparing CRC result is end |
| 17 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 16 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 15 to 13 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 12 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 11 to 10 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 9 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 8 | buf_empty | 0 | R | Buffer empty flag.<br>0: Buffer is not empty<br>1: Buffer is empty |
| 7 to 6 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (30) WCRCm KCRCm transfer enable register (WCRCm_KCRCm_EN)

WCRCm_KCRCm_EN is a register that sets enable the data transfer to/from each port for KCRCm module in FIFO.

Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | out_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | res_en | — | — | — | — | — | — | trans_en | in_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 16 | out_en | 0 | R/W | Output data from Data port in FIFO enable<br>0: Disable output from Data port<br>1: Enable output from Data port |
| 15 to 9 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | res_en | 0 | R/W | Output result data from Result port in FIFO enable<br>0: Disable output from Result port<br>1: Enable output from Result port |
| 7 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | trans_en | 0 | R/W | Transferring data from Data port in FIFO to KCRCm enable<br>0: Disable transferring data to KCRCm<br>1: Enable transferring data to KCRCm<br>When transfer to KCRCm isn't performed, this bit should be 0. |
| 0 | in_en | 0 | R/W | Input data to Data/Command/Expected data port in FIFO enable<br>0: Disable input to Data/Command/Expected data port<br>1: Enable input to Data/Command/Expected data port |

### (31) WCRCm KCRCm transfer stop register (WCRCm_KCRCm_STOP)

WCRCm_KCRCm_STOP is a register that stop transfer to KCRCm module.

Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | stop |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | stop | 0 | —/W | 1: Stop data transfer.<br>Writing 0 to this bit is ignored. |

### (32) WCRCm KCRCm transfer command enable register (WCRCm_KCRCm_CMDEN)

WCRCm_KCRCm_CMDEN is a register that sets enable command function for KCRCm module.
For detail of command function, please refer section 135.1.3(6).
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | cmd_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | cmd_en | 0 | R/W | Command function enable<br>0: Disable command function<br>1: Enable command function |

### (33) WCRCm KCRCm compare setting register (WCRCm_KCRCm_COMP)

WCRCm_KCRCm_COMP is a register that sets enable comparing CRC result from KCRCm module with expected data in FIFO.
For detail of comparing CRC result function, please refer section 135.1.3(7).
Note: m = 0, 1, 2, 3

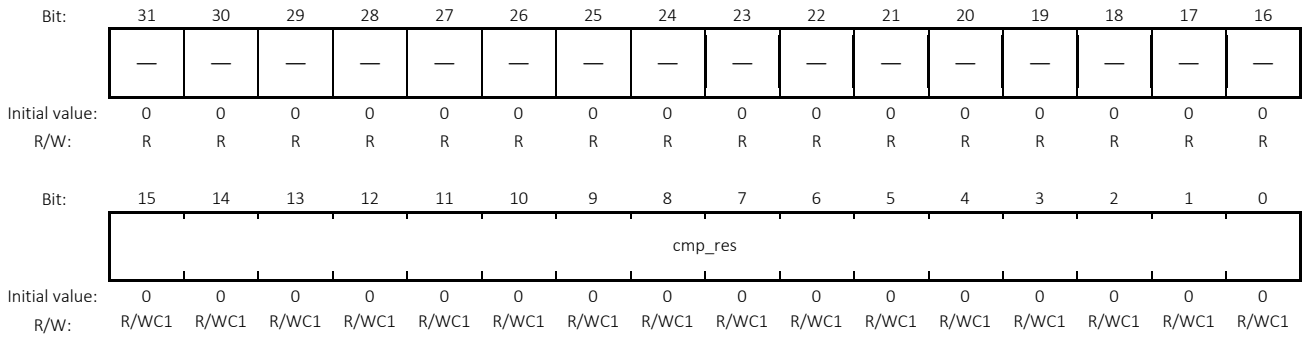
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | cmp_freq [1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | exp_reqsel | cmp_en |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 17 to 16 | cmp_freq | 00 | R/W | Set the frequency of comparing.<br>00：Compare every 16Byte CRC generated<br>01：Compare every 32Byte CRC generated<br>10：Setting prohibited<br>11：Compare every 64Byte CRC generated |
| 15 to 2 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 1 | exp_reqsel | 0 | R/W | Select DMA Transfer request source for transferring expected data.<br>0: dmareq_wcrc***m***_kcrc***n***_in<br>1: dmareq_wcrc***m***_kcrc***n***_res<br>***m***: 0 to 3, ***n***: 4 to 7 |
| 0 | cmp_en | 0 | R/W | Comparing CRC result with expected data enable<br>0: Disable comparing function<br>1: Enable comparing function |

### (34) WCRCm KCRCm compare result register (WCRCm_KCRCm_COMP_RES)

WCRCm_KCRCm_COMP_RES is a register that indicates the result of comparing CRC for KCRCm module.
Note: m = 0, 1, 2, 3

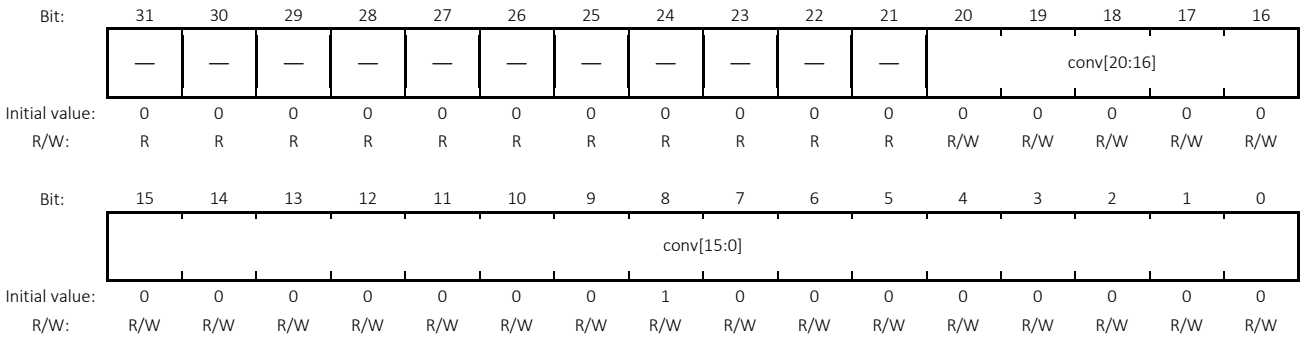



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 0 | cmp_res | H'0000 | R/WC1 | The result of comparing CRC<br>These bits indicate mismatched point of comparing data in 4-byte units.<br>0: No error<br>1: Mismatch<br>Figure 135.4 shows correspondence between comparing data and these bits. |

### (35) WCRCm KCRCm conversion setting register (WCRCm_KCRCm_CONV)

WCRCm_KCRCm_CONV is a register that sets CRC conversion size to once for KCRCm module.

Note: m = 0, 1, 2, 3

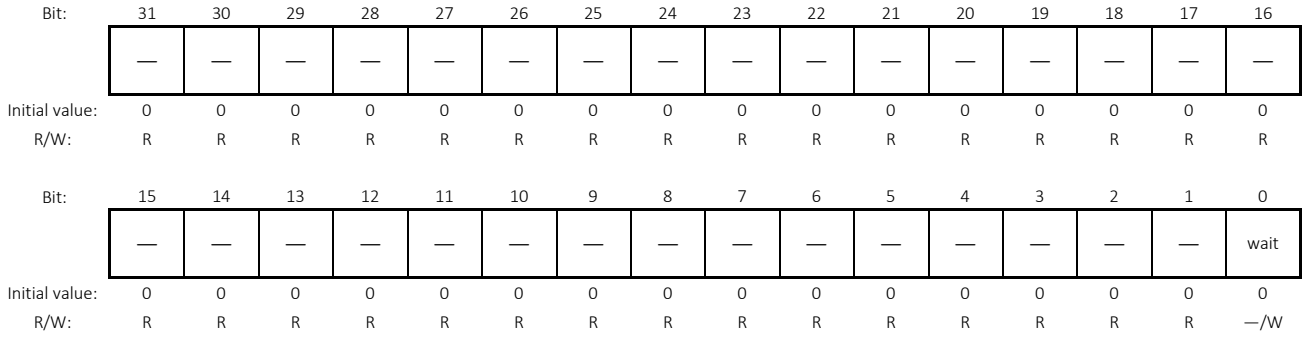



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 20 to 0 | conv | H'00 0100 | R/W | CRC conversion size specification<br>Specifies the CRC conversion size in 1-byte units.<br>Max:H'10 0000 |

### (36) WCRCm KCRCm wait register (WCRCm_KCRCm_WAIT)

WCRCm_KCRCm_WAIT is a register that wait subsequent command for KCRCm module.

Note: m = 0, 1, 2, 3



| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

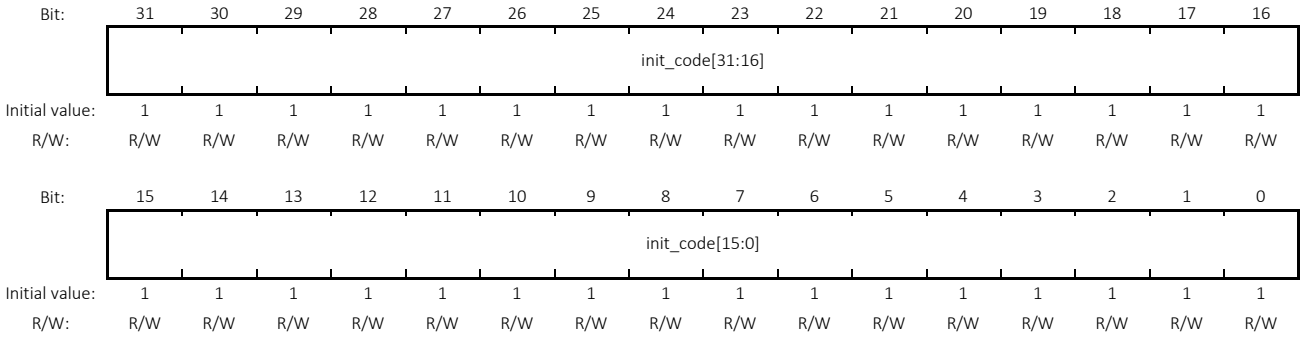
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | wait |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | —/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | wait | 0 | —/W | 1: Wait subsequent command for 128 cycles @ S0D2_RTφ.<br>This bit should be accessed by command function via command port for KCRCm in FIFO.<br>Otherwise, operation cannot be guaranteed.<br>Writing 0 to this bit is ignored. |

### (37) WCRCm KCRCm initial CRC code register (WCRCm_KCRCm_INIT_CRC)

WCRCm_KCRCm_INIT_CRC is a register that sets CRC code value to KCRCmCOUT register at auto clear.

Note: m = 0, 1, 2, 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | init_code | H'FFFF FFFF | R/W | CRC code value to KCRCmDOUT register at auto clear. |

### (38) WCRCm KCRCm status register (WCRCm_KCRCm_STS)

WCRCm_KCRCm_STS is a register that indicates the state of operation related to KCRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | stop_done | — | — | — | — | — | — | cmd_done | — | — | — | res_done | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/WC1 | R | R | R | R | R | R | R/WC1 | R | R | R | R/WC1 | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | comp_err | comp_done | — | — | — | — | — | — | — | — | — | — | — | trans_done |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/WC1 | R/WC1 | R | R | R | R | R | R | R | R | R | R | R | R/WC1 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | stop_done | 0 | R/WC1 | Indicates the state of stop operation by WCRCm_KCRCm_STOP register.<br>0: Stop operation in progress or not done.<br>1: Stop operation is done |
| 30 to 25 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 24 | cmd_done | 0 | R/WC1 | Indicates the state of command function.<br>0: Accessing to register by command in progress or not accessing<br>1: Accessing to register by command is done |
| 23 to 21 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 20 | res_done | 0 | R/WC1 | Indicates the state of result data transfer.<br>0: Transfer in progress or not transferring<br>1: Transfer is done |
| 19 to 14 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 13 | comp_err | 0 | R/WC1 | Indicates the error of comparing CRC.<br>0: No comparing error or not comparing<br>1: Mismatch has occurred |
| 12 | comp_done | 0 | R/WC1 | Indicates the state of comparing CRC.<br>0: Comparison in progress or not comparing<br>1: Comparing is done |
| 11 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 0 | trans_done | 0 | R/WC1 | Indicates the state of input data transfer.<br>0: Transfer in progress or not transferring<br>1: Transfer is done |

### (39) WCRCm KCRCm interrupt enable register (WCRCm_KCRCm_INTEN)

WCRCm_KCRCm_INTEN is a register that sets enable for the interrupt of operation related to KCRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | stop_done_ie | — | — | — | — | — | — | cmd_done_ie | — | — | — | res_done_ie | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R/W | R | R | R | R/W | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | comp_err_ie | comp_done_ie | — | — | — | — | — | — | — | — | — | — | — | trans_done_ie |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | stop_done_ie | 0 | R/W | Interrupt for the complete of stop operation by WCRCm_KCRCm_STOP register.<br>0: Disable interrupt<br>1: Enable interrupt |
| 30 to 25 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 24 | cmd_done_ie | 0 | R/W | Interrupt for the complete of command function enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 23 to 21 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | res_done_ie | 0 | R/W | Interrupt for the complete of result data transfer enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 19 to 14 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 13 | comp_err_ie | 0 | R/W | Interrupt for the error of comparing CRC enable<br>0: Disable interrupt<br>1: Enable interrupt |
| 12 | comp_done_ie | 0 | R/W | Interrupt for the complete of comparing CRC enable.<br>0: Disable interrupt<br>1: Enable interrupt |
| 11 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | trans_done_ie | 0 | R/W | Interrupt for the complete of input data transfer enable.<br>0: Disable interrupt<br>1: Enable interrupt |

### (40) WCRCm KCRCm ECM output enable register (WCRCm_KCRCm_ECMEN)

WCRCm_KCRCm_ECMEN is a register that sets enable for error notification to ECM module on operation related to KCRCm module.
Note: m = 0, 1, 2, 3

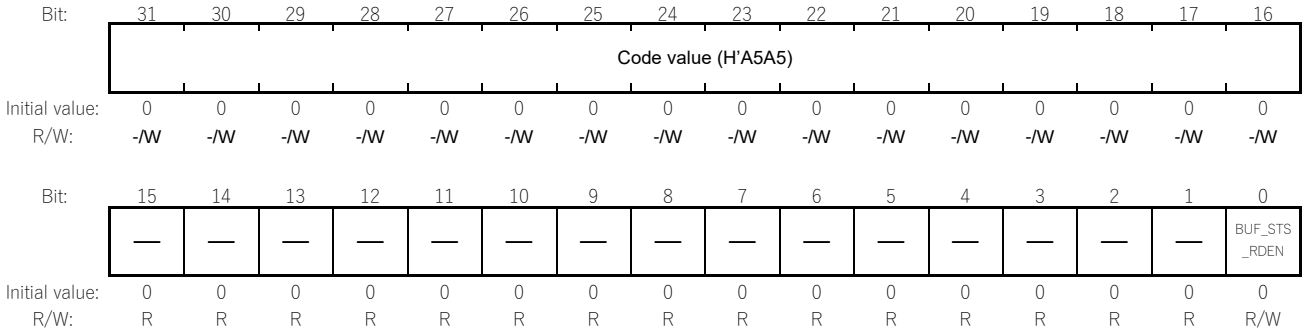
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | comp_ err_oe | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 13 | comp_err_oe | 0 | R/W | Error notification for the error of comparing CRC enable<br>0: Disable notification<br>1: Enable notification |
| 12 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (41) WCRCm KCRCm Buffer state read enable register (WCRCm_KCRCm_BUF_STS_RDEN)

WCRCm_KCRCm_BUF_STS_RDEN is a register that sets enable for the interrupt of operation related to KCRCm module.
Note: m = 0, 1, 2, 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Code value | All 0 | -/W | Code value (H'A5A5)<br>Set "Code value" to these bits when writing to this register to enable writing to this register. When read, return 0. |
| 15 to 1 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | BUF_STS_RDEN | 0 | R/W | WCRCm_KCRCm_BUF_STS enable.<br>0: Disable WCRCm_KCRCm_BUF_STS<br>1: Enable WCRCm_KCRCm_BUF_STS |

### (42) WCRCm KCRCm Buffer state read register (WCRCm_KCRCm_BUF_STS)

WCRCm_KCRCm_BUF_STS is a register that sets enable for the interrupt of operation related to KCRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | res_comp_endflag | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | buf_empty | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 19 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 18 | res_comp_endflag | 0 | R | Comparing CRC result end flag.<br>0: Comparing CRC result is not end<br>1: Comparing CRC result is end |
| 17 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 16 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 15 to 13 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 12 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 11 to 10 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 9 | — | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 8 | buf_empty | 0 | R | Buffer empty flag.<br>0: Buffer is not empty<br>1: Buffer is empty |
| 7 to 6 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (43) WCRCm common status register (WCRCm_COMMON_STS)

WCRCm_COMMON_STS is a register that indicates the state of operation related to WCRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | edc_err |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/WC1 |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |
| 16 | edc_err | 0 | R/WC1 | Indicates the error of EDC.<br>0: No EDC error<br>1: EDC error has occurred |
| 15 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value can be either 0 or 1. |

### (44) WCRCm common interrupt enable register (WCRCm_COMMON_INTEN)

WCRCm_COMMON_INTEN is a register that sets enable for the interrupt of operation related to WCRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | edc_err_ie |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 16 | edc_err_ie | 0 | R/W | Interrupt for the error of EDC enable<br>0: Disable interrupt<br>1: Enable interrupt |
| 15 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (45) WCRCm common ECM output enable register (WCRCm_COMMON _ECMEN)

WCRCm_COMMON_ECMEN is a register that sets enable for error notification to ECM module on operation related to WCRCm module.
Note: m = 0, 1, 2, 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | edc_err_oe |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

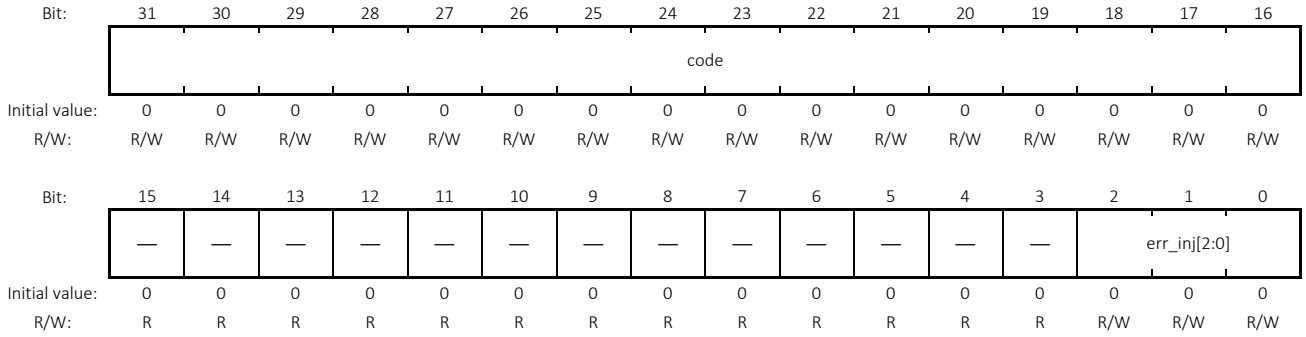
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 16 | edc_err_oe | 0 | R/W | Error notification for EDC error enable<br>0: Disable notification<br>1: Enable notification |
| 15 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (46) WCRCm error injection register (WCRCm_ERRINJ)

WCRCm_ERRINJ is a register that sets enable error injection for bus data signal or EDC code signal.

Note: m = 0, 1, 2, 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | code | H'0000 | R/W | Code value<br>Set H'A5A5 due to enable error injection. |
| 15 to 3 | — | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 2 to 0 | err_inj[2:0] | 0 | R/W | Error injection enable<br>Bit[0]:<br>0: Disables single bit error for bus data signal<br>1: Enables single bit error for bus data signal<br>Bit[1]:<br>0: Disables double bit error for bus data signal<br>1: Enables double bit error for bus data signal<br>Bit[2]:<br>0: Disables error for EDC code signal<br>1: Enables error for EDC code signal |

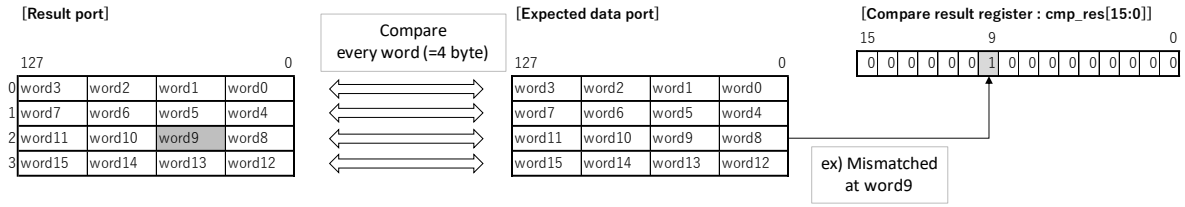



**Figure 135.4 Correspondence between comparing data and cmp_res**

### 135.1.3 Operation

#### (1) Independent CRC mode

The input data via the APB bus is directly transferred to CRCm or KCRCm module and operated on each module.
The operated data can be read via the APB bus.
For detail, refer to the chapter 135.2 CRC and 135.3 KCRC.

The procedure of this mode is as follows.

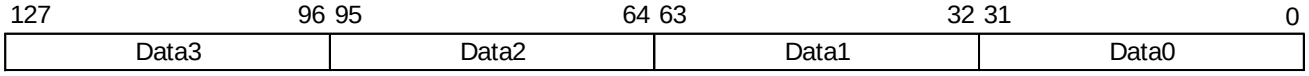
<For CRC module>
1. Set DCRAmCTL, DCRAmCTL2 registers.
2. Set initial value to DCRAmCOUT register.
3. Set input value to DCRAmCIN register.
4. Read out the operated data from DCRAmCOUT register.

<For KCRC module>
1. Set KCRCmCTL, KCRCmPOLY, KCRCmXOR registers.
2. Set initial value to KCRCmDOUT register.
3. Set input value to KCRCmDIN register.
4. Read out the operated data from KCRCmDOUT register.

### (2) E2E CRC mode

The 1 packet data is written to the data port in FIFO via the AXI bus, it is transferred to CRCm or KCRCm module according to their data size. And the operated data on each module can be read via the AXI bus.

(Data alignment on AXI Bus)




(Routing of data)

1. AXI (Write) -> FIFO (Data port) -> CRCm -> OUTPUT BUFFER -> FIFO (Result port) -> AXI (Read)
2. AXI (Write) -> FIFO (Data port) -> KCRCm -> OUTPUT BUFFER-> FIFO (Result port) -> AXI (Read)

The procedure of this mode is as follows.

1. Set CRC conversion size to once in WCRCm_XXXX_CONV register.
2. Set initial CRC code value in WCRCm_XXXX_INIT_CRC register.
3. (For CRC) Set DCRAmCTL, DCRAmCTL2, DCRAmCOUT registers.
   (For KCRC) Set KCRCmCTL, KCRCmPOLY, KCRCmXOR, KCRCmDOUT registers.
4. Set in_en=1, trans_en=1, res_en=1 in WCRCm_XXXX_EN register.
5. Set cmd_en=1 in WCRCm_XXXX_CMDEN register
6. Transfer input data to data port of FIFO by DMAC.
7. Read out result data from result port of FIFO by DMAC.
8. Set stop=1 in WCRCm_XXXX_STOP by command function.
9. Clear stop_done in WCRCm_XXXX_STS.

Note:If the data transfer must be stopped during the DMAC transfer, stop the data transfer by setting stop=1 in WCRCm_XXXX_STOP by CPU

### (3) AES-ACC mode

The 1 block data is written to the data port in FIFO via the AXI bus, it is transferred to AES-ACC module according to their data size. And the operated data on each module can be read via the AXI bus. The data alignment on AXI bus is same as E2E CRC mode.

(Routing of data)

1. AXI (Write) -> FIFO (Data port) -> (Local connect) -> AES-ACC_n (n=0,2,4,6) -> (Local connect) -> OUTPUT BUFFER -> FIFO(Result port) -> AXI (Read)

2. AXI (Write) -> FIFO (Data port) -> (Local connect) -> AES-ACC_p (p=1,3,5,7) -> (Local connect) -> OUTPUT BUFFER -> FIFO (Result port) -> AXI (Read)

The procedure of this mode is as follows.

1. Set in_en=1, trans_en=1, res_en=1 in WCRCm_XXXX_EN register.
2. Set cmd_en=1 in WCRCm_XXXX_CMDEN register
3. Transfer input data to data port of FIFO by DMAC.
4. Read out result data from result port of FIFO by DMAC.
5. Set stop=1 in WCRCm_XXXX_STOP by command function.
6. Clear stop_done in WCRCm_XXXX_STS.

Note:If the data transfer must be stopped during the DMAC transfer, stop the data transfer by setting stop=1 in WCRCm_XXXX_STOP by CPU

### (4) Data through mode

The written data to the data port in FIFO can be read directly via the AXI bus.
The data alignment on AXI bus is same as E2E CRC mode.

(Routing of data)

AXI (Write) -> FIFO (Data port) -> AXI (Read)

The procedure of this mode is as follows.

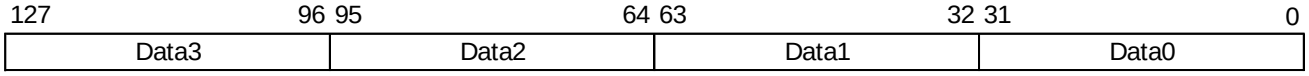
1. Set in_en=1, out_en=1 in WCRCm_XXXX_EN register.
2. Write input data to data port of FIFO by DMAC.
3. Read out input data from data port of FIFO by DMAC.
4. Set stop=1 in WCRCm_XXXX_STOP by CPU.
5. Clear stop_done in WCRCm_XXXX_STS.

Note: 1. In this mode, to set trans_en=1 in WCRCm_XXXX_EN register and use command function are prohibited.
2. The DMA transfer size (TS[3:0]) of read should be same as one of write.

### (5) E2E and data through mode

It is possible to read both the data written to the data port in FIFO and the operated data in CRCm/KCRCm/AES-ACC for that data simultaneously.

(Data alignment on AXI Bus)




(Routing of data)

1. AXI (Write) -> FIFO (Data port)
   -> CRCm -> OUTPUT BUFFER -> FIFO (Result port) -> AXI (Read)
   -> AXI (Read)

2. AXI (Write) -> FIFO (Data port)
   -> KCRCm -> OUTPUT BUFFER-> FIFO (Result port) -> AXI (Read)
   -> AXI (Read)

3. AXI (Write) -> FIFO (Data port)
   -> (Local connect)-> AES-ACC_n (n=0,2,4,6)-> (Local connect) ->Output buffer ->FIFO(Result port) ->AXI (Read)
   -> AXI (Read)

4. AXI (Write) -> FIFO (Data port)
   -> (Local connect) -> AES-ACC_p (p=1,3,5,7) -> (Local connect) ->Output buffer ->FIFO(Result port) ->AXI (Read)
   -> AXI (Read)

The procedure of this mode is as follows.

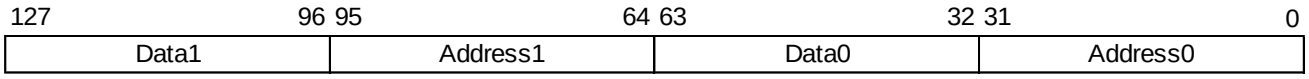
1. (For AES-ACC) Set registers in AES-ACC module.
2. (For CRC/KCRC) Set CRC conversion size to once in WCRCm_XXXX_CONV register.
3. (For CRC/KCRC) Set initial CRC code value in WCRCm_XXXX_INIT_CRC register.
4. (For CRC) Set DCRAmCTL, DCRAmCTL2, DCRAmCOUT registers.
   (For KCRC) Set KCRCmCTL, KCRCmPOLY, KCRCmXOR, KCRCmDOUT registers.
5. Set in_en=1, trans_en=1, res_en=1, out_en=1 in WCRCm_XXXX_EN register.
6. Set cmd_en=1 in WCRCm_XXXX_CMDEN register
7. Transfer input data to data port of FIFO by DMAC.
8. Read out input data from data port of FIFO by DMAC.
9. Read out result data from result port of FIFO by DMAC.
10. Set stop=1 in WCRCm_XXXX_STOP by command function.
11. Clear stop_done in WCRCm_XXXX_STS.

Note:If the data transfer must be stopped during the DMAC transfer, stop the data transfer by setting stop=1 in WCRCm_XXXX_STOP by CPU

### (6) Register access by command function

When 32bit address and 32bit data are written to the command port in FIFO via the AXI bus, the write register access is performed to the specified address for WCRC, CRC, KCRC or AES-ACC module. This function makes possible to set register with descriptor control from DMAC.

(Data alignment on AXI Bus)




(Routing of data)

1. AXI (Write) -> FIFO(Command port) -> CRCm
2. AXI (Write) -> FIFO(Command port) -> KCRCm
3. AXI (Write) -> FIFO(Command port) -> (Local connect) -> AES-ACC_n (n=0,2,4,6)
4. AXI (Write) -> FIFO(Command port) -> (Local connect) -> AES-ACC_p (p=1,3,5,7)

The procedure of this mode is as follows.

1. Set in_en=1, trans_en=1 in WCRCm_XXXX_EN register.
2. Set cmd_en=1 in WCRCm_XXXX_CMDEN register
3. Transfer command data to command port of FIFO by DMAC.

Note:If the data transfer must be stopped during the DMAC transfer, stop the data transfer by setting stop=1 in WCRCm_XXXX_STOP by CPU

### (7) Comparing CRC result

When the expected data of CRC calculation is written to the expected data port in FIFO via the AXI bus, this function compares the expected data with the CRC calculation result that was made in CRCm or KCRCm module until that time.
If they are corresponding, the both of expected data and calculated data are discarded.
And if they are difference, the interrupt occurs due to inform this mismatch.
To compare correctly, it is necessary to write the expected data after completing CRC calculation.
The data alignment on AXI bus is same as E2E CRC mode.

The procedure of this mode is as follows.

1. Write 0xa5a50001 to WCRCm_XXXX_BUF_STS_RDEN register.
2. Set CRC conversion size to once in WCRCm_XXXX_CONV register.
3. Set initial CRC code value in WCRCm_XXXX_INIT_CRC register.
4. Set cmp_en=1, exp_reqsel=1 in WCRCm_XXXX_COMP register.
5. Set cmd_done_ie in WCRCm_XXXX_INTEN register.
6. (For CRC) Set DCRAmCTL, DCRAmCTL2, DCRAmCOUT registers.
   (For KCRC) Set KCRCmCTL, KCRCmPOLY, KCRCmXOR, KCRCmDOUT registers.
7. Set in_en=1, trans_en=1, res_en=1 in WCRCm_XXXX_EN register.
8. Set cmd_en=1 in WCRCm_XXXX_CMDEN register
9. Transfer input data to data port of FIFO by DMAC.
10. Transfer expected data to expected data port of FIFO by DMAC.
11. Set stop=1 in WCRCm_XXXX_STOP by command function.
12. Wait Interrupt. After detecting Interrupt, verifying WCRCm_XXXX_STS register.
    If stop_done is set, goto No.14.
    If stop_done is not set, read WCRCm_KCRCm_BUF_STS register and wait until both BIT8 and BIT18 become “1”.
13. After both BIT8 and BIT17 become “1”, set stop=1 in WCRCm_XXXX_STOP by CPU.
14. Clear stop_done in WCRCm_XXXX_STS.
15. Read comp_err in WCRCm_XXXX_STS register.
16. If comp_err = 1, read WCRCm_XXXX_COMP_RES register.

Note:If the data transfer must be stopped during the DMAC transfer, stop the data transfer by setting stop=1 in WCRCm_XXXX_STOP by CPU

**(8)** FIFO

FIFO in this module is composed of 64bytes * 4 planes * 4 areas. Each area corresponds to AES-ACC_n (n=0,2,4,6), AES-ACC_p (p=1,3,5,7), CRCm, and KCRCm. And each plane corresponds to the port of data, command, expected data and result. Table 135.3 shows the base address of FIFO for each channel. And Table 135.4 shows the address assignment of each port in FIFO. Those are fixed address.

Table 135.3 Base address of FIFO

| Module Name | Base Address |
|---|---|
| WCRC0 | H'FFFF 0000 |
| WCRC1 | H'FFFF 4000 |
| WCRC2 | H'FFFF 8000 |
| WCRC3 | H'FFFF C000 |

Table 135.4 Address assignment of FIFO

| Communicate to | Port | Address (+Base Address) |
|---|---|---|
| AES-ACC_n (n=0,2,4,6) | Data | H'000 |
| | Command | H'100 |
| | (Reserved) | H'200 |
| | Result | H'300 |
| AES-ACC_p (p=1,3,5,7) | Data | H'400 |
| | Command | H'500 |
| | (Reserved) | H'600 |
| | Result | H'700 |
| CRCm (m=0,1,2,3) | Data | H'800 |
| | Command | H'900 |
| | Expected data | H'A00 |
| | Result | H'B00 |
| KCRCm (m=0,1,2,3) | Data | H'C00 |
| | Command | H'D00 |
| | Expected data | H'E00 |
| | Result | H'F00 |

### (9) Configuration of Descriptor Memory

The DMA transfer for each area of FIFO by using descriptor of RT-DMAC module is possible. In that case, it is necessary to write 16 Byte DMA setting to descriptor memory in RT-DMAC every step. Table 135.5 shows the correspondence of the address of descriptor memory to DMA setting.

Refer to section 30. RT-DMAC for the detail of control procedure for descriptor in RT-DMAC module.

The descriptor operates in the order descripted, so the out of order don't occur.

**Table 135.5 Correspondence of the address of descriptor memory to DMA setting**

| Offset Address | Bit | Content |
|---|---|---|
| H'0 | 31 to 0 | DMA Source Address |
| H'4 | 31 to 0 | DMA Destination Address |
| H'8 | 23 to 0 | DMA Transfer Count<br>The number of times of DMA transfer<br>H'0000 0001 : 1 time<br>H'00FF FFFF : 16,777,215 times (max) |
| H'C | 23 to 16 | DMARS[7:0]<br>For detail assignment, refer to section 30. |
| | 11, 10 | Destination address mode (DM[1:0])<br>Specify whether the DMA destination address is incremented or fixed, and the amount of increment.<br>00: Fixed destination address<br>01: Destination address is incremented<br>+ 4 in longword units transfer<br>+ 8 in 8-byte units transfer<br>+ 16 in 16-byte units transfer<br>+ 32 in 32-byte units transfer<br>+ 64 in 64-byte units transfer<br>10: Destination address is decremented<br>– 4 in longword units transfer<br>Setting prohibited in 8-/16-/32-/64-byte units transfer<br>11: Setting Prohibited |

| Offset Address | Bit | Content |
|---|---|---|
| | 9, 8 | Source Address Mode (SM[1:0])<br>Specify whether the DMA source address is incremented or fixed, and the amount of increment.<br>00: Fixed source address<br>01: Source address is incremented<br>+ 4 in longword units transfer<br>+ 8 in 8-byte units transfer<br>+ 16 in 16-byte units transfer<br>+ 32 in 32-byte units transfer<br>+ 64 in 64-byte units transfer<br>10: Source address is decremented<br>– 4 in longword units transfer<br>Setting prohibited in 8-/16-/32-/64-byte units transfer<br>11: Setting Prohibited |
| | 7, 6 | Resource selection (RS[3:2])<br>Specify which transfer requests will be sent to the RT-DMAC.<br>00: Setting prohibited<br>10: Selected by DMA extended resource selector<br>Other than above: Setting prohibited |
| | 4 | Descriptor step end interrupt enable（DSIE)<br>Specifies whether an interrupt request is generated to CPU at the time of termination of one step of Descriptor. When DSIE is set to 1, if DSE is set to 1, interruption (DEI) will be generated to CPU.<br>0: Interrupt request is disabled<br>1: Interrupt request is enabled |
| | 3 to 0 | DMA transfer size (TS[3:0])<br>(Data write to FIFO (Data port)/Data read from FIFO (Data port)/Data write to FIFO(Command port))<br>0010: Longword (4-byte) units transfer<br>0011: 16-byte units transfer<br>0100: 32-byte units transfer<br>0101: 64-byte units transfer<br>0111: 8-byte units transfer<br>Other than above: Setting prohibited<br>(Data read from FIFO (Result port))<br>0011: 16-byte units transfer<br>Other than above: Setting prohibited |

Figure 135.5 and Figure 135.6 shows the examples for operation with descriptor.

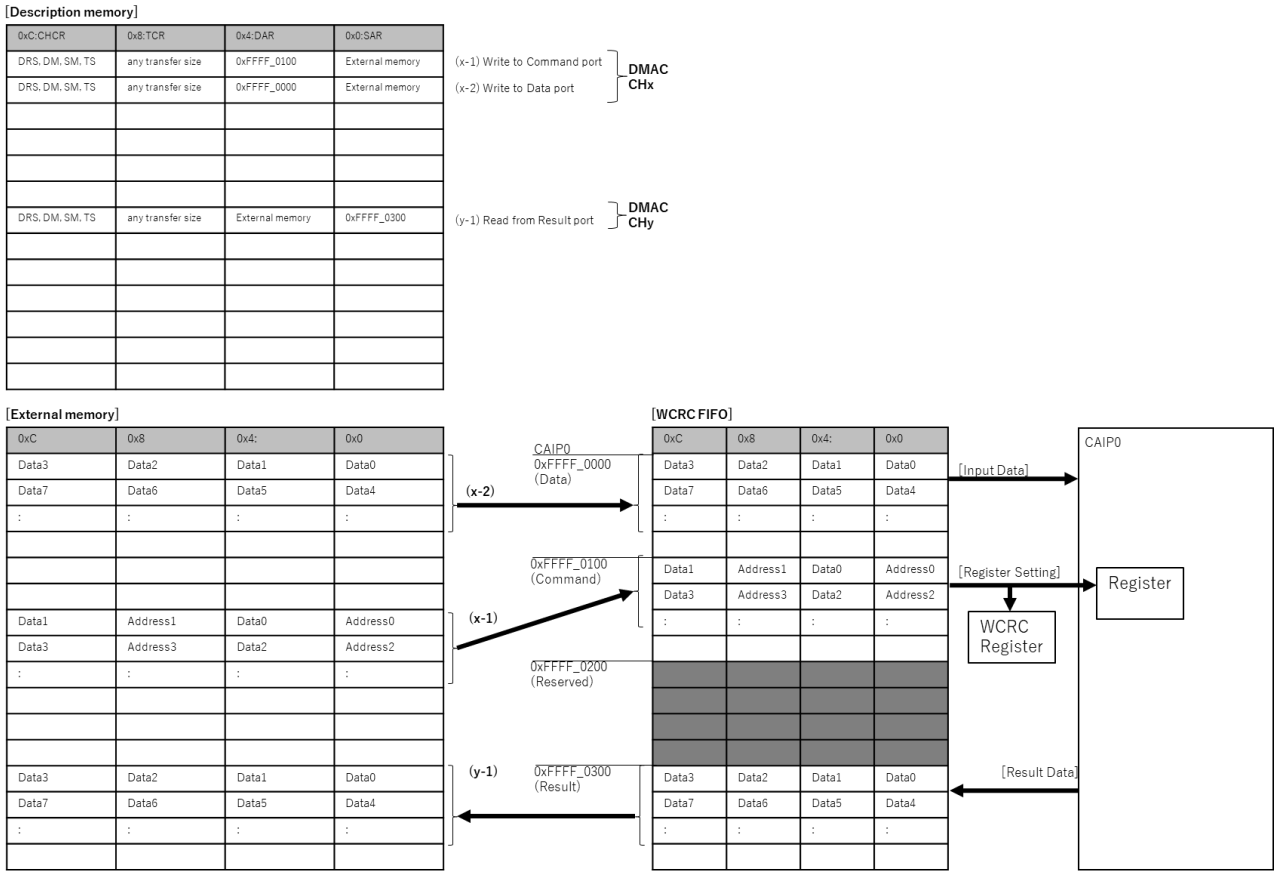



**Figure 135.5 Example of CAIP0 operation with descriptor.**

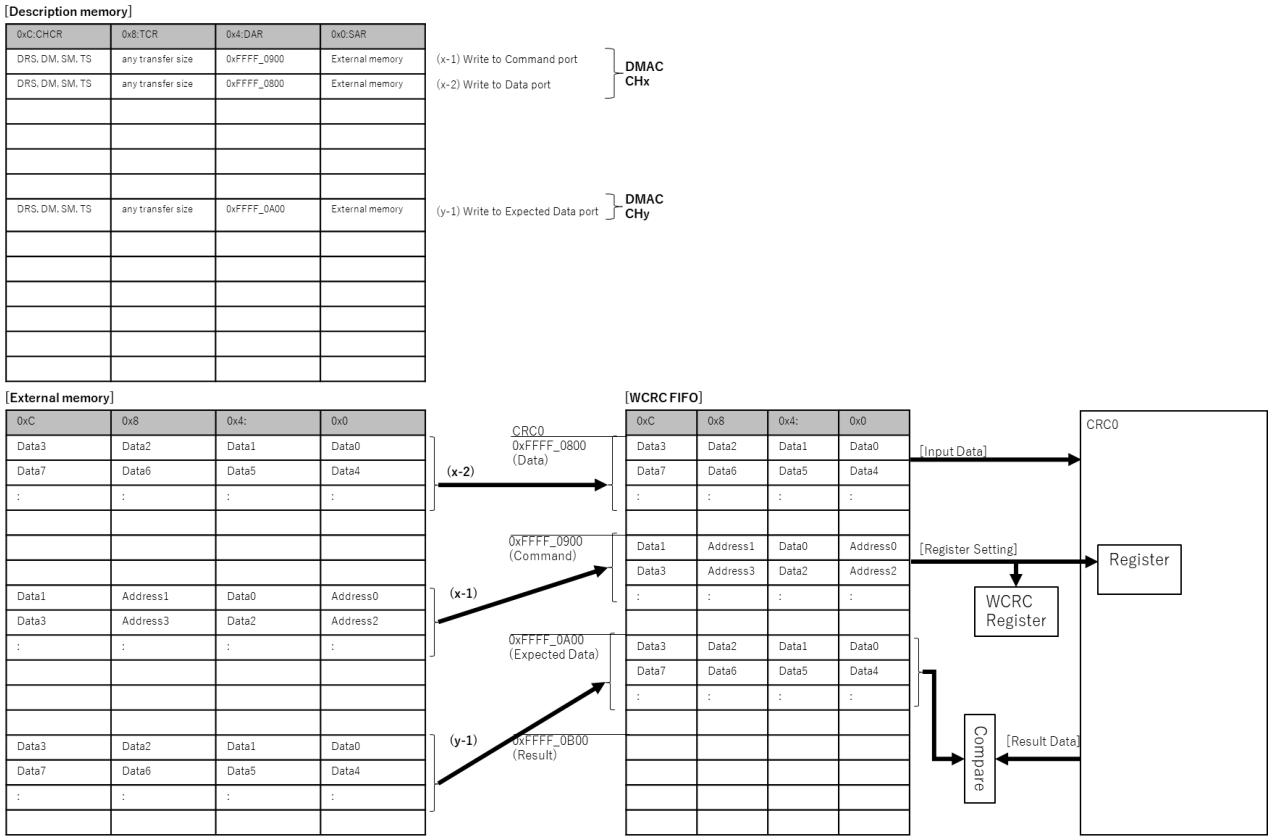



**Figure 135.6 Example of CRC comparing function with descriptor.**

### 135.1.4 Usage Notes

**(1) Note on start address for data**

Start address on memory where data is stored must be a 4-byte boundary.

**(2) Prohibition of simultaneous operation of Data through mode and Comparing CRC result**

There is no function of simultaneous operation of Data through mode (135.1.3.4) and Comparing CRC result (135.1.3.7). If you use simultaneous operation of Data through mode and Comparing CRC result, operation cannot be guaranteed.

**(3) Prohibition of simultaneous operation of Data through mode and Register access by command function**

There is no function of simultaneous operation of Data through mode (135.1.3.4) and Register access by command function (135.1.3.6). If you use simultaneous operation of Data through mode and Register access by command function, operation cannot be guaranteed.

### 135.1.5 Safety Mechanisms

WCRC is used only as module for Mechanism of FuSa.

## 135.2 CRC

### 135.2.1 Overview

The cyclic redundancy check block is a function block to generate and check CRC of input data via the APB bus.
The CRC based on this generator polynomial are stored in the register.
This module supports CRC calculation of Specification of CRC Routines of AUTOSAR Release 4.2.2.

**(1) Features**

This module supports the following 9 polynomials of below.

(1) 32-bit Ethernet polynomial (CRC-32-IEEE 802.3) (Specification of CRC Routines AUTOSAR Release 4.2.2)
$(X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1)$
The hamming distance of the code generated by this polynomial is 4 for 4KByte data.

(2) 16-bit CCITT -FALSE CRC16 (Specification of CRC Routines AUTOSAR Release 4.2.2)
$(X^{16} + X^{12} + X^5 + 1)$

(3) 8-bit SAE J1850 (8bit CRC code) (Specification of CRC Routines AUTOSAR Release 4.2.2)
$(X^8 + X^4 + X^3 + X^2 + 1)$

(4) 8-bit 0x2F polynomial (Specification of CRC Routines AUTOSAR Release 4.2.2)
$(X^8 + X^5 + X^3 + X^2 + X + 1)$

(5) 32-bit 0xF4ACFB13 polynomial (Specification of CRC Routines AUTOSAR Release 4.2.2)
$(X^{32} + X^{31} + X^{30} + X^{29} + X^{28} + X^{26} + X^{23} + X^{21} + X^{19} + X^{18} + X^{15} + X^{14} + X^{13} + X^{12} + X^{11} + X^9 + X^8 + X^4 + X^1 + 1)$
It has an advantage with respect to the Ethernet CRC - it has a Hamming Distance of 6 up to 4kB

(6) 32-bit 0x1EDC6F41 polynomial (CRC-32C) (Castagnoli)
$(X^{32} + X^{28} + X^{27} + X^{26} + X^{25} + X^{23} + X^{22} + X^{20} + X^{19} + X^{18} + X^{14} + X^{13} + X^{11} + X^{10} + X^9 + X^8 + X^6 + 1)$

(7) 21-bit 0x102899 polynomial (CRC-21)
$(X^{21} + X^{20} + X^{13} + X^{11} + X^7 + X^4 + X^3 + 1)$

(8) 17-bit 0x1685B polynomial (CRC-17)
$(X^{17} + X^{16} + X^{14} + X^{13} + X^{11} + X^6 + X^4 + X^3 + X^1 + 1)$

(9) 15-bit 0x4599 polynomial (CRC-15)
$(X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1)$

The CRC is a module for calculating the CRC code of the input data. The CPU or DMAC writes input data to the DCRAmCIN register. The CRC code is calculated for each data write cycle. By reading the DCRAmCOUT register value after all input data has been written to the DCRAmCIN register, CRC code value can be obtained.
Note: m = 0 to 3

### (1) Configuration of 32-bit Ethernet polynomial (CRC-32-IEEE 802.3) CRC calculation

Figure 135.7 shows the CRC calculation generator polynomial of the 32-bit Ethernet polynomial.
The source is AUTOSAR_SWS_CRCLibrary of AUTOSAR Release 4.2.2.

| | | |
|---|---|---|
| CRC result width | 32 bits | |
| Polynomial | 04C1 1DB7h | |
| Initial value | FFFF FFFFh | |
| Input data reflected | Yes | |
| Result data reflected | Yes | |
| XOR value | FFFF FFFFh | |

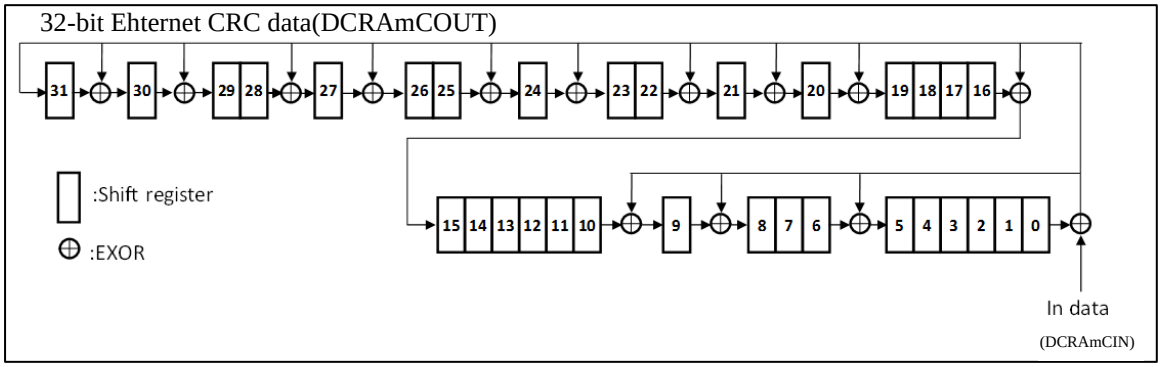



**Figure 135.7 32-Bit Ethernet**

### (2) Configuration of 16-bit CCITT-FALSE CRC16 CRC calculation

Figure 135.8 shows the CRC calculation generator polynomial of 16-bit CCITT-FALSE CRC16
The source is AUTOSAR_SWS_CRCLibrary of AUTOSAR Release 4.2.2.

| | |
|---|---|
| CRC result width | 16 bits |
| Polynomial | 1021h |
| Initial value | FFFFh |
| Input data reflected | No |
| Result data reflected | No |
| XOR value | 0000h |

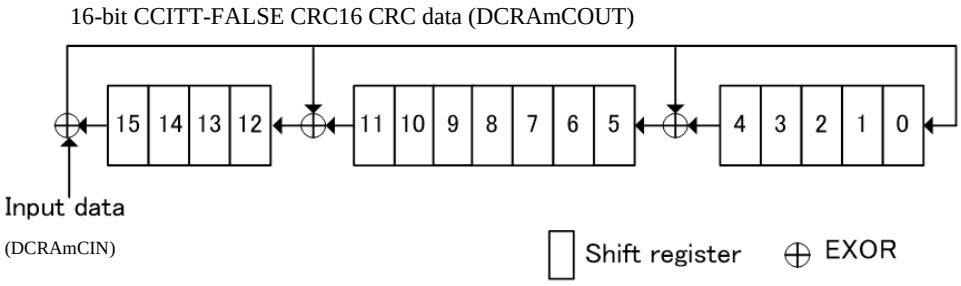



**Figure 135.8 16-bit CCITT-FALSE CRC16**

### (3) Configuration of 8-bit SAE J1850 CRCcalculation

Figure 135.9 shows the CRC calculation generator polynomial of 8-bit SAE J1850 (8bit CRC code)
The source is AUTOSAR_SWS_CRCLibrary of AUTOSAR Release 4.2.2.

| | |
|---|---|
| CRC result width | 8 bits |
| Polynomial | 1Dh |
| Initial value | FFh |
| Input data reflected | No |
| Result data reflected | No |
| XOR value | FFh |

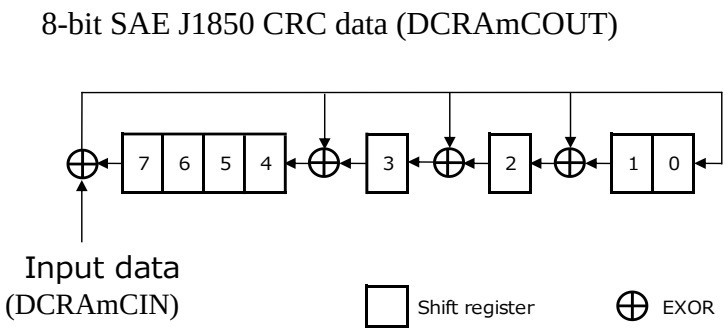



**Figure 135.9 8-bit SAE J1850 Polynomial**

### (4) Configuration of 8-bit 0x2F polynomial CRC calculation

Figure 135.10 shows the 8bit 0x2F CRC calculation generator polynomial of the 8-bit 0x2F polynomial.
The source is AUTOSAR_SWS_CRCLibrary of AUTOSAR Release 4.2.2.

| | |
|---|---|
| CRC result width | 8 bits |
| Polynomial | 2Fh |
| Initial value | FFh |
| Input data reflected | No |
| Result data reflected | No |
| XOR value | FFh |

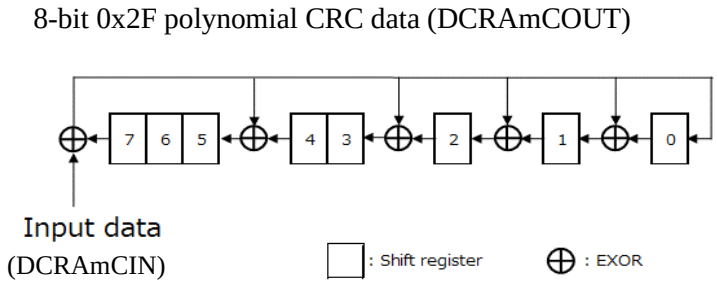



**Figure 135.10 8-bit 0x2F polynomial**

### (5) Configuration of 32-bit 0xF4ACFB13 polynomial CRC calculation

Figure 135.11 shows the CRC calculation generator polynomial of the 32-bit 0xF4ACFB13 polynomial. The source is AUTOSAR_SWS_CRCLibrary of AUTOSAR Release 4.2.2.
It has an advantage with respect to the Ethernet CRC - it has a Hamming Distance of 6 up to 4kB.

| | |
|---|---|
| CRC result width | 32 bits |
| Polynomial | F4AC FB13h |
| Initial value | FFFF FFFFh |
| Input data reflected | Yes |
| Result data reflected | Yes |
| XOR value | FFFF FFFFh |

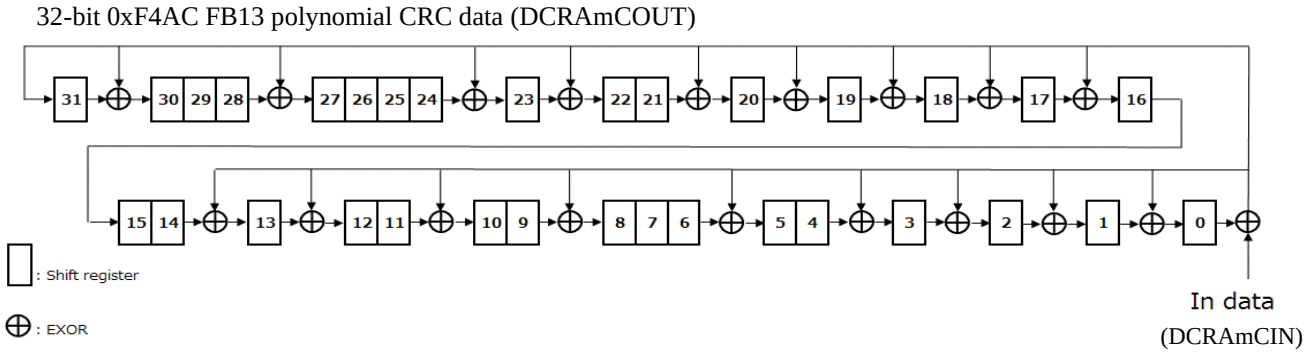



**Figure 135.11 32-bit 0xF4AC FB13 polynomial**

### (6) Configuration of 32-bit 0x1EDC6F41 polynomial (CRC-32C) (Castagnoli) CRC calculation

Figure 135.12 shows the CRC calculation generator polynomial of the 32-bit 0x1EDC6F41 polynomial.

| | |
|---|---|
| CRC result width | 32 bits |
| Polynomial | 1EDC 6F41h |
| Initial value | FFFF FFFFh |
| Input data reflected | Yes |
| Result data reflected | Yes |
| XOR value | FFFF FFFFh |

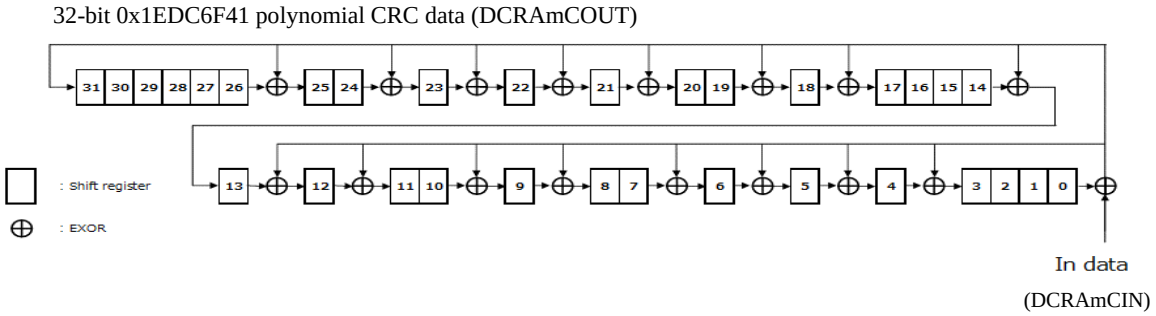



**Figure 135.12 32-bit 0x1EDC6F41 polynomial**

### (7) Configuration of 21-bit 0x102899 polynomial CRC-21 CRC calculation

Figure 135.13 shows the CRC calculation generator polynomial of the 21-bit 0x102899 polynomial.

| | |
|---|---|
| CRC result width | 21 bits |
| Polynomial | 10 2899h |
| Initial value | 1F FFFFh |
| Input data reflected | Yes |
| Result data reflected | Yes |
| XOR value | 1F FFFFh |

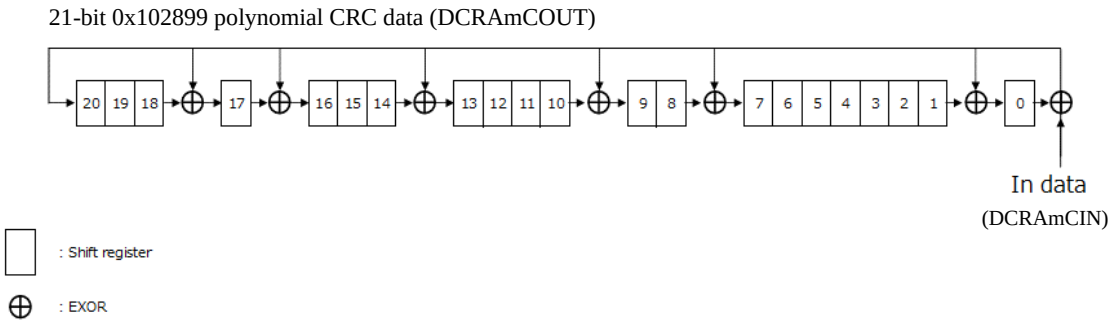



**Figure 135.13 21-bit 0x102899 polynomial**

### (8) Configuration of 17-bit 0x1685B polynomial (CRC-17) calculation

Figure 135.14 shows the CRC calculation generator polynomial of the 17-bit 0x1685B polynomial.

| | |
|---|---|
| CRC result width | 17 bits |
| Polynomial | 1 685Bh |
| Initial value | 1 FFFFh |
| Input data reflected | Yes |
| Result data reflected | Yes |
| XOR value | 1 FFFFh |

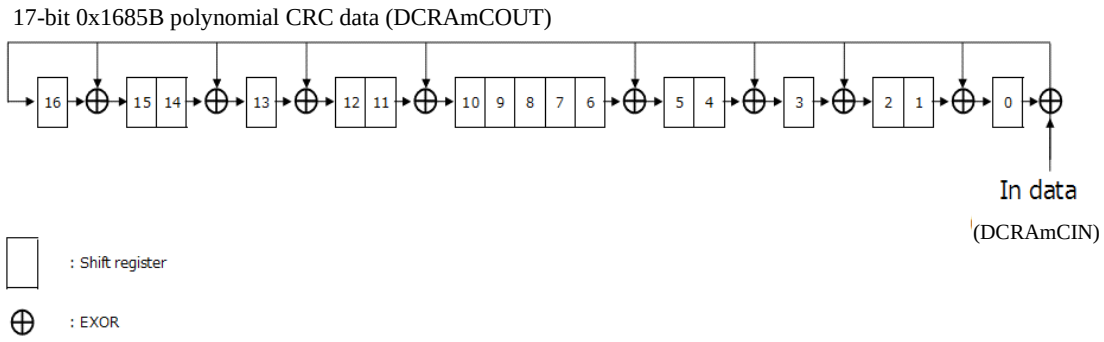



**Figure 135.14 17-bit 0x1685B polynomial**

### (9) Configuration of 15-bit 0x4599 polynomial (CRC-15) calculation

Figure 135.15 shows the CRC calculation generator polynomial of the 15-bit 0x4599 polynomial.

| | |
|---|---|
| CRC result width | 15bits |
| Polynomial | 4599h |
| Initial value | 7FFFh |
| Input data reflected | Yes |
| Result data reflected | Yes |
| XOR value | 7FFFh |

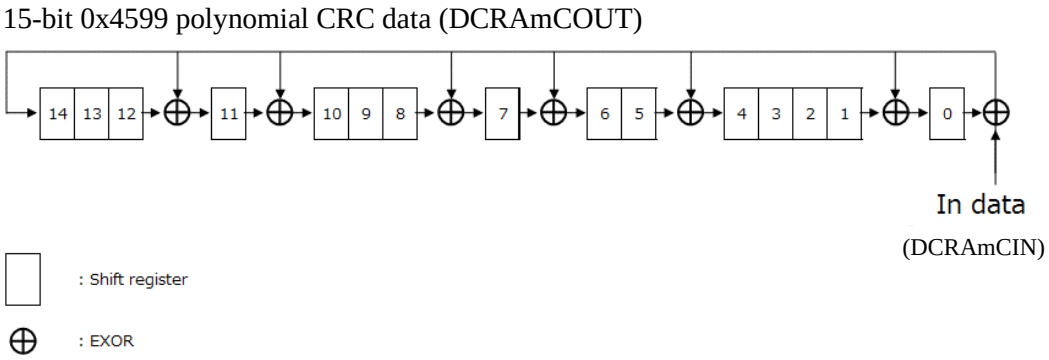



**Figure 135.15 15-bit 0x4599 polynomial**

### (2) Block Diagram

The CRC includes the internal registers and the CRC calculation block. Figure 135.16 shows the block diagram of the CRC.

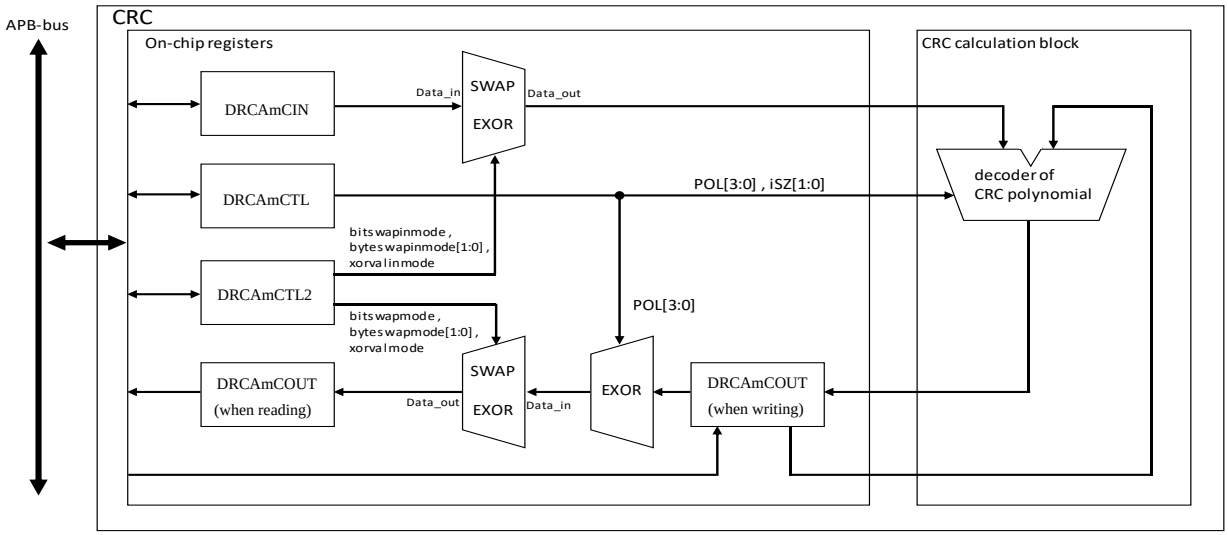



**Figure 135.16 CRC Block Diagram**

### (3) Register Configurations

The base address for each channel of registers of the CRC is as follows:

**Table 135.6 offset address Configurations**

| Module Name | Offset Base Address |
|---|---|
| CRC 0 | H'E6F0 0000 |
| CRC 1 | H'E6F1 0000 |
| CRC 2 | H'E700 0000 |
| CRC 3 | H'E701 0000 |

Each channel of the CRC has the registers below.

Table 135.7 shows the CRC register configurations and register value in each state.

**Table 135.7 Register Configurations**

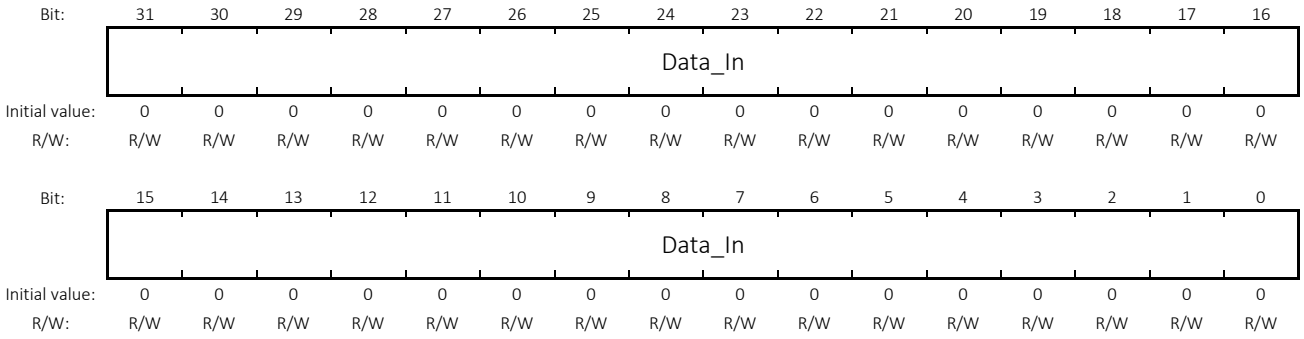
| Register Name | Abbreviation | R/W | Offset from Base Address | Initial Value (Power On Reset by PRESET# Pin) | Initial Value (Software Reset) | Access Size |
|---|---|---|---|---|---|---|
| CRCm Input Register | DCRAmCIN | R/W | H'0000 | H'0000 0000 | H'0000 0000 | 32 |
| CRCm Data Register | DCRAmCOUT | R/W | H'0004 | H'0000 0000 | H'0000 0000 | 32 |
| CRCm Control Register | DCRAmCTL | R/W | H'0020 | H'0000 0000 | H'0000 0000 | 32 |
| CRCm Control Register2 | DCRAmCTL2 | R/W | H'0040 | H'0000 0000 | H'0000 0000 | 32 |

Note: m = 0 to 3
Note: Access to the addresses other than those listed above is prohibited. Otherwise, operation cannot be guaranteed. If read, an undefined value is read.

### 135.2.2 Register Descriptions

#### (1) CRCm Input Register (DCRAmCIN)

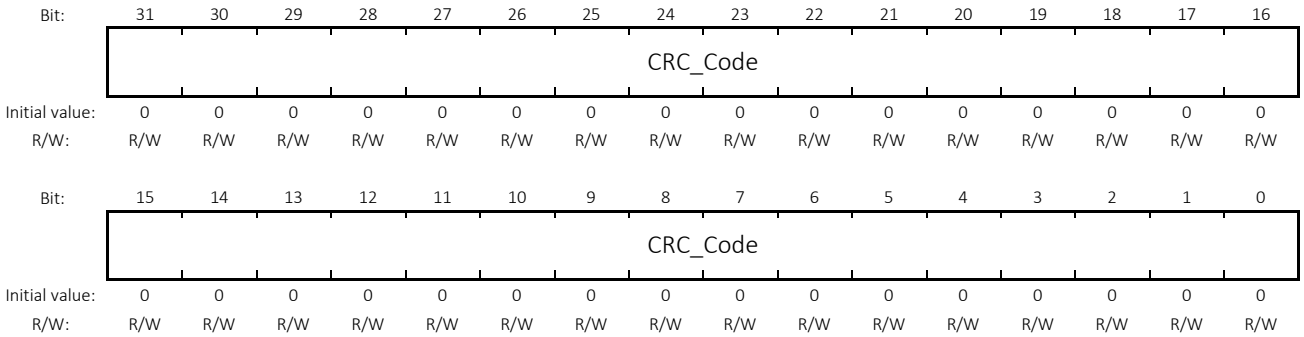
DCRAmCIN is a register that sets input data.
Note: m = 0 to 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Data_In | H'0000 0000 | R/W | Set input data<br>The valid bit position depends on the ISZ bit setting.<br>Bit 31 to 0: When 32-bit data width is enabled (ISZ[1:0] in DCRAmCTL = B'00). |

### (2) CRCm Data Register (DCRAmCOUT)

DCRAmCOUT is a register that obtains the CRC codes.
Note: m = 0 to 3




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CRC_Code | H'0000 0000 | R/W | CRC code<br>The valid bit width depends on the DCRAmCTL.POL bit setting.<br>When the 16-bit CCITT-FALSE CRC16 or<br>15bit 0x4599 polynomial (CRC-15) is enabled,<br>the bits 15 to 0 show the CRC result. The bits 31 to 16 are undefined.<br>When 8-bit SAE J1850 or 8-bit 0x2F polynomial is enabled,<br>bits 7 to 0 show the CRC result. The bits 31 to 8 are undefined.<br>On reading these bits, the value read out is the XOR of<br>each polynomial "XOR value" and the DCRAmCOUT.CRC_Code bits<br>(This does not affect the DCRAmCOUT.CRC_Code bits). |

Note: Although the hardware initial value of the CRC data register is H'FFFF FFFF, the read value is H'0000 0000. This is because the hardware value is exclusively ORed with H'FFFF FFFF before being read. Therefore, for CRC calculation, be sure to write H'FFFF FFFF to this register as default.
Note: This register must be initialized with the initial value corresponded by each polynomial before the first data of the data block is written to DCRAmCIN register.

### (3) CRCm Control Register (DCRAmCTL)

DCRAmCTL is a register that determines the valid bit width of the input data and generator polynomial for the CRC codes.

Note: m = 0 to 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ISZ[1:0] | | POL[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5, 4 | ISZ[1:0] | 00 | R/W | The value written to this field determines the valid bit width of input data.<br>2'b00: 32bit(DCRAmCIN31-0)<br>2'b01: 16bit(DCRAmCIN15-0)<br>2'b10: 8bit(DCRAmCIN7-0)<br>2'b11: Setting prohibited |
| 3 to 0 | POL[3:0] | 0000 | R/W | The value written to this field determines the generator polynomial for the CRC codes.<br>4'b0000: 32 bit Ethernet (CRC-32-IEEE 802.3)<br>4'b0001: 16-bit CCITT-FALSE CRC16<br>4'b0010: 8 bit SAE J1850<br>4'b0011: 8 bit 0x2F polynomial<br>4'b0100: 32 bit 0xF4ACFB13 polynomial<br>4'b0101: 32 bit 0x1EDC6F41 polynomial (CRC-32C) (Castagnoli)<br>4'b0110: 21 bit 0x102899 polynomial (CRC-21)<br>4'b0111: 17 bit 0x1685B polynomial (CRC-17)<br>4'b1000: 15 bit 0x4599 polynomial (CRC-15)<br>4'b1001: Reserved<br>4'b101x: Reserved<br>4'b11xx: Reserved |

Note: The value of this register should not be changed during data write to DCRAmCIN (prior to CRC code value calculation). If the value of this register is changed during data write, the generated CRC code value (DCRAmCOUT) is not guaranteed.

### (4) CRCm Control Register2 (DCRAmCTL2)

DCRAmCTL2 is a register that determines the swap, exor of input data and output data.
Note: m = 0 to 3

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | xorval mode | bitswap mode | byteswapmode [1:0] | | xorvalin mode | bitswap inmode | byteswapinmode [1:0] | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 7 | xorvalmode | 0 | R/W | The value written to this field determines EXOR ON of output data.<br>0: data out[31:0]<br>1: H'FFFF FFFF ^ data_out[31:0] |
| 6 | bitswapmode | 0 | R/W | The value written to this field determines bit swap of output data.<br>. |
| 5 to 4 | byteswapmode [1:0] | 00 | R/W | The value written to this field determines byte swap of output data. |
| 3 | xorvalinmode | 0 | R/W | The value written to this field determines EXOR ON of input data.<br>0: data_in[31:0]<br>1: H'FFFF FFFF ^ data_in[31:0] |
| 2 | bitswapinmode | 0 | R/W | The value written to this field determines bit swap of input data. |
| 1 to 0 | byteswapinmode [1:0] | 00 | R/W | The value written to this field determines byte swap of input data. |

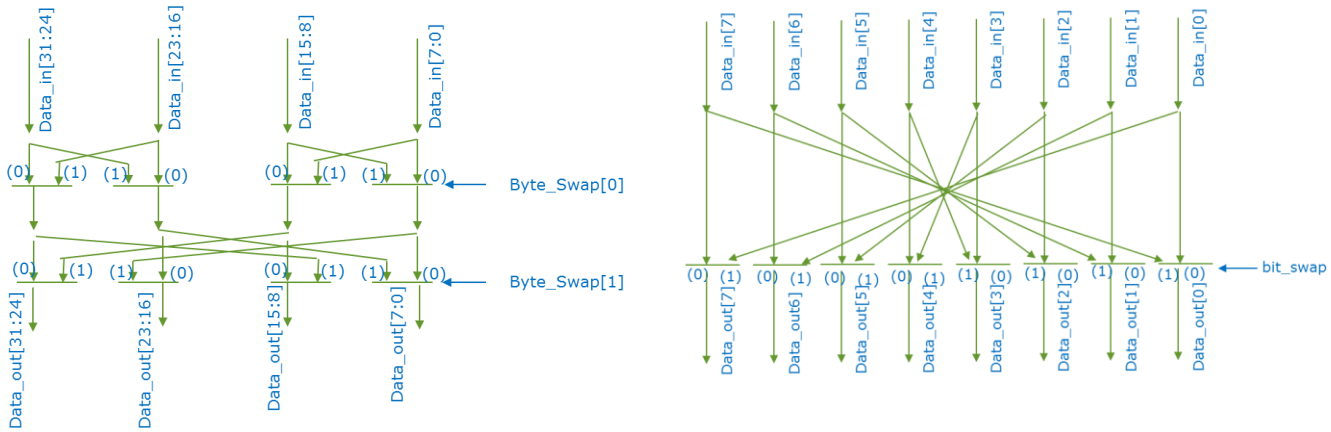



**Figure 135.17 Byte Swap and Bit Swap**

### 135.2.3 Operation

Please refer to 135.1.1.3-Operation.

### 135.2.4 Usage Notes

No description on this section

### 135.2.5 Safety Mechanisms

CRC is used only as module for Mechanism of FuSa

## 135.3 KCRC

### 135.3.1 Overview

The cyclic redundancy check block is a function block to generate and check CRC of input data via the APB bus. CRC based on this generator polynomial are stored in the register.

**(1) Features**

**Number of Units and Channels**

| | |
|---|---|
| Number of Units | 4 |
| Name | KCRCm (m = 0 to 3) |

**Function**

The KCRC can be used to verify or generate CRC protected data streams of arbitrary length and different bit widths.

- Supported CRC polynomials

(1) 32-bit Ethernet CRC
$04C11DB7_h$ : $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X^1+1$

(2) 16-bit CCITT CRC
$1021_h$ : $X^{16}+X^{12}+X^5+1$

(3) 8-bit SAE J1850 CRC
$1D_h$ : $X^8+X^4+X^3+X^2+1$

(4) 8-bit 0x2F CRC
$2F_h$ : $X^8+X^5+X^3+X^2+X+1$

(5) 32-bit CRC32C (Castagnoli)
$1EDC6F41_h$ : $X^{32}+X^{28}+X^{27}+X^{26}+X^{25}+X^{23}+X^{22}+X^{20}+X^{19}+X^{18}+X^{14}+X^{13}+X^{11}+X^{10}+X^9+X^8+X^6+X+1$

- CRC generation to an arbitrary data block length
- After initialization of the KCRC data register, every write access to the KCRC input register generates a new CRC according to the chosen polynomial and the result is stored in the KCRC data register.

### (2) Block Diagram

The KCRC includes the internal registers and the CRC calculation block. Figure 135.18 shows the block diagram of the KCRC.

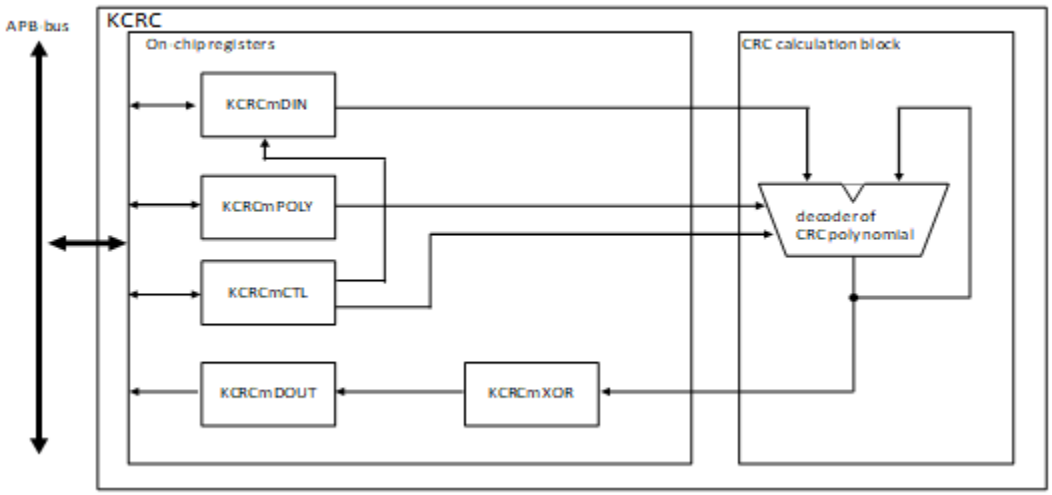



**Figure 135.18 KCRC Block Diagram**

### (3) Register Configurations

The base address for each channel of registers of the KCRC is as follows:

**Table 135.8 offset address Configurations**

| Module Name | Offset Base Address |
|---|---|
| KCRC 0 | H'E702 0000 |
| KCRC 1 | H'E703 0000 |
| KCRC 2 | H'E704 0000 |
| KCRC 3 | H'E705 0000 |

Each channel of the KCRC has the registers below.

**Table** 135.9 shows the KCRC register configurations and register value in each state.

**Table 135.9 Register Configurations**

| Register Name | Abbreviation | R/W | Offset from Base Address | Initial Value (Power On Reset by PRESET# Pin) | Initial Value (Software Reset) | Access Size |
|---|---|---|---|---|---|---|
| KCRCm Data Input Register | KCRCmDIN | R/W | H'00 | H'0000 0000 | H'0000 0000 | 32 |
| KCRCm Data Output Register | KCRCmDOUT | R/W | H'80 | H'0000 0000 | H'0000 0000 | 32 |
| KCRCm Control Register | KCRCmCTL | R/W | H'90 | H'001F 0000 | H'001F 0000 | 32 |
| KCRCm Polynomial Register | KCRCmPOLY | R/W | H'A0 | H'04C1 1DB7 | H'04C1 1DB7 | 32 |
| KCRCm XOR Mask Register | KCRCmXOR | R/W | H'B0 | H'FFFF FFFF | H'FFFF FFFF | 32 |

Note: Access to the addresses other than those listed above is prohibited. Otherwise, operation cannot be guaranteed. If read, an undefined value is read.

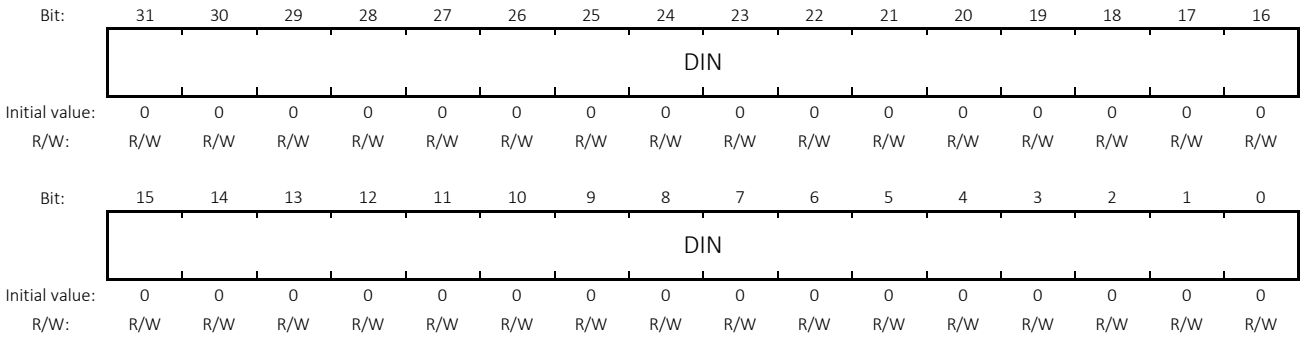
### (4) Connected Module

Table 135.10 Connected Modules

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| KCRC | APMU | Access the Register |
| | CPG | Output Clocks |
| | Module Standby | Control to stop clocks |
| | Software Reset | Execute software reset |
| | RT-DMAC | Control Direct Memory Access |

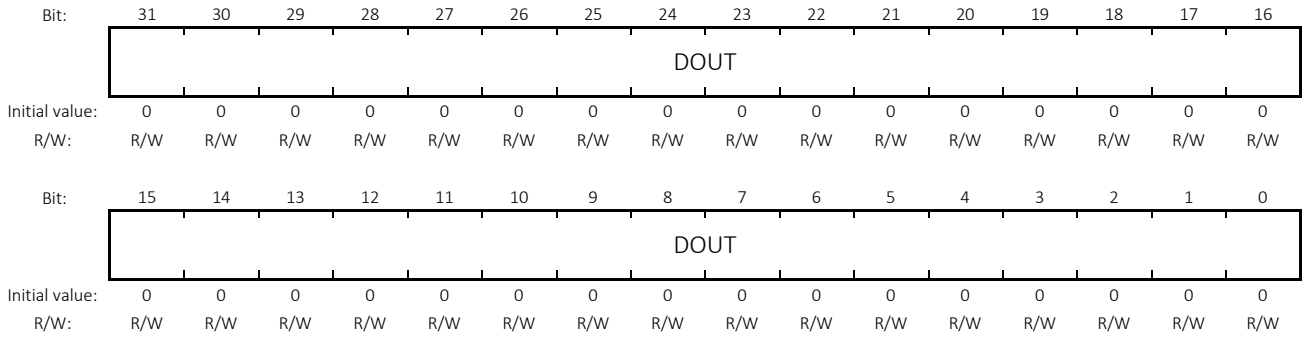
### 135.3.2 Register Descriptions

#### (1) KCRCm Data Input Register (KCRCmDIN)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DIN | H'0000 0000 | R/W | Input Data for CRC calculation.<br>The valid bits are controlled by KCRCmCTL.DW:<br>For 32 bit effective bit width: DIN[31:0]<br>For 16 bit effective bit width: DIN[15:0]<br>For 8 bit effective bit width: DIN[7:0] |

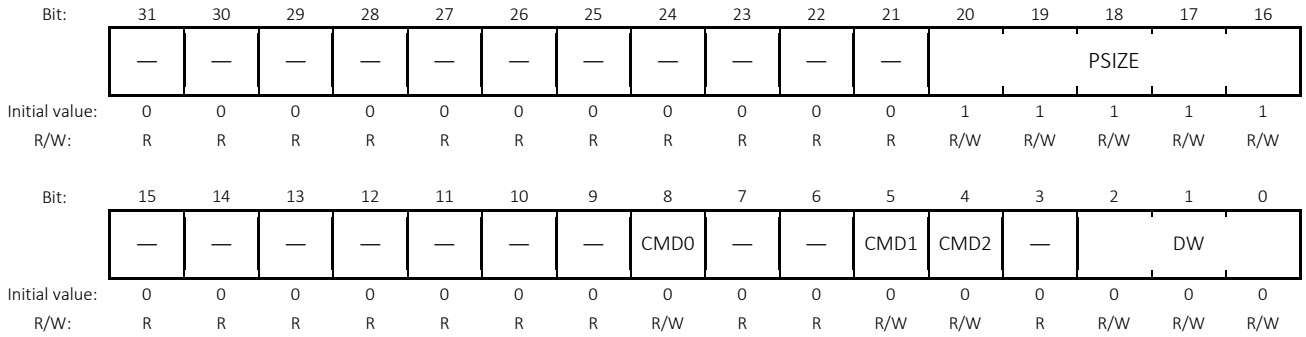
### (2) KCRCm Data Output Register (KCRCmDOUT)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DOUT | H'0000 0000 | R/W | Output Data for CRC calculation.<br>The valid bits are controlled by KCRCmCTL.DW:<br>For 32 bit effective bit width: DOUT[31:0]<br>For 16 bit effective bit width: DOUT[15:0]<br>For 8 bit effective bit width: DOUT[7:0]<br>Setting this register to initialize value. |

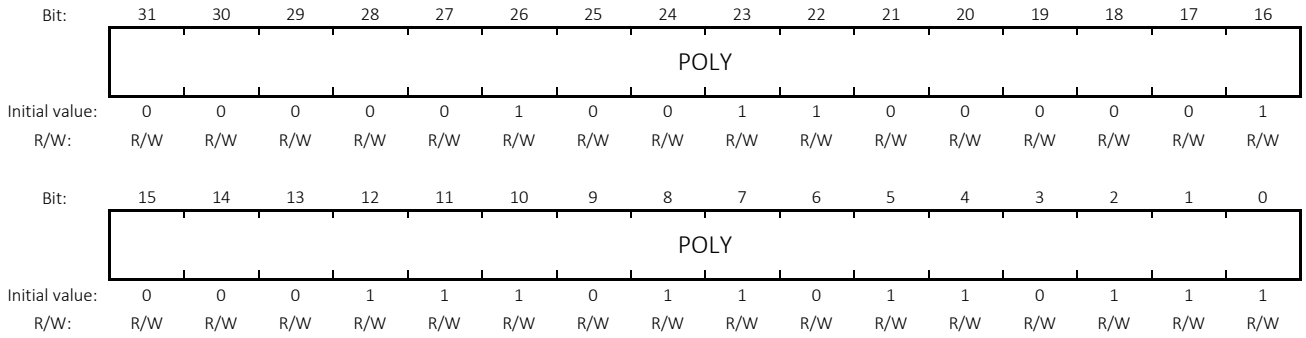
### (3) KCRCm Control Register (KCRCmCTL)

This register provided CRC calculate setting.



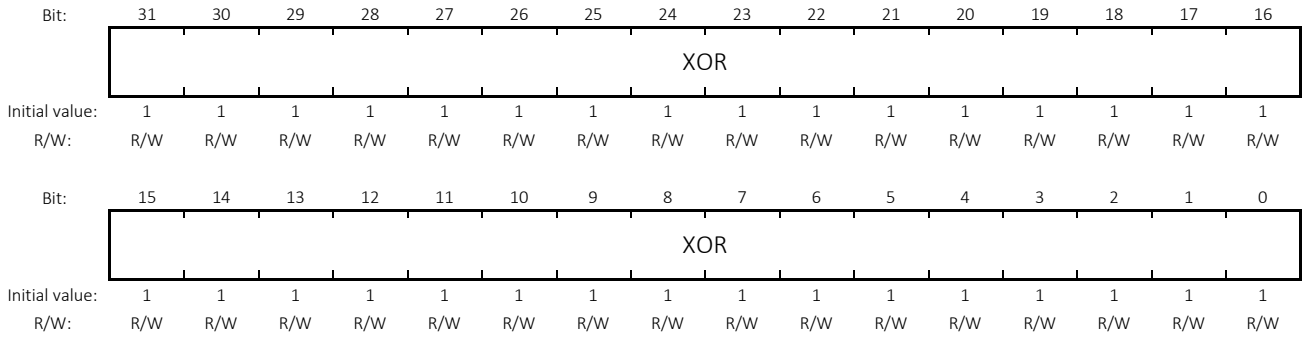

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 20 to 16 | PSIZE | H'1F | R/W | Polynomial size<br>0x7: 8 bit<br>0xF: 16 bit<br>0x1F: 32bit<br>Other: setting prohibit |
| 15 to 9 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CMD0 | 0 | R/W | Calculate Mode 0<br>0: Mode N (normal)<br>1: Mode R (output reflect) |
| 7 to 6 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 | CMD1 | 0 | R/W | Calculate Mode 1<br>0: Mode N (normal)<br>1: Mode R (input reflect) |
| 4 | CMD2 | 0 | R/W | Calculate Mode 2<br>0: Mode M (MSB shift)<br>1: Mode L (LSB shift) |
| 3 | - | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 2 to 0 | DW | 000 | R/W | Input Data size select<br>These bits define valid data width of Data input.<br>000: 32 bit fix mode<br>001: 16 bit fix mode<br>011: 8 bit fix mode<br>Other: Setting prohibit |

### (4) KCRCm Polynomial Register (KCRCmPOLY)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | POLY | H’04C1 1DB7 | R/W | Polynomial for CRC calculation. |

### (5) KCRCm XOR Mask Register (KCRCmXOR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | XOR | H'FFFF FFFF | R/W | XOR mask for Data output. |

### 135.3.3 Operation

Please refer to 135.1.1.3-Operation.

### 135.3.4 Usage Notes

No description on this section

### 135.3.5 Safety Mechanisms

KCRC is used only as module for Mechanism of FuSa

# 136. Failure Self-Detection Output(RFSO)

## 136.1 Overview

RFSO is a module which consists interval and time-out timers. This module can be used as a trigger and time-out detection for Runtime Test, trigger for Periodical Checks, internal watchdog timer, and for fault notification to ECM.

Internal clock: S0D6_PERφ

$T_{bus}$ (Period of S0D6_PERφ) = 7.50 ns
$f_{bus}$ (Frequency of S0D6_PERφ) = 133.33MHz

### 136.1.1 Features

#### 136.1.1.1 Error detection

**(1)** Output an interrupt request every specified time to initiate the routine that checks whether the target module has any failure. The interrupt request is called “Interval Timer Interrupt” that abbreviated “ITI”.

**(2)** Determine a failure in two steps either as a transient fault or a permanent fault due to a circuit failure or some other causes.

**(3)** Output a time-out interrupt (TOI) if no ITI is accepted, ITI is not completed or ITI is completed too early during the specified time window.

#### 136.1.1.2 Timers

**(1)** Interval timer for Runtime Test (Free run timer)

**(2)** Can be used to initiate the operation routine that checks whether the LSI chip has any failure at the specified time periodically.

**(3)** Time-out detection timer which detects either time-out error of Runtime Test or FTTI violation.

**(4)** Detects a time-out error when the time-out detection timer has not been stopped within the specified maximum value by Time-out detection timer time-out cycle register (CNT1_CYCm).

**(5)** Detects early timer stop when the time-out detection timer has been stopped earlier than the specified minimum value by Time-out detection timer stop minimum cycle register (CNT1_UNSm).

**(6)** Detects time-out error when time-out detection timer has not been stopped exceeding specified FTTI.

### 136.1.2 Block Diagram

This module includes the register interface and pin controller. Figure 136.1 show the block diagram of this module.

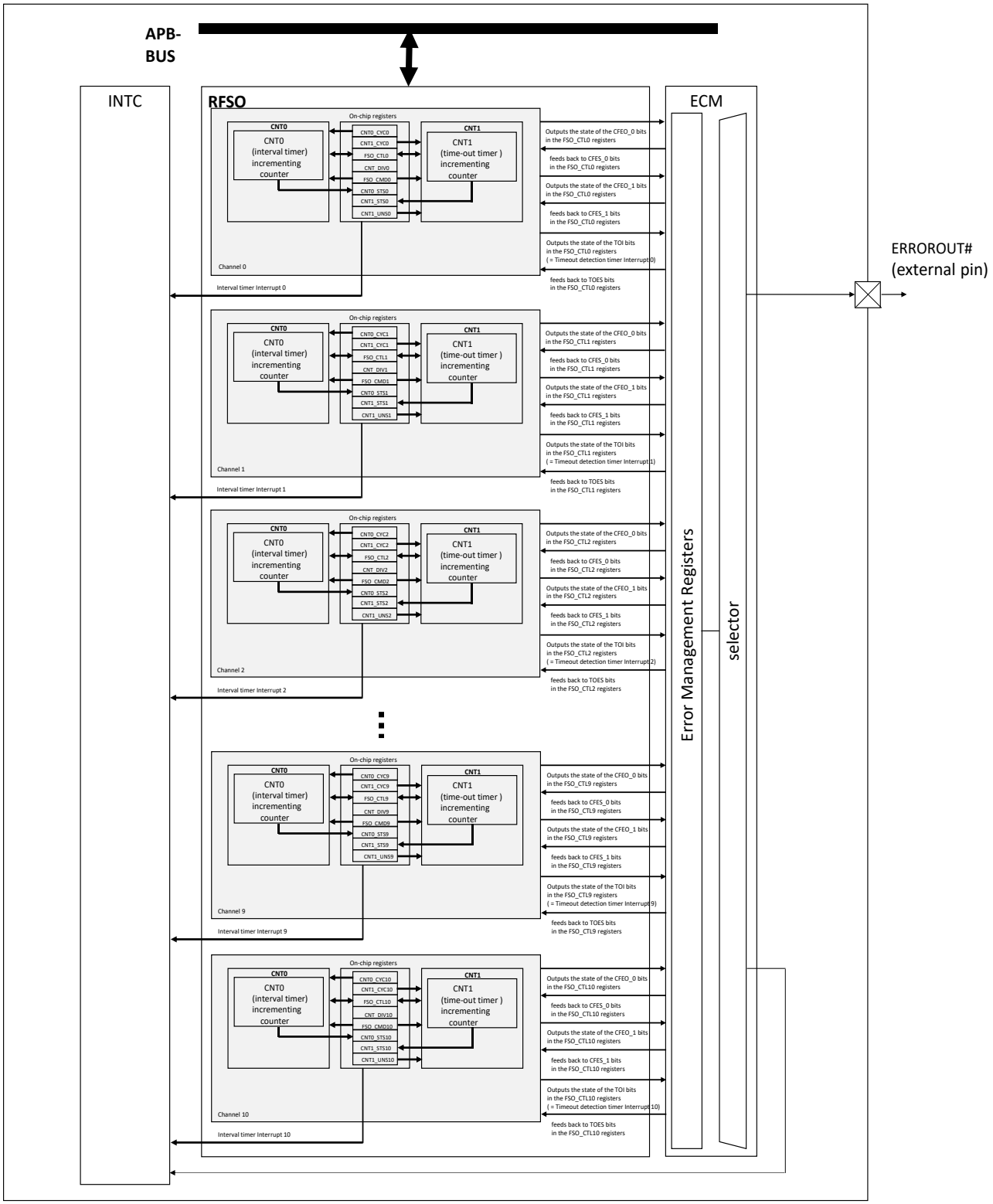



Figure 136.1 Block Diagram

### 136.1.3 Register Configurations

The base address for each channel of registers of the RFSO is as follows:

Table 136.1 Base Address Configurations

| Module Name | Base Address |
|---|---|
| RFSO 0 | H'FFE8 0000 |
| RFSO 1 | H'FFE8 1000 |
| RFSO 2 | H'FFE8 2000 |
| RFSO 3 | H'FFE8 3000 |
| RFSO 4 | H'FFE8 4000 |
| RFSO 5 | H'FFE8 5000 |
| RFSO 6 | H'FFE8 6000 |
| RFSO 7 | H'FFE8 7000 |
| RFSO 8 | H'FFE8 8000 |
| RFSO 9 | H'FFE8 9000 |
| RFSO 10 | H'FFE8 A000 |

Each channel of the RFSO has the registers below.
Table 136.2 shows the register configurations of this module.
Table 136.3 shows the register states in each operating mode.

Table 136.2 Register Configurations

| Register Name | Abbreviation | R/W | Offset from Base Address | Access Size |
|---|---|---|---|---|
| Interval timer cycle register m | CNT0_CYCm | R/W | H' 000 | 32 |
| Time-out detection timer time-out cycle register m | CNT1_CYCm | R/W | H' 004 | 32 |
| Control register m | FSO_CTLm | R/W | H' 008 | 32 |
| Timer counter clock frequency division register m | CNT_DIVm | R/W | H' 00C | 32 |
| Command register m | FSO_CMDm | -/WB | H' 010 | 32 |
| Interval timer status register m | CNT0_STSm | R | H' 014 | 32 |
| Time-out detection timer status register m | CNT1_STSm | R | H' 018 | 32 |
| Time-out detection timer operation minimum cycle register m | CNT1_UNSm | R/W | H' 01C | 32 |

Notes: 1. Access to the addresses other than those listed above is prohibited. Otherwise, operation cannot be guaranteed. If read, an undefined value is read.
2. m=0~10

Table 136.3 Register status in each operating mode

| Register Name | Abbreviation | Power On Reset by RESET# Pin |
|---|---|---|
| Interval timer cycle register m | CNT0_CYCm | H'0000 0000 |
| Time-out detection timer time-out cycle register m | CNT1_CYCm | H'0000 0000 |
| Control register m | FSO_CTLm | H'0000 0000 |
| Timer counter clock frequency division register m | CNT_DIVm | H'0000 0001 |
| Command register m | FSO_CMDm | H'0000 0000 |
| Interval timer status register m | CNT0_STSm | H'0000 0000 |
| Time-out detection timer status register m | CNT1_STSm | H'0000 0000 |
| Time-out detection timer operation minimum cycle register m | CNT1_UNSm | H'0000 0000 |

Notes: 1. To avoid wrong reset assertion, this module is designed so that reset signals are not asserted by a manual reset or module reset.
2. m=0~10

### 136.1.4 Connected Module

The base address for each channel of registers of the RFSO is as follows:
Table 136.4 shows the Connected Modules of this module.

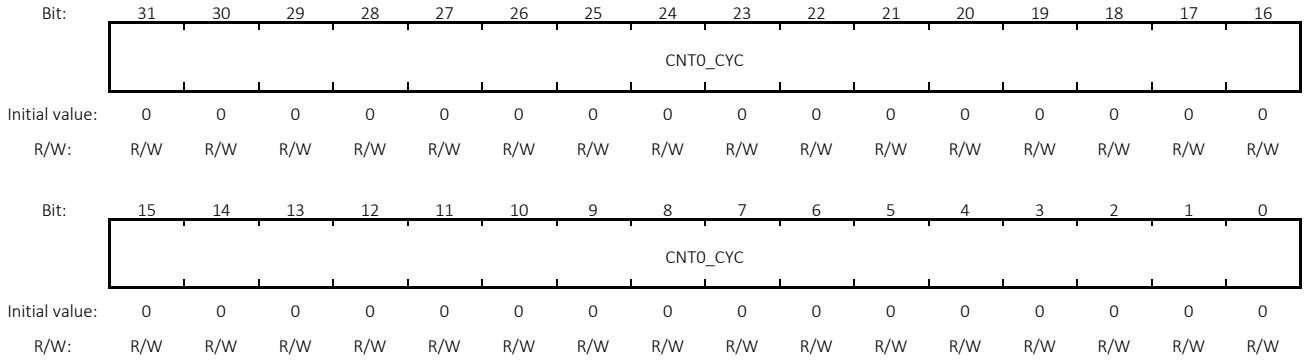
Table 136.4 Connected Modules

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| RFSO | CPG | Provide clock to the RFSO modules (Internal clock 133.33 MHz) |
| | RESET | Apply a reset to the RFSO target modules |
| | INTC | Provide interval timer interrupt to INTC |
| | ECM | Provide time-out detection timer interrupt to ECM |

## 136.2 Register Description

### 136.2.1 Interval Timer Cycle Register m (CNT0_CYCm) (m = 0 to 10)

CNT0_CYCm sets the periodical cycle value to the interval timer.

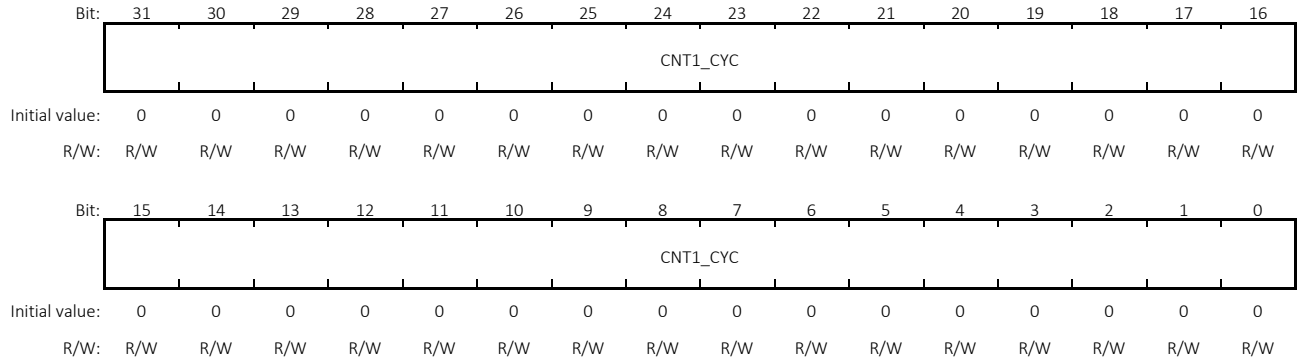



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CNT0_CYC | H'0000 0000 | R/W | Specify the periodical cycle value to the interval timer.<br>A Cycle is Timer counter clock interval.<br>H'FFFF FFFF is the Maximum cycle (4294967295=2^32-1).<br>The real time of the period interrupt as follows.<br>Timer counter clock interval × CNT0_CYC<br>= $T_{bus}$ × (CNT_DIV set value) × CNT0_CYC.<br>After this module is started, CNT0 starts incrementing the counter value. |

Note: Change of the value of CNT0_CYCm is inhibited during timers are in operation.
Note: Value H'0000 0000 is prohibited

### 136.2.2 Time-out Detection Timer Time-out Cycle Register m (CNT1_CYCm) (m = 0 to 10)

CNT1_CYCm sets the periodical cycle value to the time-out detection timer.

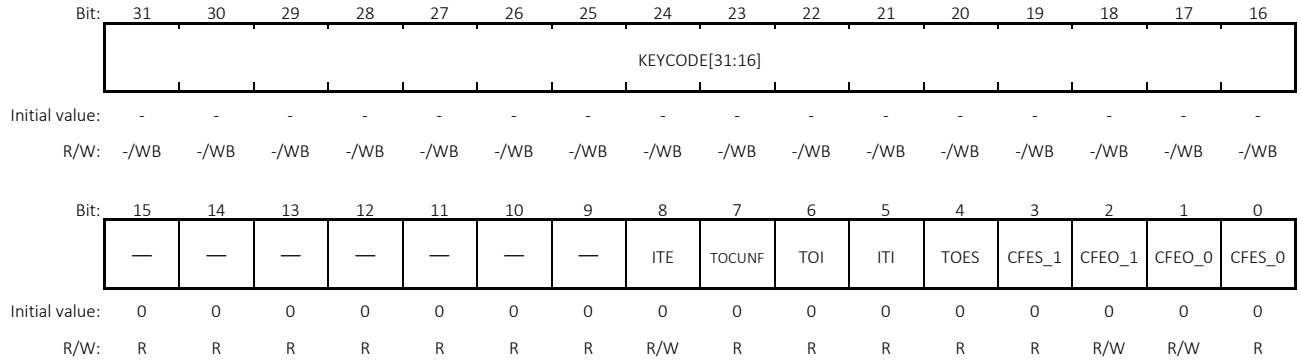



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CNT1_CYC | H'0000 0000 | R/W | Specify the periodical cycle value to the time-out detection timer.<br>A Cycle is Timer counter clock interval.<br>The initial value (H'0000 0000) is the Maximum cycle (4294967296=2^32).<br>The real time of the period interrupt as follows.<br>(When CNT1_CYC ≠ H'00000000)<br>Timer counter clock interval × CNT1_CYC<br>= $T_{bus}$ × (CNT_DIV set value) × CNT1_CYC.<br>(When CNT1_CYC = H'00000000)<br>Timer counter clock interval × H'100000000<br>= $T_{bus}$ × (CNT_DIV set value) × H'100000000.<br>After this module is started, CNT1 starts incrementing the counter value.<br>Note) CNT1_CYC should be set smaller than CNT0_CYC value. |

Note: Change of the value of CNT1_CYCm is inhibited during timers are in operation.

### 136.2.3 Control Register m (FSO_CTLm) (m = 0 to 10)

FSO_CTLm is a control register of this module.

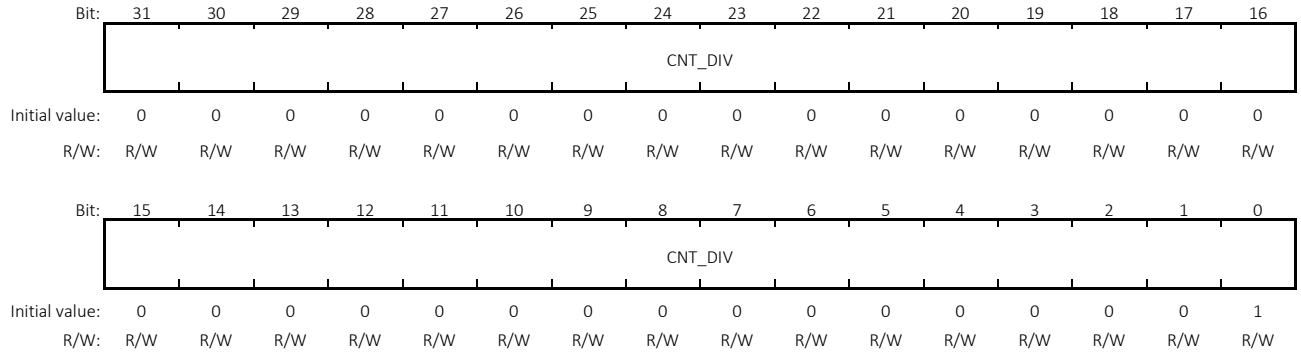



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | KEYCODE | - | -/WB | Register write enable code.<br>Set “KEYCODE” to these bits when writing to this register to enable writing to this register.<br>KEYCODE =H’779A. |
| 15 to 10 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 9 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | ITE | 0 | R/W | Enables or disables an interval timer (CNT0) interrupt request.<br>0: Disables the interrupt.<br>1: Enables the interrupt. |
| 7 | TOCUNF | 0 | R | Indicates whether a time-out detection timer (CNT1) has stopped when the value of a time-out detection timer (CNT1) is below the lower limit (CNT1_UNSm)<br>0: A time-out detection timer (CNT1) has stopped when the value of a time-out detection timer (CNT1) is greater than or equal to the lower limit (CNT1_UNSm)<br>1: A time-out detection timer (CNT1) has stopped when the value of a time-out detection timer (CNT1) is below the lower limit (CNT1_UNSm)<br>Note: When 1 is written in TOC bit of the FSO_CMDm register, this bit is cleared. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | TOI | 0 | R | Indicates whether a time-out detection timer (CNT1) has stopped under the condition that the value of a time-out detection timer (CNT1) is equal to the upper limit (CNT1_CYCm) or whether a time-out detection timer (CNT1) has stopped under the condition that the value of a time-out detection timer (CNT1) is blew the lower limit (CNT1_UNSm)<br>0: There is no interrupt.<br>1: There is an interrupt. (This bit is set when a time-out detection timer (CNT1) has stopped under the condition that the value of a time-out detection timer (CNT1) is blew the lower limit (CNT1_UNSm) or when a time-out detection timer (CNT1) has stopped under the condition that the value of a time-out detection timer (CNT1) is equal to the upper limit (CNT1_CYCm))<br>Note: When 1 is written in TOC bit of the FSO_CMDm register, this bit is cleared. |
| 5 | ITI | 0 | R | Indicates whether there is an interval timer (CNT0) interrupt request.<br>0: There is no interrupt.<br>1: There is an interrupt. (This bit is set when an interval timer (CNT0) has stopped under the condition that the value of an interval timer (CNT0) is equal to the upper limit (CNT0_CYCm))<br>Note: When 1 is written in ITC bit of the FSO_CMDm register, this bit is cleared.<br>Clear function is independent from the value of ITE. |
| 4 | TOES | 0 | R | Indicates that time-out error status is appropriately output to the ECM module.<br>This bit is cleared when a reset is applied or 1 is written to the TOC bit in the FSO_CMDm register thus clearing the time-out interrupt (TOI).<br>TOES bit is the feed back value from ECM module for TOI bit status of each channels.<br>This function is implemented in consideration of internal disconnection. |
| 3 | CFES_1 | 0 | R | Indicates that the CFEO_1 bit status is appropriately output to the ECM module.<br>CFES_1 bit is the feed back value from ECM module for CFEO_1 bit status of each channels.<br>This function is implemented in consideration of internal disconnection. |
| 2 | CFEO_1 | 0 | R/W | Sets the expected value verification result during test routine when the CFEO_0 bit is 1.<br>1 write: When result is different from the expectant value.<br>0 write: When result agrees with the expectant value. |
| 1 | CFEO_0 | 0 | R/W | Sets the expected value verification result during test routine.<br>1 write: When result is different from the expectant value.<br>0 write: When result agrees with the expectant value. |
| 0 | CFES_0 | 0 | R | Indicates that the CFEO_0 bit status is appropriately output to the ECM module.<br>CFES_0 bit is the feed back value from ECM module for CFEO_0 bit status of each channels.<br>This function is implemented in consideration of internal disconnection. |

### 136.2.4 Timer Counter Clock Frequency Division Register m (CNT_DIVm) (m = 0 to 10)

CNT_DIVm sets the interval timer/time-out detection timer counter clock frequency divisor.

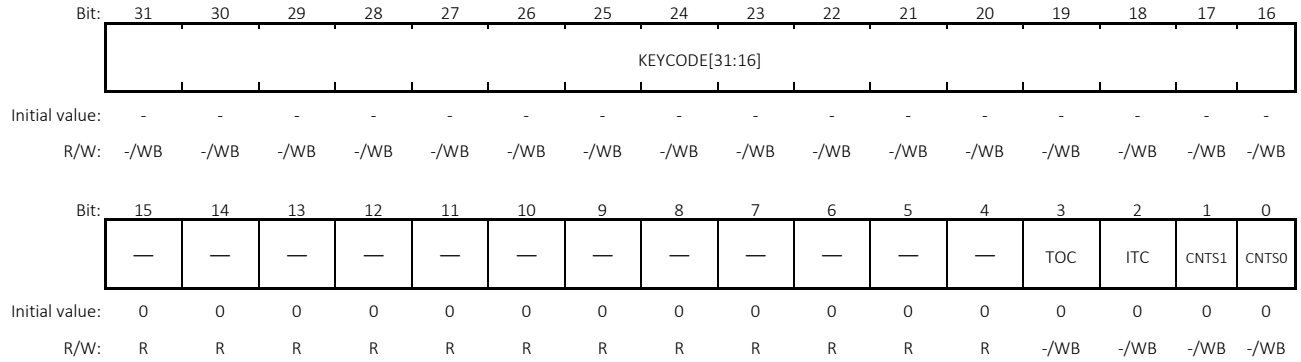



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CNT_DIV | H'0000 0001 | R/W | Set the timer counter clock frequency divisor.<br>Timer counter clock interval = operating clock of module $T_{bus}$ * (CNT_DIV set value).<br>CNT_DIV is not set to H'00000000 |

Note: Change of the value of CNT_DIVm is inhibited during timers are in operation.

### 136.2.5 Command Register m (FSO_CMDm) (m = 0 to 10)

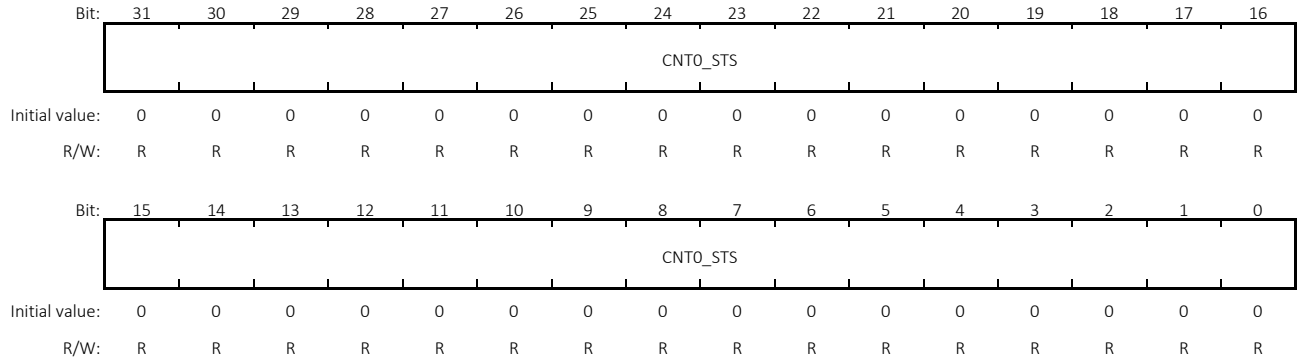
FSO_CMDm activates this module and clears the interrupts.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | KEYCODE | - | -/WB | Register write enable code.<br>Set “KEYCODE” to these bits when writing to this register to enable writing to this register.<br>KEYCODE =H’779A. |
| 15 to 4 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 3 | TOC | 0 | -/WB | Clears a time-out detection timer (CNT1) interrupt.<br>1 write: Clears the interrupt.<br>0 write: Invalid.<br>Writing 1 to this bit clears the TOI bit in the FSO_CTLm register to 0. This bit is always read as 0, not an undefined value. |
| 2 | ITC | 0 | -/WB | Clears an interval timer (CNT0) interrupt.<br>1 write: Clears the interrupt.<br>0 write: Invalid.<br>Writing 1 to this bit clears the ITI bit in the FSO_CTLm register to 0. This bit is always read as 0, not an undefined value. |
| 1 | CNTS1 | 0 | -/WB | Activates or stops the time-out detection timer (CNT1).<br>1 write: Activates the overrun error timer.<br>0 write: Stops the overrun error timer.<br>This bit is always read as 0, not an undefined value.<br>Note: When the timer is in operation, current operation is canceled and restarted loading initial value from CNT1_CYCm register by writing 1. When the timer is stopped, nothing occurs by writing 0. |
| 0 | CNTS0 | 0 | -/WB | Activates or stops the interval timer (CNT0).<br>1 write: Activates the interval timer.<br>0 write: Stops the interval timer.<br>This bit is always read as 0, not an undefined value.<br>Note: When the timer is in operation, current operation is continued even though writing 1. When the timer is stopped, nothing occurs by writing 0. |

### 136.2.6 Interval Timer Status Register m (CNT0_STSm) (m = 0 to 10)

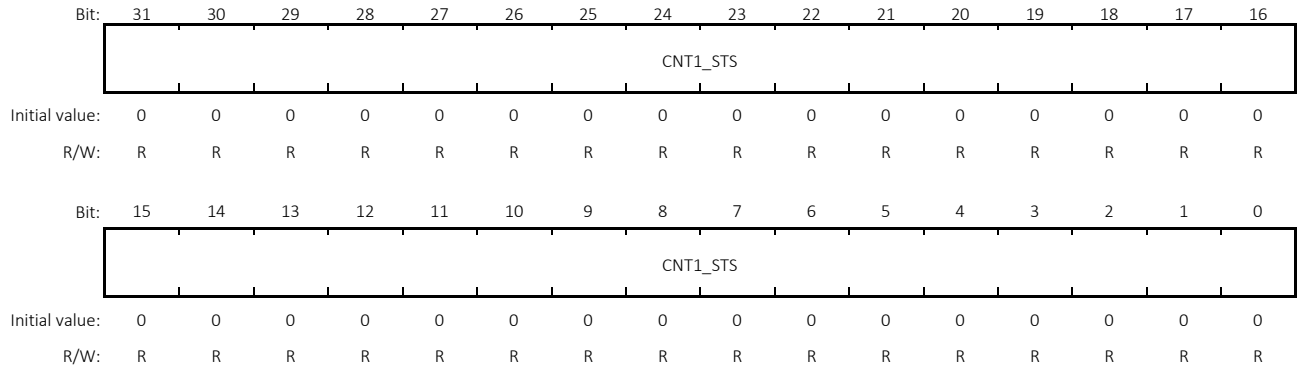
CNT0_STSm indicates the interval timer value.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CNT0_STS | H'0000 0000 | R | Indicates the interval timer value. |

### 136.2.7 Time-out Detection Timer Status Register m (CNT1_STSm) (m = 0 to 10)

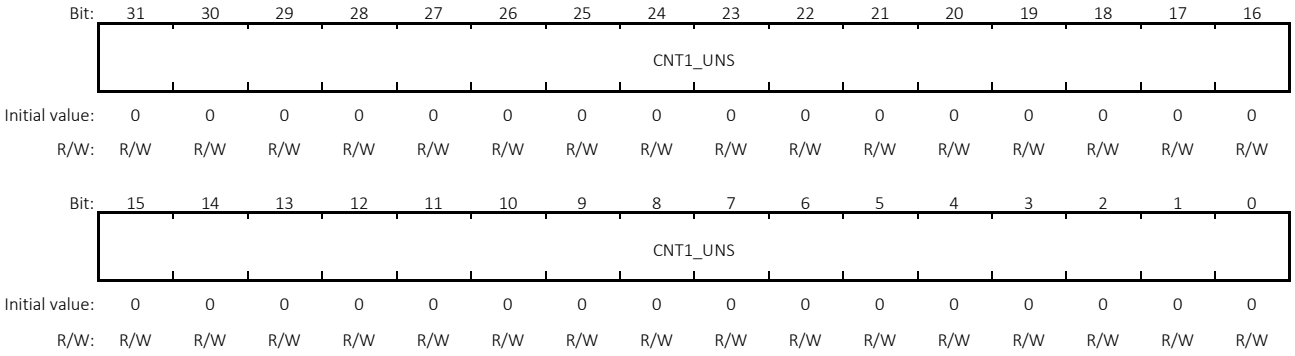
CNT1_STSm indicates the time-out detection timer value.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CNT1_STS | H'0000 0000 | R | Indicates the time-out detection timer value. |

### 136.2.8 Time-out detection timer operation minimum cycle register m (CNT1_UNSm) (m = 0 to 10)

Set time-out detection timer operation minimum cycle register.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CNT1_UNS | H'0000 0000 | R/W | Set minimum cycle value to stop Time-out detection timer.<br>The initial value is 0 cycle. CNT1_UNS is not set to 1. |

Note: Change of the value of CNT1_UNSm is inhibited during timers are in operation.

## 136.3 Operation

No description on this section

## 136.4 Usage Notes

No description on this section

## 136.5 Safety Mechanisms

### 136.5.1 Failure Self-DetectionOutoutput(RFSO)

**a) Overview**

Please refer to 136.1 Overview.

**b) Related Registers**

Please refer to 136.1.3 Register Configurations and 136.2 Register Description.

**c) Hardware Description**

Please refer to 136.3 Operation.

# 150. SD Card/MMC Interface

## 150.1 Overview

The SD Card/MMC Interface enables direct connection of SDHI or MMC to this LSI.

#### 150.1.1.1 Features

This LSI has 1 channel SD Card/MMC Interface.
The select setting of SDHI and MMC is unnecessary.

- SD memory/IO card interface (1-bit/4-bit SD bus)
- SD, SDHC, and SDXC SD memory card access supported
- Default, high-speed, UHS-I/SDR50, and SDR104 transfer modes supported
- SD clock (SDCLK) frequency = internal SDxϕ frequency/2^n (n = 0 to 9)
  SDCLK = 200 MHz (SDR104)
  SDCLK = 100 MHz (SDR50)
- Error check function: CRC7 (for command/response), CRC16 (for data)
- Interrupt request: 1
- Card detect function
- Write protect support
- MMC interface (1-bit/4-bit/8-bit MMC bus)
- e-MMC device access supported
- Backward-compatible, high-speed, HS200, and HS400 transfer modes supported
  SDCLK = 200MHz (HS200, HS400)
- High-priority interrupt (HPI) supported
- Alternative Boot operation supported
- Sequencer function (SEQ)
  The SEQ can issue maximum 8 commands sequentially.

Development of the SD-related products needs the conclusion of the following agreement.

SD Host/Ancillary Product License Agreement (SD HALA)

### 150.1.1.2 Block Diagram

Figure 150.1 shows a block diagram of the SD host interface.

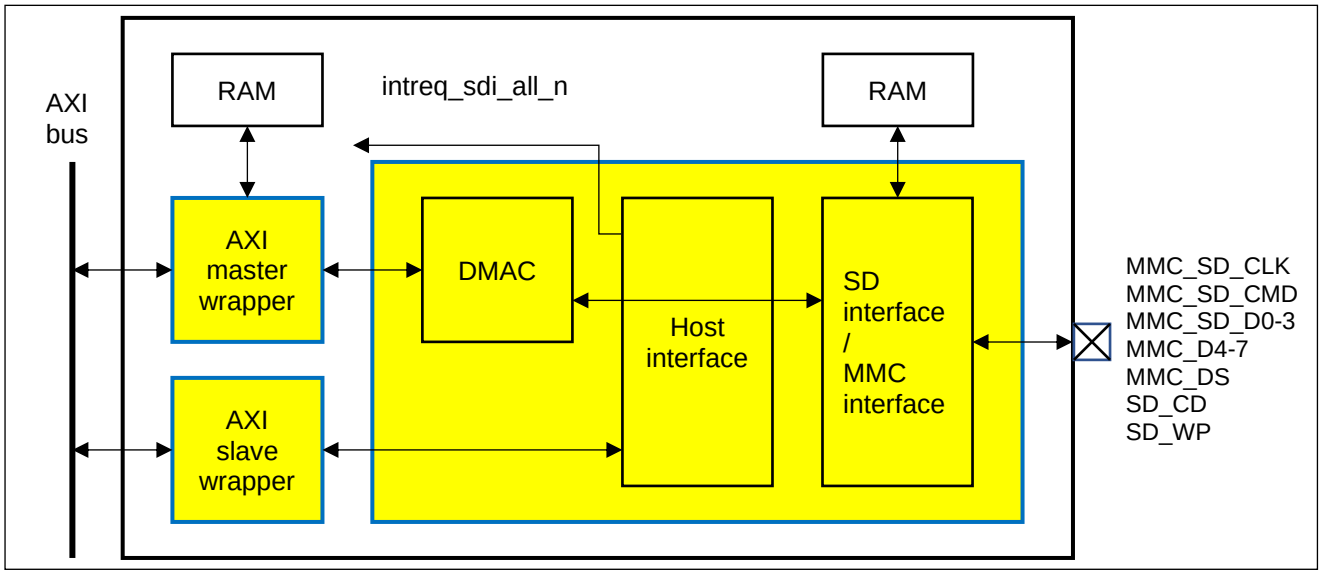



**Figure 150.1 Block Diagram of SD Host Interface**

### 150.1.1.3 External Pins

Table 150.1 lists the input and output pins used by the interface. The voltage of the SDHI terminal is variable.
Therefore a power network is separate. When choosing the function of PIO and the other ones, be careful.

**Table 150.1** Pin Configuration

| Name | Abbreviation | I/O | Function | R-Car V4H<br>R-Car V4M |
|---|---|---|---|---|
| MMC_SD_CLK | – | O | MMC clock | √*2 |
| MMC_SD_CMD | – | I/O | Command/response | √*2 |
| MMC_SD_D0 | – | I/O | Data [bit 0] | √*2 |
| MMC_SD_D1 | – | I/O | Data [bit 1] | √*2 |
| MMC_SD_D2 | – | I/O | Data [bit 2] | √*2 |
| MMC_SD_D3 | – | I/O | Data [bit 3] | √*2 |
| MMC_D4 | – | I/O | Data [bit 4] | √*3 |
| MMC_D5 | – | I/O | Data [bit 5] | √*3 |
| MMC_D6 | – | I/O | Data [bit 6] | √*3 |
| MMC_D7 | – | I/O | Data [bit 7] | √*3 |
| MMC_DS | – | I | Data Strobe | √*3 |
| SD_CD | – | I | Card detect *1 | √ |
| SD_WP | – | I | Write protect *1 | √ |

*1 When a pin is not in use, fix the corresponding signal to High.
*2 This is used by both of SDHI and MMC.
*3 This is used only MMC.

#### 150.1.1.4 Register Configuration

The base address for each interface are as follows. The select setting of SDHI and MMC is unnecessary.
The difference between SDHI and MMC is only the command type by Command type register (SD_CMD).

- SDHI: H’00 EE14 0000
- MMC: H’00 EE14 0000

Note: The pairs of interfaces (SDHI and MMC) can only be used for either the SDHI or MMC interface.

Table 150.2 summaries the registers of the interface.

**Table 150.2 Register Configurations**

| Name | Abbreviation | Address | Access Width | Mirror |
|---|---|---|---|---|
| Command type register | SD_CMD | H'0000 | 16/32/64 | |
| Port select register | SD_PORTSEL | H'0008 | 16/32/64 | |
| SD command argument registers | SD_ARG | H'0010 | 16/32/64 | |
| | SD_ARG1 | H'0018 | 16/32/64 | SD_ARG[31:16] |
| Data stop register | SD_STOP | H'0020 | 16/32/64 | |
| Block count register | SD_SECCNT | H'0028 | 16/32/64 | |
| SD card response registers | SD_RSP10 | H'0030 | 16/32/64 | |
| | SD_RSP1 | H'0038 | 16/32/64 | SD_RSP10[31:16] |
| | SD_RSP32 | H'0040 | 16/32/64 | SD_RSP10[63:32] |
| | SD_RSP3 | H'0048 | 16/32/64 | SD_RSP32[31:16] |
| | SD_RSP54 | H'0050 | 16/32/64 | |
| | SD_RSP5 | H'0058 | 16/32/64 | SD_RSP54[31:16] |
| | SD_RSP76 | H'0060 | 16/32/64 | SD_RSP54[63:32] |
| | SD_RSP7 | H'0068 | 16/32/64 | SD_RSP76[31:16] |
| SD card interrupt flag register 1 | SD_INFO1 | H'0070 | 16/32/64 | |
| SD card interrupt flag register 2 | SD_INFO2 | H'0078 | 16/32/64 | |
| SD_INFO1 interrupt mask register | SD_INFO1_MASK | H'0080 | 16/32/64 | |
| SD_INFO2 interrupt mask register | SD_INFO2_MASK | H'0088 | 16/32/64 | |
| SD clock control register | SD_CLK_CTRL | H'0090 | 16/32/64 | |
| Transfer data length register | SD_SIZE | H'0098 | 16/32/64 | |
| SD card access control option register | SD_OPTION | H'00A0 | 16/32/64 | |
| SD error status register 1 | SD_ERR_STS1 | H'00B0 | 16/32/64 | |
| SD error status register 2 | SD_ERR_STS2 | H'00B8 | 16/32/64 | |
| SD buffer read/write register | SD_BUF0 | H'00C0 | 16/32/64 | |
| SDIO mode control register | SDIO_MODE | H'00D0 | 16/32/64 | |
| SDIO interrupt flag register | SDIO_INFO1 | H'00D8 | 16/32/64 | |
| SDIO_INFO1 interrupt mask register | SDIO_INFO1_MASK | H'00E0 | 16/32/64 | |
| DMA mode enable register | CC_EXT_MODE | H'0360 | 16/32/64 | |
| Software reset register | SOFT_RST | H'0380 | 16/32/64 | |
| Version register | VERSION | H'0388 | 16/32/64 | |

| Name | Abbreviation | Address | Access Width | Mirror |
|---|---|---|---|---|
| Host interface mode setting register | HOST_MODE | H'0390 | 16/32/64 | |
| SD interface mode setting register | SDIF_MODE | H'0398 | 16/32/64 | |
| DMAC interrupt flag register 1 | DM_CM_INFO1 | H'0840 | 16/32/64 | |
| DM_CM_INFO1 interrupt mask register | DM_CM_INFO1_MASK | H'0848 | 16/32/64 | |
| DMAC interrupt flag register 2 | DM_CM_INFO2 | H'0850 | 16/32/64 | |
| DM_CM_INFO2 interrupt mask register | DM_CM_INFO2_MASK | H'0858 | 16/32/64 | |
| DMAC Sequencer Set Register | DM_CM_SEQ_REGSET | H'0800 | 16/32/64 | |
| DMAC Sequencer Control Register | DM_CM_SEQ_CTRL | H'0810 | 16/32/64 | |
| DMAC transfer mode register | DM_CM_DTRAN_MODE | H'0820 | 16/32/64 | |
| DMAC transfer control register | DM_CM_DTRAN_CTRL | H'0828 | 16/32/64 | |
| DMAC reset register | DM_CM_RST | H'0830 | 16/32/64 | |
| DMAC Status Register | DM_CM_SEQ_STAT | H'0868 | 16/32/64 | |
| DMAC transfer address register | DM_DTRAN_ADDR | H'0880 | 16/32/64 | |
| DMAC Sequencer Command Register | DM_SEQ_CMD | H'08A0 | 16/32/64 | |
| DMAC Sequencer Argument Register | DM_SEQ_ARG | H'08A8 | 16/32/64 | |
| DMAC Sequencer Size Set Register | DM_SEQ_SIZE | H'08B0 | 16/32/64 | |
| DMAC Sequencer Count Set register | DM_SEQ_SECCNT | H'08B8 | 16/32/64 | |
| DMAC Sequencer Response Register | DM_SEQ_RSP | H'08C0 | 16/32/64 | |
| DMAC Sequencer Response Check register | DM_SEQ_RSP_CHK | H'08C8 | 16/32/64 | |
| DMAC Sequencer Address Register | DM_SEQ_ADDR | H'08D0 | 16/32/64 | |

Table 150.3 summarizes the SCC registers. The base addresses of each channel are as follows.

- SDHI: H’00 EE14 1000
- MMC: H’00 EE14 1000

**Table 150.3 SCC Register Configurations**

| Name | Abbreviation | Address ([9:0]) | Access Size |
|---|---|---|---|
| Initial setting register | SCC_DTCNTL | H'000 | 32 |
| Sampling clock position setting register | SCC_TAPSET | H'008 | 32 |
| Hardware adjustment register 1 | SCC_DT2FF | H'010 | 32 |
| Sampling clock selection register | SCC_CKSEL | H'018 | 32 |
| Sampling clock position correction register | SCC_RVSCNTL | H'020 | 32 |
| Sampling clock position correction request register | SCC_RVSREQ | H'028 | 32 |
| Sampling data comparison register | SCC_SMPCMP | H'030 | 32 |
| Hardware adjustment register 2 | SCC_TMPPORT2 | H'038 | 32 |
| Hardware adjustment register 3 | SCC_TMPPORT3 | H'050 | 32 |
| Hardware adjustment register 4 | SCC_TMPPORT4 | H'058 | 32 |
| Hardware adjustment register 5 | SCC_TMPPORT5 | H'060 | 32 |
| Hardware adjustment register 6 | SCC_TMPPORT6 | H'068 | 32 |

| Name | Abbreviation | Address ([9:0]) | Access Size |
|---|---|---|---|
| Hardware adjustment register 7 | SCC_TMPPORT7 | H'070 | 32 |

## 150.2 Register Description

### 150.2.1.1 Command Type Register (SD_CMD)

The command type register (SD_CMD) is used to select the command type and response type. The command sequence is started by writing to SD_CMD.

For details on the SD_CMD setting, refer to section 150.3.6.15, Example of SD_CMD and MMC_CMD Register Setting.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 | MD7 | 0 | R/W | Multiple Block Transfer Mode (enabled at multiple block transfer) |
| 14 | MD6 | 0 | R/W | 00: CMD12 is automatically issued at multiple block transfer.<br>01: CMD12 is not automatically issued at multiple block transfer.<br>10: Setting prohibited<br>11: Setting prohibited |
| 13 | MD5 | 0 | R/W | Single/Multiple Block Transfer (enabled when the command with data is handled)<br>0: Single block transfer<br>1: Multi block transfer |
| 12 | MD4 | 0 | R/W | Write/Read Mode (enabled when the command with data is handled)<br>0: Write (SD host interface -> SD card)<br>1: Read (SD host interface <- SD card) |
| 11 | MD3 | 0 | R/W | Data Mode (Command Type)<br>0: Command without data transfer (bc, bcr, ac)<br>1: Command with data transfer (adtc) |
| 10 | MD2 | 0 | R/W | Mode/Response Type |
| 9 | MD1 | 0 | R/W | 000: Normal mode |
| 8 | MD0 | 0 | R/W | The response type and the transfer mode are selected by SD_CMD[7:0], and the SD_CMD[15:11] setting is disabled.<br>001: Setting prohibited<br>010: Setting prohibited<br>011: Extended mode/No response<br>100: Extended mode/R1, R5, R6, or R7 response from the SD card<br>101: Extended mode/R1b response from the SD card<br>110: Extended mode/R2 response from the SD card<br>111: Extended mode/R3 or R4 response from the SD card<br>Some commands cannot be used in normal mode. For details, see section 150.3.6.15, Example of SD_CMD Register Setting to select mode/response type. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | C1 | 0 | R/W | 00: CMD |
| 6 | C0 | 0 | R/W | 01: ACMD<br>10: Setting prohibited<br>11: Setting prohibited |
| 5 | CF45 | 0 | R/W | Command Index |
| 4 | CF44 | 0 | R/W | These bits specify Command Format[45:40] (command index). |
| 3 | CF43 | 0 | R/W | [Examples] |
| 2 | CF42 | 0 | R/W | CMD6: SD_CMD[7:0] = B'0000 0110 |
| 1 | CF41 | 0 | R/W | CMD18: SD_CMD[7:0] = B'0001 0010 |
| 0 | CF40 | 0 | R/W | ACMD13: SD_CMD[7:0] = B'0100 1101 |

Note: SD_CMD cannot be written to when the CBSY bit in SD_INFO2 is 1.

#### 150.2.1.2 Port Select Register (SD_PORTSEL)

The port select register (SD_PORTSEL) selects the ports for SD cards.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 12 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 11 to 10 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 9 | NP1 | 0 | R | Number of ports supported for SD cards |
| 8 | NP0 | 1 | R | [NP1,NP0] = 01: One port |
| 7 to 4 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 3 to 0 | — | All 0 | R/W | Reserved<br>The write value should always be 0. |

#### 150.2.1.3 Command Argument Register (SD_ARG)

Command arguments for SD cards are set in the SD command argument registers (SD_ARG). Set the command arguments before writing to SD_CMD.

- Note that the argument of CMD12 within command sequences is H'0000 0000 regardless of the setting of SD_ARG.
- SD_ARG

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CF39 to CF8 | All 0 | R/W | Set command format[39:8] (argument). |

- SD_ARG1 (Mirror of SD_ARG[31:16])

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Fixed 0 |
| 15 to 0 | CF39 to CF24 | All 0 | R/W | Set command format[39:24] (argument). |

### 150.2.1.4 Data Stop Register（SD_STOP）

The data stop register (SD_STOP) is used to enable or disable block counting at multiple block transfer, and to control the issuing of CMD12 within command sequences.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 19 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 18 | — | 0*1 | R/W | Reserved<br>The write value should always be 0. |
| 17 | HPIMODE | 0*1 | R/W | HPI Mode Enable<br>0: Disables HPI mode.<br>1: Enables HPI mode. |
| 16 | HPICMD | 0*1 | R/W*3 | HPI Command Issue<br>When HPICMD is set to 1 while HPIMODE is 1, the HPI command (CMD12) is issued.<br>This bit is cleared to 0 when reception of the response to CMD12 is completed.<br>The timing with which this bit is set to 1 is as follows.<br>• After reception of the response to CMD12 that was issued by setting the STP bit to 1 has been completed during the CMD6/CMD38 or CMD25 sequence.<br>• After reception of the response to CMD24/CMD25 has been completed<br>After HPICMD is set to 1, do not write 0 to this bit while the CBSY bit in SD_INFO2 is 1. Do not set this bit to 1 when the CBSY bit in SD_INFO2 is 0. |
| 15 to 9 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 8 | SEC | 0*1 | R/W | Block Count Enable*2<br>0: Disables SD_SECCNT setting value.<br>1: Enables SD_SECCNT setting value.<br>Set SEC to 1 at multiple block transfer.<br>When SD_CMD is set as follows to start the command sequence while SEC is set to 1, CMD12 is automatically issued to stop multi-block transfer with the number of blocks which is set to SD_SECCNT.<br>• 1. CMD18 or CMD25 in normal mode (SD_CMD[10:8] = 000)<br>• 2. SD_CMD[15:13] = 001 in extended mode (CMD12 is automatically issued, multiple block transfer)<br>When the command sequence is halted because of a communications error or timeout, CMD12 is not automatically issued. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | STP | 0*1 | R/W | Stop<br>• When STP is set to 1 during multiple block transfer, CMD12 is issued to halt the transfer through the SD host interface.<br>However, if a command sequence is halted because of a communications error or timeout, CMD12 is not issued. Although continued buffer access is possible even after STP has been set to 1, the buffer access error bit (ERR5 or ERR4) in SD_INFO2 will be set accordingly.<br>• When STP has been set to 1 during transfer for single block write, the access end flag is set when SD_BUF becomes empty, and CMD12 is not issued. If SD_BUF does contain data, the access end flag is set on completion of reception of the busy state without CMD12 having been issued.<br>• When STP has been set to 1 during transfer for single block read, the access end flag is set immediately after setting of the STP bit and CMD12 is not issued.<br>• When STP is set to 1 during reception of the busy state after an R1b response, the access end flag is set on completion of reception of the busy state without CMD12 having been issued.<br>• When STP is set to 1 after a command sequence has been completed, CMD12 is not issued and the access end flag is not set.<br>• Set STP to 1 after the response end flag has been set.<br>• Set STP to 0 after the response end flag has been set. |

Notes: 1. The initial value is applied at a reset and when the SDRST bit in SOFT_RST is 0.
2. Do not change the value of this bit when the CBSY bit in SD_INFO2 is set to 1.
3. It is effective only if 1 is written.

#### 150.2.1.5 Block Count Register (SD_SECCNT)

The block count register (SD_SECCNT) is used to specify the number of transfer blocks at multiple block transfer.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 0 | CNT31 to CNT0 | All 0 | R/W | Number of Transfer Blocks*<br>When H'0000 0001 is set, the number of transfer blocks is 1.<br>:<br>When H'0000 FFFF is set, the number of transfer blocks is 65535.<br>:<br>When H'FFFF FFFF is set, the number of transfer blocks is 4294967295.<br>Do not set this register to H'0000 0000 if multiple blocks are to be transferred. |

Note: * Do not change the value of this bit when the CBSY bit in SD_INFO2 is set to 1.

#### 150.2.1.6 SD Card Response Registers (SD_RSP)

The SD card response registers (SD_RSP) hold the response from the SD card.

- SD_RSP10

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 0 | R71 to R8 | All 0 | R | Hold the response from the SD card |

- SD_RSP1 (Mirror of SD_RSP10[31:16])

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 to 0 | R39 to R24 | All 0 | R | Hold the response from the SD card |

- SD_RSP32 (Mirror of SD_RSP10[63:32])

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 0 | R71 to R40 | All 0 | R | Hold the response from the SD card |

- SD_RSP3 (Mirror of SD_RSP32[31:16])

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 to 0 | R71 to R56 | All 0 | R | Hold the response from the SD card |

- SD_RSP54

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 56 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 55 to 0 | R127 to R72 | All 0 | R | Hold the response from the SD card |

- SD_RSP5 (Mirror of SD_RSP54[31:16])

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 to 0 | R103 to R88 | All 0 | R | Hold the response from the SD card |

- SD_RSP76 (Mirror of SD_RSP54[63:32])

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 24 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 23 to 0 | R127 to R104 | All 0 | R | Hold the response from the SD card |

- SD_RSP7 (Mirror of SD_RSP76[31:16])

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 8 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 7 to 0 | R127 to R120 | All 0 | R | Hold the response from the SD card |

Table 150.4 lists the response types and corresponding SD_RSP registers.

**Table 150.4 Response Types and Corresponding SD_RSP Registers**

| Response Types | SD_RSP Registers |
|---|---|
| R1, R1b[39:8] | SD_RSP10<br>SD_RSP54* |
| R2[127:8] | SD_RSP54 and SD_RSP10 |

| Response Types | SD_RSP Registers |
|---|---|
| R3[39:8] | SD_RSP10 |
| R4[39:8] | SD_RSP10 |
| R5[39:8] | SD_RSP10 |
| R6[39:8] | SD_RSP10 |
| R7[39:8] | SD_RSP10 |

Note: * The response to CMD18 and to CMD25 is stored in both R[39:8] and R[103:72].
This makes it possible to confirm the response to CMD18 and CMD25 by reading R[103:72] even if the response to automatic CMD12 is stored in R[39:8].

### 150.2.1.7 SD Card Interrupt Flag Register (SD_INFO1)

The SD card interrupt flag register 1 (SD_INFO1) indicates the response end and access end in the command sequence. This register also indicates the card detect/write protect state.

For CMD12 and CMD52 (SDIO abort) at multiple block transfer, INFO0 is not set but only INFO2 is set.

Even if the command sequence is halted because of a communications error or timeout, INFO0 or INFO2 is set.

INFO10, INFO9, and INFO8 change depending on the SDDAT3 state after a reset is released and continue to change in 4-bit transfer mode.

To clear a flag, write 0 to the bit to be cleared and 1 to the other bits.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 17 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 16 | HPIRES | 0*2 | R/W*1 | Response Reception Completion<br>[Setting condition]<br>When reception of the response to CMD12 that was issued by setting the STP bit to 1 is completed during the CMD6/CMD38 or CMD25 sequence in HPI mode.<br>[Clearing condition]<br>When 0 is written to HPIRES |
| 15 to 11 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 10 | INFO10 | Unknown | R | Indicates the SDDAT3 state.<br>0: SDDAT3 is set to 0.<br>1: SDDAT3 is set to 1. |
| 9 | INFO9 | 0 | R/W*1 | SDDAT3 Card Insertion<br>[Setting condition]<br>After change in SDDAT3 from 0 to 1, two cycles of SDxφ have elapsed with SDDAT3 held 1.<br>[Clearing condition]<br>When 0 is written to INFO9 |
| 8 | INFO8 | 0 | R/W*1 | SDDAT3 Card Removal<br>[Setting condition]<br>After change in SDDAT3 from 1 to 0, two cycles of SDxφ have elapsed with SDDAT3 held 0.<br>[Clearing condition]<br>When 0 is written to INFO8 |
| 7 | INFO7 | Unknown | R | Write Protect<br>Indicates the ISDWP state.<br>0: ISDWP is set to 1.<br>1: ISDWP is set to 0. |
| 6 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | INFO5 | Unknown | R | Indicates the ISDCD state.<br>0: Indicates that Mcycle has elapsed with ISDCD held 1.<br>1: Indicates that Mcycle has elapsed with ISDCD held 0.<br>Mcycle is set by bits 3 to 0 in SD_OPTION. |
| 4 | INFO4 | 0 | R/W*1 | ISDCD Card Insertion<br>[Setting condition]<br>After change in ISDCD from 1 to 0, Mcycle has elapsed with ISDCD held 0.<br>[Clearing condition]<br>When 0 is written to INFO4<br>Mcycle is set by bits 3 to 0 in SD_OPTION. |
| 3 | INFO3 | 0 | R/W*1 | ISDCD Card Removal<br>[Setting condition]<br>After change in ISDCD from 0 to 1, Mcycle has elapsed with ISDCD held 1.<br>[Clearing condition]<br>When 0 is written to INFO3<br>Mcycle is set by bits 3 to 0 in SD_OPTION. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | INFO2 | 0*2 | R/W*1 | Access End<br>[Setting conditions]<br>• 1. When read access to the buffer is completed in the case of transfer for single block read<br>• 2. When read access to the buffer for the last block of data is completed in the case of transfer for multiple block read<br>• 3. When read access to the buffer and reception of the response to CMD12 are completed in the case of transfer for multiple block read with automatic issuing of CMD12<br>• 4. When reception of the busy state after reception of the CRC status is completed in the case of transfer for single block write<br>• 5. When reception of the busy state after reception of the CRC status of the last block of data is completed in the case of transfer for multiple block write<br>• 6. When reception of the response busy state for CMD12 is completed in the case of transfer for multiple block write with automatic issuing of CMD12<br>• 7. When reception of the response to CMD12 that was issued by setting the STP bit to1 is completed in the case of transfer for multiple block read<br>• 8. When reception of the response busy state for CMD12 that was issued by setting the STP bit to1 is completed in the case of transfer for multiple block write<br>• 9. When reception of the response to CMD52 that was issued by setting the IOABT bit to1 is completed in the case of transfer for multiple block read<br>• 10. When reception of the response to CMD52 that was issued by setting the IOABT bit to1 is completed in the case of transfer for multiple block write<br>In addition to the above conditions, this bit is set when a command sequence is halted because of a communications error or timeout.<br>[Clearing condition]<br>When 0 is written to INFO2<br>When the access end bit is set to 1, the command sequence is terminated. |
| 1 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | INFO0 | 0*2 | R/W*1 | Response End<br>[Setting conditions]<br>• 1. When reception of the response is completed<br>• 2. When transmission of a command without response is completed<br>• 3. When reception of the busy state after R1b response is completed<br>• 4. When reception of the response to CMD52 that was issued by setting the C52PUB bit to1 is completed in the case of transfer for multiple block read<br>• 5. When reception of the response to CMD52 that was issued by setting the C52PUB bit to1 is completed in the case of transfer for multiple block write<br>In addition to the above conditions, this bit is set when a command sequence is halted because of a communications error or timeout.<br>[Clearing condition]<br>When 0 is written to INFO0<br>When a command without data is issued, the command sequence is terminated when the response end bit is set to 1. |

Notes: 1. It is effective only if 0 is written.
2. The value is initialized by a reset and also in the case of a reset by the SDRST bit in SOFT_RST.

#### 150.2.1.8 SD Card Interrupt Flag Register (SD_INFO2)

The SD card interrupt flag register 2 (SD_INFO2) indicates the access status of the SD buffer (SD_BUF) and SD card. To clear a flag, write 0 to the bit to be cleared and 1 to the other bits.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 | ILA | 0*2 | R/W*1 | Illegal Access Error<br>[Setting conditions]<br>• 1. Writing of data to SD_CMD within a command sequence (CBSY=1)<br>• 2. When SD_CMD[11] = 1 (command with data transfer) and SD_CMD[7:0]= B'0000 1100 (CMD12) are set in SD_CMD<br>[Clearing condition]<br>When 0 is written to ILA |
| 14 | CBSY | 0*2 | R | Command Type Register Busy<br>0: A command sequence has been completed.<br>1: A command sequence is being executed. |
| 13 | SCLKDIVEN | 1*2 | R | 0: The SD bus (CMD, DAT) is busy. Writing to the SCLKEN and DIV bits in SD_CLK_CTRL is not possible.<br>1: The SD bus (CMD, DAT) is not busy.<br>When a command sequence is started by writing to SD_CMD, the CBSY bit is set to 1 and, at the same time, the SCLKDIVEN bit is set to 0. The SCLKDIVEN bit is set to 1 after 8 cycles of SDCLK have elapsed after setting of the CBSY bit to 0 due to completion of the command sequence. |
| 12 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. There is no problem even if 1 is written. |
| 11 | — | 0 | R/W | Reserved<br>The write value should always be 1.<br>There is no problem even if 0 is written. |
| 10 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. There is no problem even if 1 is written. |
| 9 | BWE | 0*2 | R/W*1 | SD_BUF Write Enable<br>0: Data cannot be written in SD_BUF0.<br>1: Data can be written in SD_BUF0.<br>[Setting conditions]<br>• 1. When SD_BUF is empty at single block transfer<br>• 2. When either bank 1 or bank 2 of SD_BUF is empty at multiple block transfer<br>[Clearing conditions]<br>• 1. When 0 is written to BWE<br>• 2. Writing of a block of data to SD_BUF by DMA transfer<br>When data is written to SD_BUF0 by the CPU, clear BWE and then write amount of data specified by SD_SIZE*3,. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8 | BRE | 0*2 | R/W*1 | SD_BUF Read Enable<br>0: Data cannot be read from SD_BUF0.<br>1: Data can be read from SD_BUF0.<br>[Setting conditions]<br>• 1. When data set in SD_SIZE is stored in SD_BUF at single block transfer<br>• 2. When data set in SD_SIZE is stored in either bank 1 or bank 2 of SD_BUF at multiple block transfer<br>[Clearing conditions]<br>• 1. When 0 is written to BRE<br>• 2. Reading of a block of data from SD_BUF by DMA transfer<br>When data is read from SD_BUF0 by the CPU, clear BRE and then read amount of data specified by SD_SIZE*3.<br>Even if a CRC error or an END error occurs while block data is read, data is stored in SD_BUF and BRE is set. |
| 7 | DAT0 | Unknown | R | SDDAT0<br>Indicates the SDDAT0 state.<br>0: SDDAT0 is set to 0.<br>1: SDDAT0 is set to 1.<br>If the data timeout (ERR3) is set but the response timeout (ERR6) is not set after the Erase command has been issued, the end of the Erase sequence (DAT0 = 1) is confirmed by polling DAT0.<br>If a communications error or timeout occurs during a write sequence, the DAT0 bit may retain the value 0.<br>While the SD clock (SDCLK) is stopped, the DAT0 bit retains the value before the clock is stopped. |
| 6 | ERR6 | 0*2 | R/W*1 | Response Timeout<br>[Setting condition]<br>When a response is not received though a longer time than 640 cycles of SDCLK have elapsed (including a response to a command issued within a command sequence*5)<br>[Clearing condition]<br>When 0 is written to ERR6<br>The command sequence is halted by the response timeout.*4 |
| 5 | ERR5 | 0*2 | R/W*1 | SD_BUF Illegal Read Access<br>[Setting conditions]<br>• 1. When SD_BUF is empty while SD_BUF0 is read<br>• 2. When data with a CRC error or END error is read from SD_BUF0<br>[Clearing condition]<br>When 0 is written to ERR5 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | ERR4 | 0*2 | R/W*1 | SD_BUF Illegal Write Access<br>[Setting conditions]<br>• 1. When data is written to SD_BUF0 while it is not in the data read/write command state<br>• 2. When data is written to SD_BUF0 while SD_BUF is full<br>• 3. When data is written to SD_BUF0 while an error occurs in the CRC status or CRC status length<br>• 4. When data is written to SD_BUF0 while a busy state after the CRC status continues for longer than Ncycle<br>[Clearing condition]<br>When 0 is written to ERR4<br>Ncycle is set by bits 7 to 4 in SD_OPTION. |
| 3 | ERR3 | 0*2 | R/W*1 | Data Timeout (except response timeout)<br>[Setting conditions]<br>• 1. After R1b response, the busy state (SDDAT0 = 0) continues for longer than Ncycle.<br>• 2. After CRC status, the busy state (SDDAT0 = 0) continues for longer than Ncycle.<br>• 3. After write data, the CRC status is not received though Ncycle has elapsed.<br>• 4. After read command, read data is not received though a longer time than Ncycle has elapsed.<br>• 5. After CMD12 has been issued within a command sequence, the busy state (SDDAT0 = 0) for longer than Ncycle continues.<br>• 6. After the reception of read data, read data for the next block are not received though a longer time than Ncycle has elapsed.<br>• 7. After release of the read wait state, read data for the next block are not received though a longer time than Ncycle has elapsed.<br>[Clearing condition]<br>When 0 is written to ERR3<br>Ncycle is set by bits 7 to 4 in SD_OPTION.<br>The command sequence is halted by the data timeout. |
| 2 | ERR2 | 0*2 | R/W*1 | END Error<br>[Setting conditions]<br>• 1. When an error occurs in the response length (and the end bit has not been detected)<br>• 2. When an error occurs in the read data length (and the end bit has not been detected among the valid bits)<br>• 3. When an error occurs in the CRC status length (and the end bit has not been detected)<br>• 4. An error in the length of a response to a command issued within a command sequence*5 (i.e. the end bit has not been detected)<br>[Clearing condition]<br>When 0 is written to ERR2<br>The command sequence is halted by the End error.*4 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | ERR1 | 0*2 | R/W*1 | CRC Error<br>[Setting conditions]<br>• 1. When an error occurs in the CRC status (i.e. the received CRC status was not 010')<br>• 2. When a CRC error occurs in the read data<br>• 3. When a CRC error occurs in the response<br>• 4. A CRC error in the response to a command issued within a command sequence *5<br>[Clearing condition]<br>When 0 is written to ERR1<br>The command sequence is halted by the CRC error.*4 |
| 0 | ERR0 | 0*2 | R/W*1 | CMD Error<br>[Setting conditions]<br>• 1. The command index of the transmitted command differing from the command index of the received response<br>• 2. The command index of a command issued within a command sequence*5 differing from the command index of the received response<br>[Clearing condition]<br>When 0 is written to ERR0<br>The command sequence is halted by the CMD error.*4 |

Notes: 1. It is effective only if 0 is written.
2. The initial value is applied at a reset and when the SDRST bit in SOFT_RST is 0.
3. When the WMODE bit in HOST_MODE is 0, the single byte from the fraction of a full 16-bit unit is regarded as excess data due to an odd value for the number of bytes setting in SD_SIZE. When the WMODE bit in HOST_MODE is 1, the single byte or three bytes from the fraction of a full 64-bit unit are regarded as excess data due to an odd value for the number of bytes setting in SD_SIZE, or the two bytes from the fraction of a full 64-bit unit are regarded as excess data due if the value for the number of bytes setting in SD_SIZE is even but is not on a four-byte boundary.
4. After the C52PUB bit in SDIO_MODE has been set to 1, if a communications error or timeout for response occurs in response to the CMD52 that is issued, since the command sequence has not been completed, complete the sequence with error processing.
5. "Command issued within a command sequence" refers to CMD12 when automatic issuing is enabled for multiple block transfer by the setting in SD_CMD, CMD12 when the STP bit in SD_STOP is set to 1, or CMD52 when the C52PUB or IOABT bit in SDIO_MODE is set to 1.

#### 150.2.1.9 SD_INFO1 Interrupt Mask Register (SD_INFO1_MASK)

The SD_INFO1 interrupt mask register (SD_INFO1_MASK) is used to enable or disable the SD_INFO1 interrupt. When 0 is set in SD_INFO1_MASK while the corresponding flag in SD_INFO1 is set, an interrupt occurs.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 17 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 16 | IMASK16 | 1 | R/W | HPIRES interrupt masked |
| 15 to 10 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. There is no problem even if 1 is written. |
| 9 | IMASK9 | 1 | R/W | INFO9 interrupt masked |
| 8 | IMASK8 | 1 | R/W | INFO8 interrupt masked |
| 7 to 5 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. There is no problem even if 1 is written. |
| 4 | IMASK4 | 1 | R/W | INFO4 interrupt masked |
| 3 | IMASK3 | 1 | R/W | INFO3 interrupt masked |
| 2 | IMASK2 | 1 | R/W | INFO2 interrupt masked |
| 1 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. There is no problem even if 1 is written. |
| 0 | IMASK0 | 1 | R/W | INFO0 interrupt masked |

#### 150.2.1.10 SD_INFO2 Interrupt Mask Register (SD_INFO2_MASK)

The SD_INFO2 interrupt mask register (SD_INFO2_MASK) is used to enable or disable the SD_INFO2 interrupt. When 0 is set in SD_INFO2_MASK while the corresponding flag in SD_INFO2 is set, an interrupt occurs.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 | IMASK | 1 | R/W | ILA interrupt masked |
| 14 to 12 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. There is no problem even if 1 is written. |
| 11 | — | 1 | R/W | Reserved<br>The write value should always be 1. |
| 10 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. There is no problem even if 1 is written. |
| 9 | BMASK1 | 1 | R/W | BWE interrupt masked |
| 8 | BMASK0 | 1 | R/W | BRE interrupt masked |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. There is no problem even if 1 is written. |
| 6 | EMASK6 | 1 | R/W | ERR6 interrupt masked |
| 5 | EMASK5 | 1 | R/W | ERR5 interrupt masked |
| 4 | EMASK4 | 1 | R/W | ERR4 interrupt masked |
| 3 | EMASK3 | 1 | R/W | ERR3 interrupt masked |
| 2 | EMASK2 | 1 | R/W | ERR2 interrupt masked |
| 1 | EMASK1 | 1 | R/W | ERR1 interrupt masked |
| 0 | EMASK0 | 1 | R/W | ERR0 interrupt masked |

### 150.2.1.11 SD Clock Control Register (SD_CLK_CTRL)

The SD clock control register (SD_CLK_CTRL) is used to control the SD clock (SDCLK) output and to set the frequency. Set SCLKEN to 1 before writing to SD_CMD to issue a command. Do not write to SD_CLK_CTRL while the SCLKDIVEN bit in SD_INFO2 is set to 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 17 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 16 | — | 0 | R/W | Reserved<br>The write value should always be 0. |
| 15 to 11 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 10 | — | 0*1 | R/W | Reserved<br>The write value should always be 0. |
| 9 | SDCLK OFFEN | 0 | R/W | SD Clock (SD_CLK) Output Automatic Control Enable<br>0: Automatic control for SD clock (SD_CLK) output is disabled.<br>1: Automatic control for SD clock (SD_CLK) output is enabled.<br>This function of automatic control for SD clock (SD_CLK) output causes SD_CLK output only within a command sequence.<br>The timing with which SD_CLK output starts and stops is as follows.<br>SD_CLK output starts after writing to SD_CMD.<br>SD_CLK output stops when 8 cycles of SD_CLK have elapsed after the end of the command sequence.<br>In addition, SD_CLK is fixed to 0 while SCLKEN of SD_CLK_CTRL is 0, regardless of the value of this bit. |
| 8 | SCLKEN | 0*1 | R/W*2 | SD Clock (SD_CLK) Output Control Enable<br>0: SD clock (SD_CLK) output is disabled. The SD_CLK signal is fixed 0.<br>1: SD clock (SD_CLK) output is enabled. |
| 7 | DIV7 | 0 | R/W*2 | SD Clock (SD_CLK) |
| 6 | DIV6 | 0 | R/W*2 | B'1000 0000: SDϕ/512 |
| 5 | DIV5 | 1 | R/W*2 | B'0100 0000: SDϕ/256 |
| 4 | DIV4 | 0 | R/W*2 | B'0010 0000: SDϕ/128 |
| 3 | DIV3 | 0 | R/W*2 | B'0001 0000: SDϕ/64 |
| 2 | DIV2 | 0 | R/W*2 | B'0000 1000: SDϕ/32 |
| 1 | DIV1 | 0 | R/W*2 | B'0000 0100: SDϕ/16 |
| 0 | DIV0 | 0 | R/W*2 | B'0000 0010: SDϕ/8<br>B'0000 0001: SDϕ/4<br>B'0000 0000: SDϕ/2<br>B'1111 1111: SDϕ<br>Other settings are prohibited.<br>In addition, in the case of data transfer in DDR mode (DDR bit in SDIF_MODE = 1), do not set DIV[7:0] to B'1111 1111. |

Notes: 1. This initial value is applied at a reset and when the SDRST bit in SOFT_RST is 0.
2. Writing to SD_CLK_CTRL is impossible when the CBSY bit in SD_INFO2 is 1.

Notes on when setting the SD clock (SDCLK) to SDϕ (DIV[7:0] = B'1111 1111)

When changing the setting of bits DIV[7:0] to B'1111 1111, or from B'1111 1111 to other setting, perform the following processing before writing to SD_CMD.

(1) Set the SCLKEN bit to 0 by writing to SD_CLK_CTRL. (Do not change the setting of bits other than SCLKEN at this time.)
(2) Change the setting of bits DIV[7:0] by writing to SD_CLK_CTRL. (Do not change the setting of bits other than DIV[7:0] at this time. The SCLKEN bit should retain the value 0.)
(3) Set the SCLKEN bit to 1 by writing to SD_CLK_CTRL. (Do not change the setting of bits other than SCLKEN at this time.)

Also when changing the setting of bits DIV[7:0] to B'1111 1111 after having set the SDRST bit in SOFT_RST to 0 and then changed it to 1, perform this processing before writing to SD_CMD.

#### 150.2.1.12 Transfer Data Length Register (SD_SIZE)

The transfer data length register (SD_SIZE) is used to specify the transfer data size.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 12 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 11 to 10 | — | All 0 | R | Reserved |
| 9 to 0 | LEN9 to LEN0 | 1000000000 | R/W | Transfer Data Size*1<br>These bits specify a size between 1 and 512 bytes for single block transfer.<br>In cases of multiple block transfer with automatic issuing of CMD12 (CMD18 and CMD25), the only specifiable transfer data size is 512 bytes.<br>Furthermore, in cases of multiple block transfer without automatic issuing of CMD12, as well as 512 bytes, 32, 64, 128, and 256 bytes are specifiable. However, in the reading of 32, 64, 128, and 256 bytes for the transfer of multiple blocks, this is restricted to multiple block transfer by CMD53.<br>Additionally, if a command accompanies data transfer, do not set these bits to 0.<br>Do not specify a data size larger than 512 bytes. |

Note: * Do not change the values of these bits when the CBSY bit in SD_INFO2 is 1.

#### 150.2.1.13 SD Card Access Control Option Register (SD_OPTION)

The SD card access control option register (SD_OPTION) is used to set the bus width and timeout counter.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 | WIDTH | 0*1 | R/W | Bus width*2<br>{WIDTH,WIDTH8} = 01: 8-bit width<br>{WIDTH,WIDTH8} = 00: 4-bit width<br>{WIDTH,WIDTH8} = 10,11: 1-bit width<br>In the case of data transfer in HS400 mode (HS400 bit in SDIF_MODE = 1), do not set this bit to 1.<br>In the case of writing of one-byte block, 8-bit width cannot be specified for the bus width. Change the bus width to 4 bits or 1 bit before writing one-byte block. |
| 14 | — | 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 13 | WIDTH8 | 0*1 | R/W | Bus width*2<br>See the description of the WIDTH bit. |
| 12 to 10 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | EXTOP | 0[*1] | R/W | Timeout Mode Select<br>0: Bits TOP27 to TOP24 specify the timeout count from SDCLK*$2^{13}$ to SDCLK*$2^{27}$.<br>1: Bits TOP27 to TOP24 specify the timeout count from SDCLK*$2^{14}$ to SDCLK*$2^{28}$. |
| 8 | TOUTMASK | 0[*1] | R/W | Timeout Mask<br>0: Enables timeout.<br>1: Disables timeout. (The ERR6 and ERR3 bits in SD_INFO2 and the E6 to E0 bits in SD_ERR_STS2 are not set.)<br>If a timeout occurs while it is disabled, perform a software reset to terminate a command sequence. |
| 7 | TOP27 | 1[*1] | R/W | Timeout Counter[*2]<br>0000: SDCLK*$2^{13}$<br>0001: SDCLK*$2^{14}$<br>: :<br>1101: SDCLK*$2^{26}$<br>1110: SDCLK*$2^{27}$<br>1111: Setting prohibited |
| 6 | TOP26 | 1[*1] | R/W | |
| 5 | TOP25 | 1[*1] | R/W | |
| 4 | TOP24 | 0[*1] | R/W | |
| 3 | CTOP24 | 1[*1] | R/W | Card Detect Time Counter<br>0000: SDϕ $2^{10}$<br>0001: SDϕ *$2^{11}$<br>: :<br>1101: SDϕ *$2^{23}$<br>1110: SDϕ *$2^{24}$<br>1111: Setting prohibited |
| 2 | CTOP23 | 1[*1] | R/W | |
| 1 | CTOP22 | 1[*1] | R/W | |
| 0 | CTOP21 | 0[*1] | R/W | |

Notes: 1. The initial value is applied at a reset and when the SDRST bit in SOFT_RST is 0.
2. Do not change the values of these bits when the CBSY bit in SD_INFO2 is 1.

#### 150.2.1.14 SD Error Status Register 1 (SD_ERR_STS1)

The SD error status register 1 (SD_ERR_STS1) indicates the CRC status, CRC error, End error, and CMD error.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 15 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 14 | E14 | 0*1 | R | These bits hold the CRC status. (normal: 010) |
| 13 | E13 | 1*1 | | |
| 12 | E12 | 0*1 | | |
| 11 | E11 | 0*1 | R | Set to 1 when an error occurs in the CRC status. |
| 10 | E10 | 0*1 | R | Set to 1 when a CRC error occurs in the read data. |
| 9 | E9 | 0*1 | R | Set to 1 when a CRC error occurs in the response to a command issued within a command sequence*2. In cases where CMD12 is issued by setting a command index in SD_CMD, this is indicated in E8. |
| 8 | E8 | 0*1 | R | Set to 1 when a CRC error occurs in a response (other than a response to a command issued within a command sequence*2). |
| 7, 6 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 5 | E5 | 0*1 | R | Set to 1 when an error occurs in the CRC status length (and the end bit has not been detected). |
| 4 | E4 | 0*1 | R | Set to 1 when an error occurs in the read data length (and the end bit has not been detected among the valid bits). |
| 3 | E3 | 0*1 | R | Set to 1 when an error occurs in the response length to a command issued within a command sequence*2. In cases where CMD12 is issued by setting a command index in SD_CMD, this is indicated in E2. |
| 2 | E2 | 0*1 | R | Set to 1 when an error occurs in the response length (other than a response to a command issued within a command sequence*2). |
| 1 | E1 | 0*1 | R | Set to 1 when an error occurs in the command index of the response to a command issued within a command sequence*2. In cases where CMD12 is issued by setting a command index in SD_CMD, this is Indicated in E0. |
| 0 | E0 | 0*1 | R | Set to 1 when an error occurs in the command index of a response (other than a response to a command issued within a command sequence*2). |

Notes: 1. The initial value is applied at a reset and when the SDRST bit in SOFT_RST is 0.
2. "Command issued within a command sequence" refers to CMD12 when automatic issuing is enabled for multiple block transfer by the setting in SD_CMD, CMD12 when the STP bit in SD_STOP is set to 1, or CMD52 when the C52PUB or IOABT bit in SDIO_MODE is set to 1.

#### 150.2.1.15 SD Error Status Register 2 (SD_ERR_STS2)

The SD error status register 2 (SD_ERR_STS2) indicates the timeout state. Ncycle is set by bits 7 to 4 in SD_OPTION.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 7 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 6 | E6 | 0*1 | R | Set to 1 when the busy state continues for longer than Ncycle after the CRC status |
| 5 | E5 | 0*1 | R | Set to 1 when the CRC status is not received though a longer time than Ncycle has elapsed after data writing |
| 4 | E4 | 0*1 | R | Set to 1 when read data is not received though a longer time than Ncycle has elapsed after read command.<br>Set to 1 when read data for the next block are not received though a longer time than Ncycle has elapsed after the reception of read data.<br>Set to 1 when read data for the next block are not received though a longer time than Ncycle has elapsed after release of the read wait state. |
| 3 | E3 | 0*1 | R | Set to 1 when the busy state for longer than Ncycle continues after CMD12 has been issued within a command sequence. In cases where CMD12 is issued by setting a command index in SD_CMD, this is indicated in E2. |
| 2 | E2 | 0*1 | R | Set to 1 when the busy state for longer than Ncycle continues after R1b response. |
| 1 | E1 | 0*1 | R | Set to 1 when the response to a command issued within a command sequence*2 is not received though a longer time than 640 cycles of SDCLK have elapsed. In cases where CMD12 is issued by setting a command index in SD_CMD, this is indicated in E0. |
| 0 | E0 | 0*1 | R | Set to 1 when the response (other than a response to a command issued within a command sequence*2) is not received though a longer time than 640 cycles of SDCLK have elapsed. |

Notes: 1. The initial value is applied at a reset and when the SDRST bit in SOFT_RST is 0.
2. "Command issued within a command sequence" refers to CMD12 when automatic issuing is enabled for multiple block transfer by the setting in SD_CMD, CMD12 when the STP bit in SD_STOP is set to 1, or CMD52 when the C52PUB or IOABT bit in SDIO_MODE is set to 1.

#### 150.2.1.16 SD Buffer Read/Write Register (SD_BUF0)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 0 | BUF31 to BUF0 | Unknown | R/W | When writing to the SD card, the write data is written to this register.<br>When reading from the SD card, the read data is read from this register.<br>This register is internally connected to two 512-byte buffers (SD_BUF).<br>When both buffers are not empty at multiple block read, suspend data reception by stopping the SD clock. When either buffer becomes empty, restart data reception by starting supply of the SD clock. |

Note: When using the DMAC, the bus width should be fixed at 64 bits.

#### 150.2.1.17 SDIO Mode Control Register (SDIO_MODE)

The SDIO mode control register (SDIO_MODE) controls the CMD52 issuance and the read wait state at multiple block transfer, and the reception of SDIO interrupt. C52PUB and IOABT should not be set to 1 simultaneously.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 10 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 9 | C52PUB | 0 | R/W | SDIO None Abort<br>• When C52PUB is set to 1 in the CMD53 (multiple block) write sequence, CMD52 is automatically issued between blocks if SD_BUF becomes empty.C52PUB is automatically cleared to 0 after reception of the response to CMD52 is completed. Additionally, if C52PUB is set to 1 while the last block is being transferred, CMD52 is not issued. In this case, C52PUB is automatically cleared to 0 after the access end flag has been set to 1.<br>• When C52PUB and RWREQ are set to 1 in the CMD53 (multiple block) read sequence, the block transfer enters the read wait state between blocks and CMD52 is automatically issued. C52PUB is automatically cleared to 0 after reception of the response to CMD52 is completed. Additionally, if C52PUB is set to 1 while the last block is being transferred, CMD52 is not issued. In this case, C52PUB is automatically cleared to 0 after the access end flag has been set to 1.<br>• If C52PUB is set to 1 in the CMD53 (multiple block) read sequence, be sure to set RWREQ to 1 as well as C52PUB.<br>• Set SD_ARG before setting C52PUB to 1.<br>• Set C52PUB to 1 after the response end flag has been set. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8 | IOABT | 0 | R/W | SDIO Abort<br>• When IOABT is set to 1 in the CMD53 (multiple block) sequence, the CMD53 sequence is halted and CMD52 is issued.<br>However, if a command sequence is halted because of a communications error or timeout, CMD52 is not issued.<br>Although continued buffer access is possible even after IOABT has been set to 1, the buffer access error bit (ERR5 or ERR4) in SD_INFO2 will be set accordingly. Set SD_ARG before setting IOABT to 1.<br>• When IOABT has been set to 1 during transfer for single block write, the access end flag is set when SD_BUF becomes empty, and CMD52 is not issued. If SD_BUF does contain data, the access end flag is set on completion of reception of the busy state without CMD52 having been issued.<br>• When IOABT has been set to 1 during transfer for single block read, the access end flag is set immediately after setting of IOABT and CMD52 is not issued.<br>• When IOABT is set to 1 during reception of the busy state after an R1b response, the access end flag is set on completion of reception of the busy state without CMD52 having been issued.<br>• When IOABT is set to 1 after a command sequence has been completed, CMD52 is not issued and the access end flag is not set.<br>• Set IOABT to 1 after the response end flag has been set.<br>• Set IOABT to 0 after the access end flag has been set. |
| 7 to 3 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 2 | RWREQ | 0 | R/W | Read Wait Request<br>When RWREQ is set to 1 in the CMD53 (multiple block) read sequence, the block transfer enters the read wait state between blocks.<br>[Read wait state releasing]<br>(1) The read wait state is released, when RWREQ is cleared to 0 in the read wait state.<br>(2) When IOABT is set to 1 in the read wait state, RWREQ is automatically cleared to 0 after CMD52 has been issued, and then the read wait state is released.<br>(3) When C52PUB and RWREQ are set to 1 simultaneously in the CMD53 (multiple block) read sequence, the read wait state is not automatically released. Therefore, after the CMD52 response is received, clear RWREQ. (Be sure to set RWREQ and C52PUB simultaneously.)<br>When RWREQ is set to 1 while the last block in the CMD53 (multiple block) read sequence is transferred, the read wait state is not entered and RWREQ is automatically cleared to 0 by setting access end.<br>Set RWREQ to 1 after the response end flag has been set. |
| 1 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | IOMOD | 0 | R/W | SDIO Mode *<br>0: Disables the SD host interface to receive SDIO interrupt from the SDIO card<br>1: Enables the SD host interface to receive SDIO interrupt from the SDIO card |

Note: * Do not change the value of this bit when the CBSY bit in SD_INFO2 is set to 1.

### 150.2.1.18 SDIO Interrupt Flag Register (SDIO_INFO1)

The SDIO interrupt flag register (SDIO_INFO1) indicates the status regarding to the SDIO card access. To clear a flag, write 0 to the bit to be cleared and 1 to the other bits.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 | EXWT | 0*2 | R/W*1 | [Setting condition]<br>While the last block in the CMD53 (multiple block) read sequence is transferred, RWREQ in SDIO_MODE is set to 1.<br>[Clearing condition]<br>When 0 is written to EXWT |
| 14 | EXPUB52 | 0*2 | R/W*1 | [Setting conditions]<br>• 1. While the last block in the CMD53 (multiple block) sequence is transferred, C52PUB in SDIO_MODE is set to 1.<br>• 2. While C52PUB is set to 1 in the CMD53 (multiple block) write sequence, the last block is transferred.<br>[Clearing condition]<br>When 0 is written to EXPUB52 |
| 13 to 3 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 2, 1 | — | All 0*2 | R/W | Reserved<br>The write value should always be 1. The value may change during operation. |
| 0 | IOIRQ | 0*2 | R/W*1 | [Setting condition]<br>When SDIO interrupt from an SDIO card is received while IOMOD in SDIO_MODE is set to 1.<br>[Clearing condition]<br>When 0 is written to IOIRQ*3 |

Notes: 1. It is effective only if 0 is written.
2. The initial value is applied at a reset and when the SDRST bit in SOFT_RST is 0.
3. Before clearing this bit, access the SDIO card to negate the SDIO interrupt signal from the SDIO card. If the interrupt signal is not negated, this bit may be set again.

#### 150.2.1.19 SDIO_INFO1 Interrupt Mask Register (SDIO_INFO1_MASK)

The SDIO_INFO1 interrupt mask register (SDIO_INFO1_MASK) enables or disables the SD_INFO1 interrupt. When 0 is set in SDIO_INFO1_MASK while the corresponding flag in SD_INFO1 is set, an interrupt occurs.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 | MEXWT | 1 | R/W | EXWT interrupt masked |
| 14 | MEXPUB52 | 1 | R/W | EXPUB52 interrupt masked |
| 13 to 3 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 2 | — | 1 | R/W | Reserved<br>The write value should always be 1. |
| 1 | — | 1 | R/W | Reserved<br>The write value should always be 1. |
| 0 | IOMSK | 1 | R/W | IOIRQ interrupt masked |

#### 150.2.1.20 DMA Mode Enable Register (CC_EXT_MODE)

The DMA mode enable register (CC_EXT_MODE) enables the DMA transfer.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 13 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 12 | — | 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. There is no problem even if 0 is written. |
| 11, 10 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 9, 8 | — | All 0 | R/W | Reserved<br>The write value should always be 0. |
| 7, 6 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 5 | — | 0 | R/W | Reserved<br>The write value should always be 0. |
| 4 | — | 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. There is no problem even if 0 is written. |
| 3, 2 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | DMASDRW | 0 | R/W | SD_BUF Read/Write DMA Transfer *<br>0: The SD_BUF read/write DMA transfer is disabled.<br>1: The SD_BUF read/write DMA transfer is enabled.<br>At the SD_BUF read/write DMA transfer, set DMASDRW to 1 before setting SD_CMD. |
| 0 | — | 0 | R/W | Reserved<br>The write value should always be 0. |

Note: * Do not change the value of this bit when the CBSY bit in SD_INFO2 is set to 1.

#### 150.2.1.21 Software Reset Register (SOFT_RST)

The software reset register (SOFT_RST) sets a software reset. Also use this register to check that release from the reset state has been completed before attempting to use the SD host interface and before attempting access to the other registers.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 3 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 2 | — | 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. There is no problem even if 0 is written. |
| 1 | — | 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. There is no problem even if 0 is written. |
| 0 | SDRST | 1 | R/W | Software Reset of SD Interface Unit<br>0: Reset<br>1: Reset released |

#### 150.2.1.22 Version Register (VERSION)

The version register (VERSION) indicates the version of the SD host interface.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 | UR7 | 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 14 | UR6 | 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 13, 12 | UR5 to UR4 | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 11 to 8 | UR3 to UR0 | H'D | R | Version of Renesas' IP |
| 7 to 0 | IP7 to IP0 | H'10 | R | Version of introductory IP |

#### 150.2.1.23 Host Interface Mode Setting Register (HOST_MODE)

The host interface mode setting register (HOST_MODE) selects the width for access to the data bus.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 9 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 8 | BUSWIDTH | 0 | R/W | Width for Access to SD_BUF*1*2<br>Read or write access to SD_BUF0 can be performed with the specified width for access.<br>0: 16-bit access<br>1: 32-bit access<br>This bit is enabled while the WMODE bit is set to 1. |
| 7 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 1 | ENDIAN | 0 | R/W | SD_BUF0 data swap |
| 0 | WMODE | 0 | R/W | Width for Access to SD_BUF*1*2<br>Read or write access to SD_BUF0 can be performed with the specified width for access.<br>0: 64-bit access<br>1: 16-bit or 32-bit access |

Notes: 1. Do not change the value of this bit when the CBSY bit in SD_INFO2 is set to 1.
2. When using the DMAC, the bus width should be fixed at 64 bits.

#### 150.2.1.24 SD Interface Mode Setting Register (SDIF_MODE)

The SD interface mode setting register (SDIF_MODE) specifies HS400 mode.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 10 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 9 | — | 0*1 | R/W | Reserved<br>The write value should always be 0. |
| 8 | NOCHKCR | 0*1 | R/W | CRC Check Mask (test command for MMC supported)<br>Enables or disables checking of the CRC16 and CRC status.<br>0: Enables the CRC check.<br>1: Disables the CRC check (the CRC16 value is ignored at read, and the CRC status is not detected at write) |
| 7 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | HS400 | 0*1 | R/W | HS400 Mode Select*2<br>0: Normal mode (default, high speed, or SDR)<br>1: HS400 mode (DDR mode)<br>Set this bit to 0 when the SD clock division ratio is specified as 1:1 (bits DIV[7:0] in SD_CLK_CTRL are set to 11111111). |

Notes: 1. The initial value is applied at a reset and when the SDRST bit in SOFT_RST is 0.
2. Do not change the value of this bit when the CBSY bit in SD_INFO2 is set to 1.

#### 150.2.1.25 DMAC Interrupt Register 1 (DM_CM_INFO1)

The DMAC interrupt register 1 (DM_CM_INFO1) indicates the status of DMAC and a sequencer. To clear a flag, write 0 to the bit to be cleared and 1 to the other bits.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 21 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 20 | DTRANEND1 | 0[*1] | R/W | DMAC Channel 1 Transfer End<br>[Setting conditions]<br>1. When transfer of DMAC channel 1 is completed<br>2. When an error occurs on DMAC channel 1<br>[Clearing condition]<br>When 0 is written to DTRANEND1 |
| 19, 18 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 17 | — | 0[*1] | R/W | Reserved<br>The write value should always be 0. |
| 16 | DTRANEND0 | 0[*2] | R/W | DMAC Channel 0 Transfer End<br>[Setting conditions]<br>1. When transfer of DMAC channel 0 is completed<br>2. When an error occurs on DMAC channel 0<br>[Clearing condition]<br>When 0 is written to DTRANEND0 |
| 15 to 9 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 8 | SEQSUSPEND | 0 | R/W | Interrupt of command sequencer process suspend<br>This bit is asserted, when SDIP goes to suspend mode by DM_CM_CTRL.SEQ_SUSPEND at DM_CM_SEQ_CTRL.SEQ_TYPE = 01. |
| 7 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | SEQEND | 0[*3] | R/W | Sequencer Operation End<br>[Setting conditions]<br>1. When operation of a sequencer is completed<br>2. When a sequencer error occurs<br>[Clearing condition]<br>When 0 is written to SEQEND |

Notes: 1. The initial value is applied at a reset and when the DTRANRST1 bit in DM_CM_RST is 0.
2. The initial value is applied at a reset and when the DTRANRST0 bit in DM_CM_RST is 0.
3. The initial value is applied at a reset and when the SEQRST bit in DM_CM_RST is 0.

#### 150.2.1.26 DM_CM_INFO1 Interrupt Mask Register (DM_CM_INFO1_MASK)

The DM_CM_INFO1 interrupt mask register (DM_CM_INFO1_MASK) enables or disables the DM_CM_INFO1 interrupt. When 0 is set in DM_CM_INFO1_MASK while the corresponding flag in DM_CM_INFO1 is set, an interrupt occurs.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 21 | — | All 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 20 | DTRANEND1_MASK | 1 | R/W | DTRANEND1 interrupt masked |
| 19, 18 | — | All 1 | R/W | Reserved<br>The write value should always be 1. |
| 17 | — | 0 | R/W | Reserved<br>The write value should always be 1. |
| 16 | DTRANEND0_MASK | 1 | R/W | DTRANEND0 interrupt masked |
| 15 to 9 | — | All 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 8 | SEQSUSPEND_MASK | 0 | R/W | Interrupt mask for DM_CM_INFO1.SEQSUSPEND |
| 7 to 1 | — | All 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 0 | SEQEND_MASK | 1 | R/W | SEQEND interrupt masked |

#### 150.2.1.27 DMAC Interrupt Register 2 (DM_CM_INFO2)

The DMAC interrupt register 2 (DM_CM_INFO2) indicates the status of DMAC and a sequencer. To clear a flag, write 0 to the bit to be cleared and 1 to the other bits.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 20 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 19, 18 | — | All 0 | R | Reserved<br>The write value should always be 0. |
| 17 | DTRANERR1 | 0*1 | R/W | DMAC Channel 1 Error<br>[Setting condition]<br>When an error occurs on DMAC channel 1<br>[Clearing condition]<br>When 0 is written to DTRANERR1 |
| 16 | DTRANERR0 | 0*2 | R/W | DMAC Channel 0 Error<br>[Setting condition]<br>When an error occurs on the DMAC channel 0<br>[Clearing condition]<br>When 0 is written to DTRANERR0 |
| 15 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | SEQERR | 0*3 | R/W | Sequencer error<br>[Setting condition]<br>When a sequencer error occurs<br>[Clearing condition]<br>When 0 is written to SEQERR |

Notes: 1. The initial value is applied at a reset and when the DTRANRST1 bit in DM_CM_RST is 0.
2. The initial value is applied at a reset and when the DTRANRST0 bit in DM_CM_RST is 0.
3. The initial value is applied at a reset and when the SEQRST bit in DM_CM_RST is 0.

#### 150.2.1.28 DM_CM_INFO2 Interrupt Mask Register (DM_CM_INFO2_MASK)

The DM_CM_INFO2 interrupt mask register (DM_CM_INFO2_MASK) enables or disables the DM_CM_INFO2 interrupt. When 0 is set in DM_CM_INFO2_MASK while the corresponding flag in DM_CM_INFO2 is set, an interrupt occurs.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 20 | — | All 1 | R | Reserved bits<br>These bits are always read as 1 The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 19, 18 | — | All 1 | R | Reserved<br>The write value should always be 1. |
| 17 | DTRANERR1_MASK | 1 | R/W | DTRANERR1 interrupt masked |
| 16 | DTRANERR0_MASK | 1 | R/W | DTRANERR0 interrupt masked |
| 15 to 1 | — | All 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 0 | SEQERR_MASK | 1 | R/W | SEQERR interrupt masked |

#### 150.2.1.29 DMAC Sequencer Set Register (DM_CM_SEQ_REGSET)

This register indicates a DMAC(SEQ) context, table setting register.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 9 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 8 | TABLE_NUM | All 0 | R/W | Sequencer Table number<br>0 : sequencer table 0<br>1 : sequencer table 1<br>Software sets the number of sequencer table storing the value of DM_SEQ_***. |
| 7 to 2 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 1 to 0 | CTXT_NUM | All 0 | R/W | Context number (0~3)<br>Software sets the number of context storing the value of DM_SEQ_***. |

#### 150.2.1.30 DMAC Sequencer Control Register (DM_CM_SEQ_CTRL)

This register indicates a DMAC(SEQ) control register.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 29 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 28 | SEQ_TABLE | 0 | R/W | This bit indicates sequencer table number for queuing command. |
| 27 to 25 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 24 | T_NUM | All 0 | R/W | Maximum Table number for SDmultiple command stack<br>This bit indicates the maximum number of sequencer table (SEQ_TYPE=01). |
| 23 to 18 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 17 to 16 | SEQ_TYPE | All 0 | R/W | Sequencer mode selector<br>00: Not use Sequencer<br>01: SD multiple command stack<br>10: - (Setting prohibit)<br>11: - (Setting prohibit) |
| 15 to 14 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 13 to 12 | START_NUM | All 0 | R/W | These bits indicate start number of context for queuing command (0-3).<br>Set START_NUM is less than or equal to END_NUM.<br>SDIP stores START_NUM to the table indicated by SEQ_TABLE, when software sets these bits. SDIP outputs START_NUM of the table indicated by SEQ_TABLE, when software reads these bits. |
| 11 to 10 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 9 to 8 | END_NUM | All 0 | R/W | These bits indicate end number of context for queuing command (0-3).<br>Set END_NUM is greater than or equal to START_NUM.<br>SDIP stores END_NUM to the table indicated by SEQ_TABLE, when software sets these bits. SDIP outputs END_NUM of the table indicated by SEQ_TABLE, when software reads these bits. |
| 7 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 6 | PRIORITY_MODE | 0 | R/W | Priority control mode selector for SD multiple command stack *1<br>0: SDIP executes in order, from table0 to table1 (table 0 -> table 1).<br>1: SDIP executes in order of priority (table1 is higher priority than table0).<br>Don't switch this mode during sequencer process. |
| 5 to 4 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 3 | SEQ_SUSPEND | 0 | R/W | This bit triggers to suspend sequencer process<br>SDIP clears this bit automatically, when SDIP goes to suspend mode or DM_CM_INFO1.SEQEND interrupt is occurred. |
| 2 | SEQ_RESUME | 0 | R/W | This bit triggers to resume sequencer process<br>SDIP clears this bit automatically, when SDIP recovers from suspend mode. Don't set this bit to 1 when SDIP is not in suspend mode. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | — | 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | SEQ_START | 0 | R/W | This bit triggers to queuing command.<br>SDIP clears this bit automatically, when SDIP queues command.<br>SDIP queues command to the table whose number is set by SEQ_TABLE at SEQ_TYPE=01.<br>Start number and end number of context for queuing command are set by START_NUM and END_NUM. |

### 150.2.1.31 DMAC Transfer Mode Register (DM_CM_DTRAN_MODE)

This register indicates a DMAC(DTRAN) mode select register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 18 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 17 to 16 | CH_NUM | All 0 | R/W | DMAC channel selector<br>00 : SD down stream<br>01 : SD up stream<br>10, 11: - (Setting prohibit) |
| 15 to 6 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 5 to 4 | BUS_WIDTH | All 0 | R/W | Bus width selector<br>00: - (Setting prohibit)<br>01: - (Setting prohibit)<br>10: - (Setting prohibit)<br>11: 64-bit [default]<br>These bits are common set for all channels. |
| 3 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | ADDR_MODE | 0 | R/W | System address mode selector<br>0: fixed address<br>1: address increment<br>These bits are common for all channels. |

### 150.2.1.32 DMAC Transfer Control Register (DM_CM_DTRAN_CTRL)

This register indicates a DMAC(DTRAN) control register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 9 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 8 | — | 0 | R/W | Reserved<br>The write value should always be 0. |
| 7 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | DM_START | 0 | R/W | DMAC Start<br>Writing 1 to this bit starts DMAC operation. This bit is automatically cleared when DMA transfer is started. |

#### 150.2.1.33 DMAC Reset Register (DM_CM_RST)

The Initial value should be written on each bit not in use now for the future extended specification.
High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | — | All 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 9 to 8 | DTRANRST | All 0 | R/W | When software sets this bit to 0, SDIP executes soft reset to control FF of each channel. When software sets this bit to 1, SDIP releases soft reset.<br>DTRANRST[0]: control FF of channel 0, 1<br>Set this bit to 0 for Error recovery (for DM_CM_INFO2.DTRANERR).<br>Set this bit to 0 only for the purpose mentioned above, otherwise the SDIP will not accurately operate. |
| 7 to 1 | — | All 1 | R | Reserved bits<br>These bits are always read as 1. The write value should always be 1. Operation is not guaranteed if a value other than 1 is written to these bits. |
| 0 | SEQRST | 0 | R/W | When software sets this bit to 0, SDIP executes soft reset to control FF of sequencer. When software sets this bit to 1, SDIP releases soft reset.<br>Set this bit to 0 for Error recovery (for DM_CM_INFO2.SEQERR).<br>Set this bit to 0 only for the purpose mentioned above, otherwise the SDIP will not accurately operate. |

#### 150.2.1.34 DMAC Status Register (DM_CM_SEQ_STAT)

This register outputs the status register of sequencer process. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 2 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 1 to 0 | SEQTBSTS | All 0 | R/W | busy flag of sequencer process<br>These bits indicate that sequencer process is busy or not for each table (SEQTBSTS[1 :0]={table1, table0}). These bits are asserted from software sets DM_CM_SEQ_CTRL.SEQ_START=1 to command sequencer process completion.<br>0: not busy<br>1: busy |

#### 150.2.1.35 DMAC Transfer Address Register (DM_DTRAN_ADDR)

This register indicates system memory address(forwarding/source). This register is indirect register. SDIP stores in context indicated by DM_CM_DTRAN_MODE.CH_NUM, when software sets this register. SDIP outputs the value of context indicated by DM_CM_DTRAN_MODE.CH_NUM, when software reads this register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 7 | DADDR | All 0 | R/W | destination address / source address (128 byte unit)<br>Note that the value of DM_DTRAN_ADDR + transfer data length (more detail, refer to chapter 5) is less than or equal to 2^32. |
| 6 to 0 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

### 150.2.1.36 DMAC Sequencer Command Register (DM_SEQ_CMD)

This register indicates a command controller register for sequencer in DM_CM_SEQ_CTRL.SEQ_TYEP=01. This register is indirect register. SDIP stores in context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software sets this register. SDIP outputs the value of context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software reads this register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 17 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 16 | PRIORITY_MASK | 0 | R/W | Priority Mask for SD multiple command stack<br>When software sets this bit to 1, SDIP goes to high priority sequencer table after current context is completed. |
| 15 to 14 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 13 | MULTI | 0 | R/W | Selects Single or Multiple block transfer<br>0: single block transfer ( byte mode )<br>1: multiple block transfer ( block mode ) |
| 12 | DIO | 0 | R/W | Selects Write or Read transfer<br>0: write transfer<br>1: read transfer |
| 11 | CMDTYP | 0 | R/W | Selects the command with data or without data.<br>0: without data<br>1: with data |
| 10 to 8 | RESTYP | All 0 | R/W | Response Type<br>These bits select Response type as shown below.<br>000 : Not Specified<br>011 : No Response<br>100 : R1, R5, R6, R7<br>101 : R1b, R5b<br>110 : R2<br>111 : R3, R4<br>Others : Reserved(Setting prohibit) |
| 7 | NONAUTOSTP | 0 | R/W | UHS-I only<br>0: issue Auto CMD12<br>1: not issue Auto CMD12 |
| 6 | APP | 0 | R/W | indicator for application command<br>0: regular command<br>1: application command |
| 5 to 0 | INDEX | All 0 | R/W | Command Index |

#### 150.2.1.37 DMAC Sequencer Argument Register (DM_SEQ_ARG)

This register indicates a command argument register for sequencer in DM_CM_SEQ_CTRL.SEQ_TYPE=01. This register is indirect register. SDIP stores in context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software sets this register. SDIP outputs the value of context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software reads this register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 0 | ARG | All 0 | R/W | command argument |

#### 150.2.1.38 DMAC Sequencer Size Set Register (DM_SEQ_SIZE)

This register indicates the number of bytes per sector register for sequencer in DM_CM_SEQ_CTRL.SEQ_TYPE=01. This register is indirect register. SDIP stores in context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software sets this register. SDIP outputs the value of context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software reads this register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 0 | SIZE | All 0 | R/W | Number of bytes per sector (byte unit)<br>Set the value which is equal to or more than 8 and a power-of-two to this register at DM_SEQ_CMD.MULTI=1.<br>Note that DM_SEQ_SIZE * DM_SEQ_SECCNT is less than 2^32. |

#### 150.2.1.39 DMAC Sequencer Count Set register (DM_SEQ_SECCNT)

This register indicates the number of sector counts register for sequencer in DM_CM_SEQ_CTRL.SEQ_TYPE=01. This register is indirect register. SDIP stores in context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software sets this register. SDIP outputs the value of context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software reads this register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 0 | CNT | All 0 | R/W | The number of sector counts<br>This register is invalid at DM_SEQ_CMD.MULTI=0.<br>Note that DM_SEQ_SIZE * DM_SEQ_SECCNT is less than 2^32. |

#### 150.2.1.40 DMAC Sequencer Response Register (DM_SEQ_RSP)

This register indicates the expected value of argument of command response for sequencer in DM_CM_SEQ_CTRL.SEQ_TYPE=01. This register is indirect register. SDIP stores in context indicated by

DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software sets this register. SDIP outputs the value of context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software reads this register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 0 | RSP | All 0 | R/W | The expected value for the argument of response.<br>When DM_SEQ_CMD.{MULTI,CMDTYP,NONAUTOSTP}=110, SDIP uses the value of this register to the expected value for the argument of response both of data transfer command and Auto CMD12. |

This register is not supported to R2 type response.

### 150.2.1.41 DMAC Sequencer Response Check register (DM_SEQ_RSP_CHK)

This register indicates the bit of comparing expected value of argument of command response for sequencer in DM_CM_SEQ_CTRL.SEQ_TYPE=01. This register is indirect register. SDIP stores in context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software sets this register. SDIP outputs the value of context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software reads this register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 0 | RSP_CHK | All 0 | R/W | This register indicates the bit which performs comparison with expected value for the argument of response.<br>0: SDIP does not compare. [default]<br>1: SDIP compares DM_SEQ_RSP and actual read data.<br>When DM_SEQ_CMD.{MULTI,CMDTYP,NONAUTOSTP}=110, SDIP uses the value of this register to the expected value for the argument of response both of data transfer command and Auto CMD12. |

### 150.2.1.42 DMAC Sequencer Address Register (DM_SEQ_ADDR)

This register indicates system memory address (forwarding/source) for sequencer in DM_CM_SEQ_CTRL.SEQ_TYPE=01. This register is indirect register. SDIP stores in context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software sets this register. SDIP outputs the value of context indicated by DM_CM_SEQ_REGSET.{TABLE_NUM, CTXT_NUM}, when software reads this register. High 32 bits (bit 63-32) are read only 0.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 63 to 32 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 31 to 7 | SADDR | All 0 | R/W | destination address / source address (128 byte unit)<br>Note that the value of DM_SEQ_ADDR + transfer data length (more detail, refer to chapter 5) is less than 2^32. |
| 6 to 0 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

#### 150.2.1.43 Initial Setting Register (SCC_DTCNTL)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 23 to 16 | TAPNUM7 to TAPNUM0 | H'0A | R/W | SCC Sampling Clock Selection Width<br>When the SD clock frequency obtained by setting DIV7 to DIV0 in the SD_CLK_CTRL register to 0xFF (1:1 mode) is 200 MHz, set these bits to H'08. |
| 15 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | TAPEN | 0 | R/W | SCC Sampling Clock Operation Enable<br>0: SCC sampling clock operation is disabled.<br>1: SCC sampling clock operation is enabled. |

#### 150.2.1.44 Sampling Clock Position Setting Register (SCC_TAPSET)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 7 to 0 | TAPSET7 to TAPSET0 | H'00 | R/W | SCC Sampling Clock Position<br>▪ Set the tuning result in the range from 0 to TAPNUM-1. |

#### 150.2.1.45 Sampling Clock Selection Register (SCC_CKSEL)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | DTSEL | 0 | R/W | Sampling Clock Selection<br>0: An SCC sampling clock is not used (other than SDR104, HS200, HS400).<br>1: An SCC sampling clock is used (SDR104, HS200, HS400).<br>• For SDR104, HS200 and HS400, set DTSEL to 1.<br>• For SDR104 and HS200, set DIV7 to DIV0 in the SD_CLK_CTRL register to H'FF (1:1 mode).<br>• For HS400, set DIV7 to DIV0 in the SD_CLK_CTRL register to H'00 (SDϕ/2 mode).<br>• When this bit is switched, stop the SD clock output from the SD host interface (set SCLKEN in SD_CLK_CTRL to 0). |

### 150.2.1.46 Sampling Clock Position Correction Register (SCC_RVSCNTL)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 to 8 | TAPSEL7 to TAPSEL0 | H'00 | R | SCC Sampling Clock Position Display<br>• Displays the SCC sampling clock position selected by hardware.<br>• After RVSEN has been set to 1, the value may differ from that of TAPSET. |
| 7 to 2 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 1 | RVSW | 0 | R/W | This bit is read as 0. The write value should be 0. |
| 0 | RVSEN | 0 | R/W | SCC Sampling Clock Position Correction Enable<br>0: SCC sampling clock position correction is disabled.<br>1: SCC sampling clock position correction is enabled.<br>• When RVSEN is set to 1 after tuning has been performed, this module corrects the SCC sampling clock position each time of a command sequence of the SD host interface.<br>• However, when RVSERR is 1, this module does not correct the SCC sampling clock position.<br>• While tuning is being performed, set RVSEN to 0. |

### 150.2.1.47 Sampling Clock Position Correction Request Register (SCC_RVSREQ)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 2 | RVSERR | 0 | R/W | SCC Sampling Clock Position Correction Error<br>0: There is no correction error.<br>1: There is a correction error.<br>• If this bit is set to 1 after a command sequence, write 0 to this bit and perform tuning again.<br>• Ignore this bit while tuning is being performed.<br>• Writing 1 to this bit is disabled and writing 0 to this bit is only enabled. |
| 1 | REQTAPUP | 0 | R/W | SCC Sampling Clock Position Positive Direction Correction Request<br>0: There is no correction request.<br>1: There is a correction request.<br>• If this bit is set to 1 after a command sequence, write 0 to this bit and rewrite TAPSET in the positive direction (when TAPSEL = TAPNUM-1, set 0 to TAPSET).<br>• When RVSEN is 1, this bit is disabled (this bit is not set to 1).<br>• Writing 1 to this bit is disabled and writing 0 to this bit is only enabled. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | REQTAPDWN | 0 | R/W | SCC Sampling Clock Position Negative Direction Correction Request<br>0: There is no correction request.<br>1: There is a correction request.<br>• If this bit is set to 1 after a command sequence, write 0 to this bit and rewrite TAPSET in the negative direction (when TAPSEL = 0, set TAPNUM-1 to TAPSET).<br>• When RVSEN is 1, this bit is disabled (this bit is not set to 1).<br>• Writing 1 to this bit is disabled and writing 0 to this bit is only enabled. |

### 150.2.1.48 Hardware Adjustment Register 1 (SCC_DT2FF)

This register makes a setting that SD_DATA, which has been fetched by the sampling clock at each TAP position, is used in the appropriate timing.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 15 to 8 | DT2NESET7 to DT2NESET0 | H'04 | R/W | Hardware Adjustment 1<br>The following is the recommended setting value for each SD_CLK frequency. This bit must be changed from its initial value.<br>SDR104 mode ： H'03<br>HS200 mode ： H'03<br>HS400 mode ： H’03 |
| 7 to 0 | DT2NSSET7 to DT2NSSET0 | H'01 | R/W | Hardware Adjustment 2<br>The following is the recommended setting value for each SD_CLK frequency. This bit must be changed from its initial value.<br>SDR104 mode ：H'00<br>HS200 mode ： H'00<br>HS400 mode ： H'00 |

### 150.2.1.49 Hardware Adjustment Register 2 (SCC_TMPPORT2)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | HS400EN | 0 | R/W | Set this bit to 1 to select operation of this module in HS400 mode.<br>0: Disables HS400 mode.<br>1: Enables HS400 mode. |
| 30 to 5 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 4 | HS400OSEL | 0 | R/W | Set this bit to 1 to select operation of this module in HS400 mode.<br>0: Disables HS400 data output timing.<br>1: Enables HS400 data output timing. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 to 0 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |

#### 150.2.1.50 Sampling data comparison register (SCC_SMPCMP)

Data comparison register indicates the result of the comparison of the sampling data. The subject of the comparison is the before and the behind TAP.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 25 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 24 to 16 | CMPNGU | All 0 | R/W | Comparison of sampling data with the previous TAP Clock. Bit 16-23 is the comparison result of data 0-7. Bit 24 is the comparison result of CMD.<br>0: Match<br>1: Mismatch<br>< Clear conditions.><br>The start of the command sequence<br>Write to SCC_TAPSET register |
| 15 to 9 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 8 to 0 | CMPNGD | All 0 | R/W | Comparison of sampling data with the after TAP Clock. Bit 0-7 is the comparison result of data 0-7. Bit 8 is the comparison result of CMD.<br>0: Match<br>1: Mismatch<br>< Clear conditions.><br>The start of the command sequence<br>Write to SCC_TAPSET register |

#### 150.2.1.51 Hardware Adjustment Register 3 (SCC_TMPPORT3)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 1 to 0 | ofsel[1-0] | 10 | R/W | Select offset value of 90 degree phase shifter for HS400:<br>00：max (offset=3)<br>11：min (offset=0) |

#### 150.2.1.52 Hardware Adjustment Register 4 (SCC_TMPPORT4)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 0 | dll_acc_start | All 0 | R/W | DLL(90 degree phase shifter) register Access Start<br>0：register access stop<br>1：register access start |

#### 150.2.1.53 Hardware Adjustment Register 5 (SCC_TMPPORT5)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 8 | dll_rw_sel | All 0 | R/W | DLL(90 degree phase shifter) register read/write select<br>0：Write<br>1：Read |
| 7 to 6 | — | All 0 | R | Reserved bits<br>These bits are always read as 0. The write value should always be 0. Operation is not guaranteed if a value other than 0 is written to these bits. |
| 5 to 0 | dll_addr[5-0] | All 0 | R/W | DLL(90 degree phase shifter) register address |

#### 150.2.1.54 Hardware Adjustment Register 6 (SCC_TMPPORT6)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | dll_wdata[31-0] | All 0 | R/W | DLL(90 degree phase shifter) register write data |

#### 150.2.1.55 Hardware Adjustment Register 7 (SCC_TMPPORT7)

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | dll_rdata[31-0] | All 0 | R/W | DLL(90 degree phase shifter) register read data |

## 150.3 Operation

### 150.3.1 SD Interface

**(1) SD Data Format**

When data is read from the SD card, the procedure is as follows.

1) The SD host interface receives data from the SD card via the SDDAT signal. (SDDAT signal: see figures 150.2, 150.3 and 150.4.)
2) The receive data is stored in SD_BUF of the SD host interface. (SD_BUF store data: see Figure 150.5)
3) The data stored in SD_BUF is read from SD_BUF0. (Reading from SD_BUF0: see Figure 150.6)

When data is written to the SD card, the above procedure will be reversed.

When accessing SD_BUF0, caution should be taken for the transfer order in SDDAT and the store order in SD_BUF. In addition, data stored in SD_BUF0 can be replaced in bytes with the HOST_MODE. (See Table 150.5)

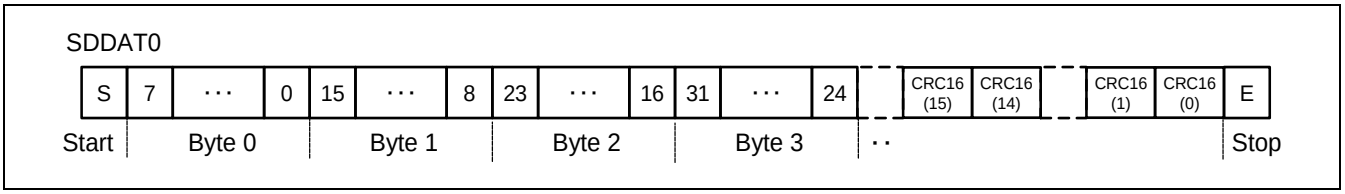



**Figure 150.2 SDDAT in 1-Bit Width Mode**

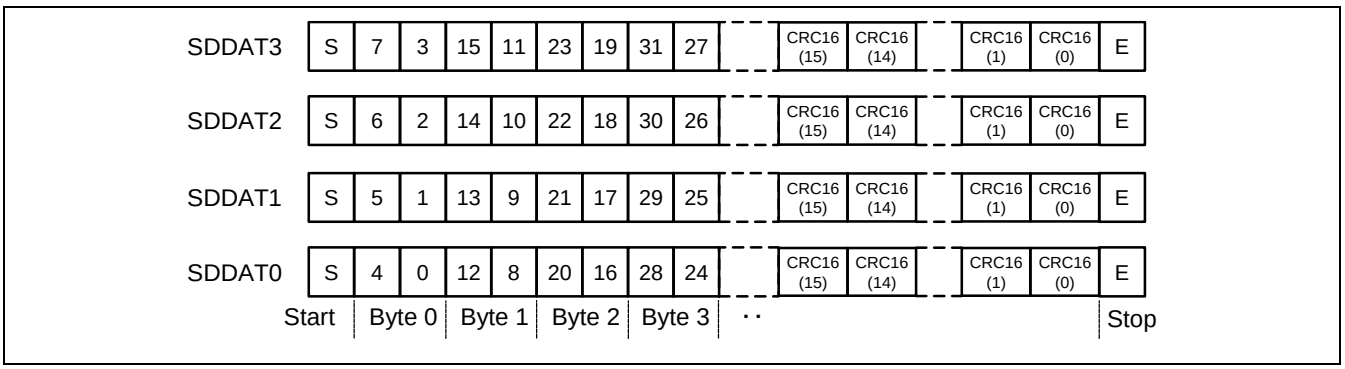



**Figure 150.3 SDDAT in 4-Bit Width Mode**

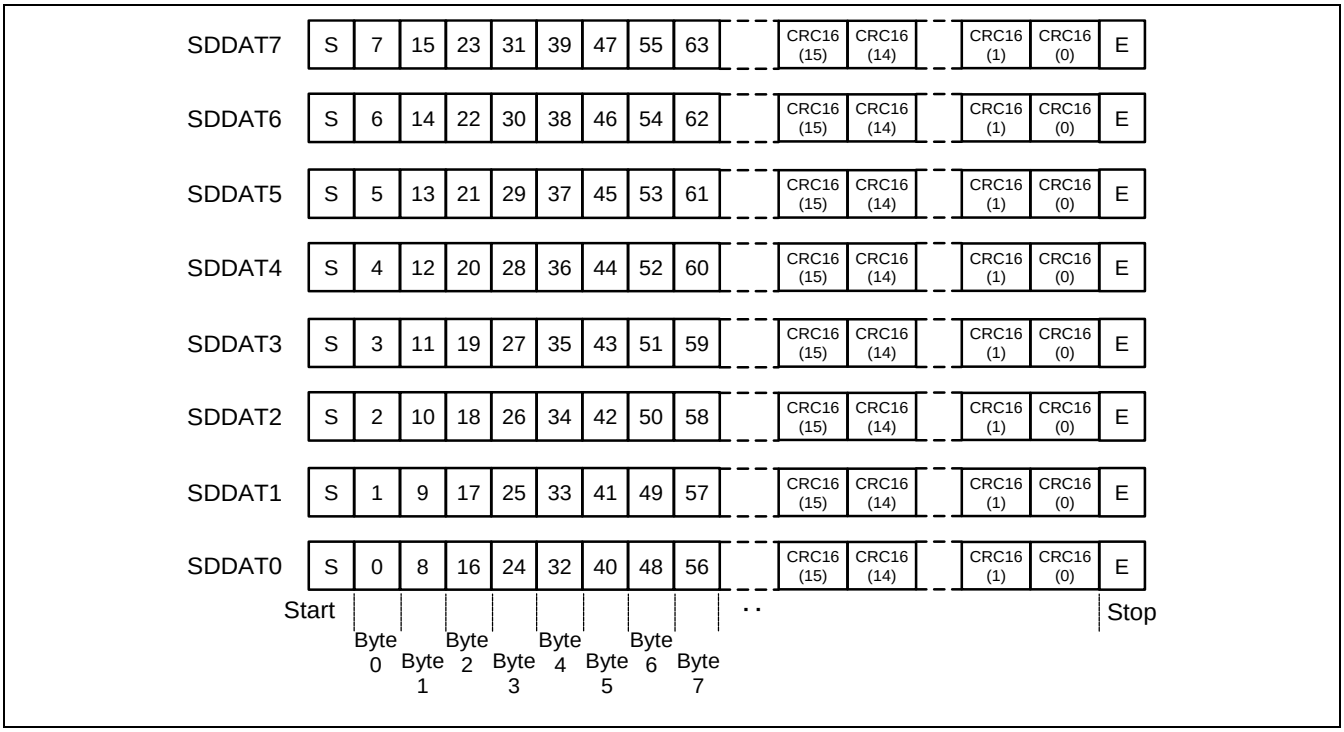



**Figure 150.4 SDDAT in 8-Bit Width Mode**

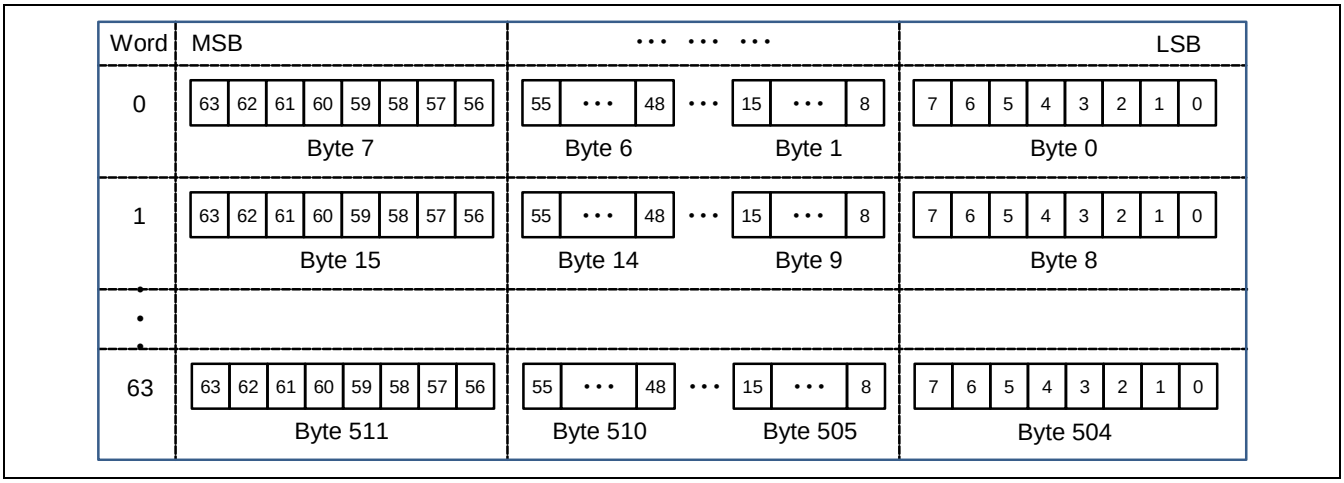



**Figure 150.5 SD_BUF Store Data**

**Table 150.5 Reading from SD_BUF0**

| WMODE*1 | BUSWIDTH*1 | ENDIAN*1 | Read Data*2 |
|---|---|---|---|
| 0 | 0 | 0 | H'0123456789ABCDEF |
| 0 | 0 | 1 | H'EFCDAB8967452301 |
| 1 | 1 | 0 | H'89ABCDEF (1st)<br>H'01234567 (2nd) |
| 1 | 1 | 1 | H'EFCDAB89 (1st)<br>H'67452301 (2nd) |
| 1 | 0 | 0 | H'CDEF (1st)<br>H'89AB (2nd)<br>H'4567 (3rd)<br>H'0123 (4th) |
| 1 | 0 | 1 | H'EFCD (1st)<br>H'AB89 (2nd)<br>H'6745 (3rd)<br>H'2301 (4th) |

Notes: 1. The name of a bit in HOST_MODE.
2. When the data stored in SD_BUF is H'0123456789ABCDEF

**(2) Bus Signal Voltage Switch**

Change the electric potential of the bus signal in the following procedure after checking that the SD card supports 1.8 V.

(i) Issuing CMD11

Perform command sequence processing of CMD11.

(ii) Stopping the SD clock (a)

Set the SCLKEN bit in the SD_CLK_CTRL to 0 to stop* the output of the SD clock. When the SDCLKOFFEN bit in the SD_CLK_CTRL register is 1, the SDCLKOFFEN bit is also set to 0.

Note: When the SDCLKOFFEN bit in the SD_CLK_CTRL register is 1, the SD clock has automatically been stopped.

(iii) Checking the value of SDDAT

Check that the DAT0 bit in the SD_INFO2 register is 0.

(iv) Changing the electric potential of the host device

Change the electric potential of the LSI, which is supplied to the VDDQVA_SD pin, from 3.3 V to 1.8 V.

(v) Starting supply of the SD clock (b)

After the SD clock has been stopped ((a) above) and 5 ms or more has elapsed, set the SCLKEN bit in the SD_CLK_CTRL register to 1 and allow the output of the SD clock. The SDCLKOFFEN bit must be 0.

(vi) Checking the value of SDDAT

After supplying the SD clock has been started ((b) above) and 1 ms or more has elapsed, check that the DAT0 bit in the SD_INFO2 register is 1. It is possible to set the SDCLKOFFEN bit in the SD_CLK_CTRL register to 1 and allow SD Clock (SDCLK) Output Automatic Control Enable.

### 150.3.2 Card Detect/Write Protect

**(1) Card Detect**

The SD host interface has two types of card detect functions as described in the following.

- Card detect with ISDCD

  Figure 150.6 shows the timing chart of card detect using ISDCD. ISDCD is connected to the card socket and pulled up on the host device. The resistance of the pull-up resistor is decided by the specification of the SD host device.

  [Card insertion]

  ISDCD is pulled down when a card is inserted. At this time, if ISDCD has been pulled down for the Mcycle period (set in SD_OPTION), INFO4 in SD_INFO1 is set to 1. (It is cleared to by writing 0.)

  [Card removal]

  ISDCD is pulled up when a card is removed. At this time, if ISDCD has been pulled up for the Mcycle period (set in SD_OPTION), INFO3 in SD_INFO1 is set to 1. (It is cleared to by writing 0.)

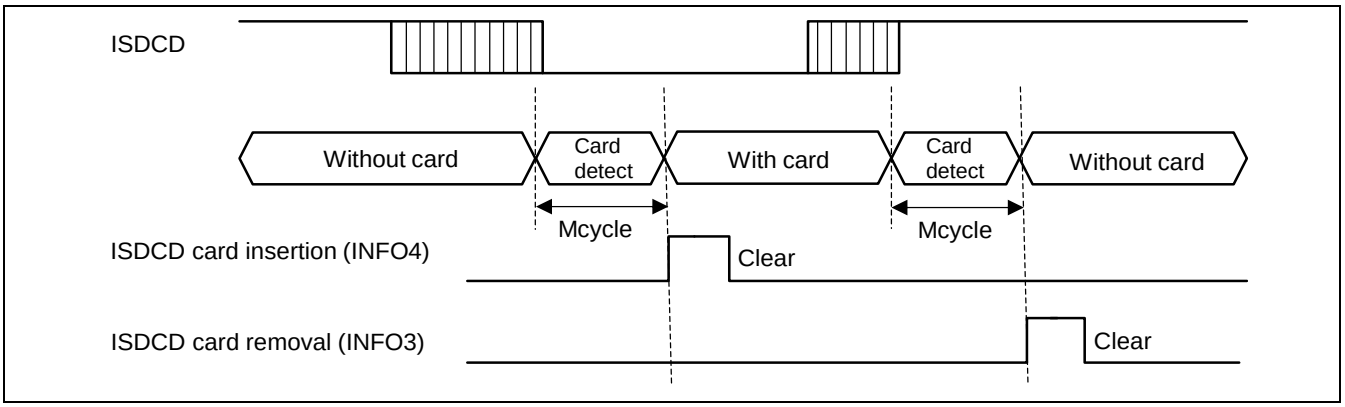



**Figure 150.6 Example of Card Detect with ISDCD**

- SD card detect with SDDAT3

  Figure 150.7 shows the timing chart when the SD card is detected using SDDAT3. In addition, SDDAT3 is pulled down on the host device. The resistance of the pull-down resistor is decided by the specification of the SD host device.

  [Card insertion]

  When an SD card is inserted, SDDAT3 is pulled up. Accordingly, INFO9 in SD_INFO1 is set to 1. (It is cleared to by writing 0.)

  [Card removal]

  When an SD card is removed, SDDAT3 is pulled down. Accordingly, INFO8 in SD_INFO1 is set to 1. (It is cleared to by writing 0.)

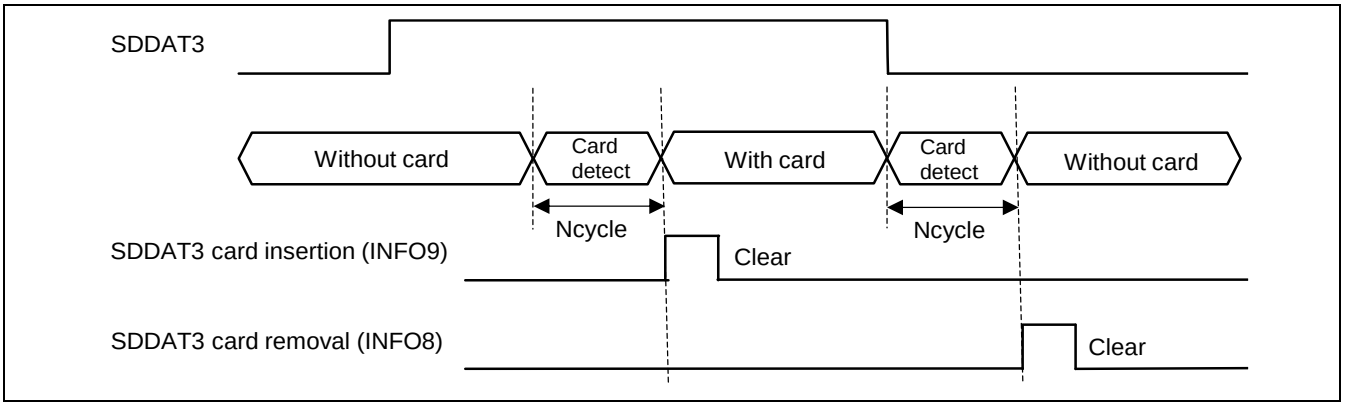



**Figure 150.7 SD Card Detect with SDDAT3**

**(2) Write Protest**

The SD host interface has two types of write protect functions.

- Write protect with ISDWP

  ISDWP is connected to the card socket, and pulled up or pulled down by the card insertion. The selection of pulling up or pulling down and the resistance value is decided by the specifications of the SD host device. As the ISDWP state is reflected to INFO7 in SD_INFO1, the write protect is decided after the SD card is inserted.
- Write protect with command

  The card's internal write protection and the card lock/unlock operation are realized by the command.

### 150.3.3 Interrupt Request

#### (1) Interrupt Request

The SD host interface has the interrupt requests shown in Table 150.6. Table 150.6 shows the relationship between the interrupt flag registers and the interrupt mask registers. When a bit in an interrupt mask register is set to 0, an interrupt occurs by setting the corresponding bit in the interrupt flag register to 1.

To clear a flag, write 0 to the bit to be cleared and 1 to the other bits.

**Table 150.6 Interrupt Request**

| | Interrupt Flag Register | | Interrupt Mask Register | |
|---|---|---|---|---|
| **Interrupt Request** | **Register Name** | **Bit Name** | **Register Name** | **Bit Name** |
| Card access interrupt | SD_INFO1 | INFO2 | SD_INFO1_MASK | IMASK2 |
| | | INFO0 | | IMASK0 |
| | SD_INFO2 | ILA | SD_INFO2_MASK | IMASK |
| | | BWE | | BMASK1 |
| | | BRE | | BMASK0 |
| | | ERR6 | | EMASK6 |
| | | ERR5 | | EMASK5 |
| | | ERR4 | | EMASK4 |
| | | ERR3 | | EMASK3 |
| | | ERR2 | | EMASK2 |
| | | ERR1 | | EMASK1 |
| | | ERR0 | | EMASK0 |
| SDIO access interrupt | SDIO_INFO1 | EXWT | SDIO_INFO1_MASK | MEXWT |
| | | EXPUB52 | | MEXPUB52 |
| | | IOIRQ | | IOMSK |
| Card detect interrupt | SD_INFO1 | INFO9 | SD_INFO1_MASK | IMASK9 |
| | | INFO8 | | IMASK8 |
| | | INFO4 | | IMASK4 |
| | | INFO3 | | IMASK3 |
| DMAC interrupt | DM_CM_INFO1 | DTRANEND1 | DM_CM_INFO1_MASK | DTRANEND1_MASK |
| | | DTRANEND0 | | DTRANEND0_MASK |
| | | SEQEND | | SEQEND_MASK |
| | DM_CM_INFO2 | DTRANERR1 | DM_CM_INFO2_MASK | DTARERR1_MASK |
| | | DTRANERR0 | | DTARERR0_MASK |
| | | SEQERR | | SEQERR_MASK |

### 150.3.4 Communications Errors and Timeouts

- Communications Errors and Timeouts

Tables 150.7 and 150.8 show the relationships between the SD card interrupt flag register and SD error status register for communications errors and timeouts, respectively. When a bit in the SD card interrupt flag register is set to 1, the corresponding bit in the SD error status register is set to 1. The values of the SD error status register are cleared by writing to SD_CMD or writing 0 to the SDRST bit in SOFT_RST.

When an error or timeout occurs during boot mode, the DTRAN1ERR or SEQERR bit in DM_CM_INFO2 is set to 1. Confirm the source of error or timeout by reading SD_INFO2, SD_ERR_STS1, and SD_ERR_STS2.

**Table 150.7 Communications Errors**

| Communication Error | SD Card Interrupt Flag Register | | SD Error Status Register | | Description |
|---|---|---|---|---|---|
| | Register Name | Bit Name | Register Name | Bit Name | |
| END error | SD_INFO2 | ERR2 | SD_ERR_STS1 | E5 | When an error occurs in the CRC status length |
| | | | | E4 | When an error occurs in read data length |
| | | | | E3 | When an error occurs in the response length to a command issued within a command sequence |
| | | | | E2 | When an error occurs in the response length (other than a response to a command issued within a command sequence) |
| CRC error | | ERR1 | | E11 | When an error occurs in the CRC status |
| | | | | E10 | When a CRC error occurs in the read data |
| | | | | E9 | When a CRC error occurs in the response to a command issued within a command sequence |
| | | | | E8 | When a CRC error occurs in the response (other than a response to a command issued within a command sequence) |
| CMD error | | ERR0 | | E1 | The command index of the transmitted command differed from the command index of the received response (for a command issued within a command sequence) |
| | | | | E0 | The command index of the transmitted command differed from the command index of the received response (for a command issued other than within a command sequence) |

**Table 150.8 Timeouts**

| Timeout | SD Card Interrupt Flag Register | | SD Error Status Register | | Description |
|---|---|---|---|---|---|
| | Register Name | Bit Name | Register Name | Bit Name | |
| Response timeout | SD_INFO2 | ERR6 | SD_ERR_STS2 | E1 | When the response to a command issued within a command sequence is not received though a longer time than 640 cycles of SDCLK have elapsed |
| | | | | E0 | When the response (other than a response to a command issued within a command sequence) is not received though a longer time than 640 cycles of SDCLK have elapsed |
| Data timeout (other than response timeout) | | ERR3 | | E6 | When the busy state (SDDAT0 = 0) continues for longer than Ncycle* after the CRC status |
| | | | | E5 | When the CRC status is not received though a longer time than Ncycle* has elapsed after data writing |
| | | | | E4 | When read data is not received though a longer time than Ncycle* has elapsed after read command |
| | | | | | When read data for the next block are not received though a longer time than Ncycle* has elapsed after the reception of read data |
| | | | | | When read data for the next block are not received though a longer time than Ncycle* has elapsed after release of the read wait state |
| | | | | E3 | When the busy state (SDDAT0 = 0) for longer than Ncycle* continues after CMD12 has been issued within a command sequence. |
| | | | | E2 | When the busy state (SDDAT0 = 0) for longer than Ncycle* continues after R1b response |

Note: * Ncycle is set by bit 7 to bit 4 in SD_OPTION.

### 150.3.5 Multiple commands serial issue

The SEQ stacks multiple commands by the sequencer table.

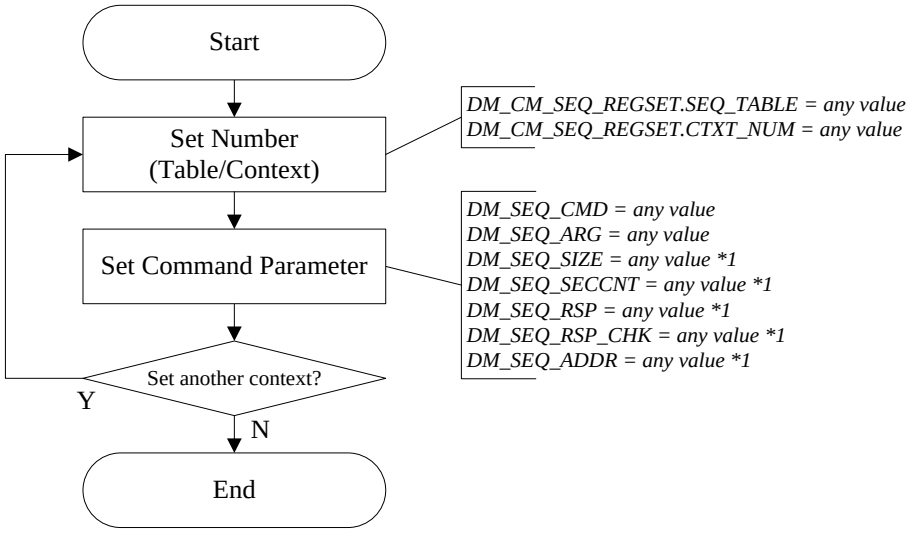
And the SEQ issues (SDMEM, SDIO) command sequentially and checks the response (including clear of response interrupt). The SEQ has command stack by the sequencer table and the priority control of command issuing. The SEQ has 2 sequencer tables and each sequencer table has 4 contexts. The SEQ can issue maximum 8 commands sequentially.

#### (1) Context setting

The following flowchart indicates context setting procedure.

1) Software sets DM_CM_SEQ_REGSET to the number of sequencer table (TABLE_NUM) and the number of context (CTXT_NUM).

2) Software sets command parameters (DM_SEQ_***) for context of step 1.




*1 Set parameters as needed.

#### (2) SEQ startup / suspend / resume

The SEQ starts up sequencer for the table indicated by DM_CM_SEQ_CTRL.SEQ_TABLE when software sets DM_CM_SEQ_CTRL.SEQ_START to 1.

Software can not access registers except for DM_*** during sequencer process. If software accesses registers except for DM_*** during sequencer process, suspend it using suspend/resume function. Don't overwrite context during sequencer process (from software sets DM_CM_SEQ_CTRL.SEQ_START=1 to SDIP asserts DM_CM_INFO1.SEQEND).

When software sets DM_CM_SEQ_CTRL.SEQ_SUSPEND=1 during sequencer process, SDIP suspends it by the command (context).

When software sets DM_CM_SEQ_CTRL.SEQ_RESUME=1, SDIP continues sequencer process.

#### (3) Interrupt during sequencer process

Interrupts from OXMNIRQ occurred during sequencer process are as follows.

| | |
|---|---|
| output interrupts | DM_CM_INFO1.<each factor>,<br>DM_CM_INFO2.<each factor> |
| masked interrupts | SD_INFO1.<each factor>,<br>SD_INFO2.<each factor>,<br>SDIO_INFO.<each factor>, |

Interrupts from OXASIOIRQ are independent from sequencer process.

### (4) Priority control

SDIP has 2 priority control modes (fix mode, interrupt mode).

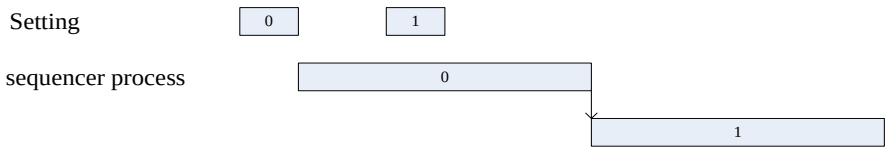
In fix mode (DM_CM_SEQ_CTRL.PRIORITY_MODE=0), SDIP executes in order, from table0 to table1.

Software can set context for table1 during sequencer process for table0. Below is example flow of sequencer process in fix mode.

"setting" indicates process from context setting to DM_CM_SEQ_CTRL.SEQ_START=1.

"sequencer process" indicates during execution of command (context) process.

DM_CM_SEQ_CTRL.PRIORITY_MODE=0

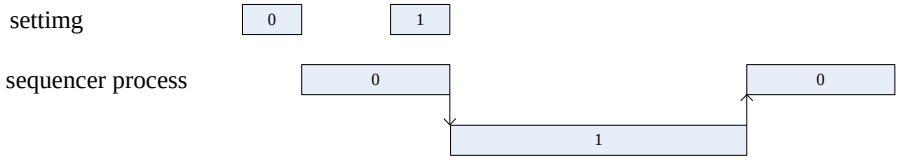



In interrupt mode (DM_CM_SEQ_CTRL.PRIORITY_MODE=1), SDIP executes in order of priority (table1 is higher priority than table0). SDIP switches table by the command (context). In the case that one process consists of some commands, SDIP can switch table by that process unit(Set DM_SEQ_CMD.PRIORITY_MASK at context setting). Below is example flow of sequencer process in interrupt mode.

"setting" indicates process from context setting to DM_CM_SEQ_CTRL.SEQ_START=1.

"sequencer process" indicates during execution of command (context) process.

DM_CM_SEQ_CTRL.PRIORITY_MODE=1



### 150.3.6 Usage Example

#### 150.3.6.1 Command without Data Transfer

**(1) Flowchart**

Figures 150.8 and 150.9 show flowchart examples.

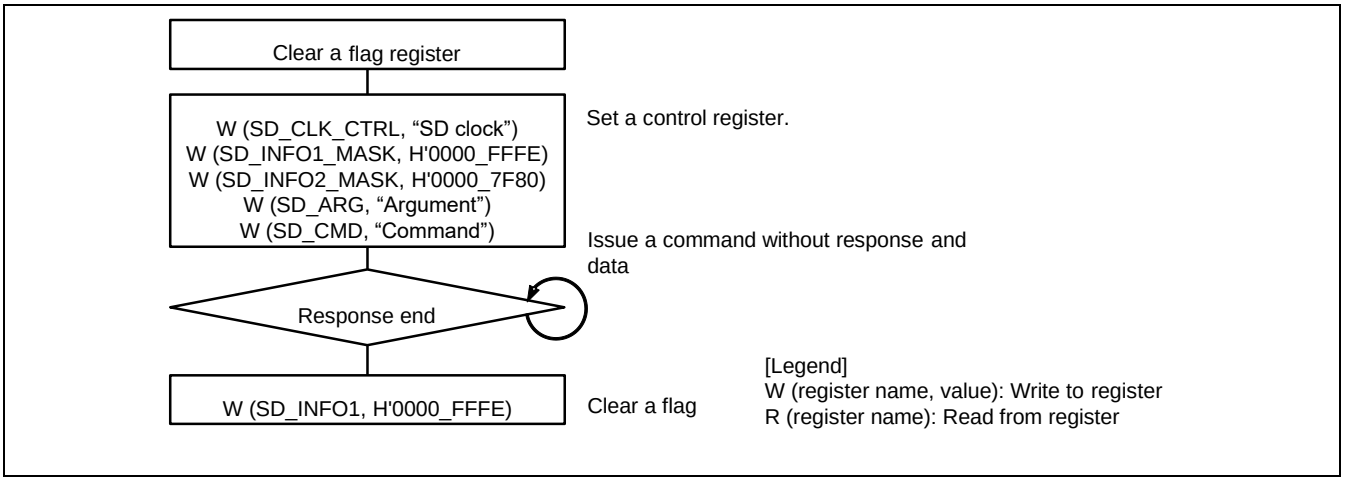



**Figure 150.8 Flow Example of Command without Response and Data**

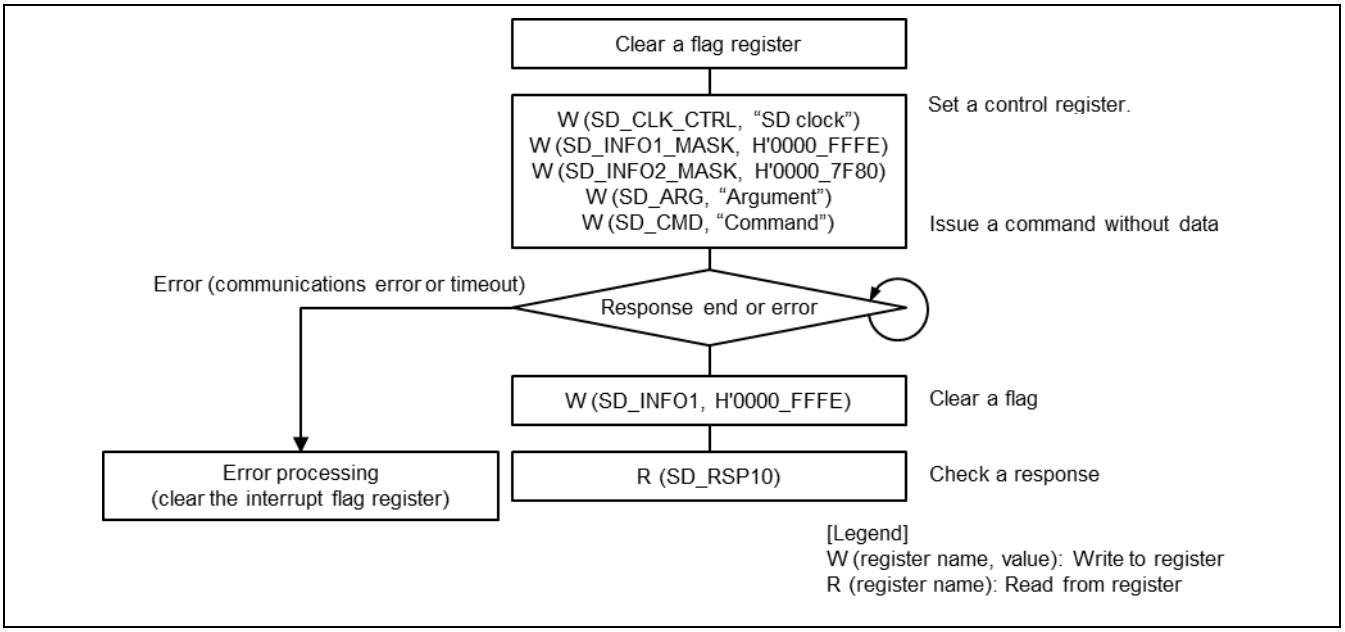



**Figure 150.9 Flow Example of Command without Data**

### (2) Operation for Command without Data Transfer

The legend below is used for description of register read/write.

W (register name, value): Write to register
R (register name): Read from register

The operation is described as below.

#### (a) Command without response and data

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)
2) Control register set
   Set the SD clock (SDCLK), interrupt mask, and so on. (SD_CLK_CTRL, SD_INFO1_MASK, and SD_INFO2_MASK)
3) Command issue
   Set CMD Argument in SD_ARG and write to SD_CMD.
   Accordingly, CMD is issued, and the operation is started.
4) Flag clear
   When transmission of a command is completed, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0.

#### (b) Command without data

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)
2) Control register set
   Set the SD clock (SDCLK), interrupt mask, and so on. (SD_CLK_CTRL, SD_INFO1_MASK, and SD_INFO2_MASK)
3) Command issue
   Set CMD Argument in SD_ARG and write to SD_CMD.
   Accordingly, CMD is issued, and the operation is started.
4) Flag clear
   When a response is received, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0.
5) Read a response from SD_RSP10.
   Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

### 150.3.6.2 Single Block Read

#### (1) Flowchart

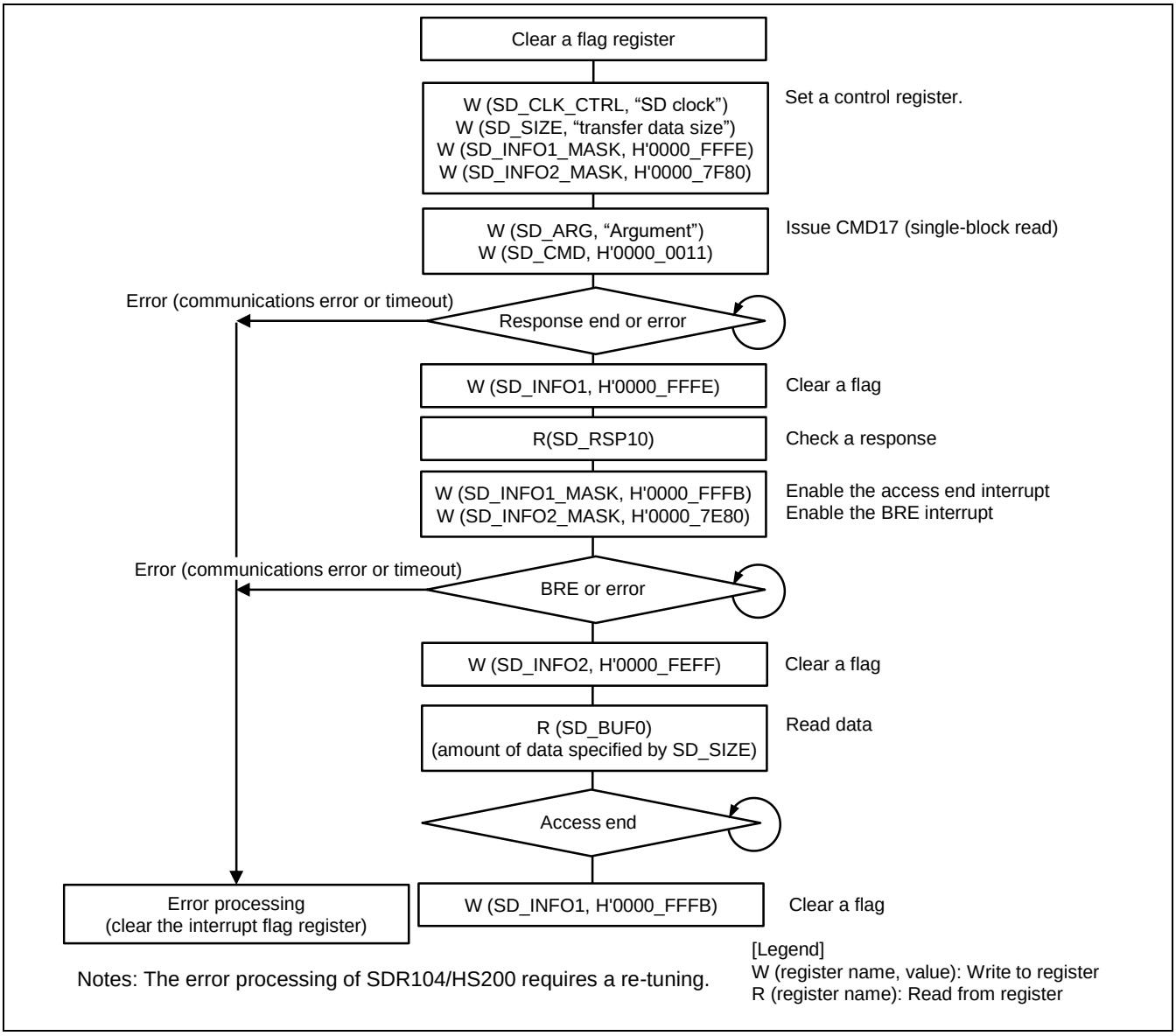



**Figure 150.10 Single Block Read Flowchart Example**

### (2) Operation for Single Block Read

The operation of the single block read is described below.

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)
2) Control register set
   Set the SD clock (SDCLK), transfer data size, interrupt mask, and so on. (SD_CLK_CTRL, SD_SIZE, SD_INFO1_MASK, and SD_INFO2_MASK)
3) Command issue (CMD17)
   Set CMD17 Argument in SD_ARG and write H'0000 0011 to SD_CMD.
   Accordingly, CMD17 is issued, and the single block read operation is started.
4) Response check
   On receiving the response, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0 and read the response from SD_RSP10.
   If the result of response decoding is an error, the command sequence can be halted by setting the STP bit in SD_STP or the IOABT bit in SDIO_MODE to 1. Furthermore, this causes CMD12 and CMD52 to not be issued. If the command sequence is halted, the INFO2 bit (access end) in SD_INFO1 is set to 1 to generate an interrupt.
5) Data receive from SD card and data read
   Write H'0000 FFFB to SD_INFO1_MASK to enable the access end interrupt. In addition, write H'0000 7E80 to SD_INFO2_MASK to enable the BRE interrupt. When the data receive from the SD card is completed, the BRE bit in SD_INFO2 is set to 1 to generate an interrupt. Clear the BRE bit to 0 and read the amount of data specified by SD_SIZE from SD_BUF0.
   However, a communications error or timeout may be generated if data are being received while reading of SD_BUF0 is in progress.
6) Operation complete
   When the data read from SD_BUF0 is completed, INFO2 (access end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO2 to 0 to end the single block read operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

### 150.3.6.3 Single Block Write

#### (1) Flowchart

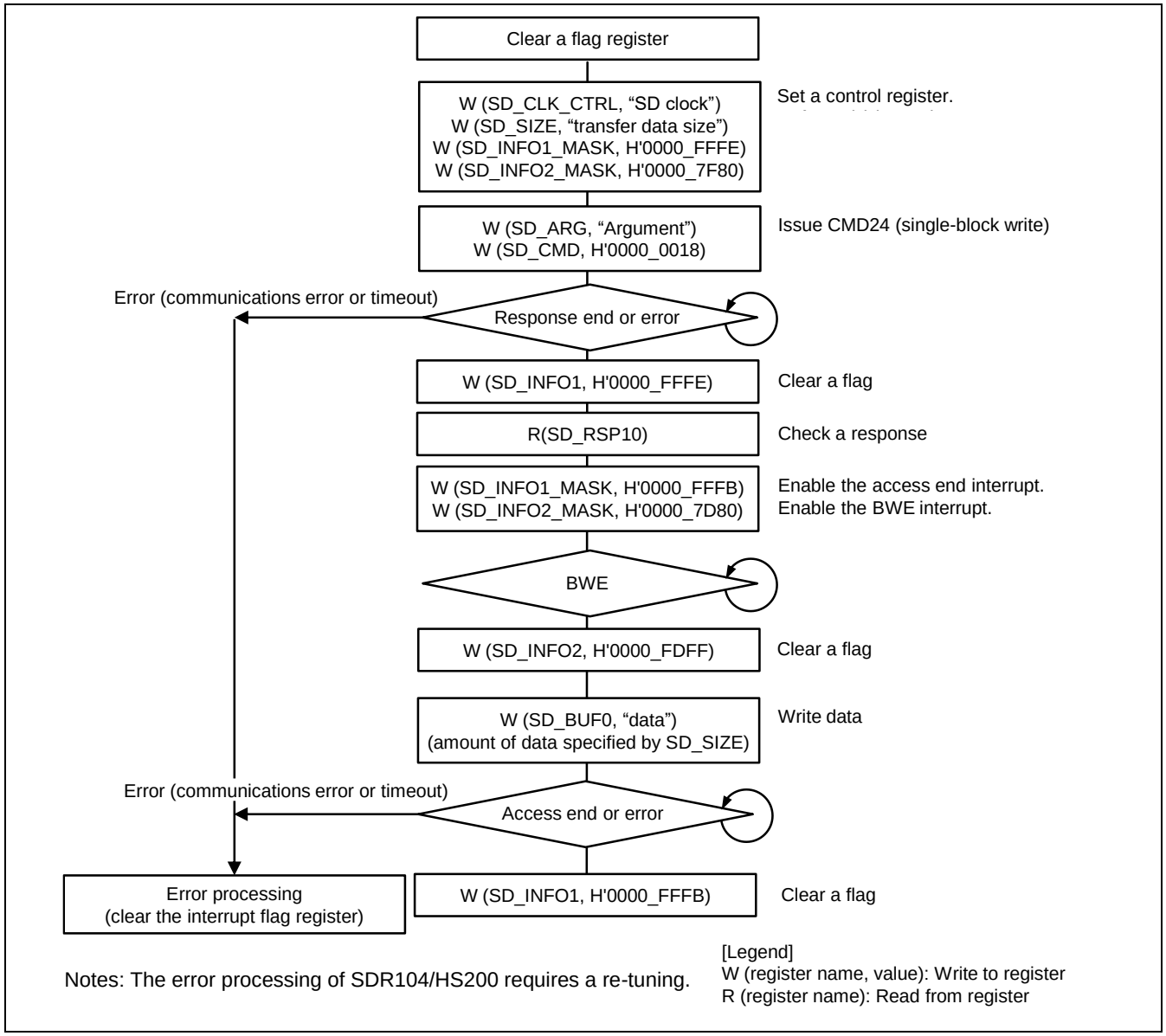



**Figure 150.11 Single Block Write Flowchart Example**

### (2) Operation for Single Block Write

The operation of the single block write is described below.

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)
2) Control register set
   Set the SD clock (SDCLK), transfer data size, interrupt mask, and so on. (SD_CLK_CTRL, SD_SIZE, SD_INFO1_MASK, and SD_INFO2_MASK)
3) Command issue (CMD24)
   Set CMD24 Argument in SD_ARG and write H'0000 0018 to SD_CMD.
   Accordingly, CMD24 is issued, and the single block write operation is started.
4) Response check
   On receiving the response, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0 and read the response from SD_RSP10.
   If the result of response decoding is an error, the command sequence can be halted by setting the STP bit in SD_STP or the IOABT bit in SDIO_MODE to 1. Furthermore, this causes CMD12 and CMD52 to not be issued. If the command sequence is halted, the INFO2 bit (access end) in SD_INFO1 is set to 1 to generate an interrupt.
5) Data write and data transmit to SD card
   Write H'0000 FFFB to SD_INFO1_MASK to enable the access end interrupt. In addition, write H'0000 7D80 to SD_INFO2_MASK to enable the BWE interrupt. When SD_BUF0 is ready for the data to be written, the BWE bit in SD_INFO2 is set to 1 to generate an interrupt. Clear the BWE bit to 0 and write the amount of data specified by SD_SIZE to SD_BUF0. When the data write to SD_BUF0 is completed, data is transmitted to the SD card. Then, the CRC status and busy state are received from the SD card.
   However, a communications error or timeout may be generated if data are being transmitted after writing to SD_BUF0.
6) Operation complete
   When the CRC status and busy state are received from the SD card, INFO2 (access end) in SD_INFO1 is set to 1 to generate an interrupt. Clear the INFO2 bit to 0 to end the single block write operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

### 150.3.6.4 Multiple Block Read

#### (1) Flowchart

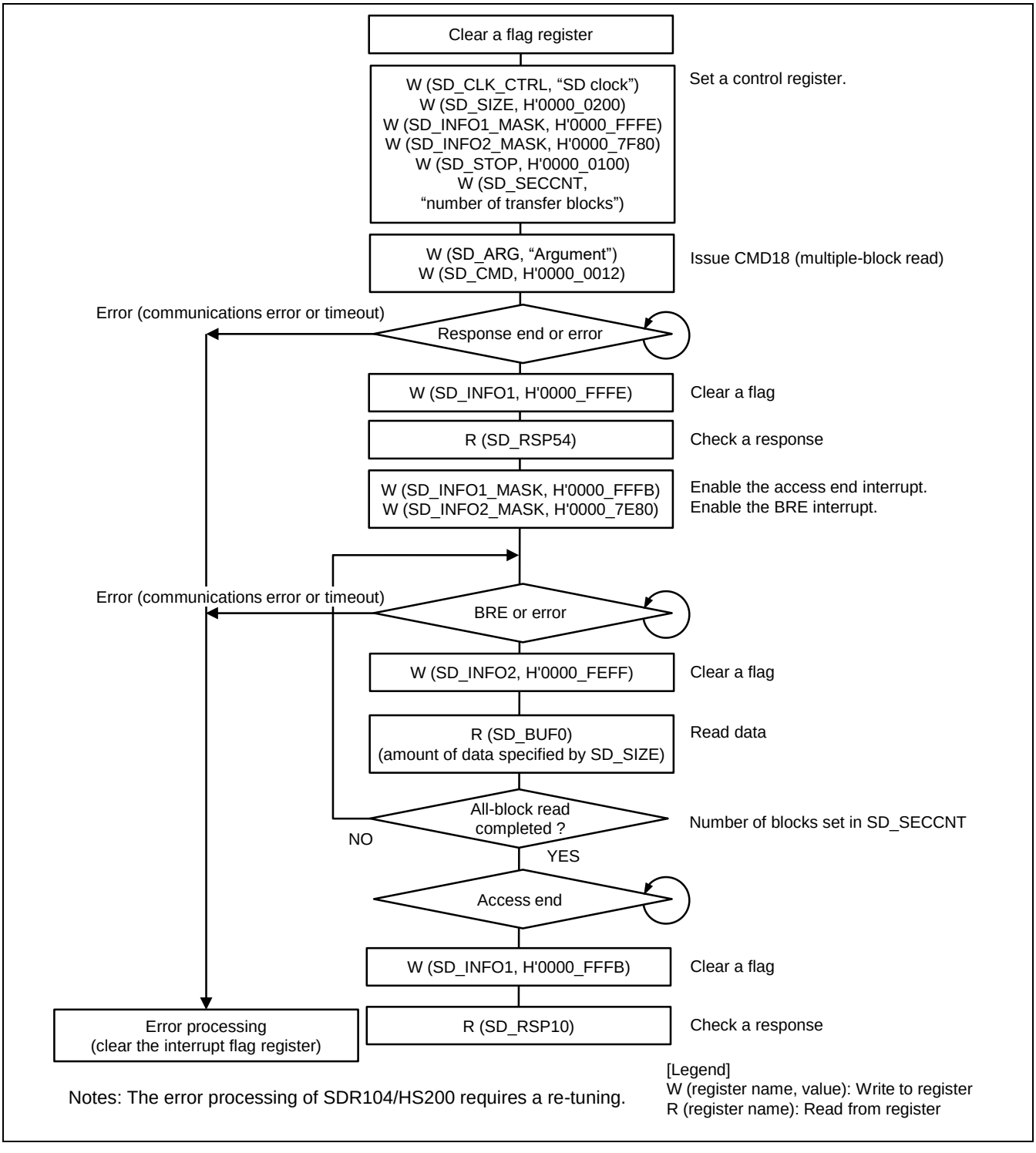



**Figure 150.12 Multiple Block Read Flowchart Example**

#### (2) Operation for Multiple Block Read

The operation of the multiple block read is described below.

1) Flag register clear

First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)

2) Control register set

   Set the SD clock (SDCLK), transfer data size, interrupt mask, and so on. (SD_CLK_CTRL, SD_SIZE, SD_INFO1_MASK, and SD_INFO2_MASK)

   Set the SEC bit in SD_STOP to 1, and set the number of transfer blocks in SD_SECCNT.

3) Command issue (CMD18)

   Set CMD18 Argument in SD_ARG and write H'0000 0012 to SD_CMD.

   Accordingly, CMD18 is issued, and the multiple block read operation is started.

4) Response check

   On receiving the response, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0 and read the response from SD_RSP54.

   If the result of response decoding is an error, the command sequence can be halted by setting the STP bit in SD_STP to 1. Setting the STP bit to 1 also causes CMD12 to be issued and the response received. If the command sequence is halted because the access end interrupt has been enabled, an interrupt will be generated by setting of the INFO2 bit (access end) in SD_INFO1 to 1 when reception of the response has been completed. Clear the INFO2 bit to 0 and read the response.

5) Data receive from SD card and data read

Write H'0000 FFFB to SD_INFO1_MASK to enable the access end interrupt. In addition, write H'0000 7E80 to SD_INFO2_MASK to enable the BRE interrupt. When one-block data receive from the SD card is completed, the BRE bit in SD_INFO2 is set to 1 to generate an interrupt. Clear the BRE bit to 0 and read the amount of data specified by SD_SIZE from SD_BUF0. Doing this repeats transfer of the number of blocks set in SD_SECCNT. However, a communications error or timeout may be generated if data are being received while reading of SD_BUF0 is in progress. CMD12 is automatically issued to stop multi-block transfer with the number of blocks which is set to SD_SECCNT and the response is received. At this point, CMD12 Argument is automatically set to H'0000 000.

6) Operation complete

   When all-block data read and the CMD12 response receive are completed, INFO2 (access end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO2 to 0 to read the response. This is the end of multiple block read operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

### 150.3.6.5 Multiple Block Write (when Using Internal Timer)

#### (1) Flowchart

Figure 150.13 shows the flowchart when using an internal timer.

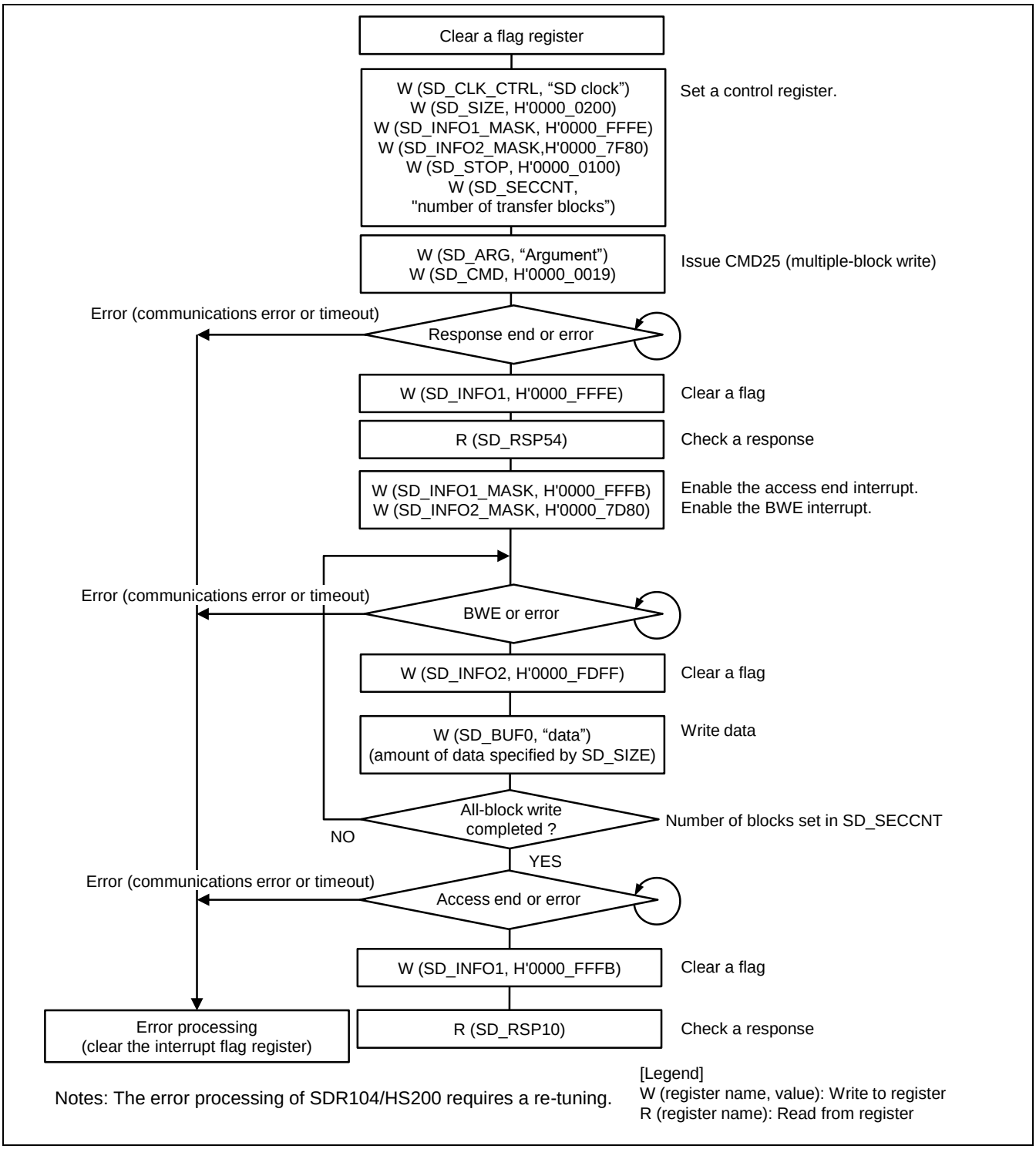



**Figure 150.13 Multiple Block Write Flowchart Example (when Using Internal Timer)**

### (2) Operation for Multiple Block Write

The operation of the multiple block write is described below.

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)
2) Control register set
   Set the SD clock (SDCLK), transfer data size, interrupt mask, and so on. (SD_CLK_CTRL, SD_SIZE, SD_INFO1_MASK, and SD_INFO2_MASK)
   Set the SEC bit in SD_STOP to 1, and set the number of transfer blocks in SD_SECCNT.
3) Command issue (CMD25)
   Set CMD25 Argument in SD_ARG and write H'0000 0019 to SD_CMD.
   Accordingly, CMD25 is issued, and the multiple block write operation is started.
4) Response check
   On receiving the response, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0 and read the response from SD_RSP54.
   If the result of response decoding is an error, the command sequence can be halted by setting the STP bit in SD_STP to 1. Setting the STP bit to 1 also causes CMD12 to be issued and the response received. If the command sequence is halted because the access end interrupt has been enabled, an interrupt will be generated by setting of the INFO2 bit (access end) bit in SD_INFO1 to 1 when reception of the response has been completed. Clear the INFO2 bit to 0 and read the response.
5) Data write and data transmit to SD card
   Write H'0000 FFFB to SD_INFO1_MASK to enable the access end interrupt. In addition, write H'0000 7D80 to SD_INFO2_MASK to enable the BWE interrupt. When SD_BUF0 is ready for the data to be written, the BWE bit in SD_INFO2 is set to 1 to generate an interrupt. Clear the BWE bit to 0 and write the amount of data specified by SD_SIZE to SD_BUF0. When the data write to SD_BUF0 is completed, data is transmitted to the SD card. Then, the CRC status and busy state are received from the SD card. Doing this repeats transfer of the number of blocks set in SD_SECCNT.

However, a communications error or timeout may be generated if data are being transmitted while writing to SD_BUF0 is in progress. CMD12 is automatically issued to stop multi-block transfer with the number of blocks which is set to SD_SECCNT and the response is received. At this point, CMD12 Argument is automatically set to H'0000 0000.

6) Operation complete
   When all-block data transmit and the CRC status receive are completed, the INFO2 bit (access end) in SD_INFO1 is set to 1 to generate an interrupt. Clear the INFO2 bit to 0 to read the response. This is the end of multiple block write operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

### 150.3.6.6 Multiple Block Write (when Using External Timer)

#### (1) Flowchart

The flowchart when using an external timer instead of an internal timer of IP is shown below.

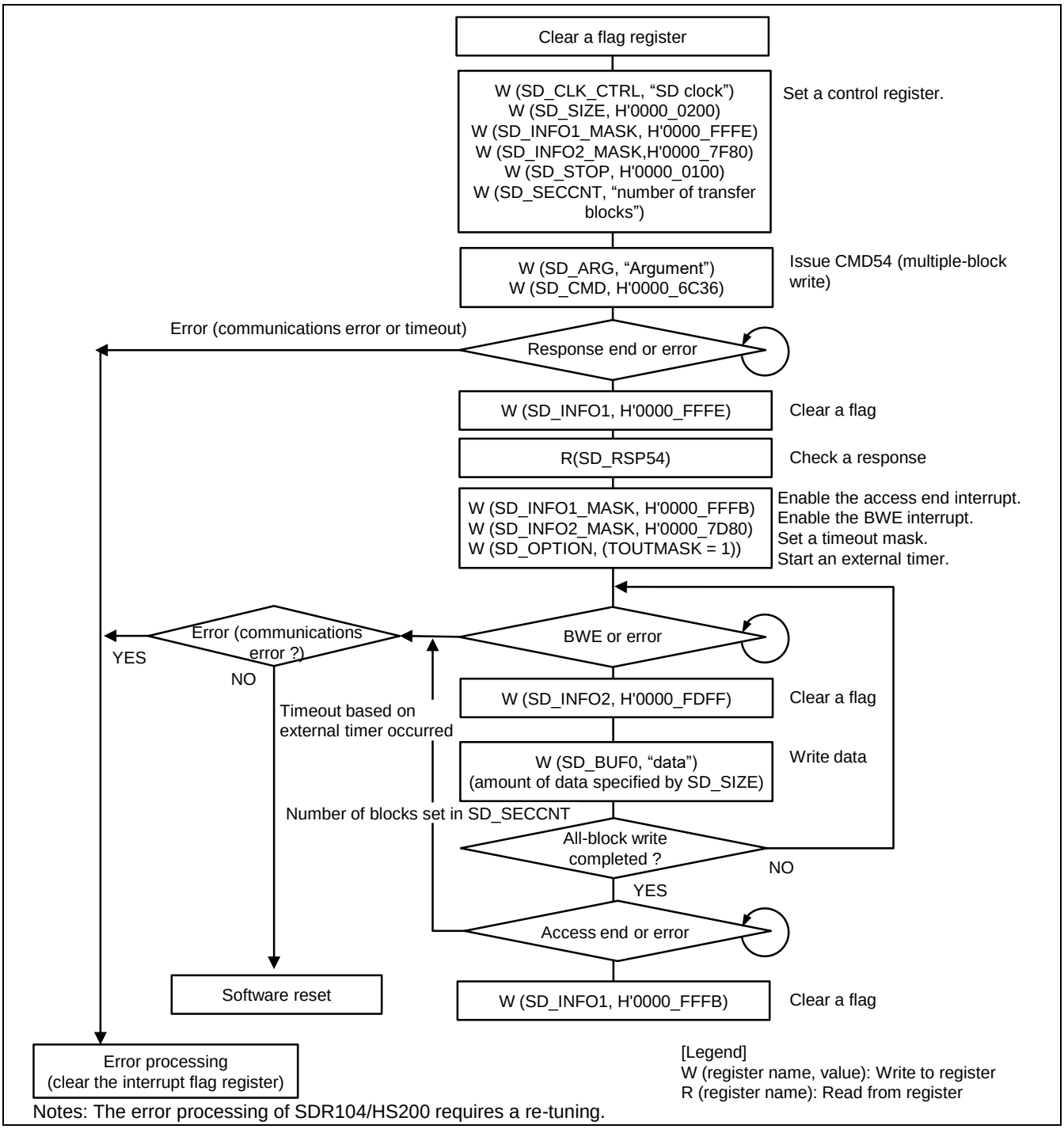



**Figure 150.14 Multiple Block Write Flowchart Example (when Using External Timer)**

#### (2) Operation for Multiple Block Write

The operation of the multiple block write is described below.

1) Flag register clear

   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)

2) Control register set

   Set the SD clock (SDCLK), transfer data size, interrupt mask, and so on. (SD_CLK_CTRL, SD_SIZE, SD_INFO1_MASK, and SD_INFO2_MASK)

   Set the SEC bit in SD_STOP to 1, and set the number of transfer blocks in SD_SECCNT.

3) Command issue (CMD54)

   Set CMD54 Argument in SD_ARG and write H'0000 6C36 to SD_CMD.

   Accordingly, CMD54 is issued, and the multiple block write operation is started.

4) Response check

   On receiving the response, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0 and read the response from SD_RSP54.

   If the result of response decoding is an error, the command sequence can be halted by setting the STP bit in SD_STP to 1. Setting the STP bit to 1 also causes CMD12 to be issued and the response received. If the command sequence is halted because the access end interrupt has been enabled, an interrupt will be generated by setting of the INFO2 bit (access end) bit in SD_INFO1 to 1 when reception of the response has been completed. Clear the INFO2 bit to 0 and read the response.

5) Data write and data transmit to SD card

   Write H'0000 FFFB to SD_INFO1_MASK to enable the access end interrupt. In addition, write H'0000 7D80 to SD_INFO2_MASK to enable the BWE interrupt. Set the TOUTMASK bit in SD_OPTION to disable timeout and start an external timer.

   When SD_BUF0 is ready for the data to be written, the BWE bit in the SD_INFO2 resister is set to 1 to generate an interrupt. Clear the BWE bit to 0 and write the amount of data specified by SD_SIZE to SD_BUF0. When the data write to SD_BUF0 is completed, data is transmitted to the SD card. Then, the CRC status and busy state are received from the SD card. Doing this repeats transfer of the number of blocks set in SD_SECCNT. However, a communications error may be generated if data are being transmitted while writing to SD_BUF0 is in progress.

6) Operation complete

   When all-block data transmit and the CRC status receive are completed, the INFO2 bit (access end) in SD_INFO1 is set to 1 to generate an interrupt. Clear the INFO2 bit to 0 to read the response. This is the end of multiple block write operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs at response reception (a communications error or timeout) or at data transmission. Perform a software reset if a timeout occurs at data transmission based on an external timer.

### 150.3.6.7 IO_RW_DIRECT Commend (CMD52)

#### (1) Flowchart

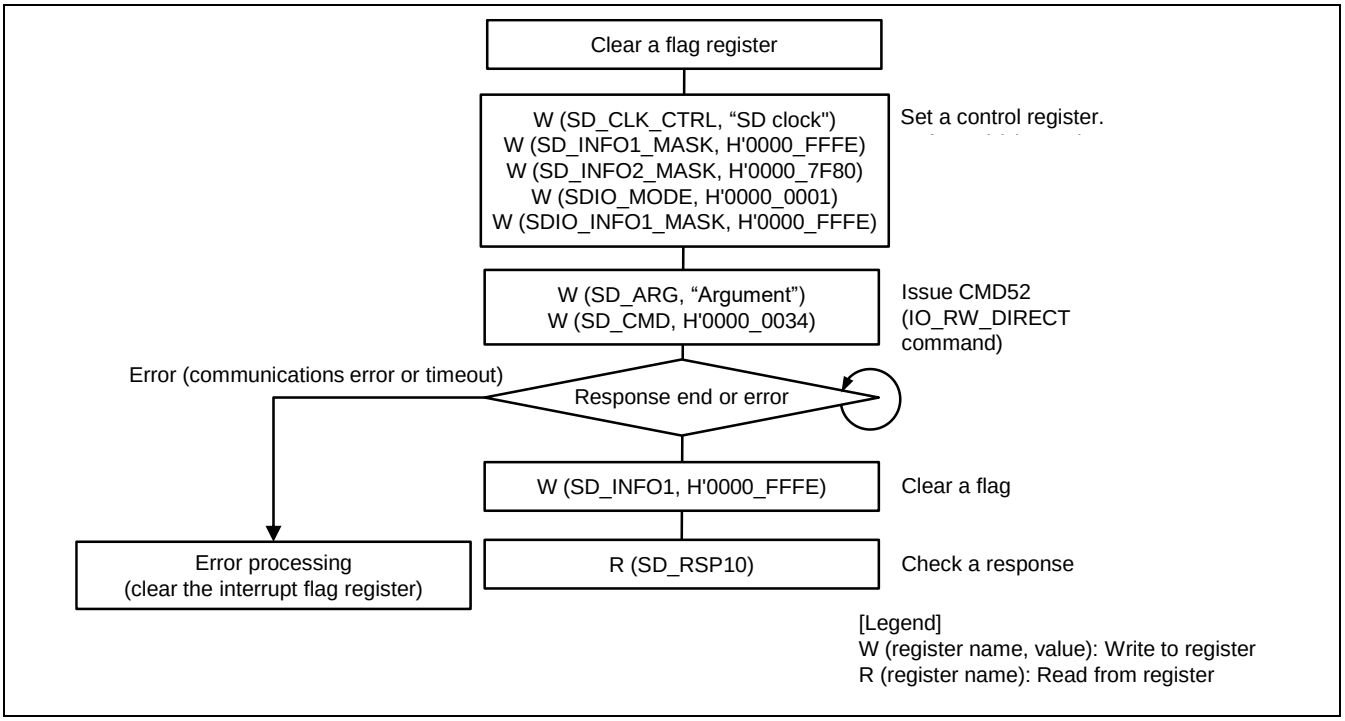



**Figure 150.15 IO_RW_DIRECT Command (CMD52) Flowchart Example**

### 150.3.6.8 IO_RW_EXTENDED (CMD53/Multiple Block Read)

#### (1) Flowchart

Figure 150.16 shows a flowchart example for CMD53 (multiple block read).

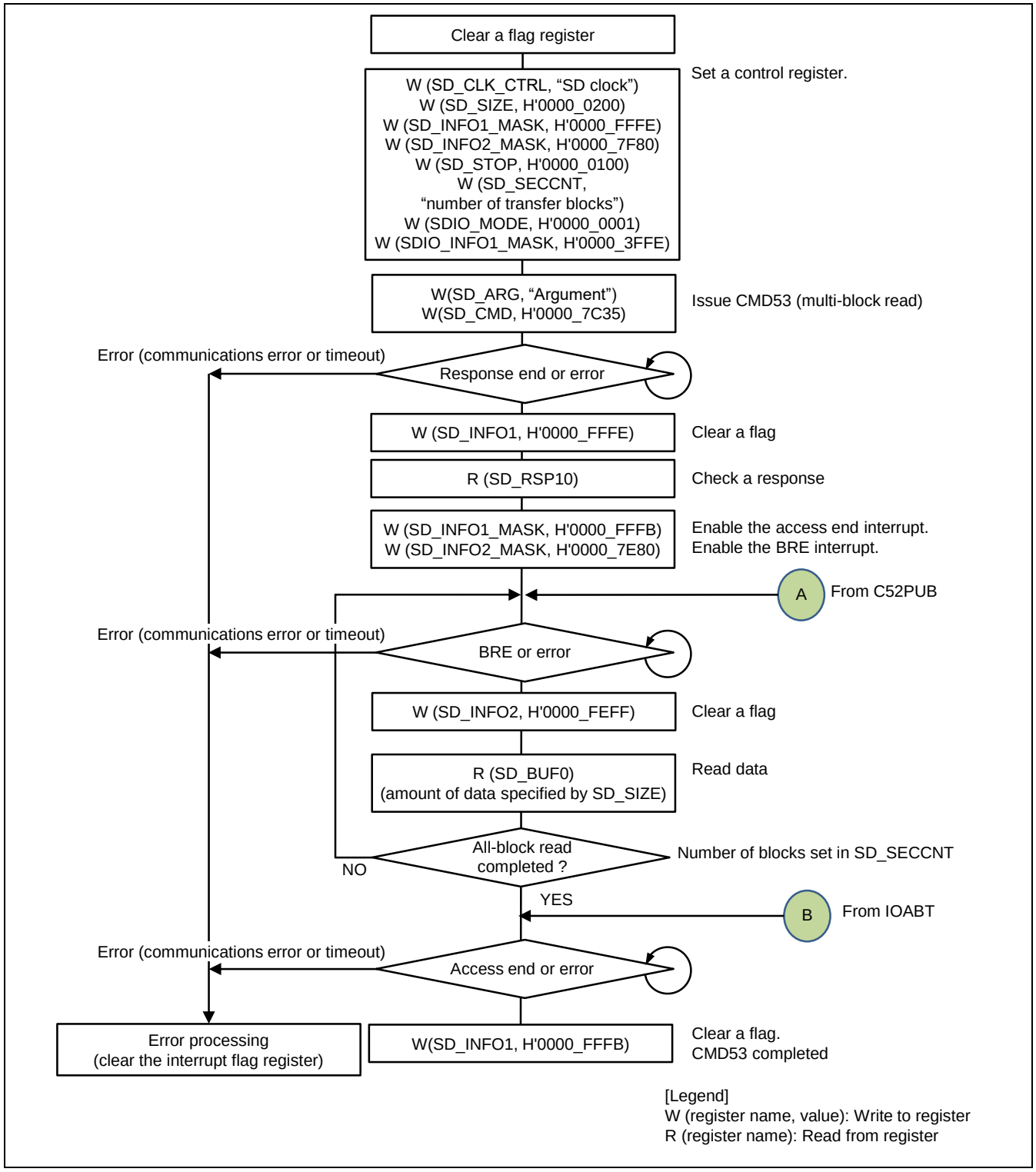



**Figure 150.16 CMD53 (Multiple Block Read) Flowchart Example**

Figure 150.17 shows a flowchart example when CMD52 (SDIO abort) is issued at CMD53 (multiple block read).

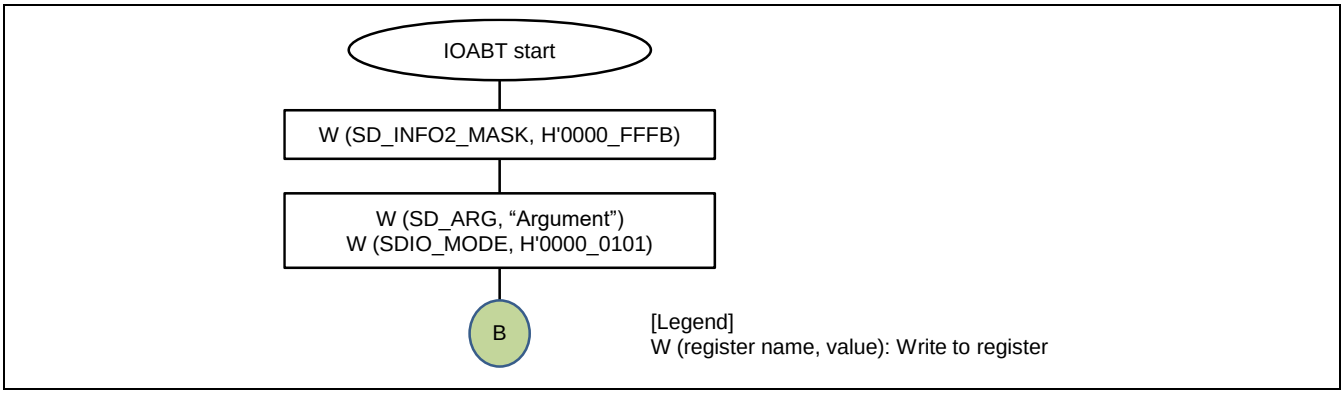



**Figure 150.17 Flowchart Example when CMD52 (SDIO Abort) is Issued at CMD53 (Multiple Block Read)**

Figure 150.18 shows a flowchart example when CMD52 (SDIO none abort) is issued at CMD53 (multiple block read) while the SD host interface is in the read wait state.

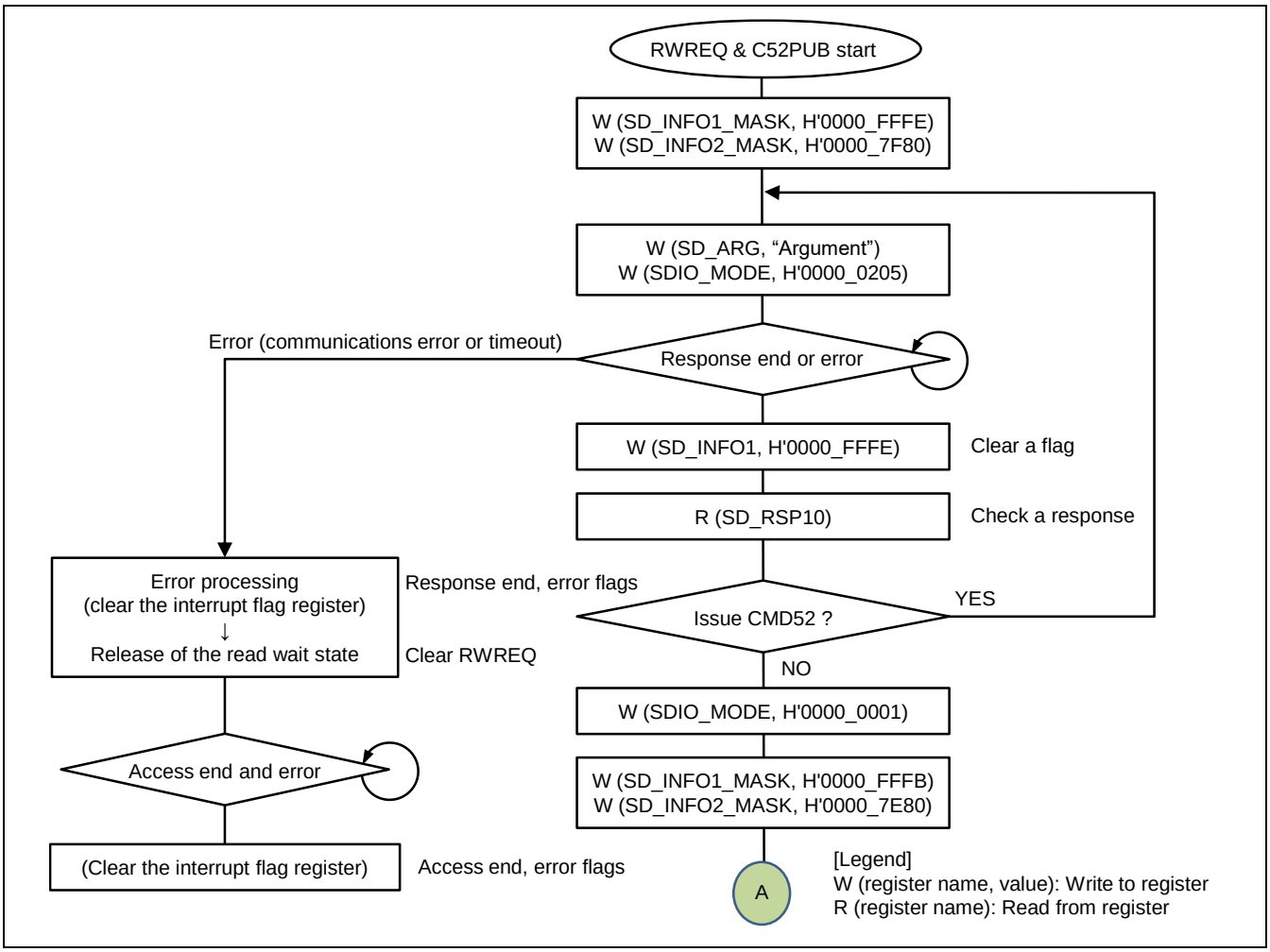



**Figure 150.18 Flowchart Example when CMD52 (SDIO None Abort) is Issued after Read Wait State is Entered at CMD53 (Multi Block Read)**

### 150.3.6.9 IO_RW_EXTENDED (CMD53/Multiple Block Write)

### (1) Flowchart

Figure 150.19 shows a flowchart example for CMD53 (multiple block write).

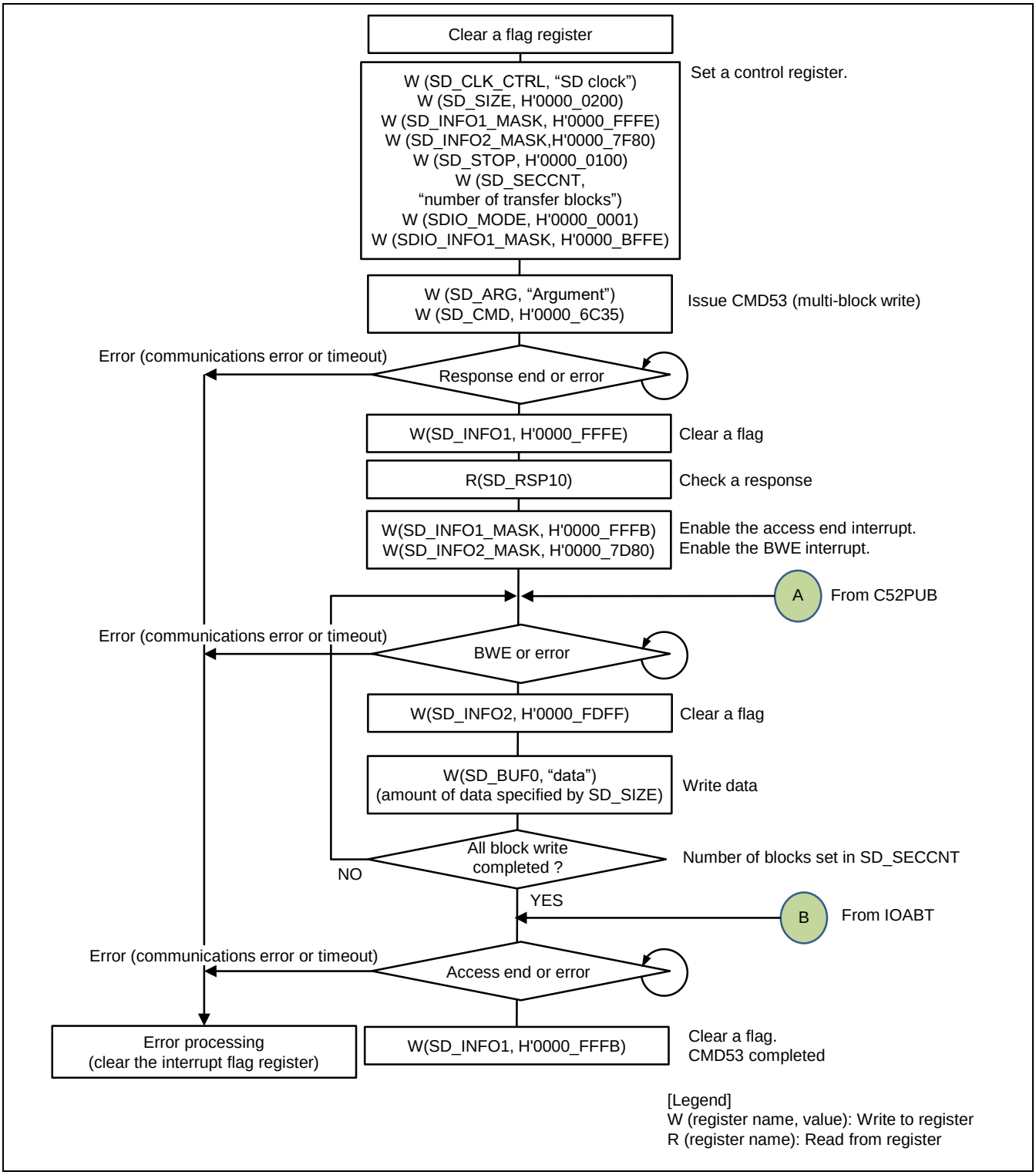



**Figure 150.19 CMD53 (Multiple Block Write) Flowchart Example**

Figure 150.20 shows a flowchart example when CMD52 (SDIO abort) is issued at CMD53 (multiple block write).

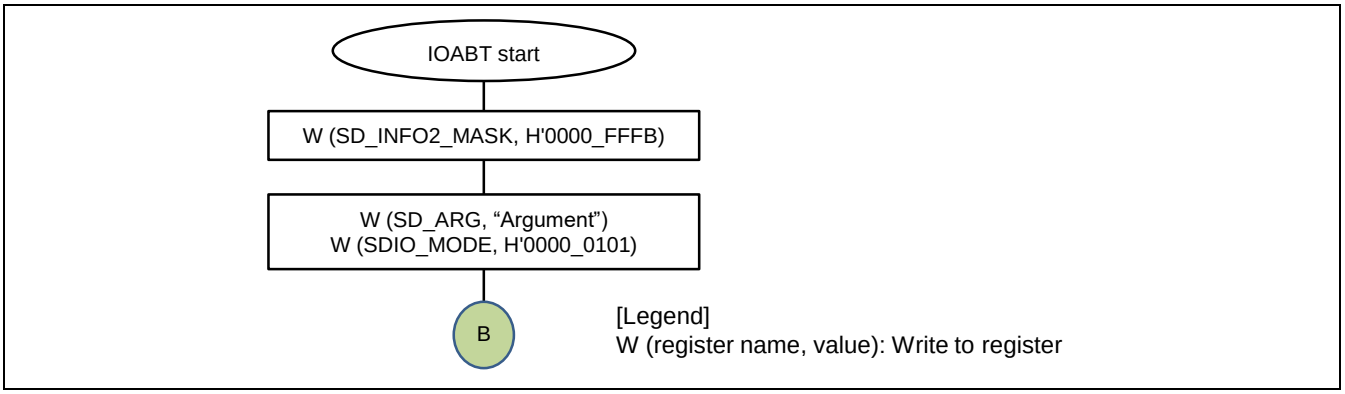



**Figure 150.20 Flowchart Example when CMD52 (SDIO Abort) is Issued at CMD53 (Multiple Block Write)**

Figure 150.21 shows a flowchart example when CMD52 (SDIO none abort) is issued at CMD53 (multiple block write).

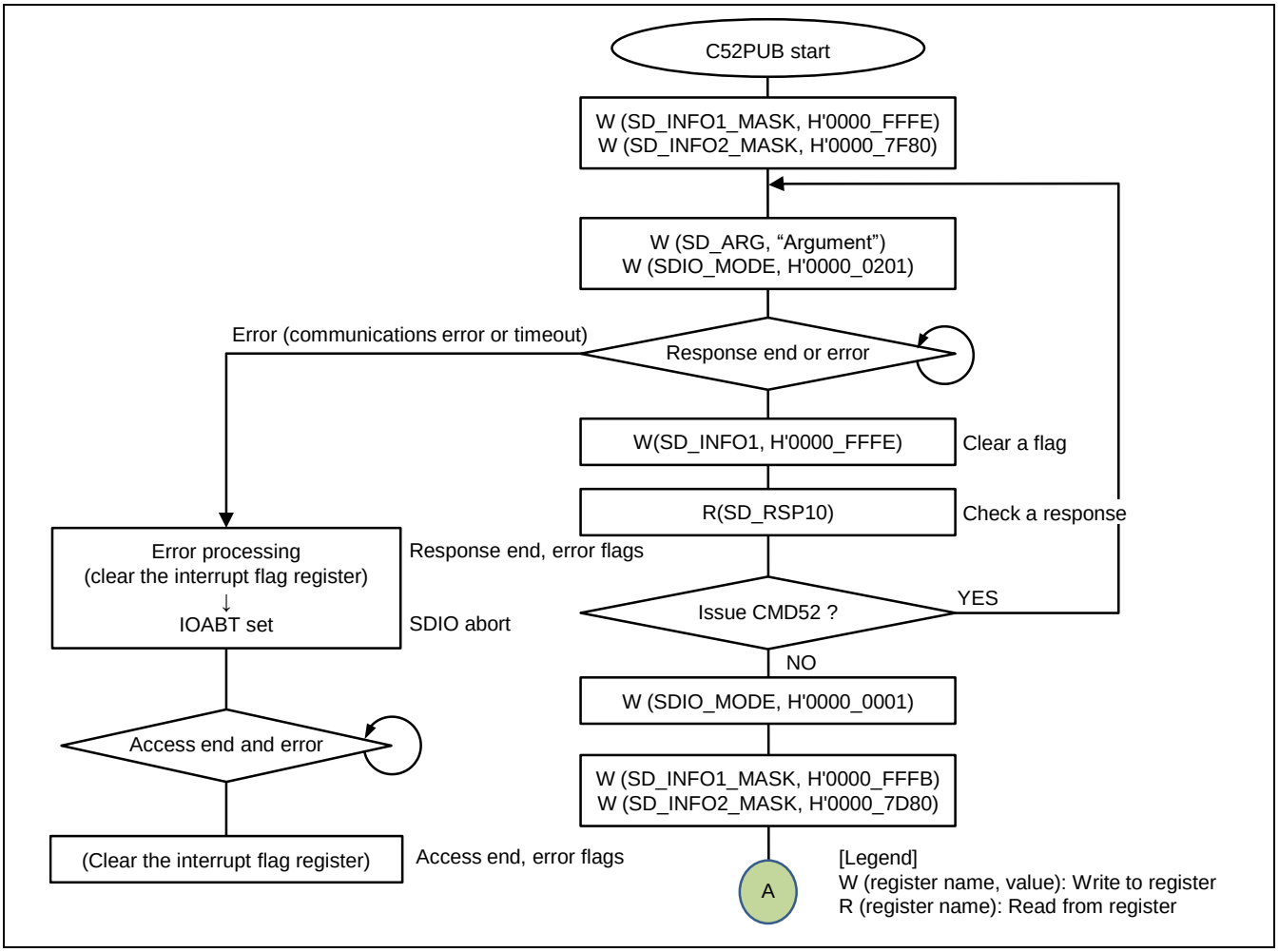



**Figure 150.21 Flowchart Example when CMD52 (SDIO None Abort) is Issued at CMD53 (Multiple Block Write)**

### 150.3.6.10 DMA Transfer

#### (1) SD_BUF DMA Transfer

Figure 150.22 shows a flowchart example for SD_BUF DMA read when CMD18 (multiple block read) is issued.

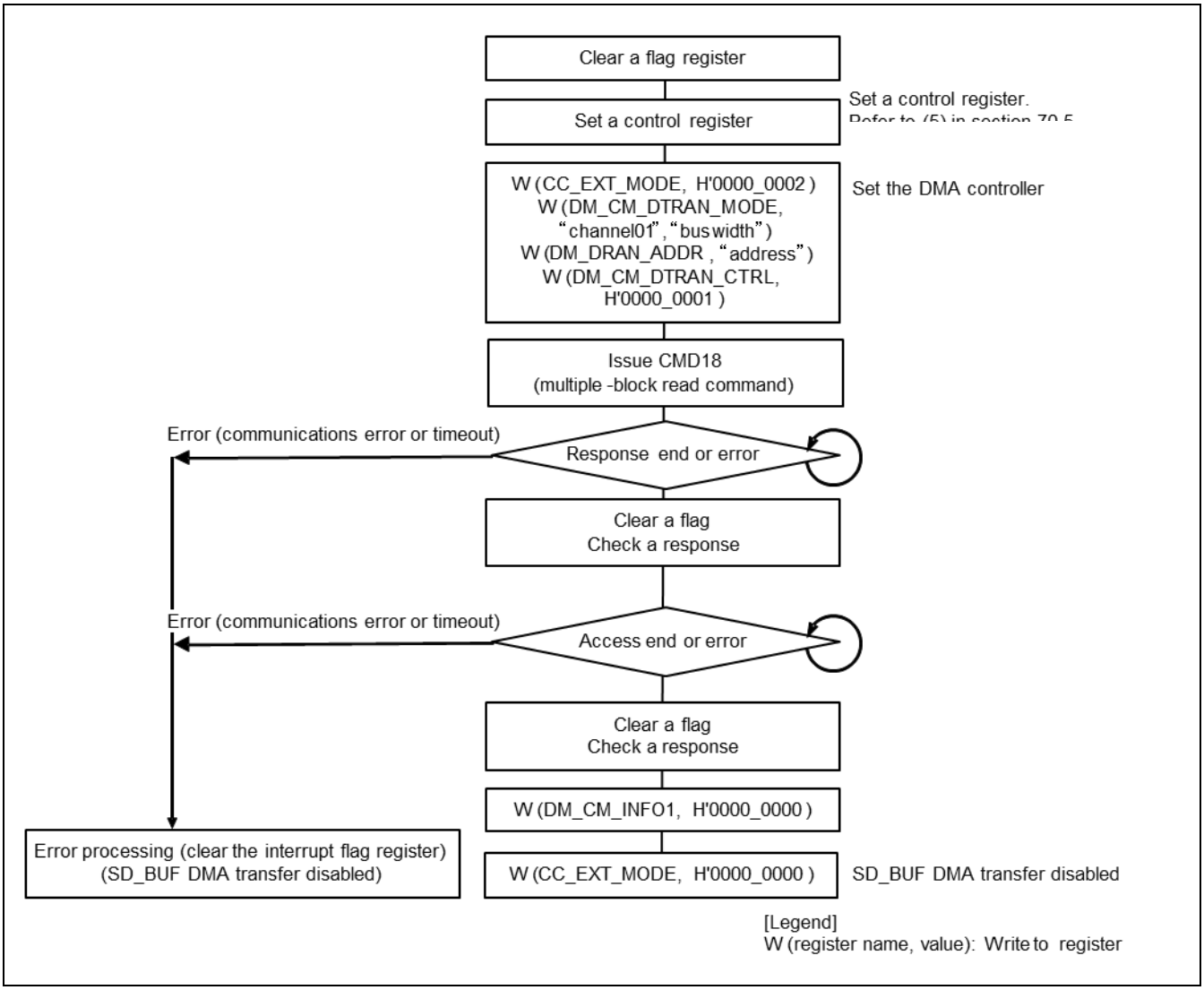



**Figure 150.22 SD_BUF DMA Read Flowchart Example**

Figure 150.23 shows a flowchart example for SD_BUF DMA write when CMD25 (multiple block write) is issued.

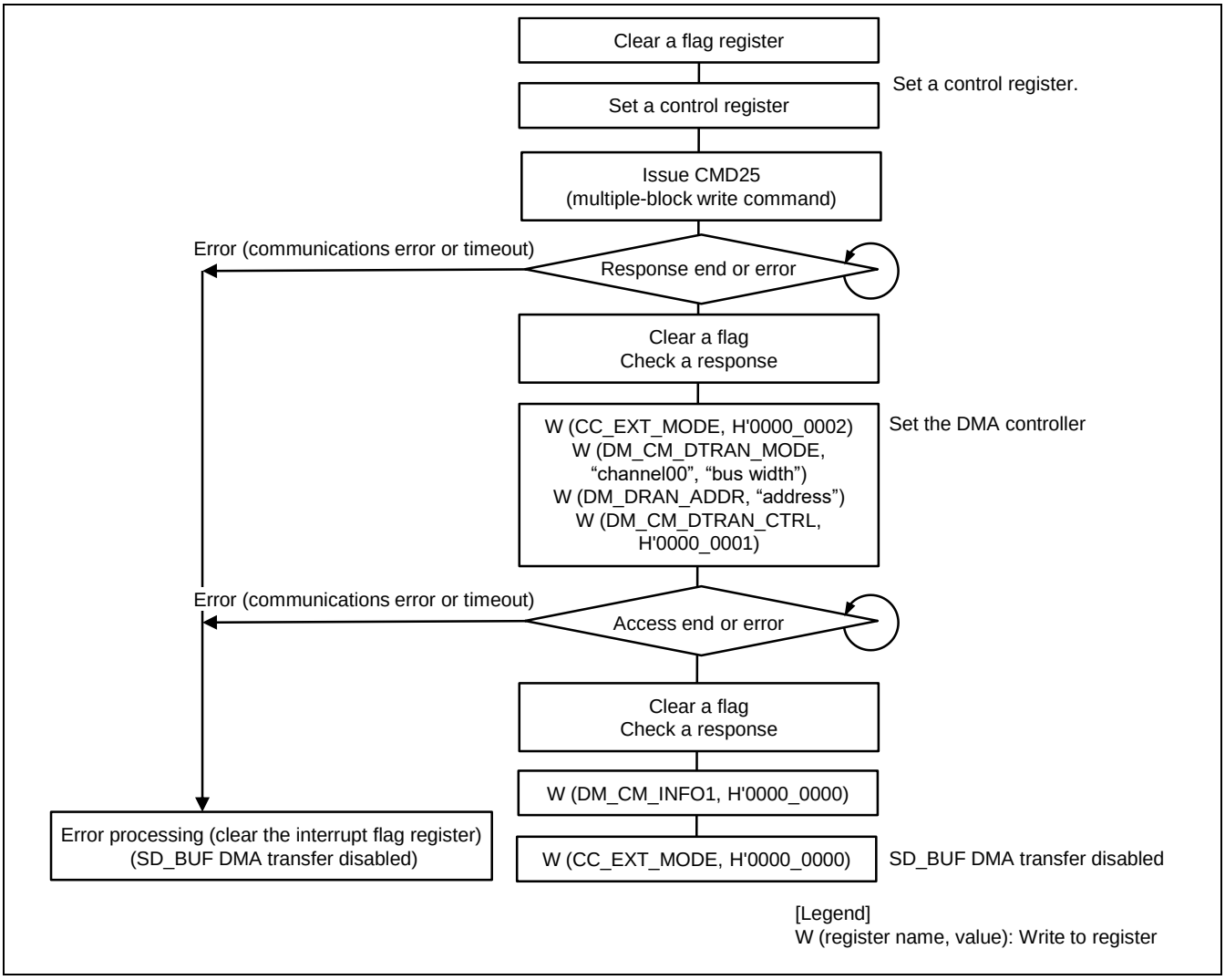



**Figure 150.23 SD_BUF DMA Write Flowchart Example**

### 150.3.6.11 Alternative Boot Operation

**(1) Flowchart**

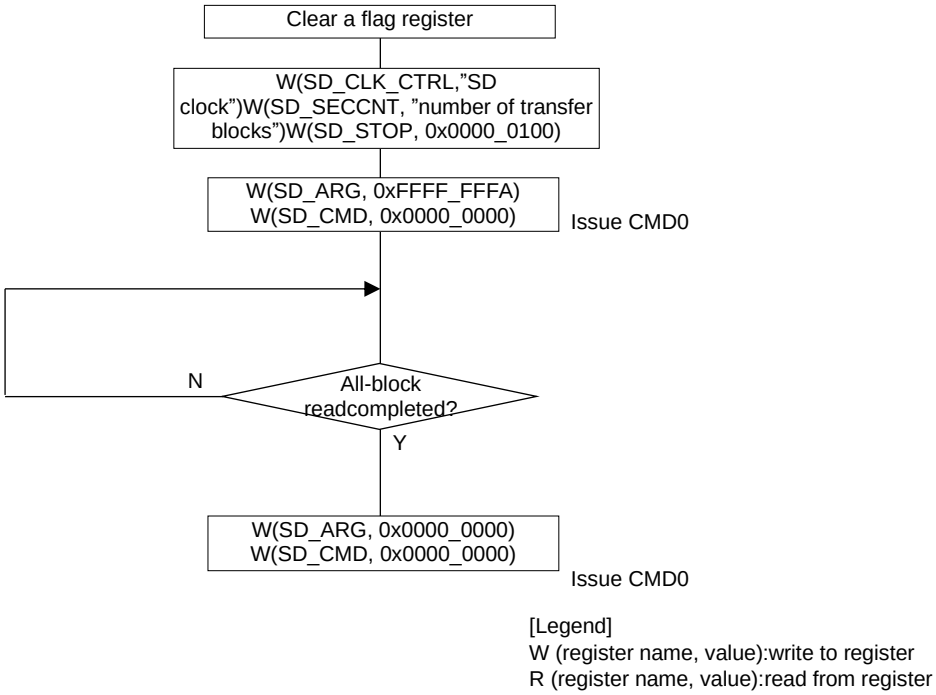



**Figure 150.24 Alternative Boot Operation Flowchart Example**

### 150.3.6.12 High-Priority Interrupt (without Data Transfer)

#### (1) Flowchart

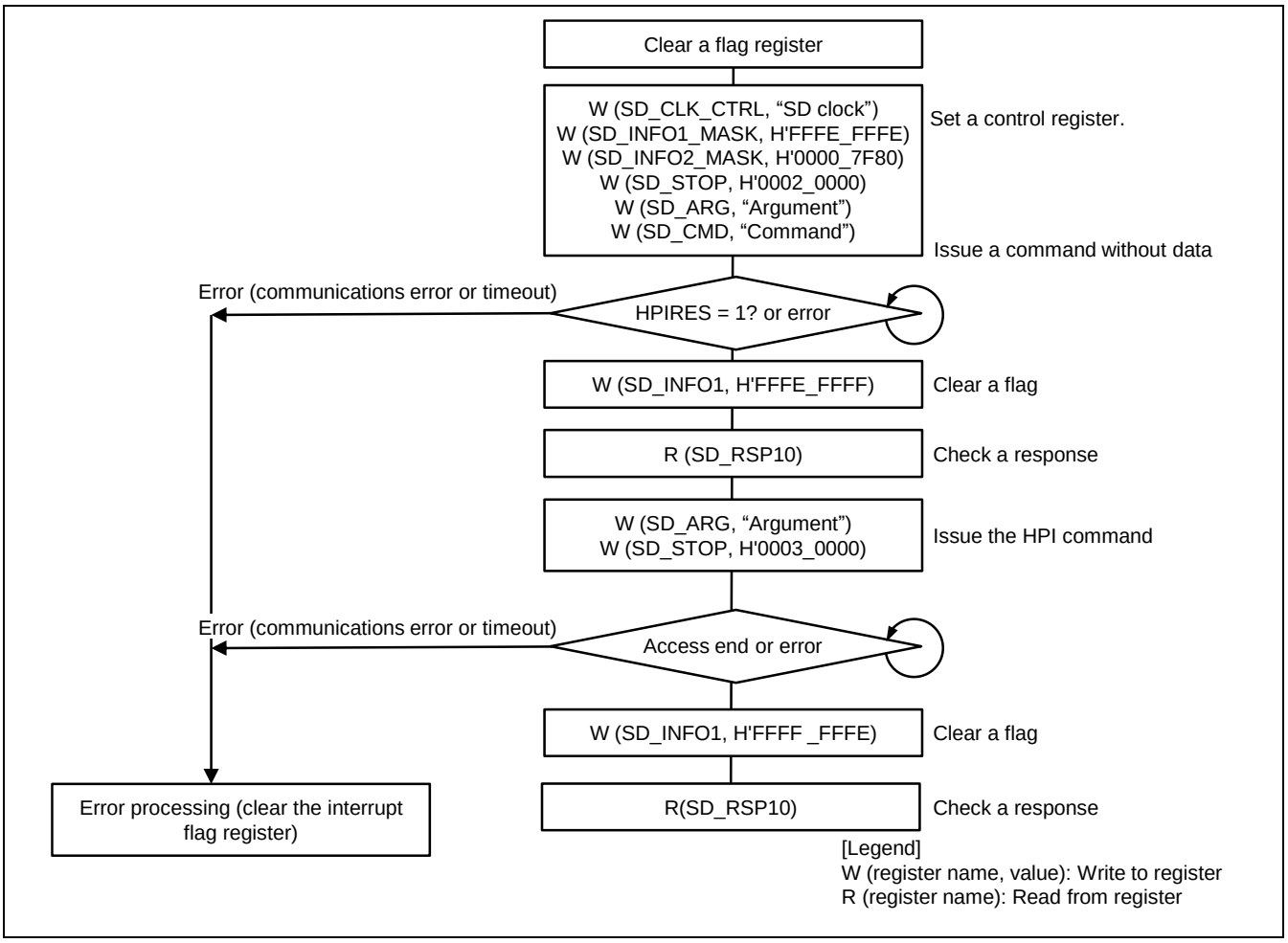



**Figure 150.25 Example of the High-Priority Interrupt (without Data Transfer) Flowchart**

### (2) Operation for High-Priority Interrupt without Data Transfer

The operation of the high-priority interrupt (HPI) without data transfer is described below.

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)
2) Control register set
   Set the SD clock (SDCLK), HPI enable, interrupt mask, and so on. (SD_CLK_CTRL, SD_INFO1_MASK, and SD_INFO2_MASK)
3) Command issue
   Set the CMD Argument in SD_ARG and write to SD_CMD.
   Accordingly, CMD is issued, and the operation is started.
4) Flag clear
   On receiving the response, the HPIRES bit in SD_INFO1 is set to 1 to generate an interrupt. Clear the HPIRES bit to 0.
5) Read the response from SD_RSP10.
6) HPI command issue
   Set the HPI command argument in SD_ARG and set the HPIMODE and HPICMD bits in SD_STOP to 1.
7) Operation complete
   When reception of the response to the HPI command is completed and the busy state is released, the INFO0 bit in SD_INFO1 is set to 1 to generate an interrupt. Clear the INFO0 bit to 0 to read the response from SD_RSP10. This is the end of HPI operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

### 150.3.6.13 High-Priority Interrupt (at Single Block Write)

#### (1) Flowchart

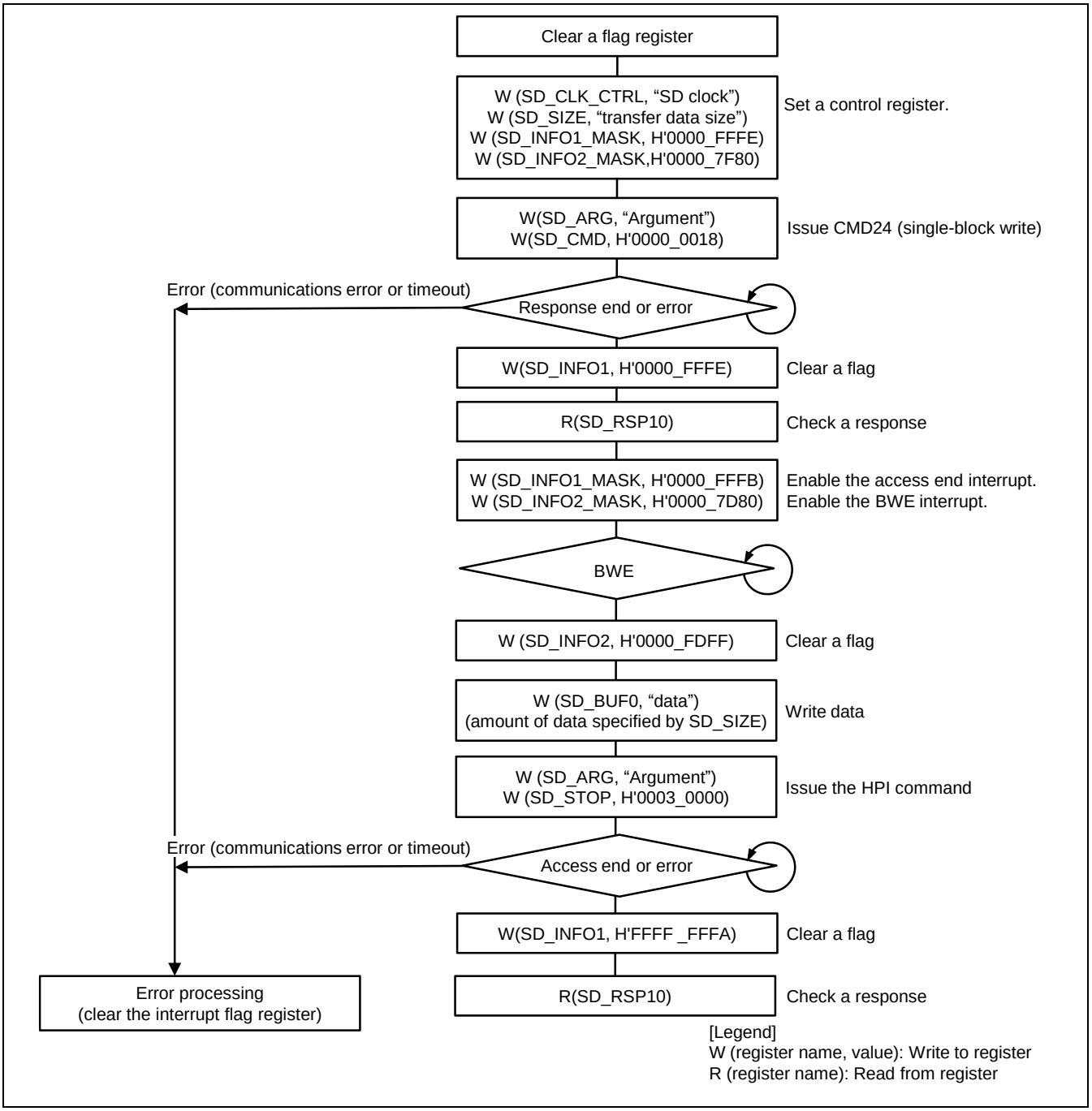



**Figure 150.26 Example of the High-Priority Interrupt (at Single Block Write) Flowchart**

### (2) Operation for HPI at Single Block Write

The operation of HPI at single block write is described below.

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)
2) Control register set
   Set the SD clock (SDCLK), transfer data size, interrupt mask, and so on. (SD_CLK_CTRL, SD_SIZE, SD_INFO1_MASK, and SD_INFO2_MASK)
3) Command issue (CMD24)
   Set CMD24 Argument in SD_ARG and write H'0000 0018 to SD_CMD.
   Accordingly, CMD24 is issued, and the single block write operation is started.
4) Response check
   On receiving the response, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0 and read the response from SD_RSP10.
   If the result of response decoding is an error, the command sequence can be halted by setting the STP bit in SD_STP or the IOABT bit in SDIO_MODE to 1. Furthermore, this causes CMD12 and CMD52 to not be issued. If the command sequence is halted, the INFO2 bit (access end) in SD_INFO1 is set to 1 to generate an interrupt.
5) Data write and data transmit to SD card

Write H'0000 FFFB to SD_INFO1_MASK to enable the access end interrupt. In addition, write H'0000 7D80 to SD_INFO2_MASK to enable the BWE interrupt. When SD_BUF0 is ready for the data to be written, the BWE bit in SD_INFO2 is set to 1 to generate an interrupt. Clear the BWE bit to 0 and write the amount of data specified by SD_SIZE to SD_BUF0. When the data write to SD_BUF0 is completed, data is transmitted to the SD card.

6) HPI command issue
   Set the HPI command argument in SD_ARG and set the HPIMODE and HPICMD bits in SD_STOP to 1.
7) Operation complete
   When reception of the response to the HPI command is completed and the busy state is released, the INFO2 and INFO0 bits in SD_INFO1 are set to 1 to generate an interrupt. Clear the INFO2 and INFO0 bits to 0 to read the response from SD_RSP10. This is the end of HPI operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

### 150.3.6.14 High-Priority Interrupt (at Multiple Block Write)

#### (1) Flowchart

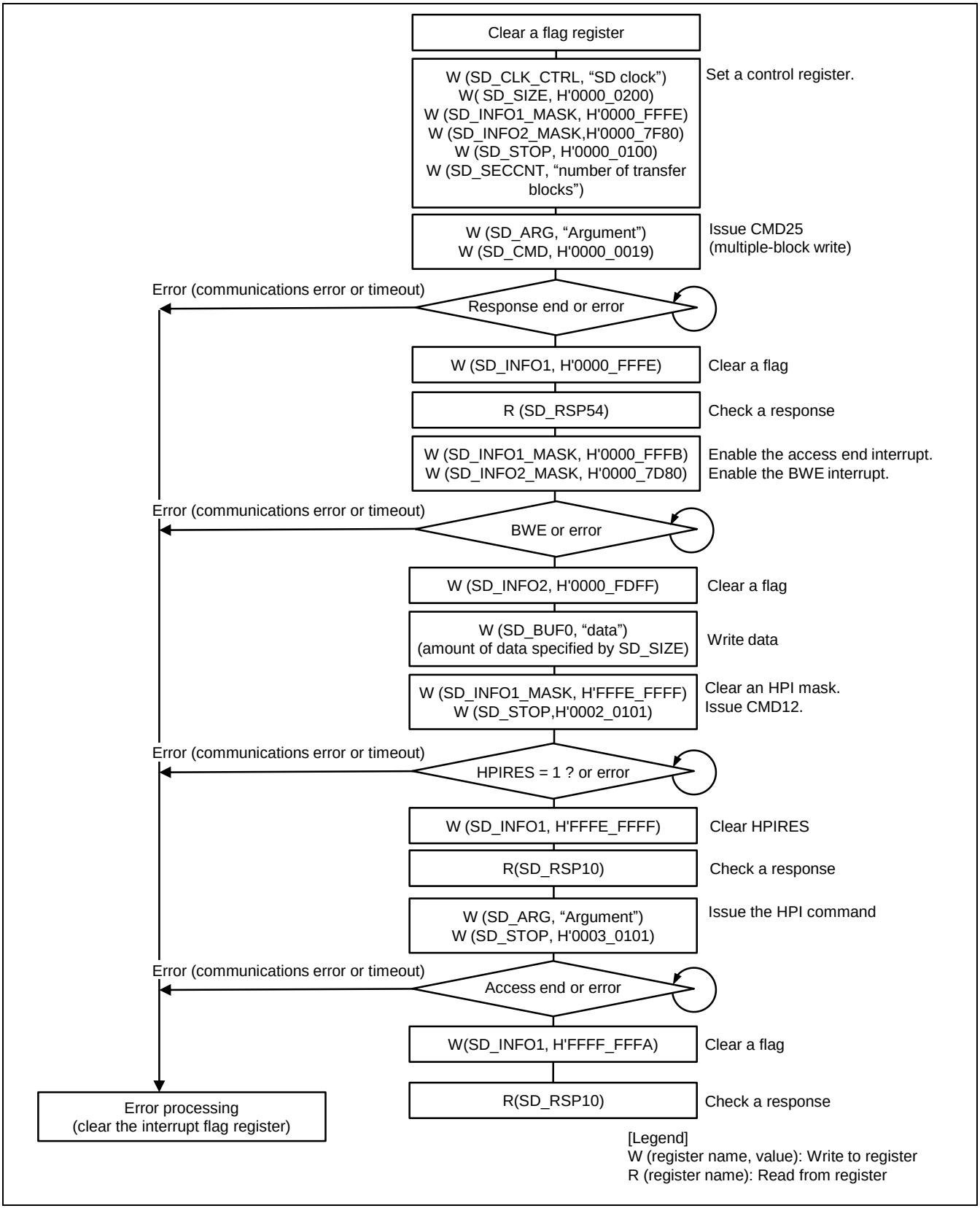



**Figure 150.27 Example of the High-Priority Interrupt (at Multiple Block Write) Flowchart (a)**

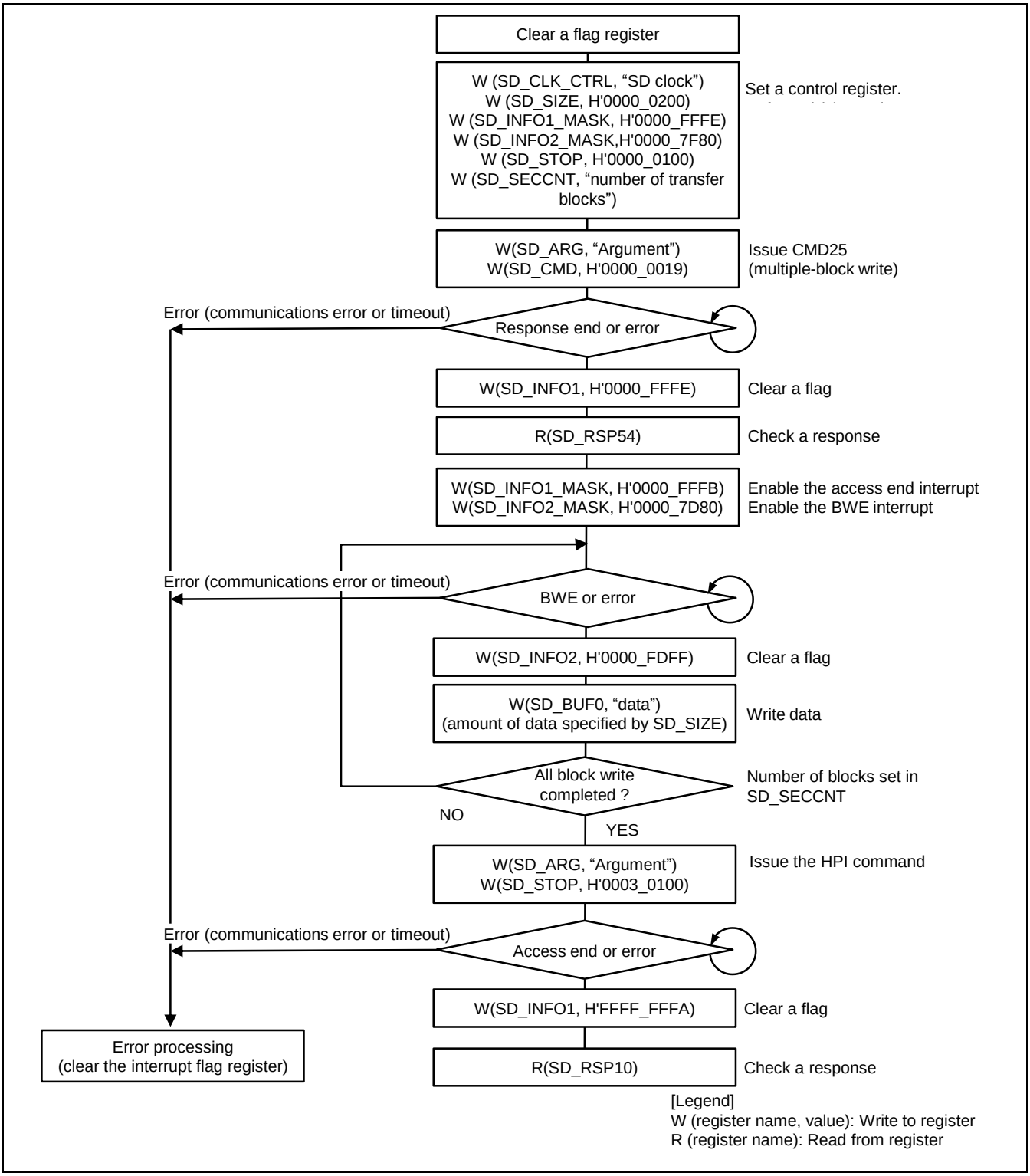



**Figure 150.28 Example of the High-Priority Interrupt (at Multiple Block Write) Flowchart (b)**

### (2) Operation for HPI at Multiple Block Transfer

The operation of HPI at the multiple block write is described below.

#### (a) When not all the data has been written to SD_BUF

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)
2) Control register set
   Set the SD clock (SDCLK), transfer data size, interrupt mask, and so on. (SD_CLK_CTRL, SD_SIZE, SD_INFO1_MASK, and SD_INFO2_MASK)
   Set the SEC bit in SD_STOP to 1, and set the number of transfer blocks in SD_SECCNT.
3) Command issue (CMD25)
   Set CMD25 Argument in SD_ARG and write H'0000 0019 to SD_CMD.
   Accordingly, CMD25 is issued, and the multiple block write operation is started.
4) Response check
   On receiving the response, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0 and read the response from SD_RSP54.
   If the result of response decoding is an error, the command sequence can be halted by setting the STP bit in SD_STP to 1. Setting the STP bit to 1 also causes CMD12 to be issued and the response received. If the command sequence is halted because the access end interrupt has been enabled, an interrupt will be generated by setting of the INFO2 bit (access end) bit in SD_INFO1 to 1 when reception of the response has been completed. Clear the INFO2 bit to 0 and read the response.
5) Data write and data transmit to SD card
   Write H'0000 FFFB to SD_INFO1_MASK to enable the access end interrupt. In addition, write H'0000 7D80 to SD_INFO2_MASK to enable the BWE interrupt. When SD_BUF0 is ready for the data to be written, the BWE bit in SD_INFO2 is set to 1 to generate an interrupt. Clear the BWE bit to 0 and write the amount of data specified by SD_SIZE to SD_BUF0. When the data write to SD_BUF0 is completed, data is transmitted to the SD card.
   Then, the CRC status and busy state are received from the SD card.
   However, a communications error or timeout may be generated if data are being transmitted while writing to SD_BUF0 is in progress.
6) Command issue (CMD12)
   Write H'FFFE FFFF to SD_INFO1_MASK to enable the HPIRES interrupt.

Write H'0002 0101 to SD_STOP, which causes CMD12 to be issued.

7) Response check
   When the response is received, the HPIRES bit in SD_INFO1 is set to 1 to generate an interrupt. Clear the HPIRES bit to 0 to read the response from SD_RSP10.
8) HPI command issue (CMD25)
   Set the HPI command argument in SD_ARG and set the HPIMODE and HPICMD bits in SD_STOP to 1.
9) Operation complete
   When reception of the response to the HPI command is completed and the busy state is released, the INFO2 and INFO0 bits in SD_INFO1 are set to 1 to generate an interrupt. Clear the INFO2 and INFO0 bits to 0 to read the response from SD_RSP10. This is the end of HPI operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

#### (b) When all the data has been written to SD_BUF

1) Flag register clear
   First, clear the bits in the flag register. (SD_INFO1 and SD_INFO2)

2) Control register set

   Set the SD clock (SDCLK), transfer data size, interrupt mask, and so on. (SD_CLK_CTRL, SD_SIZE, SD_INFO1_MASK, and SD_INFO2_MASK)

   Set the SEC bit in SD_STOP to 1, and set the number of transfer blocks in SD_SECCNT.

3) Command issue (CMD25)

   Set CMD25 Argument in SD_ARG and write H'0000 0019 to SD_CMD.

   Accordingly, CMD25 is issued, and the multiple block write operation is started.

4) Response check

   On receiving the response, INFO0 (response end) in SD_INFO1 is set to 1 to generate an interrupt. Clear INFO0 to 0 and read the response from SD_RSP54.

   If the result of response decoding is an error, the command sequence can be halted by setting the STP bit in SD_STP to 1. Setting the STP bit to 1 also causes CMD12 to be issued and the response received. If the command sequence is halted because the access end interrupt has been enabled, an interrupt will be generated by setting of the INFO2 bit (access end) bit in SD_INFO1 to 1 when reception of the response has been completed. Clear the INFO2 bit to 0 and read the response.

5) Data write and data transmit to SD card

   Write H'0000 FFFB to SD_INFO1_MASK to enable the access end interrupt. In addition, write H'0000 7D80 to SD_INFO2_MASK to enable the BWE interrupt. When SD_BUF0 is ready for the data to be written, the BWE bit in SD_INFO2 is set to 1 to generate an interrupt. Clear the BWE bit to 0 and write the amount of data specified by SD_SIZE to SD_BUF0. When the data write to SD_BUF0 is completed, data is transmitted to the SD card.

   Then, the CRC status and busy state are received from the SD card. Doing this repeats transfer of the number of blocks set in SD_SECCNT.

   However, a communications error or timeout may be generated if data are being transmitted while writing to SD_BUF0 is in progress.

6) HPI command issue

   Set the HPI command argument in SD_ARG and set the HPIMODE and HPICMD bits in SD_STOP to 1.

7) Operation complete

   When reception of the response to the HPI command is completed and the busy state is released, the INFO2 and INFO0 bits in SD_INFO1 are set to 1 to generate an interrupt. Clear the INFO2 and INFO0 bits to 0 to read the response from SD_RSP10. This is the end of HPI operation.

Furthermore, perform error processing (clear the interrupt flag register) if an error occurs (a communications error or timeout).

### 150.3.6.15 Example of SD_CMD Register Setting

Table 150.9 and Table 150.10 list the example of SD_CMD and MMC_CMD register setting.

**Table 150.9 Example of SD_CMD Register Setting (SD)**

| Type | Command | Example of SD_CMD Register Setting | Remark |
|---|---|---|---|
| CMD | CMD0 | H'0000 0000 | |
| | CMD2 | H'0000 0002 | |
| | CMD3 | H'0000 0003 | |
| | CMD4 | H'0000 0004 | |
| | CMD5 | H'0000 0705 or<br>H'0000 0005 | |
| | CMD6 | H'0000 1C06 or<br>H'0000 0006 | |
| | CMD7 | H'0000 0007 | When the card is placed in the deselected state, the response timeout flag will be set since there is no response. |
| | CMD8 | H'0000 0408 or<br>H'0000 0008 | |
| | CMD9 | H'0000 0009 | |
| | CMD10 | H'0000 000A | |
| | CMD11 | H'0000 040B or<br>H'0000 000B | |
| | CMD12 | H'0000 000C | |
| | CMD13 | H'0000 000D | |
| | CMD15 | H'0000 000F | |
| | CMD16 | H'0000 0010 | |
| | CMD17 | H'0000 0011 | |
| | CMD18 | H'0000 0012 | With auto CMD12 enabled (other than in SDR104 mode) |
| | | H'0000 7C12 | With auto CMD12 disabled (only in SDR104 mode) |

| Type | Command | Example of SD_CMD Register Setting | Remark |
|---|---|---|---|
| | CMD19 | H'0000 1C13 or H'0000 0013 | Setting prohibited in HS400 mode |
| | CMD20 | H'0000 0514 or H'0000 0014 | |
| | CMD23 | H'0000 0417 or H'0000 0017 | |
| | CMD24 | H'0000 0018 | |
| | CMD25 | H'0000 0019 | With auto CMD12 enabled (other than in SDR104 mode) |
| | | H'0000 6C19 | With auto CMD12 disabled (only in SDR104 mode) |
| | CMD27 | H'0000 001B | |
| | CMD28 | H'0000 001C | |
| | CMD29 | H'0000 001D | |
| | CMD30 | H'0000 001E | |
| | CMD32 | H'0000 0020 | |
| | CMD33 | H'0000 0021 | |
| | CMD38 | H'0000 0026 | |
| | CMD42 | H'0000 002A | |
| | CMD48 | H'0000 1C30 | |
| CMD | CMD49 | H'0000 0C31 | |
| | CMD52 | H'0000 0434 or H'0000 0034 | |
| | CMD53 | H'0000 1C35 | Single read |
| | | H'0000 0C35 | Single write |
| | | H'0000 7C35 | Multiple read |

| Type | Command | Example of SD_CMD Register Setting | Remark |
|---|---|---|---|
| | | H'0000 6C35 | Multiple write |
| | | H'0000 0035 | The value on the left can be set irrespective of whether single or multi. However, the CF39 bit in SD_ARG must be set as follows.<br>Read: 0<br>Write: 1 |
| | CMD55 | H'0000 0037 | |
| | CMD56 | H'0000 0038 | |
| | CMD58 | H'0000 7C3A | |
| | CMD59 | H'0000 6C3B | |
| ACMD | ACMD6 | H'0000 0046 | |
| | ACMD13 | H'0000 004D | |
| | ACMD22 | H'0000 0056 | |
| | ACMD23 | H'0000 0057 | |
| | ACMD41 | H'0000 0069 | |
| | ACMD42 | H'0000 006A | |
| | ACMD51 | H'0000 0073 | |

**Table 150.10 Example of MMC_CMD Register Setting**

| Type | Command | Example of SD_CMD Register Setting | Remark |
|---|---|---|---|
| CMD | CMD0 | H'0000 0000 | |
| | CMD1 | H'0000 0701 | |
| | CMD2 | H'0000 0002 | |
| | CMD3 | H'0000 0003 | |
| | CMD4 | H'0000 0004 | |
| | CMD5 | H'0000 0505 | |
| | CMD6 | H'0000 0506 | In the response busy state |
| | | H'0000 0406 | Not in the response busy state |
| | CMD7 | H'0000 0007 | When the card is placed in the deselected state, the response timeout flag will be set since there is no response. |
| | CMD8 | H'0000 1C08 | |
| | CMD9 | H'0000 0009 | |
| | CMD10 | H'0000 000A | |
| | CMD12 | H'0000 000C | |
| | CMD13 | H'0000 000D | |
| | CMD14 | H'0000 1C0E | SDIF_MODE must be set to H'0100 (with CRC16 disabled). |
| | CMD15 | H'0000 000F | |
| | CMD16 | H'0000 0010 | |
| | CMD17 | H'0000 0011 | |
| | CMD18 | H'0000 7C12 | Pre-defined |
| | CMD19 | H'0000 0C13 | SDIF_MODE must be set to H'0100 (with CRC16 disabled). |
| | CMD21 | H'0000 1C15 | Setting prohibited in HS400 mode |

| Type | Command | Example of SD_CMD Register Setting | Remark |
|---|---|---|---|
| | CMD23 | H'0000 0017 | |
| | CMD24 | H'0000 0018 | |
| | CMD25 | H'0000 6C19 | Pre-defined |
| | CMD26 | H'0000 0C1A | |
| | CMD27 | H'0000 001B | |
| | CMD28 | H'0000 001C | |
| | CMD29 | H'0000 001D | |
| | CMD30 | H'0000 001E | |
| | CMD31 | H'0000 1C1F | |
| | CMD35 | H'0000 0423 | |
| | CMD36 | H'0000 0424 | |
| | CMD38 | H'0000 0026 | |
| | CMD39 | H'0000 0427 | |
| | CMD40 | H'0000 0428 | |
| | CMD42 | H'0000 002A | |
| | CMD49 | H'0000 0C31 | |
| | CMD53 | H'0000 7C35 | |
| | CMD54 | H'0000 6C36 | |
| | CMD55 | H'0000 0037 | |
| | CMD56 | H'0000 0038 | |

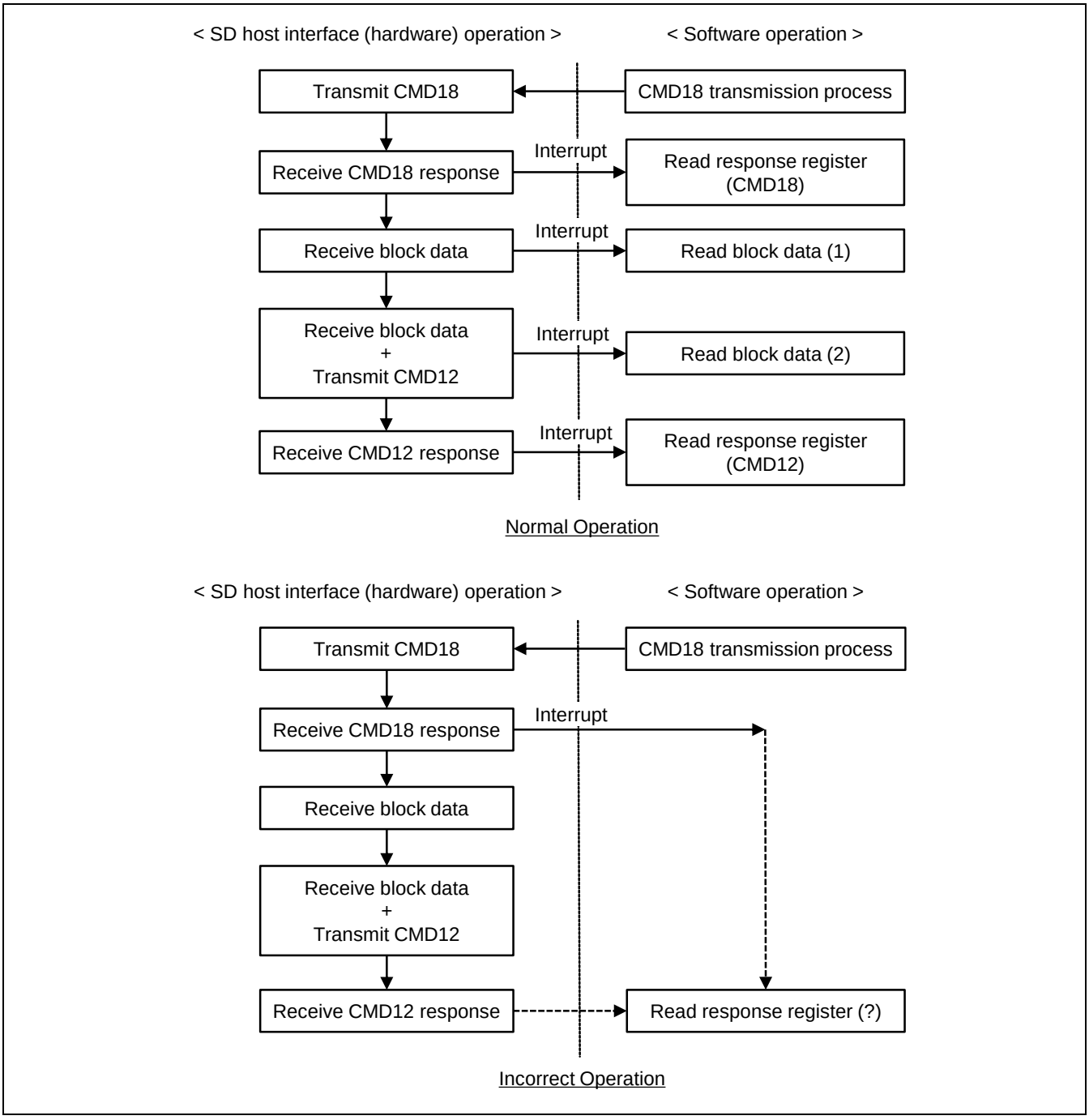



**Figure 150.29 Flowcharts for Multiple Block Read Operation (Two Blocks)**

### (1) Automatic Control of SDCLK Output

In the SD Card standard, 74 cycles of SDCLK must be output before initialization of the card. For this reason, use automatic control of SDCLK output after 74 cycles of SDCLK have been output. Furthermore, if automatic control of SDCLK output was in use, SDCLK output is stopped on completion of the sequence for a communications error or timeout. Thus, in cases where state transitions within the SD card are necessary and so on after completion of the sequence, release automatic control of SDCLK output and restart supply of SDCLK to the SD card.

### (2) Control of the C52PUB Setting for Multiple Block Write

If the C52PUB bit in SDIO_MODE is set to 1 during a sequence of multiple block write due to CMD53, CMD52 is not issued until SD_BUF becomes empty. For this reason, set the C52PUB bit after suspending writing to SD_BUF by following the appropriate procedure below.

- When DMA transfer is not in use

1. Before setting the C52PUB bit, suspend writing to SD_BUF by making the setting in SD_INFO2 to disable BWE interrupts.
2. Set the C52PUB bit in SDIO_MODE to 1 (so that CMD52 is issued when SD_BUF becomes empty).
3. After the INFO0 interrupt processing in SD_INFO1 due to the issuing of CMD52 has been completed, restart writing to SD_BUF by making the setting in SD_INFO2 to enable BWE interrupts.

- When DMA transfer is in use

1. Every time DMA transfer of the value set in SD_SIZE × n blocks (where n = 1, 2, ...) proceeds, suspend writing to SD_BUF by DMA transfer before the C52PUB bit is set.
2. Set the C52PUB bit in SDIO_MODE to 1 (so that CMD52 is issued when SD_BUF becomes empty).
3. After the INFO0 interrupt processing in SD_INFO1 due to the issuing of CMD52 has been completed, restart writing to SD_BUF by DMA transfer.

### (3) Notes on SD_CLK_CTRL Register Settings

When the SCLKDIVEN bit in SD_INFO2 is 0, SD_CLK_CTRL cannot be written to. Before writing to SD_CLK_CTRL, be sure to check that the SCLKDIVEN bit in SD_INFO2 is 1.

### (4) Restrictions on specifications

1. The SDIO suspend/resume is not supported.
2. The SPI bus is not supported.
3. The shared bus and 8-bit SD bus for embedded SDIO are not supported.
4. The stream transfer for MMC cards is not supported.
5. The open-ended multiple-block transfer for MMC cards is not supported.

### (5) STP bit setting during multiple block read

During execution of multiple block read with automatic CMD12 execution by setting the SEC bit in SD_STOP to 1, even if the STP bit in SD_STOP is set to 1 to forcibly stop the execution, the command sequence may not stop depending on the timing of setting the STP bit.

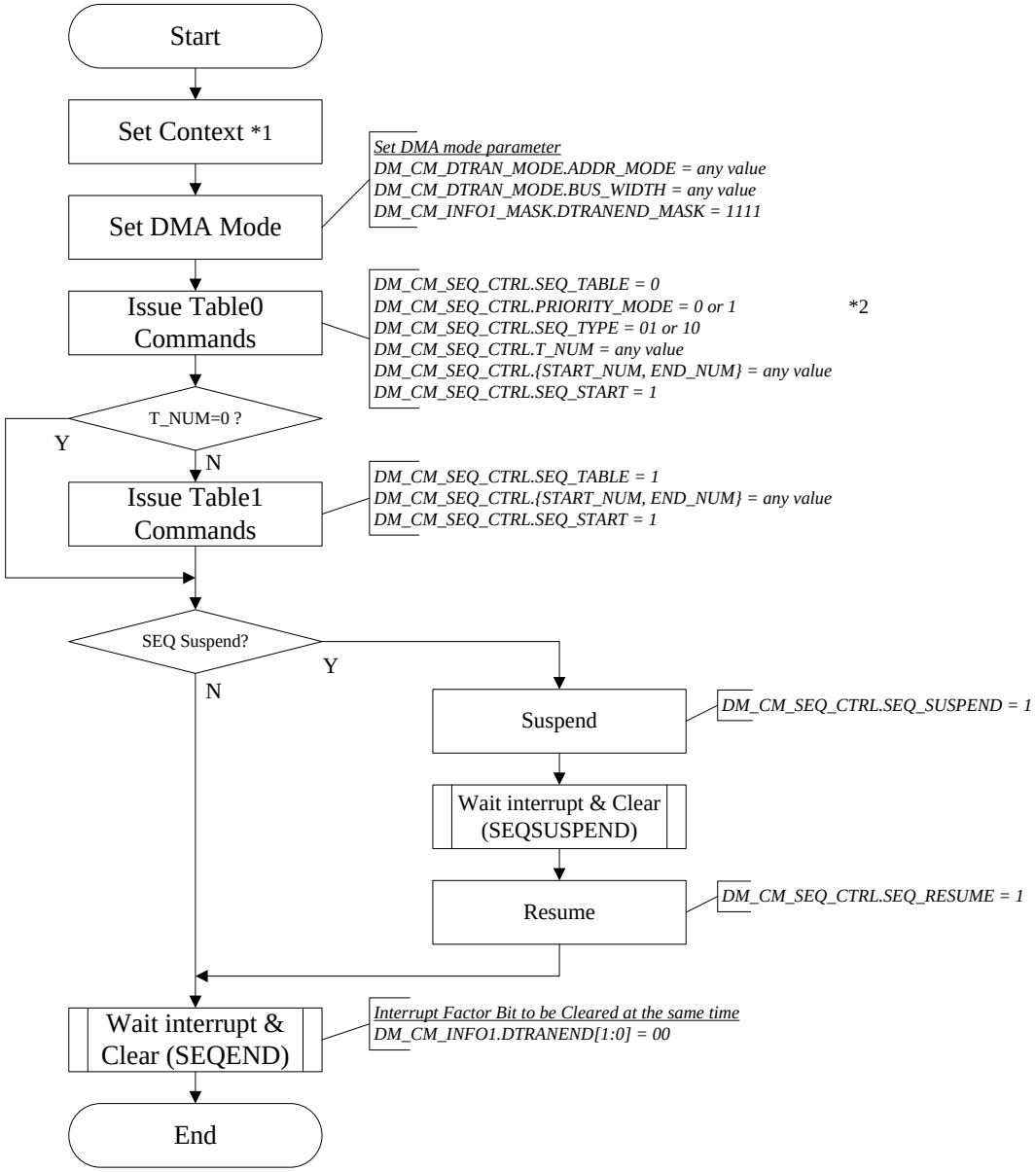
To avoid this, when setting the STP bit in SD_STOP to 1 during multiple block transfer, clear the SEC bit in SD_STOP to 0 at the same time. (Even when the SCLKDIVEN bit in SD_INFO2 is 0, change the SEC bit from 1 to 0.)

When the command sequence has not stopped because the SEC bit was not cleared to 0, the command sequence can be stopped by clearing the SDRST bit in SOFT_RST to 0.

When forcibly terminating the CMD53 multiple block transfer through the IOABT bit in SDIO_MODE, be sure to leave the SEC bit in SD_STOP as 1.

### 150.3.6.16 Multiple Commands issuing with SEQ

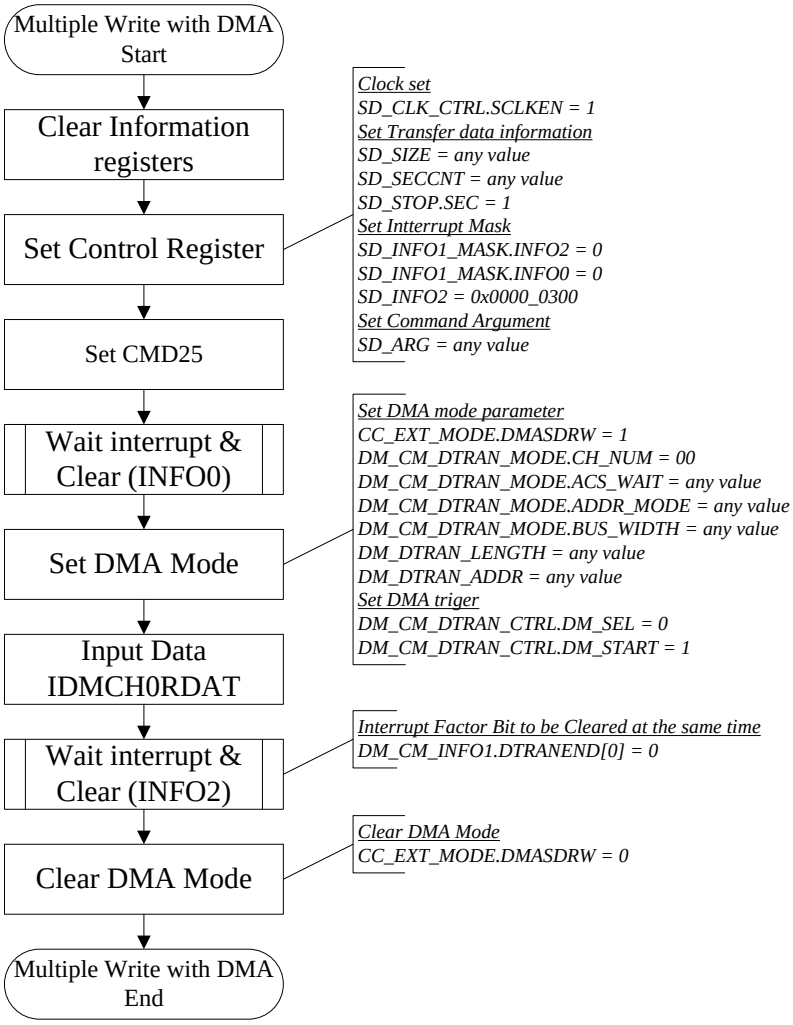
The following sequence can be executed after Card detection and Card initialization.




*1 SDIP keeps the value of context (DM_SEQ_***) after the completion of sequencer process.
Software can skip "set context" in the case of repetitive same sequencer process.
*2 Software must execute sequencer table in order of table number (table0 -> table1)

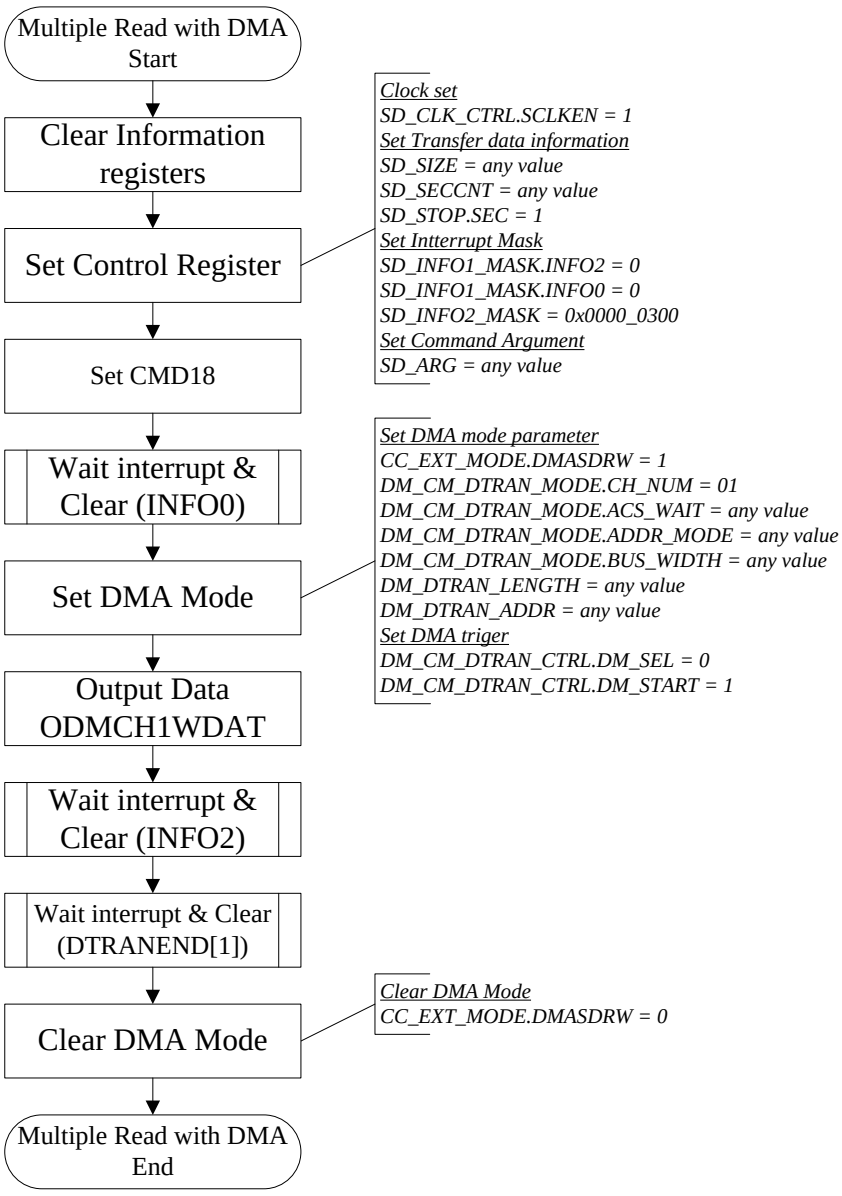
### 150.3.6.17 SD buffer DMA Write without SEQ

The following sequence can be executed after Card detection and Card initialization.



### 150.3.6.18 SD buffer DMA Read without SEQ

The following sequence can be executed after Card detection and Card initialization.



### 150.3.7 Sampling Clock Controller (SCC)

#### 150.3.7.1 Features

This module controls a sampling clock (hereafter referred to as the SCC sampling clock) that is used for SD UHS-I/SDR104, MMC HS200 and MMC HS400. When this module is used with the LSI, SD UHS-I/SDR104, MMC HS200 and MMC HS400 can be supported.

The total number of TAPs in HS400 mode for each product is 8.

#### 150.3.7.2 SCC Block Diagram

Figure 150.30 and 150.30B shows a block diagram of the sampling clock controller.

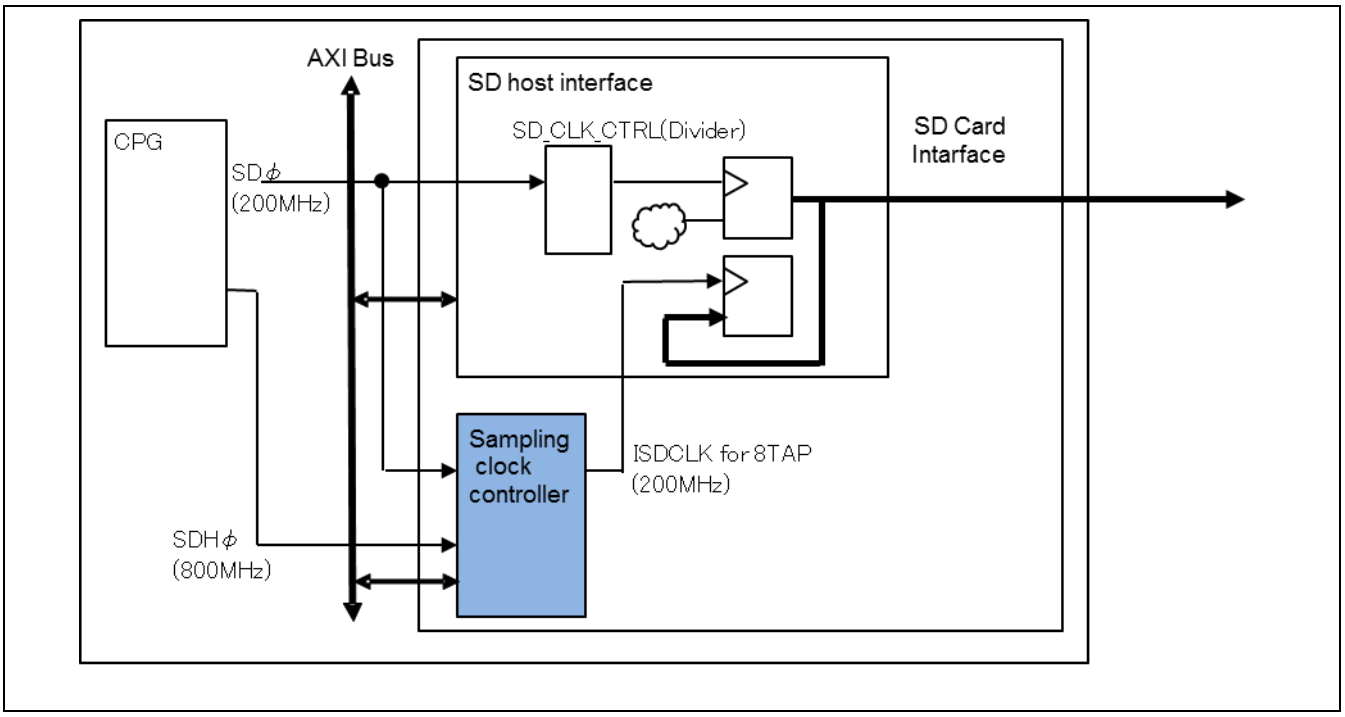



**Figure 150.30 Block Diagram of the Sampling Clock Controller**

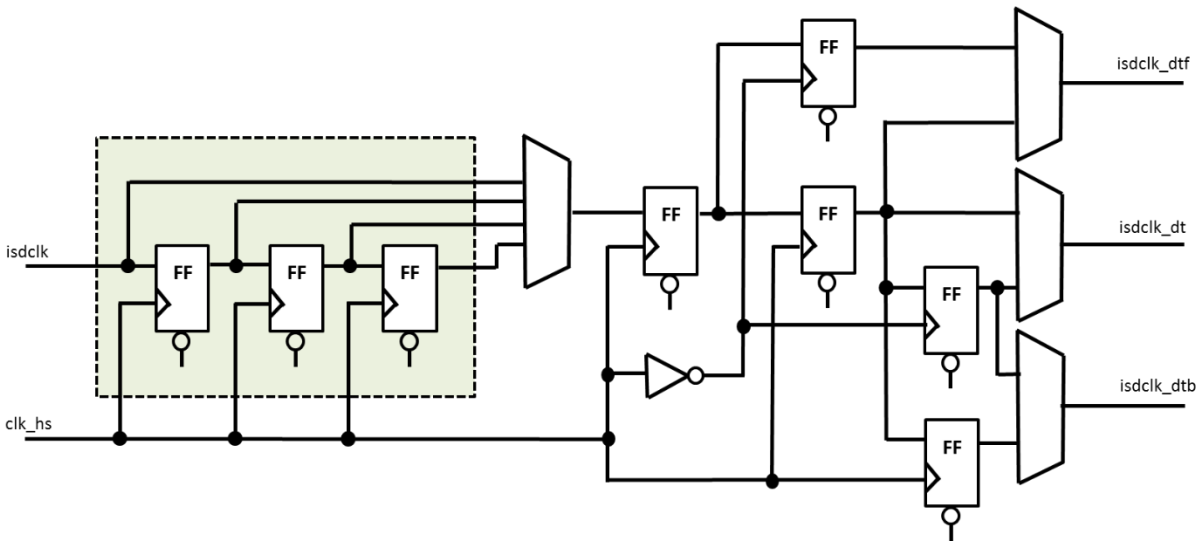


**Figure 150.31 Detail of Sampling clock controller**

The Delay Tap generates a plurality of clocks for sampling the input data (CMD / DAT) for each different phase (TAP position), and controls the phase (TAP position) of the sampling clock. Figure 150.31 shows a schematic diagram of delay Tap. This circuit shifts the edge of OSDCLK at both edges of the high-speed clock (clk_hs), and generates a sampling clock (isdclk_dt) having a plurality of phases and a correction clock (isdclk_dtf, isdclk_dtb).

The procedure for adjusting the appropriate TAP position is as follows.

(1) A sampling clock and a correction clock having a plurality of phases are generated at both edges of the high-speed clock.

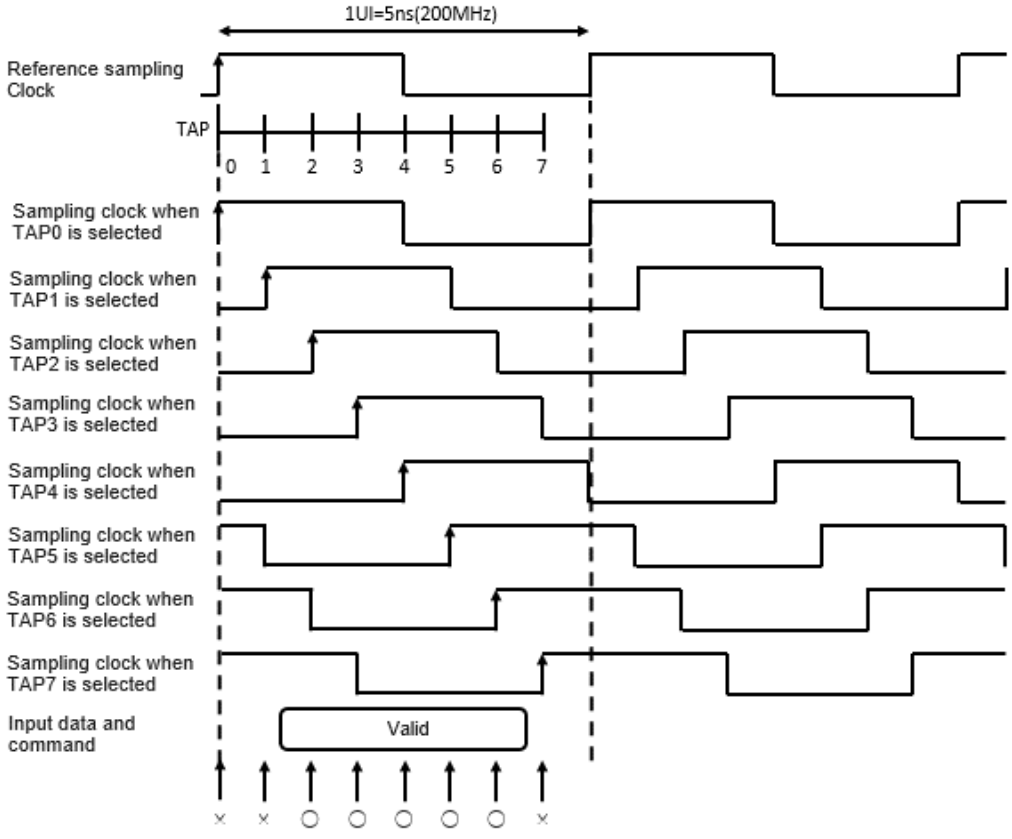
(2) Capture data with sampling clock and preceding and succeeding correction clocks.

(3) Compare the three values.

(4) When a mismatch occurs in the above comparison result, adjust the position by shifting the position of the sampling clock.

#### 150.3.7.3 Sampling clock selection (TAP position selection) function

Delay Tap divides the "sampling clock cycle" by the number of TAPs and generates sampling clocks having different phases at each divided timing. Since this transfer mode samples input data (CMD / DAT) at an appropriate timing, it is necessary to select an appropriate TAP position. Figure 150.32 shows an example of the clock generated by Delay Tap when the sampling clock is 200 MHz and the TAP number is 8 and the sampling timing adjustment method at the time of host interface initialization.



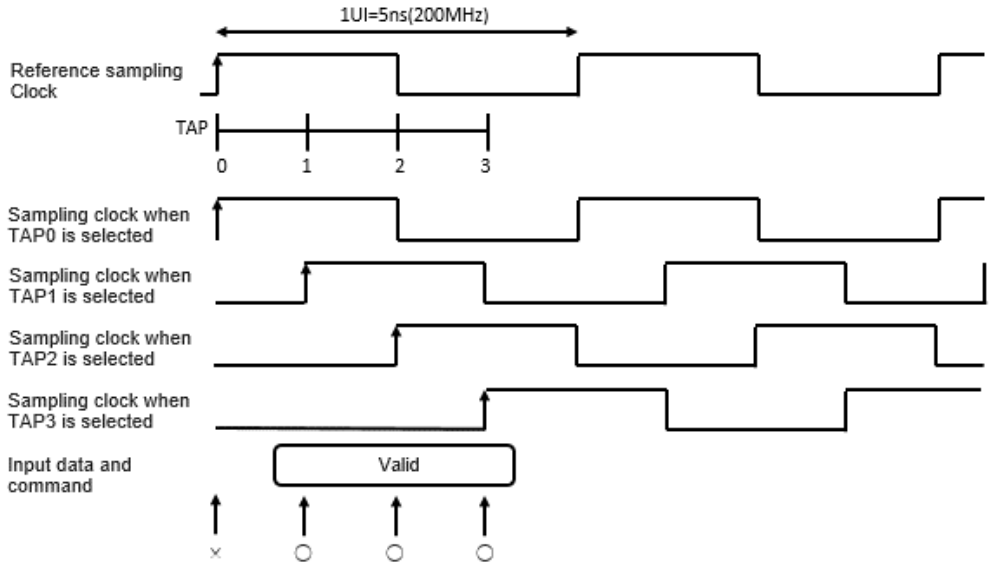
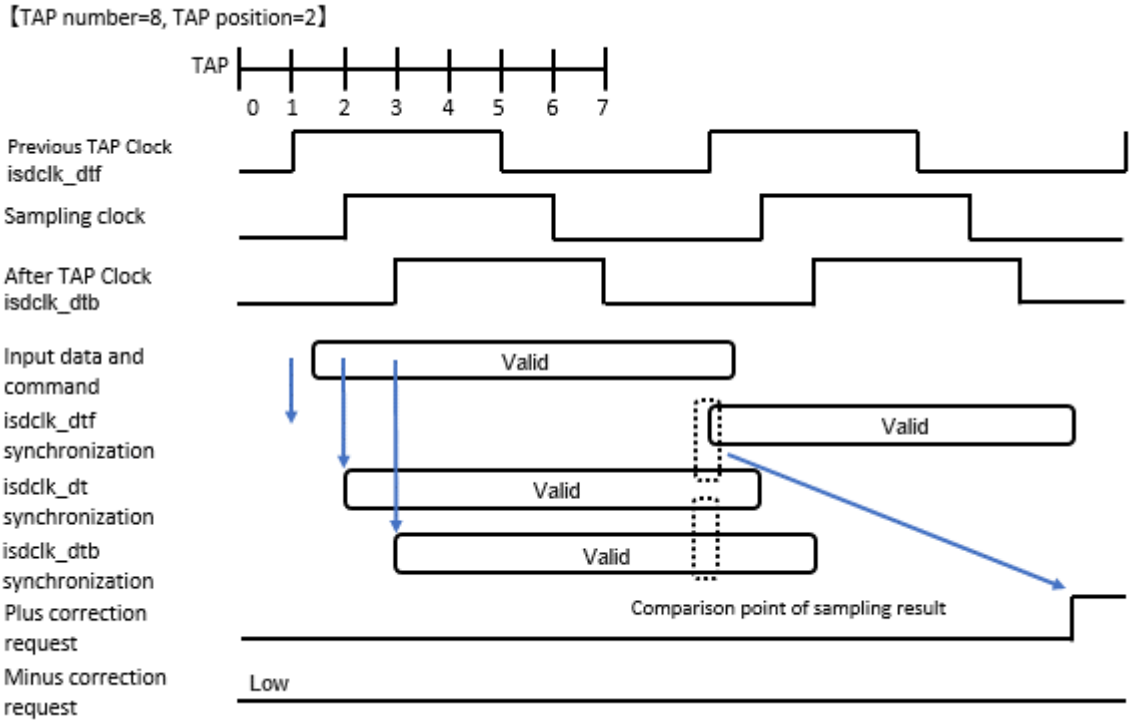



**Figure 150.32 Clock Generated by Delay Tap**

### 150.3.7.4 TAP position correction request

Figure 150.33 shows the waveform when plus correction request is generated. Figure 150.34 shows the waveform when minus correction request is generated. The value sampled by the selected TAP is compared with the value sampled by the selected preceding and succeeding TAP to judge the correction of the TAP position. When correction is requested, the TAP position is changed in accordance with the correction direction.



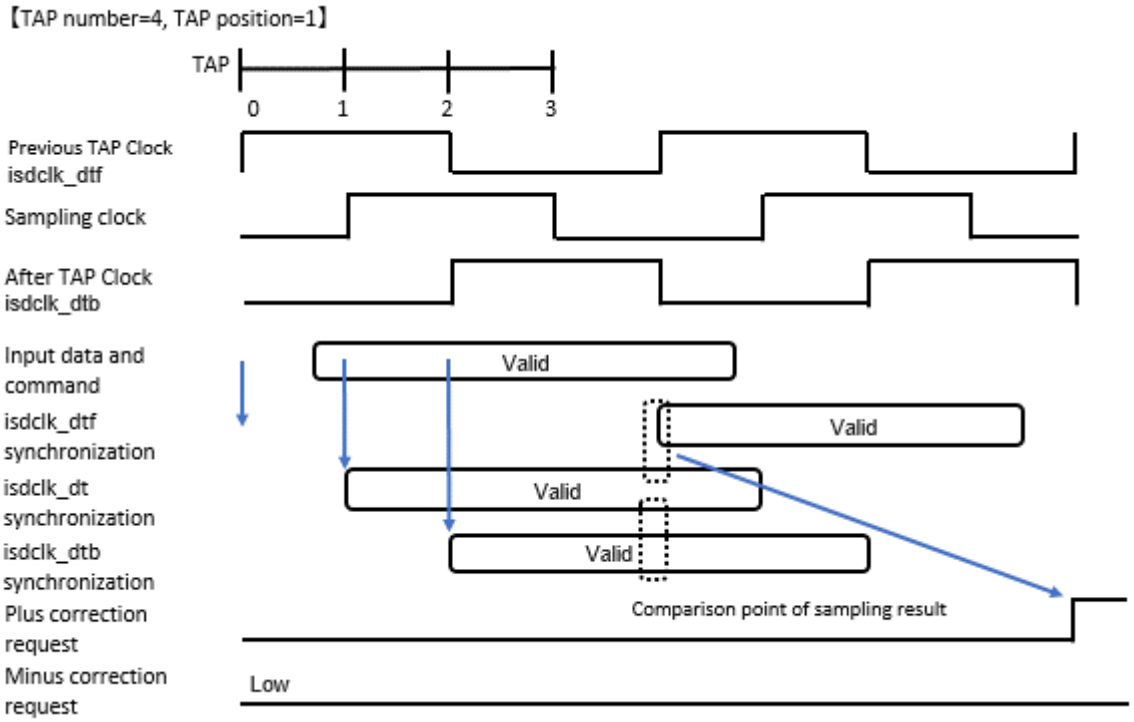



**Figure 150.33 Plus correction request generation**

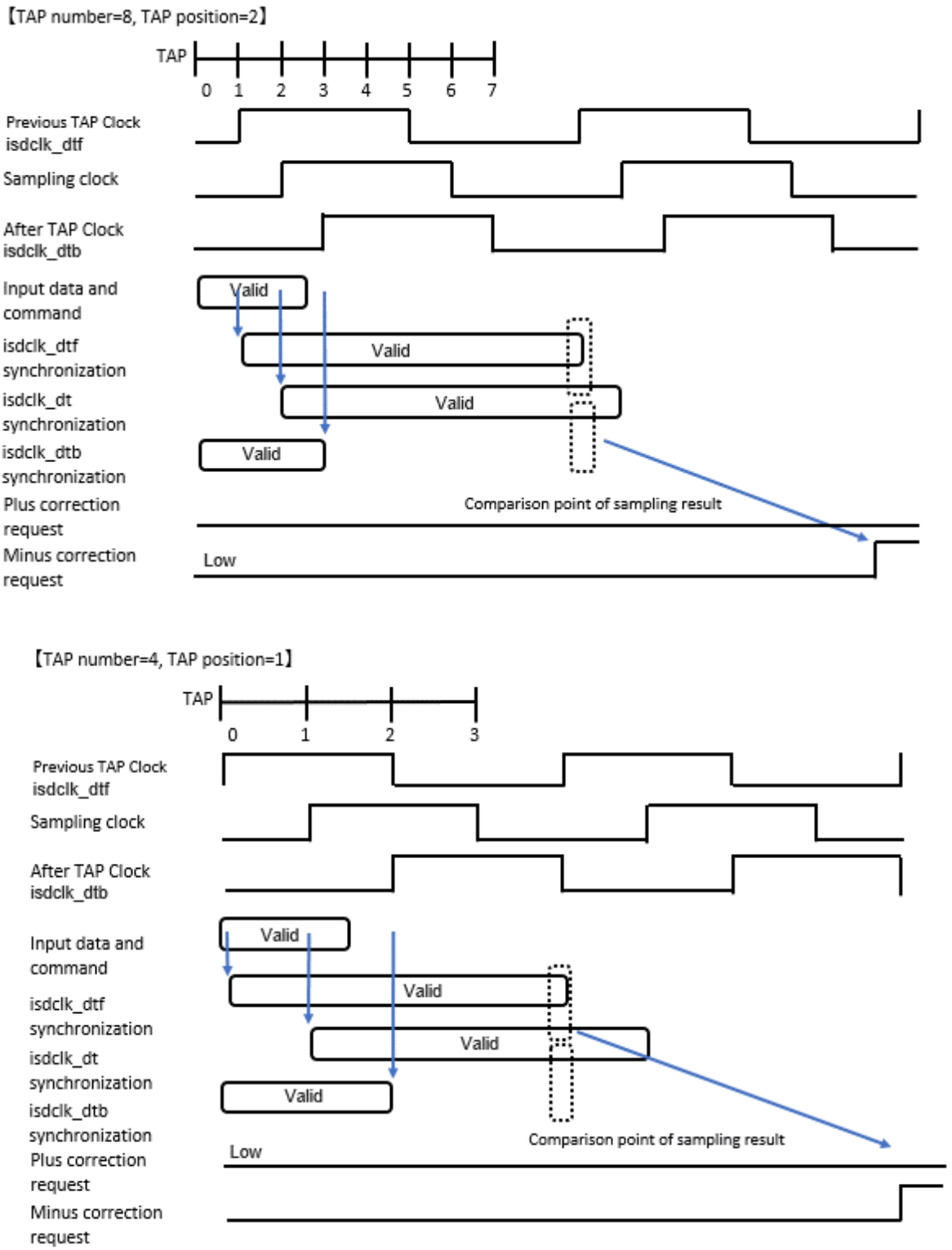



**Figure 150.34 Minus correction request generation**

#### 150.3.7.5 TAP position automatic correction

When TAP position automatic correction is enabled (the RVSEN bit of the SCC_RVSCNTL register is 1), the TAP position correction request is not set and the TAP position is automatically changed each time the command sequence is started. The TAP position after automatic correction is not reflected in the TAPSET of the SCC_TAPSET register but reflected in the TAPSEL of the SCC_RVSCNTL register. When automatic correction is performed, when the TAP position is corrected to the plus side with the maximum value, it returns to the minimum value, and when the TAP position is corrected to the minus side with the minimum value, it becomes the maximum value.

#### 150.3.7.6 TAP position correction error

When the TAP position automatic correction is invalid (the RVSEN bit of the SCC_RVSCNTL register is 0), both the plus correction request of the TAP position (the REQTAPUP bit of the SCC_RVSREQ register) and the minus correction request (the REQTAPDWN bit of the SCC_RVSREQ register) within the same command sequence When it is set, Delay Tap correction request error occurs (1 set in RVSERR of SCC_RVSREQ register).
When the TAP position automatic correction is enabled (the RVSEN bit of the SCC_RVSCNTL register is 1), the plus correction request (the REQTAPUP bit of the SCC_RVSREQ register) and the minus correction request (the REQTAPDWN bit of the SCC_RVSREQ register) of the TAP position are not set, If one condition is satisfied for both the plus correction request and the minus correction request of the TAP position in the TAP position, a Delay Tap correction request error occurs (one set in the RVSERR of the SCC_RVSREQ register). If there is a Delay Tap correction request error, it is necessary to perform tuning again.

#### 150.3.7.7 SCC Register Configuration

Following DLL Register Configuration table is the SCC registers that can access by SCC_TMPPORT4/5/6/7.

**DLL Register Configuration**

| Name | Abbreviation | dll_addr[5-0]* | Access Size |
|---|---|---|---|
| Mode register | MD | H'00 | 32 |
| Shifter code setting register | DQSRSETH0 | H'22 | 32 |
| Shifter code monitor register | DQSRMONH0 | H'26 | 32 |

Note: (*) Refer to 150.2.53 Hardware Adjustment Register 5 (SCC_TMPPORT5).

### 150.3.8 Usage Example of SCC

#### 150.3.8.1 Overview

Below is an overview of SCC settings.

Figure 150.35 shows the example of setting flow when using SCC.

When using SCC, it is necessary to perform tuning to determine the tap position of the received clock. Tuning is performed on the HS200 even when operating on the HS400. When tuning is completed, set the clock and mode to HS400 when operating with HS400.

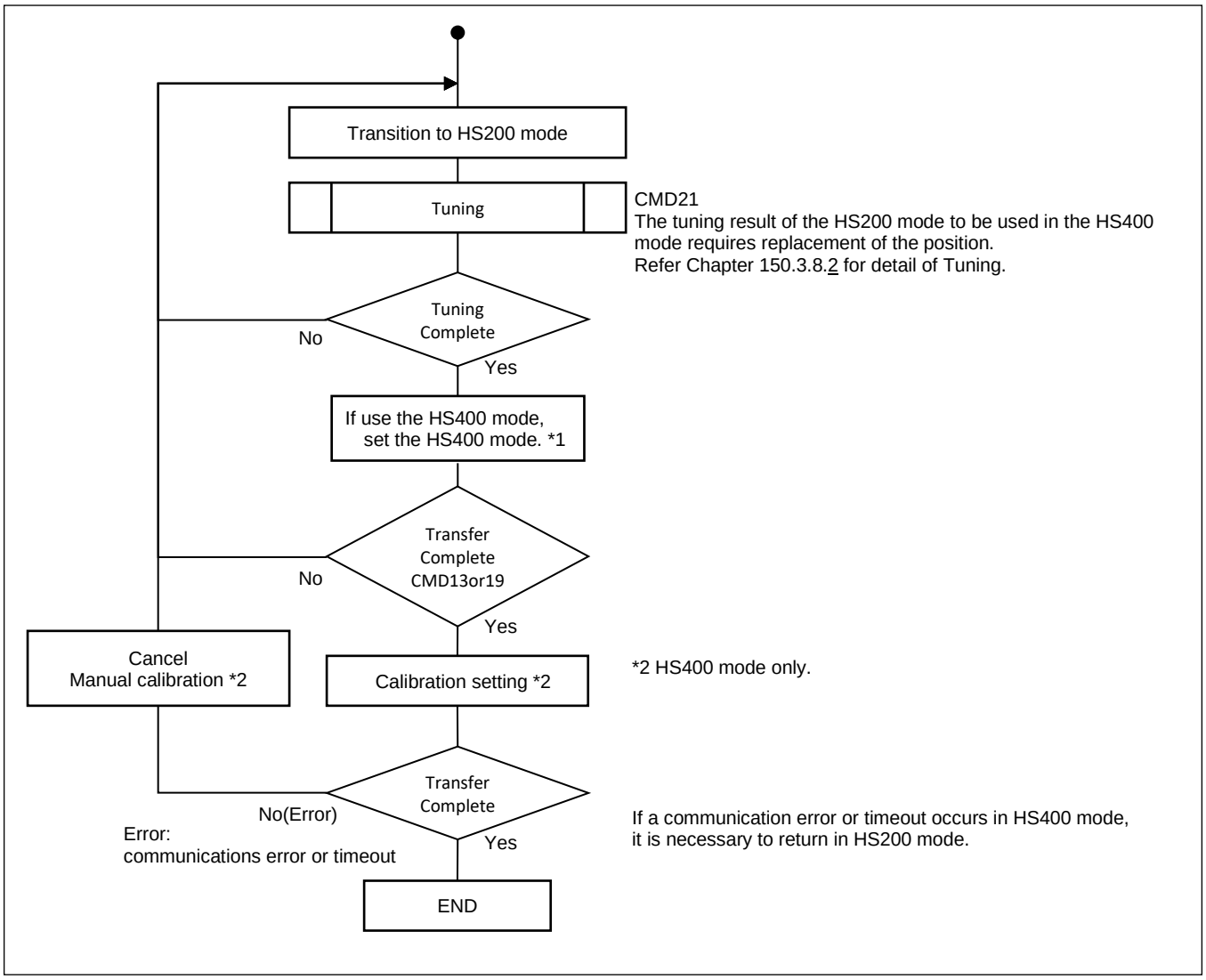



**Figure 150.35 Example of setting flow when using SCC.**

*1 Automatic Correction is a function to adjust the TAP position automatically. When operating with HS400, set Calibration.

### 150.3.8.2 Tuning

SCC is tuned by using operation of single-block reading.

Tuning is performed on the HS200 even when operating on the HS400.

As shown in Figure 150.36, check whether the single-block read command normally ends when the sampling clock position is changed from 0 to TAPNUM-1 and save the result. After checking, confirm that there exists the range which has three or more continuous normal ends (OK). Then, the median value within the continuous range is determined as the final adjustment value.

Figure 150.37 and Table 150.11 show the detailed tuning flow and the method how to select the sampling clock position (example when TAPNUM = 8).

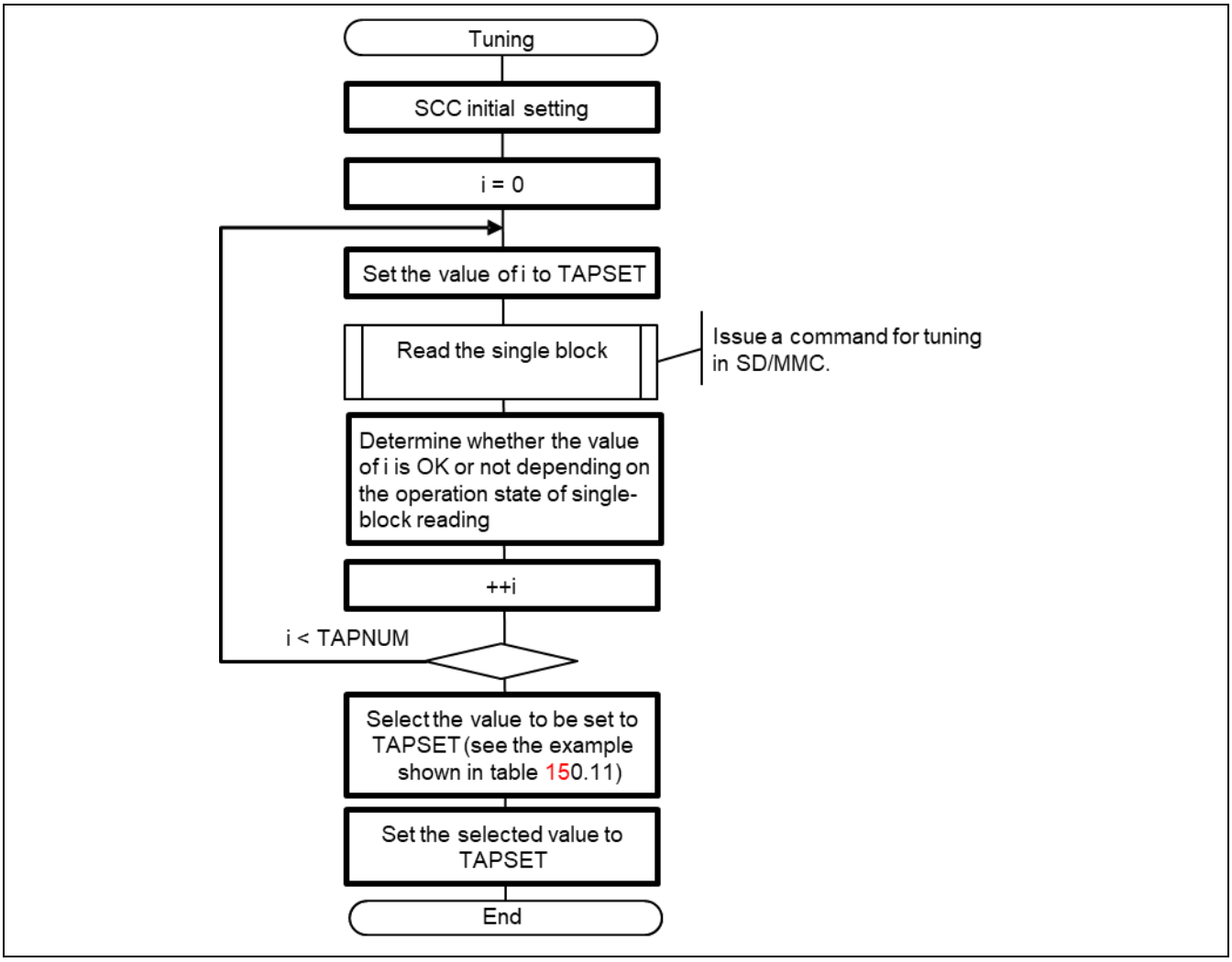



**Figure 150.36 Example of Tuning Flow (Outline)**

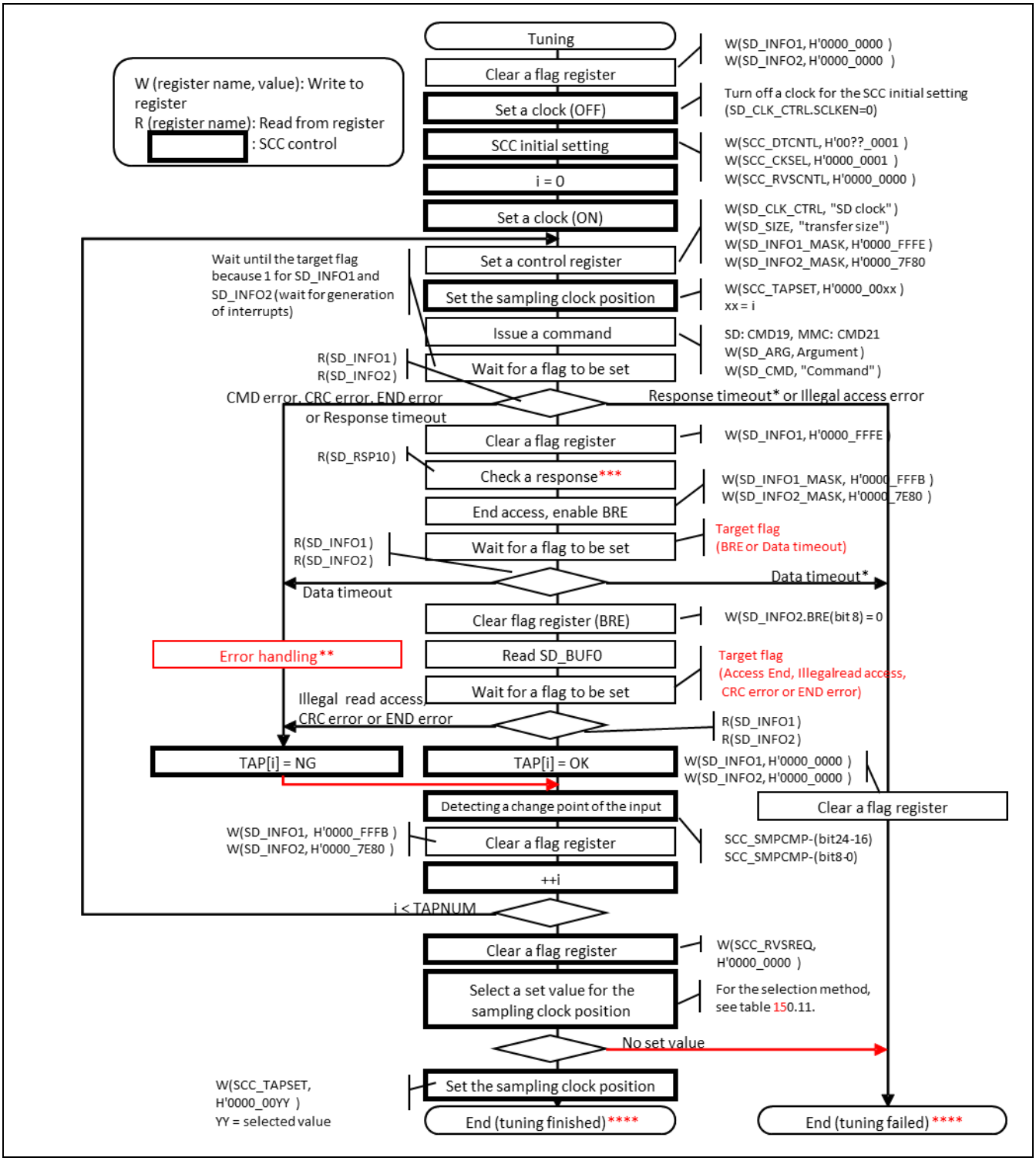



**Figure 150.37 Example of Tuning Flow (Detailed)**

*: If all of the TAP[i] has become "NG".
**: Perform one of the following for error handling.
- Issue STOP command(CMD12). (CMD error, CRC error, END error and Response timeout can be ignored.)
- Wait for a device to be ready. (Please decide waiting time by checking datasheet of memory device, and please set maximum delay value of data for tuning command as waiting time.)

***: Illegal command shall be ignored.
****: Clean-up is expected by issuing CMD13 and discard the response.

**Table 150.11 Example of the Method How to Select the Sampling Clock Position (when TAPNUM = 8)**

| Item | i | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 *1 |
|---|---|---|---|---|---|---|
| TAP[i] | 0 | NG | OK | NG | OK | OK |
| | 1 | OK | OK | NG | NG | OK |
| Max. value → | 2 | OK | NG | OK | NG | OK |
| | 3 | OK(←) | NG | OK | NG | OK |
| | 4 | OK | NG | NG | NG | OK |
| | 5 | OK | OK | NG | OK | OK |
| | 6 | NG | OK | NG | OK(←) | OK |
| | 7 | NG | OK(←) | NG | OK | OK |
| | (0) | NG | OK | NG | OK | OK |
| | (1) | OK | OK | NG | NG | OK |
| | (2) | OK | NG | OK | NG | OK |
| | (3) | OK | NG | OK | NG | OK |
| | (4) | OK | NG | NG | NG | OK |
| | (5) | OK | OK | NG | OK | OK |
| | (6) | NG | OK | NG | OK | OK |
| | (7) | NG | OK | NG | OK | OK |
| Selected value | | i = 3 | i = 7 | Fail | i = 6 or 7 | i = 0 to 7 |

Symbols: (←): Example of selection, (x): repeated display of index x of TAP[x]

*1 If all of the TAP [i] is OK, the sampling clock position is selected by identifying the change point of data. Change point of the data can be found in the value of SCC_SMPCMP register. Usage example is 150.3.8.3 Change point of the input data

(a) The sampling clock position is selected by considering a margin in the range which has three or more continuous 'TAP[i] = OK'.

(b) The sampling clock position is repeated from 0 after the maximum value (TAP number-1). In case 2 above, that position is continued in the order of 5→6→7→0→1.

### 150.3.8.3 Sampling Clock Position Correction after Tuning (Excluding HS400 mode)

After tuning, correction of the sampling clock position may be required when a command is issued.

There are manual and automatic correction methods. After a command sequence, if the CMD, CRC, END error or time out occurs or the correction error occurs, tuning will be performed again. The following shows examples of manual and automatic correction methods.

**(1) Manual correction of the sampling clock position**

Figure 150.38 shows the flow of manual correction of the sampling clock position. If manual correction is out and a communication failure occurs, require performed re-tuning.

Table 150.12 shows set values determined when correction is required.

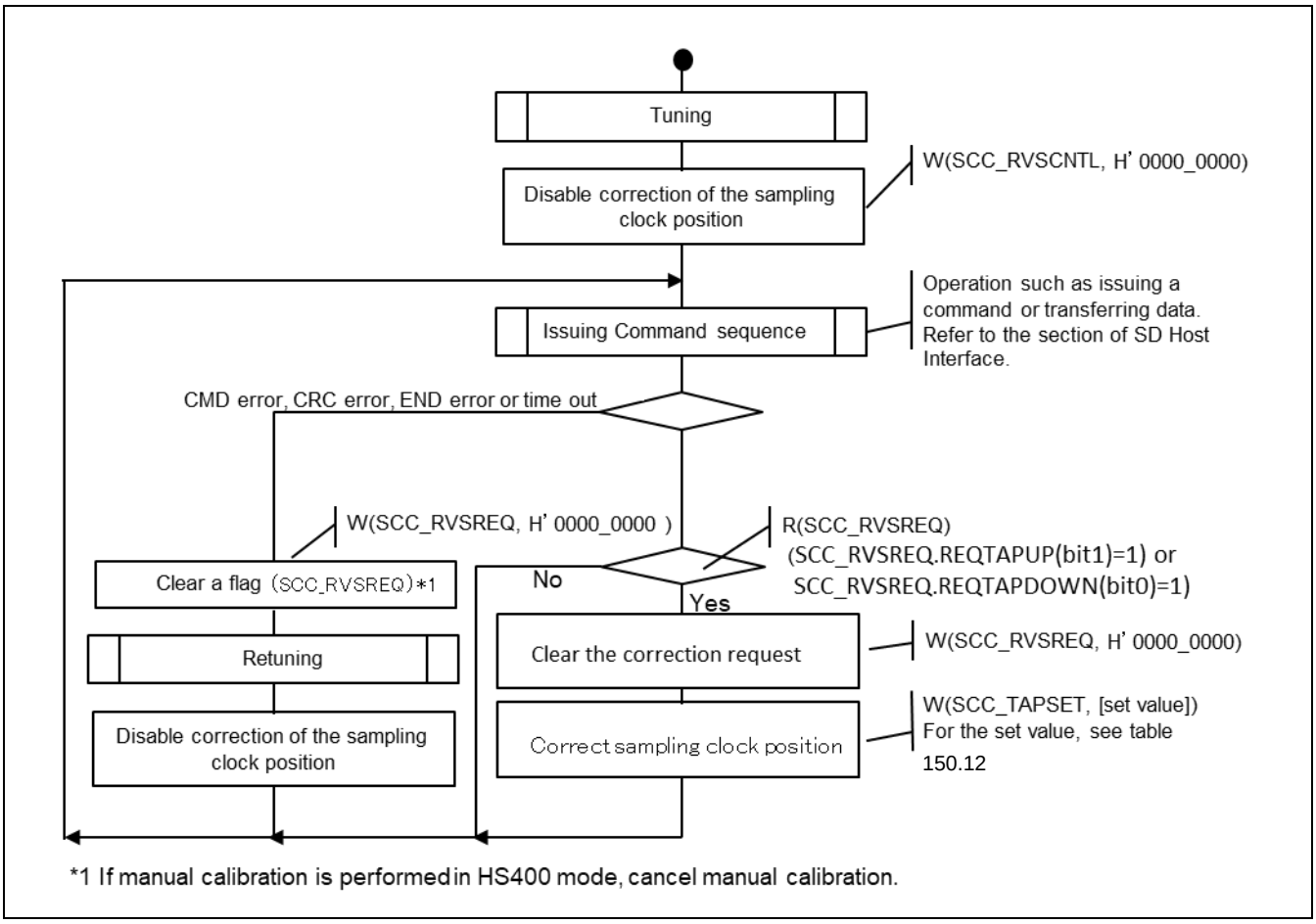



**Figure 150.38 Flow of Manual Correction of the Sampling Clock Position (Example)**

**Table 150.12 Set Values for TAPSET when Correction is Required**

| No. | Current value of TAPSET | Value set to TAPSET when REQTAPUP = 1 | Value set to TAPSET when REQTAPDWN = 1 |
|---|---|---|---|
| 1 | 0 | 1 | 7 |
| 2 | 1 | 2 | 0 |
| 3 | 2 | 3 | 1 |
| 4 | 3 | 4 | 2 |
| 5 | 4 | 5 | 3 |

| No. | Current value of TAPSET | Value set to TAPSET when REQTAPUP = 1 | Value set to TAPSET when REQTAPDWN = 1 |
|---|---|---|---|
| 6 | 5 | 6 | 4 |
| 7 | 6 | 7 | 5 |
| 8 | 7 | 0 | 6 |

Note: As is the case in the tuning selection method, the sampling clock position is 0 after the maximum value (TAPNUM-1).

### (2) Automatic correction of the sampling clock position

Figure 150.39 shows the flow of automatic correction of the sampling clock position. If automatic correction is out and a communication failure occurs, require performed re-tuning.

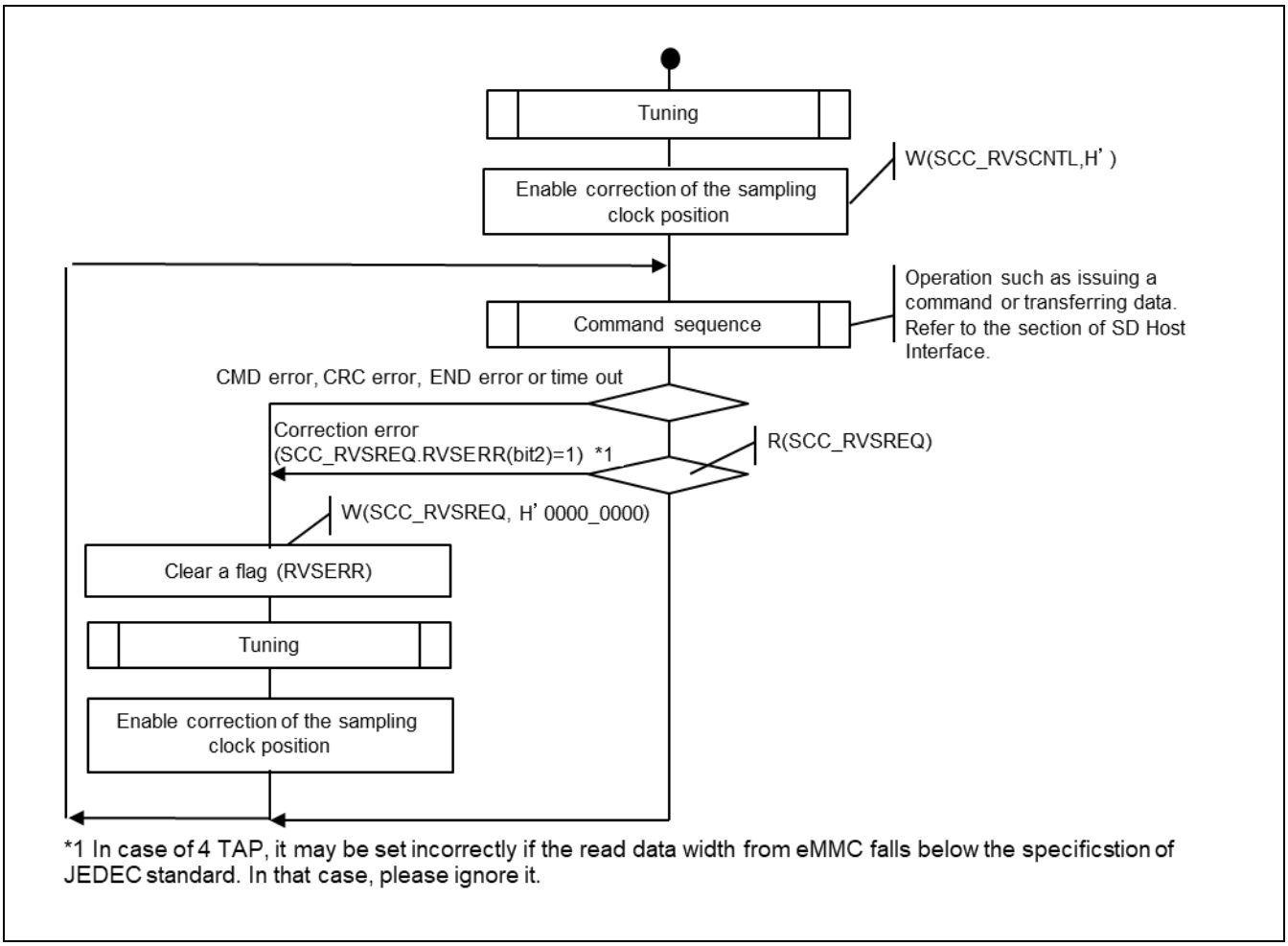



**Figure 150.39 Flow of Automatic Correction of the Sampling Clock Position (Example)**

#### 150.3.8.4 Sampling Clock Position Correction after Tuning (HS400 mode)

After tuning, correction of the sampling clock position may be required when a command is issued.

There are manual and automatic correction methods. After a command sequence, if the CMD, CRC, END error or time out occurs or the correction error occurs, tuning will be performed again. The following shows examples of manual and automatic correction methods.

When operating in HS400 mode, there are TAPs whose settings are prohibited. It is necessary to keep the result of tuning in order to determine whether the operable TAP has changed since tuning. The prohibited TAPs for each product are as follows.

**(1) Manual correction of the sampling clock position**

Figure 150.38 shows the flow of manual correction of the sampling clock position. If manual correction is out and a communication failure occurs, require performed re-tuning.

Table 150.12 shows set values determined when correction is required.

a. Setting procedure

1. After a command sequence, if the CMD, CRC, END error or time out occurs or the correction error occurs, tuning will be performed again. The following is an example of how to correct for HS400.

2. Is it a true Correction request? If the value read from SCC_RVSREQ is not 'H0000_0000, check the detailed information. Only the command signal status of bit24 and bit8 of SCC_SMPCMP is referenced. If these 2 bits are 00, there is no command signal error and the transfer continues. If it is not 00, move to Correction.

3. Correction:

When above 2bit is 11, it will be retuning as a correction error.

If not, determine the destination TAP according to the value of 2 bits above.

In the case of 01, since it is a Down request, the destination TAP is set to TAP [i-1].

In the case of 10, since it is an Up request, the destination TAP is set to TAP [i + 1].

If the destination TAP is not a prohibited TAP, move the TAP. If the destination TAP is a prohibited TAP, check if the TAP that caused the correction request was passing during tuning.

If it originally failed, the status does not change and the transfer continues. If it was passed at the time of tuning, the condition deteriorated and retuning is performed. If any bit of SCC_RVSREQ is set after issuing the command, read SCC_SMPCMP to get detailed information. The read value determines whether to retune, move the TAP position, or continue at the same TAP position.

Figure 150.40 shows the flow of manual correction of the sampling clock position. (TAP number = 8)

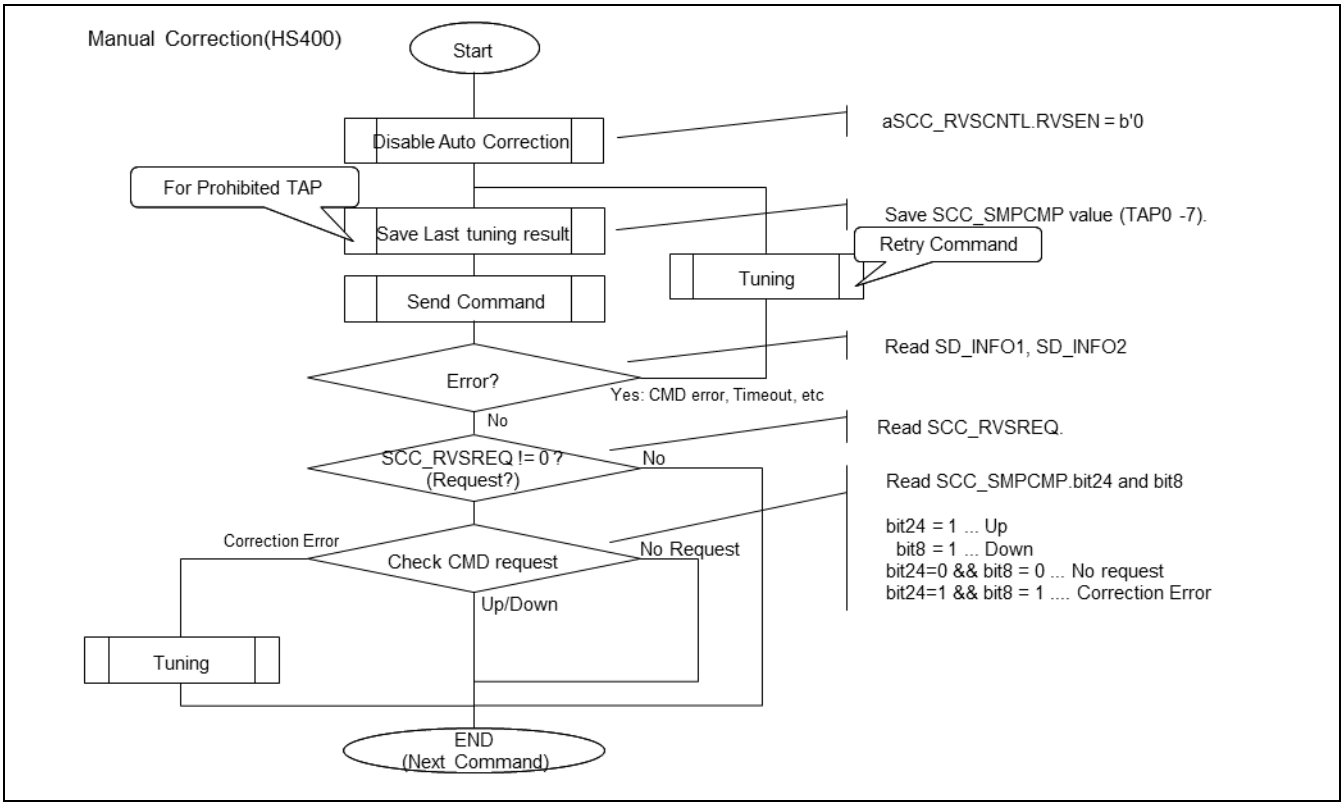



**Figure 150.40 Flow Example of Sampling clock position correction(TAP number=8)**

**(2) Automatic correction of the sampling clock position**

Figure 150.41 shows the flow of automatic correction of the sampling clock position. If automatic correction is out and a communication failure occurs, require performed re-tuning.

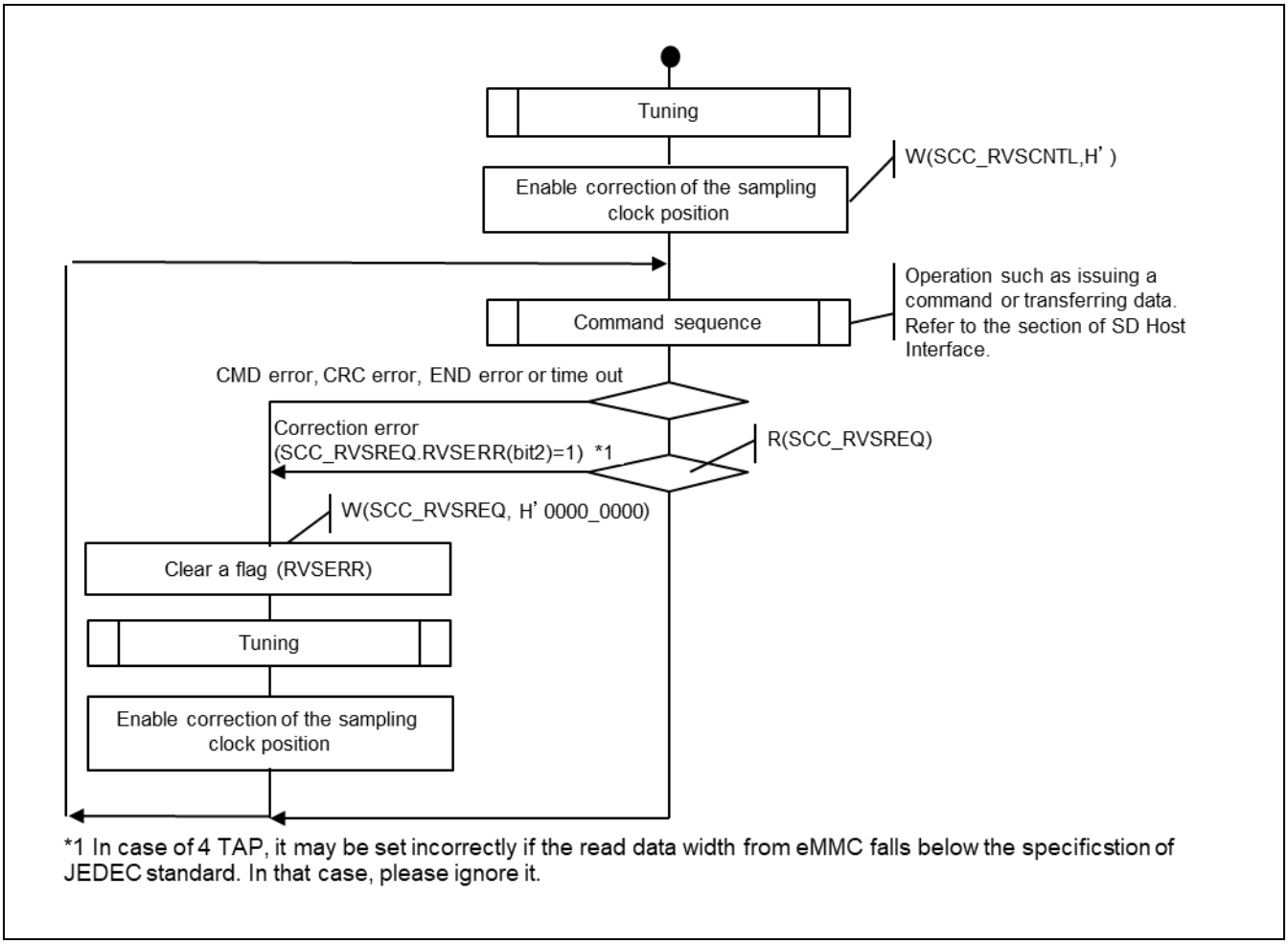



**Figure 150.41 Flow of Automatic Correction of the Sampling Clock Position (Example)**

### 150.3.8.5 Calibration (Only HS400 mode)

With the HS400, data is captured by the clock generated by shifting the phase of the DS signal.
Adjust the delay value for the DS signal to capture data.
There are two ways to determine the delay value.

(1) Auto calibration
Auto calibration is set by default.

(2) Manual calibration
The manual calibration procedure is shown below.

a. Manual calibration setting
1. Transition to HS400 mode(HS400EN=1)
Issue of CMD13(SEND_STATUS)

2. Write Protect release of DLL register
W(SCC_TMPPORT5, H'0000_0000) Write instruction
W(SCC_TMPPORT6, H'A500_0000) Write Protect release

W(SCC_TMPPORT4, H'0000_0001) Register access start
W(SCC_TMPPORT4, H'0000_0000) Register Access End

3. Reading the calibration code value
W(SCC_TMPPORT5, H'0000_0126) Read instruction
W(SCC_TMPPORT4, H'0000_0001) Register access start
W(SCC_TMPPORT4, H'0000_0000) Register Access End
R(SCC_TMPPORT7) Read the calibration code value

4. Manual calibration of internal DS signal
W(SCC_TMPPORT5, H'0000_0022) Write instruction
W(SCC_TMPPORT6, H'0000_00m) Set the DQSRSETH0.Set_enable bit and "calibration code value"
W(SCC_TMPPORT4, H'0000_0001) Register access start
W(SCC_TMPPORT4, H'0000_0000) Register Access End
W(SCC_TMPPORT3, H'0000_000n) Delay Line Offset value change (n=3)

i: read calibration code
m: calibration code
n: offset value

b. Cancel manual calibration
If retuning is required due to communication error, cancel manual calibration.
Please note that the calibration value is not cleared by software reset (SOFT_RST) because the register that sets the calibration value is a local register in the DLL.

1. DLL built-in register Write Protect release
W(SCC_TMPPORT5, H'0000_0000) Write instruction
W(SCC_TMPPORT6, H'A500_0000) Write Protect release
W(SCC_TMPPORT4, H'0000_0001) Register access start
W(SCC_TMPPORT4, H'0000_0000) Register Access End

2. Cancel manual correction of internal DS signal
W(SCC_TMPPORT5, H'0000_0022) Write instruction
W(SCC_TMPPORT6, H'0000_0000) Cancel manual correction
W(SCC_TMPPORT4, H'0000_0001) Register access start
W(SCC_TMPPORT4, H'0000_0000) Register Access End
W(SCC_TMPPORT3, H'0000_0000) Offset value Initial state set

#### 150.3.8.6 Setting Example (HS400 mode)

(HS400 mode setting)
The HS400 mode setting using the SCC sampling clock are shown below.

(1)SDSRCφ is 800 a (MHz).
Register is SDHI of CPG Clock Frequency Control Register (SD0CKCR).
SDSRCFC[2:0]：3'b000
SD0FC[1:0]：2'b01

(2)The output clock 1: 1 mode.
Register is SD Clock Register (SD_CLK_CTRL).
DIV[7:0]：8'b1111 1111

(3)Setting of SCC_DT2FF for HS200
Register is Hardware Adjustment Register 1(SCC_DT2FF)
DT2NESET7 to DT2NESET0: 8'H03
DT2NSSET7 to DT2NSSET0: 8'H00

(4)Enable the SCC sampling clock.
Register is the sampling clock select register (SCC_CKSEL).

DTSEL：1’b1

(5)It will conduct the 8TAP tuning of CMD terminal and data terminal in the HS200 mode.

(6)SDSRCφ = 800 a (MHz).
Register is SDHI of CPG Clock Frequency Control Register (SD0CKCR).

SDSRCFC[2:0]：3’b000(SD0H = 800MHz)

SD0FC[1:0]：2’b01(SD0 = 200MHz)

(7)The output clock is the 1/2 frequency mode.
Register is SD Clock Register (SD_CLK_CTRL).
DIV[7:0]：8’b0000 0000

(8)Setting of SCC_DT2FF for HS400
Register is Hardware Adjustment Register 1(SCC_DT2FF)

DT2NESET7 to DT2NESET0: 8’H03
DT2NSSET7 to DT2NSSET0: 8’H00

(9)Enable the HS400 mode.
Register is Interface Mode Setting Register. (SDIF_MODE)
HS400 : 1’b1
Register is a hardware adjustment register 2. (SCC_TMPPORT2)
HS400EN：1’b1

HS400OSEL：1’b1

(10)Manual calibration
The manual calibration procedure is shown below.

① Transition to HS400 mode(HS400EN=1)
Issue of CMD13(SEND_STATUS)

② Write Protect release of DLL register
W(SCC_TMPPORT5, H’0000_0000) Write instruction
W(SCC_TMPPORT6, H’A500_0000) Write Protect release
W(SCC_TMPPORT4, H’0000_0001) Register access start
W(SCC_TMPPORT4, H’0000_0000) Register Access End

③ Reading the calibration code value
W(SCC_TMPPORT5, H’0000_0126) Read instruction
W(SCC_TMPPORT4, H’0000_0001) Register access start
W(SCC_TMPPORT4, H’0000_0000) Register Access End
R(SCC_TMPPORT7) Read the calibration code value

④ Manual correction of internal DS signal
W(SCC_TMPPORT5, H’0000_0022) Write instruction
W(SCC_TMPPORT6, H’0000_00m) Set the DQSRSETH0.Set_enable bit and write the calibration code value
W(SCC_TMPPORT4, H’0000_0001) Register access start
W(SCC_TMPPORT4, H’0000_0000) Register Access End
W(SCC_TMPPORT3, H’0000_000n) Delay Line Offset value change(n=3)

i: read calibration code

m: calibration code
n: offset value

(11) Cancel manual calibration
If retuning is required due to communication error, cancel manual calibration.

① DLL built-in register Write Protect release
W(SCC_TMPPORT5, H'0000_0000) Write instruction
W(SCC_TMPPORT6, H'A500_0000) Write Protect release
W(SCC_TMPPORT4, H'0000_0001) Register access start
W(SCC_TMPPORT4, H'0000_0000) Register Access End

② Cancel manual correction of internal DS signal
W(SCC_TMPPORT5, H'0000_0022) Write instruction
W(SCC_TMPPORT6, H'0000_0000) Cancel manual correction
W(SCC_TMPPORT4, H'0000_0001) Register access start
W(SCC_TMPPORT4, H'0000_0000) Register Access End
W(SCC_TMPPORT3, H'0000_0000) Offset value Initial state set

### 150.3.9 Change clock source from PLL5 to PLL4 for HS400/HS200 transfer clock

R-Car V4M can select PLL4 as source clock of this module to enable fine-grained frequency change.

Note that on using PLL4 for source clock, there are following restrictions.
- Clock frequency for 3D Graphics Engine also changes via PLL setting for this module
- Cannot use SSCG and Fractional multiplication (PLL4CR0.SSMODE[2:0] shall be B'000 only in this use)

Figure 150.42 shows how to change clock source from PLL5 to PLL4 and change divider ratio of PLL4.
Table 150.13 shows an example of PLL4CR0.NI [7:0] setting to make clock frequency as SD0Hφ/SD0φ = around 760/190MHz ~ 772/193MHz, respectively.

This flow changes registers SD0CKCR, PLL4CR0, CKSRCSELCR, PLLECR, MSTPCR0, and MSTPCR7 described in 8. Clock Pulse Generator (CPG) and 9. Module Standby, Software Reset section.
To write these registers, writing inverted value to CPGWPR is required before writing target value.
This flowchart omits this CPGWPR write step.

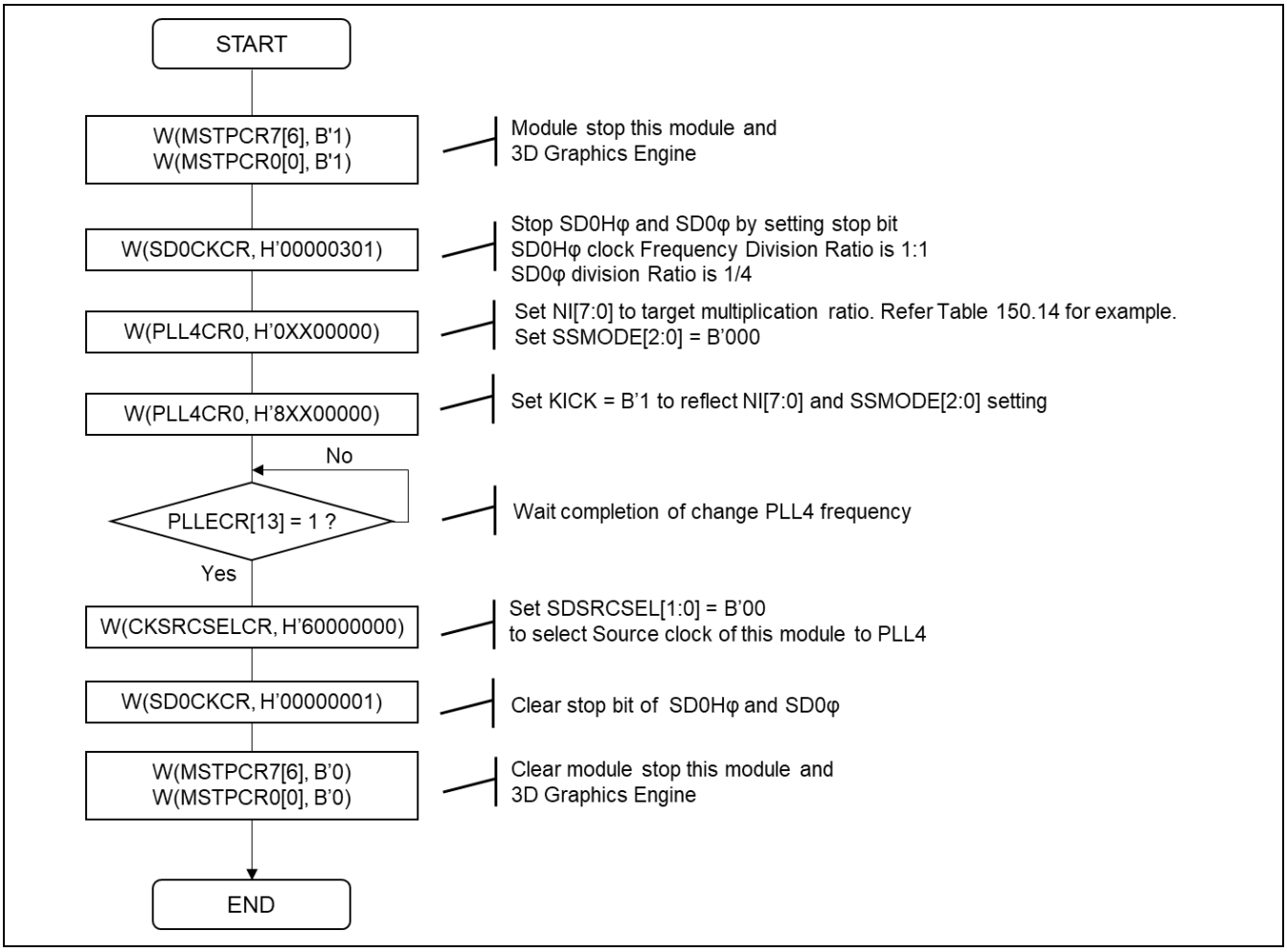



**Figure 150.42 Flow to change clock source from PLL5 to PLL4**

**Table 150.13 Example of PLL4CR0.NI [7:0] to select around SD0Hφ/SD0φ=760/190MHz ~ 772/193MHz**

| PLL4 reference clock [MHz] *1 | PLL4CR0.NI [7:0] | SD0Hφ [MHz] *2 | SD0φ [MHz] *2 |
|---|---|---|---|
| 16.66 | H'71 | 760 | 190 |
| | H'72 | 766.66 | 191.66 |
| | H'73 | 773.33 | 193.33 |
| 20.00 | H'60 | 776 | 194 |
| | H'5F | 768 | 192 |
| | H'5E | 760 | 190 |

Note.1: Please refer Table 8.1.5.4 PLL4 initial Parameter Setting for "PLL4 reference clock" information
Note.2: SD0Hφ/SD0φ value is the case of SD0CKCR.SDSRCFC[2:0] = B'000 and SD0CKCR.SD0FC[1:0] = B'01

## 150.4 Usage Notes

### 150.4.1 Designing the board

SDHI and MMC interface transfer performance has been improved. When designing the board, run the SI simulation using the IBIS model.

### 150.4.2 SD_BUF Illegal Write Access

When writing data to SD_BUF0 after the single block write or multi block write command is issued, the data of the size specified by SD_SIZE must be written to.

If the data of the size which exceeds the size specified by SD_SIZE is written to, the ERR4 bit in SD_INFO2 is set to 1. In addition, the data written to SD_BUF0 may not be transmitted and it causes the SCLKDIVEN bit in SD_INFO2 to hold the value of 0. In such cases, clearing the SDRST bit in SOFT_RST to 0 and then restoring its value to 1 clears the SCLKDIVEN bit to 1.

However, for the single byte (in the case of 16- or 32-bit access) or three bytes (in the case of 32-bit access) when the number of bytes setting in SD_SIZE is odd, or the fraction of bytes when the number of bytes setting in SD_SIZE is even (in the case of 32-bit access), since the portion of dummy data writing is regarded as excess data and ignored, it is not within the scope of the above description (the fraction of bytes: the two bytes that are not in a four-byte unit).

### 150.4.3 Block Number Limitation for Multiple Block Read

When performing a multiple block read of one or two blocks, depending on the timing with which the response register is read, the response value may not be read properly. This must be avoided by either of the following countermeasures.

1) When receiving one or two blocks of data, use single block reading.
2) Read the response to CMD18 from SD_RSP54.

<Mechanism of incorrect reading>

Figure 150.29 shows the processing flows of SD host interface (hardware) operation and software operation when a multiple block read is performed on two blocks. As shown in the incorrect operation of Figure 150.29, when an interrupt is generated on reception of the CMD18 response and the timing with which the SD card response register (SD_RSP10) is read by the interrupt is delayed, the data during the CMD12 response reception or the CMD12 response may be read. In the case of a multiple block read of three or more blocks, CMD12 is not issued until the block of data has been read, so this problem does not arise. Furthermore, in the case of a multiple block write, since the CMD25 response is read before the block of data is sent, the problem does not arise.

### 150.4.4 Clock register setting of the HS400 mode

When HS400 mode is used, setting of clock control register is bellow.

1) SD-IFn Clock Frequency Control Register(SD0CKCR)

SDSRCFC[2:0] = B’000(SD0H = 800MHz) , SD0FC[1:0] = B’01(SD0 = 200MHz)

2) SD Clock Control Register(SD_CLK_CTRL)

DIV[7:0] = B’0000 0000

### 150.4.5 Stop and restart during data communication

If stopping and restarting IP during data communication, data is not guaranteed.

## 150.5 Safety Mechanisms

This module is treated as non-safety-related function in case of the products supporting functional safety.

# 154. Watchdog Timer (RCLKWDT, WindowWDT, SystemWDT, ...)

Timer Unit module is a module composed of RWDT, WWDT, and SWDT.

Refer to following section for the detailed function.

- RWDT:154.1 RCLK Watchdog Timer (RWDT)
- WWDT:154.2 Window Watchdog Timer (WWDT)
- SWDT:154.3 System Watchdog Timer (SWDT)

# 154.1 RCLK Watchdog Timer (RWDT)

## 154.1.1 Overview

This LSI includes the RCLK watchdog timer (RWDT).

This LSI can be reset by the overflow of the counter when the value of the counter has not been updated because of a system runaway.

The RWDT is a single-channel timer that uses the RCLK as an input and can be used as a watchdog timer. RCLK is a clock, which is generated by the clock pulse generator (CPG).

Frequency of RCLK: 32.8 kHz.

### 154.1.1.1 Features

- One channel is provided.
- Can be used as a watchdog timer. Reset is generated when the counter overflows.
- A counter input clock can be chosen from:
  Clocks (RCLK/1 to RCLK/ (128 × 60 × 30)) that are obtained by dividing the RCLK.

### 154.1.1.2 Block Diagram

Figure 154.1.1.1 shows block diagrams of the RWDT.

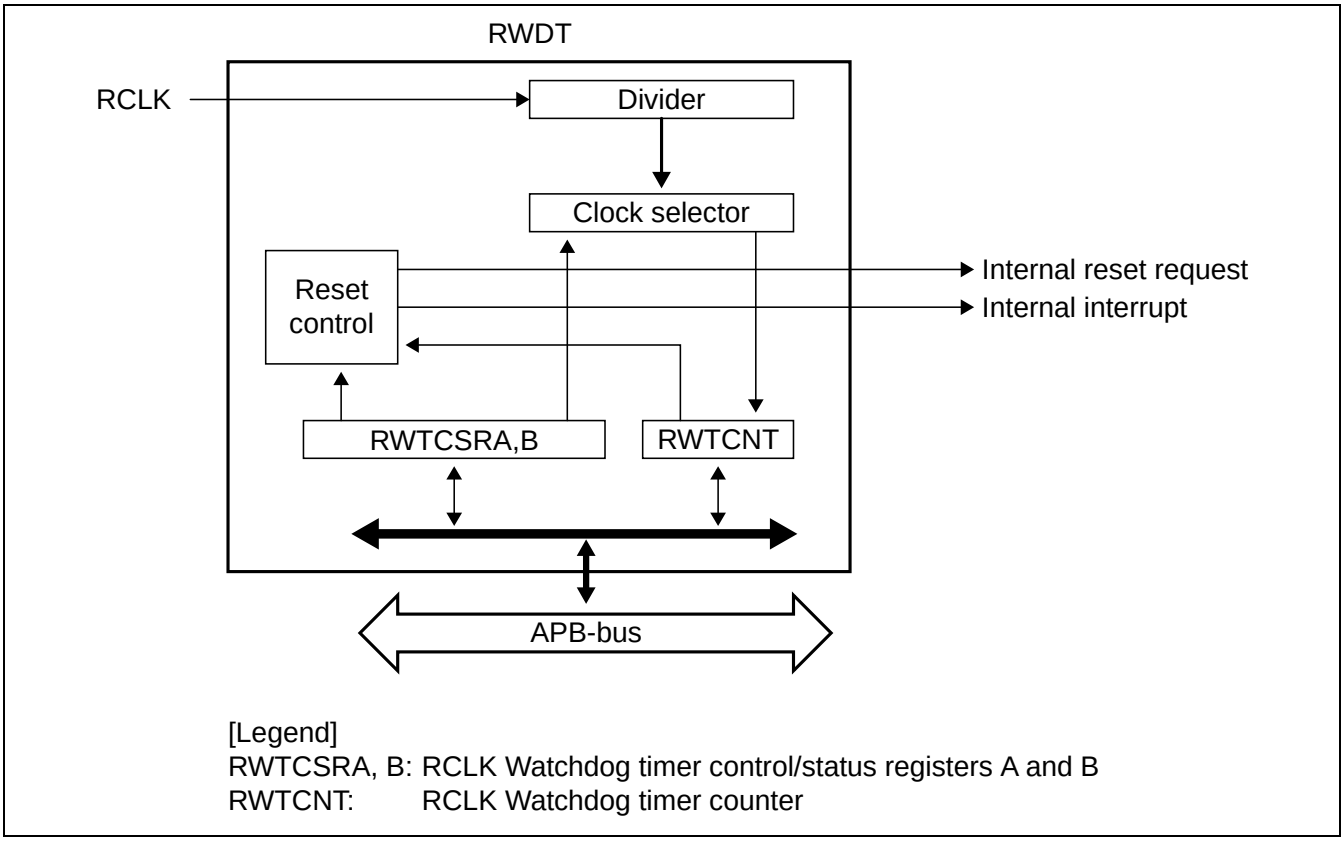



Figure 154.1.1.1 Block Diagram of RWDT

### 154.1.1.3 Register Configuration

Table 154.1.1.1 shows the RWDT register configuration. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined.
Table 154.1.1.2 shows the register state in each processing mode.

**Table 154.1.1.1 Register Configuration of RWDT**

| Name | Abbreviation | R/W | Address | Access Size |
|---|---|---|---|---|
| RCLK watchdog timer counter | RWTCNT | R/W | H'E602 0000 | 16/32*1 |
| RCLK watchdog timer control/status register A | RWTCSRA | R/W*3 | H'E602 0004 | 8/32*2 |
| RCLK watchdog timer control/status register B | RWTCSRB | R/W | H'E602 0008 | 8/32*2 |

Notes: 1. Write is performed in 32 bits and read in 16 bits. Operation cannot be guaranteed if the register is not written as a longword (32bits) and not read as a word (16bits).
2. Write is performed in 32 bits and read in 8 bits. Operation cannot be guaranteed if the register is not written as a longword (32bits) and not read as a byte (8bits).
3. Except for bit [4] that is R/WC0 register.

**Table 154.1.1.2 Register State of RWDT in Each Processing Mode**

| Register Abbreviation | | Reset Caused by Other than Overflow | Reset Caused by Overflow* |
|---|---|---|---|
| **Register** | **Bit** | | |
| RWTCNT | All | Initialized | Initialized |
| RWTCSRA | WOVF | Initialized | Retained |
| | Other than WOVF | Initialized | Initialized |
| RWTCSRB | All | Initialized | Initialized |

Note: * Section 154.3. System Watchdog Timer (SWDT) and RWDT overflow reset.

### 154.1.2 Register Description

[Legend for Register Description]

Initial value: Register value after a reset.

—: Undefined value

R/W: Bit or field is readable and writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

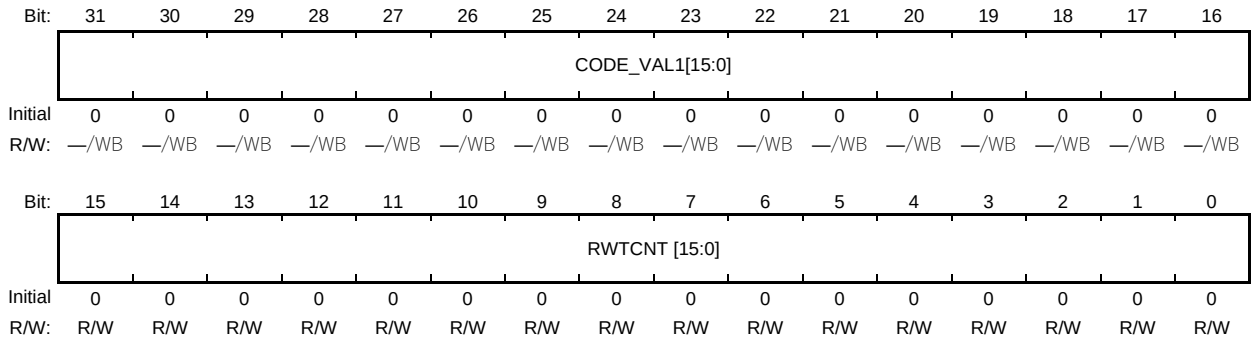
R: Read-only. The write value should always be initial value.

—/WB: Write-only. The read value is undefined.

#### 154.1.2.1 RCLK Watchdog Timer Counter (RWTCNT)

RWTCNT is a 16-bit readable/writable register that increments on the selected clock. When an overflow occurs, it generates a Reset. The RWTCNT counter is initialized to H'0000 by a power-on reset and RWDT software reset.

Use a long word access to write to the RWTCNT counter, with H'5A5A in the upper byte. Use a word access to read RWTCNT.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | CODE_VAL1 [15:0] | H’0000 | —/WB | Code value 1<br>When writing this register, write H’5A5A to this bit field.<br>When reading, these bits are always read as same as RWTCNT [15:0]. |
| 15 to 0 | RWTCNT [15:0] | H’0000 | R/W | Timer Counter Bits |

### 154.1.2.2 RCLK Watchdog Timer Control/Status Register A (RWTCSRA)

RWTCSRA is an 8-bit readable/writable register composed of bits to select the clock used for the count, overflow flag, and enable bit.

Use a long word access to write to RWTCSRA, with H'A5 A5A5 in the upper byte. Use a byte access to read RWTCSRA.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CODE_VAL2[23:8] | | | | | | | | | | | | | | | |
| Initial | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CODE_VAL2[7:0] | | | | | | | | TME | — | WRFLG | WOVF | WOVFE | CKS0[2:0] | | |
| Initial | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | R/W | R | R | R/WC0 | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | CODE_VAL2 [23:0] | H'0F 0F0F | —/WB | Code value 2<br>When writing this register, write H'A5 A5A5 to this bit field.<br>When reading, these bits are always read as {RWTCSRA [7:0], RWTCSRA [7:0], RWTCSRA [7:0]}. |
| 7 | TME | 0 | R/W | Starts and stops timer operation.<br>0: Timer disabled: Count-up stops and RWTCNT value is retained<br>1: Timer enabled |
| 6 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5 | WRFLG | 0 | R | Write Status Flag<br>When this bit is 1, write access to RWTCNT is prohibited. If it isn't maintained, operation isn't secured. This bit indicates the period that the writing to RWTCNT is masked for synchronization after the writing to RWTCNT Confirm that this flag is 0 when RWTCNT is written to continuously. |
| 4 | WOVF | 0 | R/WC0 | Indicates that the RWTCNT has overflowed. This bit isn't initialized by a generated reset by an overflow of RWDT and section 154.3. System Watchdog Timer (SWDT). Write 0 to this bit before using the RWDT.<br>0: No overflow<br>1: RWTCNT has overflowed<br>Note: Only 0 can be written to clear the flag. Writing 1 to this bit is ignored. |
| 3 | WOVFE | 1 | R/W | Overflow Interrupt Disable/Enable<br>0: Disables interrupts due to overflow<br>1: Enables interrupts due to overflow |

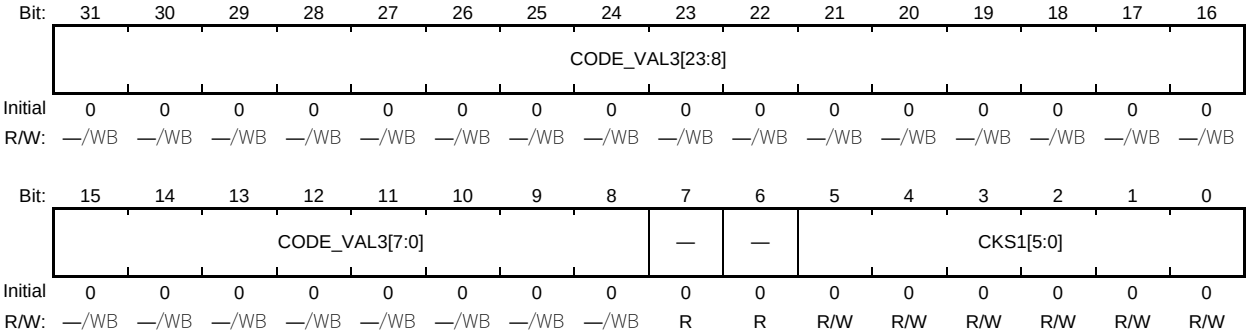
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 to 0 | CKS0[2:0] | 111 | R/W | RTC Clock Select<br>These bits select the clock to be used for the RWTCNT count from the eight types obtainable by dividing the RCLK. The overflow period that is shown inside the parenthesis in the table is the approximate value when the RCLK is 32.8 kHz.<br>000: RCLK (RWTCNT: H'0000 = 1.9 s (RWTCNT: H'FF00 = 7.7 ms))<br>001: RCLK/4 (RWTCNT: H'0000 = 7.9 s (RWTCNT: H'FF00 = 31.1 ms))<br>010: RCLK/16 (RWTCNT: H'0000 = 31.9 s (RWTCNT: H'FF00 = 124.5 ms))<br>011: RCLK/32 (RWTCNT: H'0000 = 63.9 s (RWTCNT: H'FF00 = 249.0 ms))<br>100: RCLK/64 (RWTCNT: H'0000 = 127.8 s (RWTCNT: H'FF00 = 498.0 ms))<br>101: RCLK/128 (RWTCNT: H'0000 = 255.7 s (RWTCNT: H'FF00 = 0.9 s))<br>110: RCLK/1024 (RWTCNT: H'0000 = 2046.0 s (RWTCNT: H'FF00 = 7.9 s))<br>111: RCLK select expanded mode<br>The clock cycle varies depending on the CKS1 bit in RWTCSRB. If RWTCSRB.CKS1[3:0] is H'0, division ratio is RCLK/4096 and values of overflow time is RWTCNT: H'0000 = 8192.0 s (RWTCNT: H'FF00 = 32.0 s). If RWTCSRB.CKS1[3:0] is other than H'0, values of overflow time is calculated with formula in chapter 154.1.3.3.<br>Note. These overflow periods are calculated from the rounded value of RCLK = 32.8 kHz as example. |

Note: If bits CKS0 [2:0] are modified when the RWDT is operating, the up-count may not be performed correctly. Ensure that the bits CKS0 [2:0] are modified only when the RWDT is not operating.

### 154.1.2.3 RCLK Watchdog Timer Control/Status Register B (RWTCSRB)

RWTCSRB is an 8-bit readable/writable register composed of bits to select the clock used for the count.

Use a word access to write to RWTCSRB, with H'A5 A5A5 in the upper byte. Use a byte access to read RWTCSRB.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | CODE_ VAL3[23:0] | H’00 0000 | — /WB | Code value 3<br>When writing this register, write H’A5 A5A5 to this bit field.<br>When reading, these bits are always read as {RWTCSRB [7:0], RWTCSRB [7:0], RWTCSRB [7:0]}. |
| 7, 6 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | CKS1[5:0] | All 0 | R/W | RCLK Select for RCLK Select Expanded Mode<br>Selects the clock used for the RWTCNT count when the CKS0 bit in RWTCSRA is B'111.<br>Table 154.1.2.1 shows the time values (approximate value) used to cause an overflow from the counter H'0000(Overflow time from counter H'FF00). Assuming the overflow cycle in the table is calculated for the case when the RCLK is 32.8 kHz. For more values of overflow time, refer chapter 154.1.3.3. |

Note: If the CKS1 [5] to CKS1 [0] bits are modified when the RWDT is running, the up-count may not be performed correctly. Ensure that these bits are modified only when the RWDT is not running.

Table 154.1.2.1 Example of Overflow time

| CKS1[3:0] \ CKS1[5:4] | B'00 | B'01 | B'10 | B'11 |
|---|---|---|---|---|
| H'0 | 8184.0s (31.8 s) | | | |
| H'1 | 255.7 s<br>(0.9 s) | 1278.7 s<br>(4.9 s) | 255.7 min<br>(0.9 min) | 1278.7 min<br>(4.9 min) |
| H'2 | 511.5 s<br>(1.9 s) | 2557.5 s<br>(9.9 s) | 511.5 min<br>(1.9 min) | 2557.5 min<br>(9.9 min) |
| H'3 | 767.2 s<br>(2.9 s) | 3836.2 s<br>(14.9 s) | 767.2 min<br>(2.9 min) | 3836.2 min<br>(14.9 min) |
| H'4 | 1023.0 s<br>(3.9 s) | 5115.0 s<br>(19.9 s) | 1023.0 min<br>(3.9 min) | 5115.0 min<br>(19.9 min) |
| H'5 | 1278.7 s<br>(4.9 s) | 6393.7 s<br>(24.8 s) | 1278.7 min<br>(4.9 min) | 6393.7 min<br>(24.8 min) |
| H'6 | 1534.5 s<br>(5.9 s) | 7672.5 s<br>(29.8 s) | 1534.5 min<br>(5.9 min) | 7672.5 min<br>(29.8 min) |
| H'7 | 1790.2 s<br>(6.9 s) | 8951.2 s<br>(34.8 s) | 1790.2 min<br>(6.9 min) | Prohibited<br>(Prohibited) |
| H'8 | 2046.0 s<br>(7.9 s) | 10230.0 s<br>(39.8 s) | 2046.0 min<br>(7.9 min) | Prohibited<br>(Prohibited) |
| H'9 | 2301.7 s<br>(8.9 s) | 11508.8 s<br>(44.7 s) | 2301.7 min<br>(8.9 min) | Prohibited<br>(Prohibited) |
| H'A | 2557.5 s<br>(9.9 s) | 12787.5 s<br>(49.7 s) | 2557.5 min<br>(9.9 min) | Prohibited<br>(Prohibited) |
| H'B | Prohibited<br>(Prohibited) | 14066.3 s<br>(54.7 s) | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) |
| H'C | Prohibited<br>(Prohibited) | 15345.0 s<br>(59.7 s) | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) |
| from H'D | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) |

Note. These overflow periods are calculated from the rounded value of RCLK = 32.8 kHz as example.

### 154.1.3 Operation

#### 154.1.3.1 Control of System Runaway

After a reset, RWDT is disabled. When the counter overflow occurs after the counter starts, an internal reset is again generated. By this function, a reset can be automatically generated even when this LSI has caused a system runaway. While this LSI is operating correctly, write H'0000 (or the set value) to RWTCNT periodically so that RWTCNT does not overflow.

Take the following steps to use the RWDT.

1. Clear the TME bit in RWTCSRA to 0 to temporarily stop counting.
   (Confirm that RCLK was input more than 2 cycles in case RWTCNT is not initialized)
2. Write H'0000 or the set value in RWTCNT
3. Clear the WOVF bit in RWTCSRA to 0.
4. Set the kind of count clock to the CKS0 [2:0] bits in RWTCSRA, and CKS1 [5:0] bits in RWTCSRB.
5. Confirm that RWTCSRA.WRFLG becomes 0.
   If WRFLG is 1, wait until it'll be 0.
6. Start the counting by setting the TME bit in RWTCSRA to 1.
   (TME and CKS0 can't be set at the same time. When setting that at the same time, behavior isn't guaranteed.)
7. While this LSI is operating correctly, write H'0000 (or the set value) to RWTCNT periodically so that RWTCNT does not overflow.
8. When RWTCNT overflows, Reset is generated because the RWDT sets the WOVF flag in RWTCSRA to 1. At this time, RWTCNT, RWTCSRA (excluding the WOVF bit), and RWTCSRB are initialized.

Execute a power-on reset or a software reset at that time and initialize RWDT.

When not initializing and using RWDT, behavior isn't guaranteed.

#### 154.1.3.2 Module Stop

A setup of a stop of the Clock by a Module Stop signal is possible when RWDT is an idle state. The Idle state of RWDT is when the condition of following all was satisfied.

- When RWTCSRA.TME bit is 0
- When RWTCSRA.WRFLG is 0
- When 3 or more cycles have passed in RCLK since the last Register access

#### 154.1.3.3 Formula to calculate overflow time

This is the formula to calculate the overflow time for RWDT when choosing Expanded Mode.

Overflow time = 128 × T_RCLK × opset_val × (65536 – RWTCNT)

T_RCLK: is the period of RCLK clock.

RWTCNT: is value of RWTCNT in decade.

opset_val: refer to Table 154.1.3.1.

Table 154.1.3.1 Value of opset_val

| CKS1[5:0] | opset_val | CKS1[5:0] | opset_val | CKS1[5:0] | opset_val | CKS1[5:0] | opset_val |
|---|---|---|---|---|---|---|---|
| H'00 | 32 | H'10 | 32 | H'20 | 32 | H'30 | 32 |
| H'01 | 1 | H'11 | 5 | H'21 | 60 | H'31 | 300 |
| H'02 | 2 | H'12 | 10 | H'22 | 120 | H'32 | 600 |
| H'03 | 3 | H'13 | 15 | H'23 | 180 | H'33 | 900 |
| H'04 | 4 | H'14 | 20 | H'24 | 240 | H'34 | 1200 |
| H'05 | 5 | H'15 | 25 | H'25 | 300 | H'35 | 1500 |
| H'06 | 6 | H'16 | 30 | H'26 | 360 | H'36 | 1800 |
| H'07 | 7 | H'17 | 35 | H'27 | 420 | Other case | 1 |
| H'08 | 8 | H'18 | 40 | H'28 | 480 | | |
| H'09 | 9 | H'19 | 45 | H'29 | 540 | | |
| H'0A | 10 | H'1A | 50 | H'2A | 600 | | |
| Other case | 1 | H'1B | 55 | Other case | 1 | | |
| | | H'1C | 60 | | | | |
| | | Other case | 1 | | | | |

#### 154.1.3.4 Notes on Register Access

RWTCSRA: Bits TME, CKS0

RWTCSRB: Bits CKS1

After writing a value in this register, it is possible to read that value 2 cycles in counter input clock (RCLK) after writing has finished. And 2 cycles of RCLK need to be inputted until the set-up value is reflected.

RWTCNT: Bits 15 to 0

After writing a value in this register, it is possible to read that value 2 cycles in counter input clock (RCLK) after writing has finished. And 2 cycles of RCLK need to be inputted until the set-up value is reflected. Writing to this register, once started, is protected from further write operations until the writing operation has been completed. Read RWTCSRA. If WRFLG is 0, the next value can be written in RWTCNT.

The writing procedure to RWTCNT, RWTCSRA, and RWTCSRB differs from that of other registers with the purpose of preventing an unintended write. The procedure for writing to these registers is given below.

**Writing to RWTCNT, RWTCSRA, and RWTCSRB:**

- These registers must be written by a long word transfer instruction. They cannot be written by a byte or word transfer instruction.
- When writing to RWTCNT, set the upper byte to H'5A5A and transfer the lower byte as the write data. When writing to RWTCSRA and RWTCSRB, set the upper byte to H'A5A5A5 and transfer the lower byte as the write data.

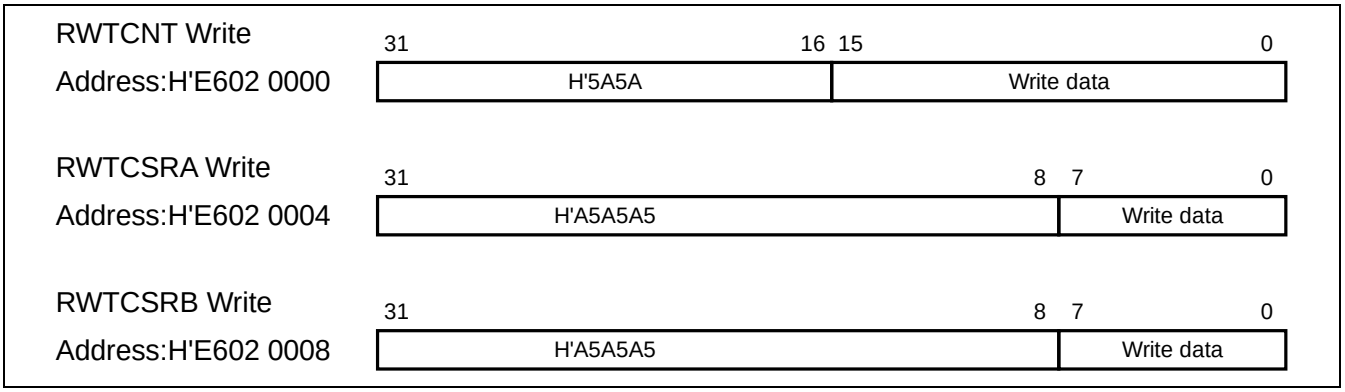



Figure 154.1.3.1 Writing to RWTCNT, RWTCSRA, and RWTCSRB

### 154.1.4 Usage Notes

No description on this section.

### 154.1.5 Safety Mechanisms

#### 154.1.5.1 Power-On test

This module is protected by Power-On test.

# 154.2 Window Watchdog Timer (WWDT)

### 154.2.1 Overview

The Window Watchdog Timer (hereinafter referred to as the "WWDT") is a module of Watchdog timer with window that makes monitoring-period by CPU varying.

#### 154.2.1.1 Features

The WWDT has the following features.

- Window watchdog function;
  - — When counter reach overflow value then module will detect error.
  - — When the start trigger register is written during the window is close then module will detect error.
  - — When counter reach 75% value of overflow value then module will create 75% interrupt.
- Generates Reset or NMI on error detection.
- Support Software trigger mode.
- 7 channels.

### 154.2.1.2 Block Diagram

Figure 154.2.1.1 is a block diagram of the WWDT.

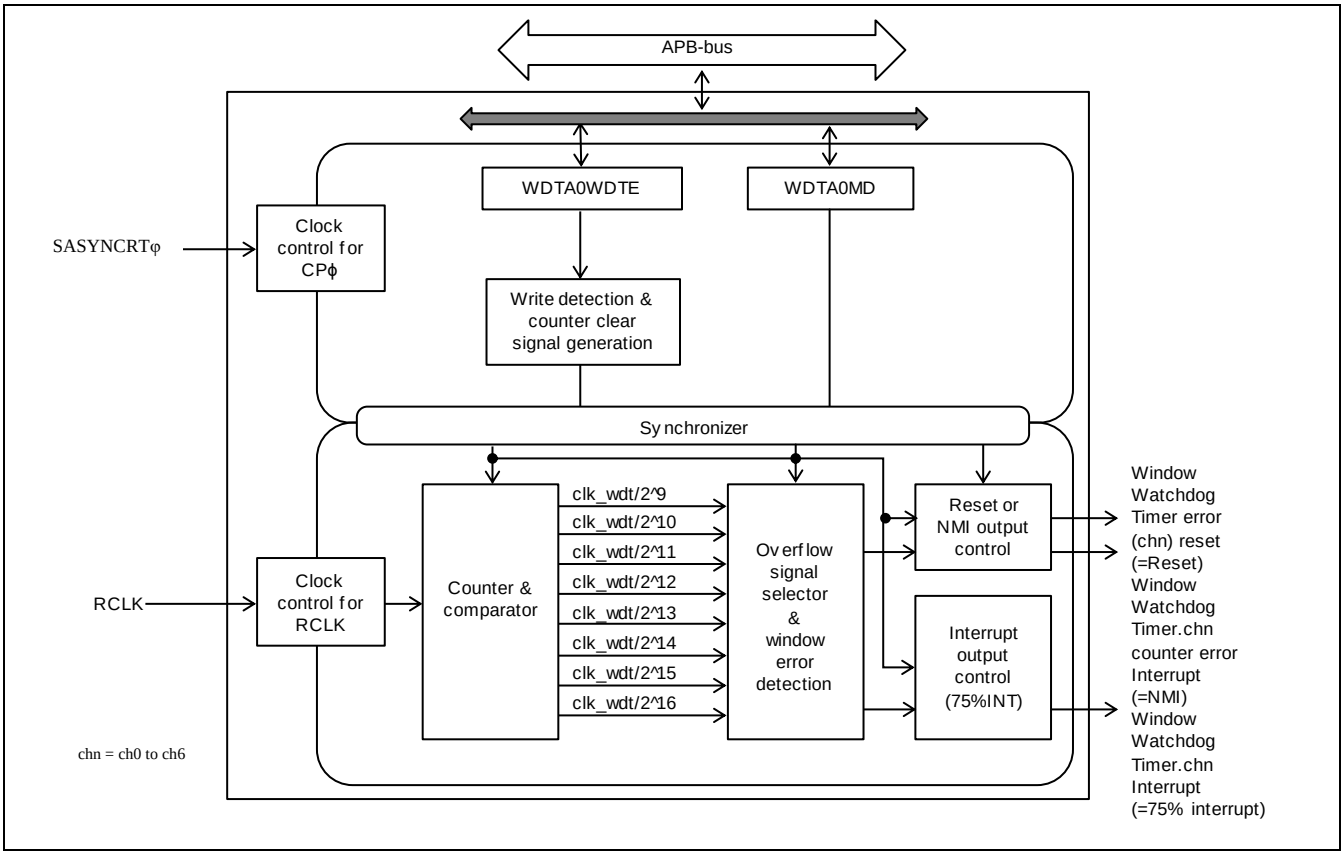



Figure 154.2.1.1 Block Diagram of the WWDT

### 154.2.1.3 Register Configuration

Table 154.2.1.1 shows the WWDT register configuration. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined. Access the register as a byte (8 bits). Operation cannot be guaranteed if the register is not accessed as a byte.

Table 154.2.1.1 Register Configuration

| Channel n<br>n = 0 to 6 | Name | Abbreviation | R/W | Address | Size | Reset |
|---|---|---|---|---|---|---|
| 0 | Watchdog Timer Enable register 0 | WDTA0WDTE0 | R/W | H'FFC9 0000 | 8 | H'2C |
| | Watchdog Timer Mode register 0 | WDTA0MD0 | R/W | H'FFC9 000C | 8 | H'04 |
| 1 | Watchdog Timer Enable register 1 | WDTA0WDTE1 | R/W | H'FFCA 0000 | 8 | H'2C |
| | Watchdog Timer Mode register 1 | WDTA0MD1 | R/W | H'FFCA 000C | 8 | H'04 |
| 2 | Watchdog Timer Enable register 2 | WDTA0WDTE2 | R/W | H'FFCB 0000 | 8 | H'2C |
| | Watchdog Timer Mode register 2 | WDTA0MD2 | R/W | H'FFCB 000C | 8 | H'04 |
| 3 | Watchdog Timer Enable register 3 | WDTA0WDTE3 | R/W | H'FFCC 0000 | 8 | H'2C |
| | Watchdog Timer Mode register 3 | WDTA0MD3 | R/W | H'FFCC 000C | 8 | H'04 |
| 4 | Watchdog Timer Enable register 4 | WDTA0WDTE4 | R/W | H'FFCF 0000 | 8 | H'2C |
| | Watchdog Timer Mode register 4 | WDTA0MD4 | R/W | H'FFCF 000C | 8 | H'04 |
| 5 | Watchdog Timer Enable register 5 | WDTA0WDTE5 | R/W | H'FFEF 0000 | 8 | H'2C |
| | Watchdog Timer Mode register 5 | WDTA0MD5 | R/W | H'FFEF 000C | 8 | H'04 |
| 6 | Watchdog Timer Enable register 6 | WDTA0WDTE6 | R/W | H'FFF1 0000 | 8 | H'2C |
| | Watchdog Timer Mode register 6 | WDTA0MD6 | R/W | H'FFF1 000C | 8 | H'04 |

#### 154.2.1.4 Connected Module

Table 154.2.1.2 Connected modules

| Module name | Connected module name | Function of connected module |
|---|---|---|
| WWDT | CPG | Supply clock and reset |
| | INTC-SYS, INTC-RT | Control to interrupt |
| | MFIS | Control error happen |

### 154.2.2 Register Description

[Legend for Register Description]

Initial value: Register value after a reset.

—: Undefined value

R/W: Bit or field is readable and writable. The written value can be read.

R: Read-only. The write value should always be initial value.

All registers are accessed in byte units.

#### 154.2.2.1 Watchdog Timer Enable Register (WDTA0WDTEn)

The register is a trigger register.

To clear (= trigger) the counter, 0xAC must be written on this register. Writing wrong value or writing while window is window closed leads to error detection.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WDTA0RUN | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | WDTA0RUN | 0 | R/W | Watchdog timer count start or stop.<br>WDTA0RUN bit is the set only register and it can only be cleared by PRESETZ.<br>When "1" is written on WDTA0RUN, the watchdog timer starts counting. |
| 6 to 0 | — | H'2C | R | Reserved |

### 154.2.2.2 Watchdog Timer Mode register (WDTA0MDn)

This register is timer mode register.

This register can be updated only once after any resets and before first write to trigger register.

The updated new value will be effective from next period (= when the counter value becomes H'0000).

Updating this register after the first writing of trigger leads to error detection.

Updating this register more than once leads to error detection.

The new or initial value of WDTA0MD applies after the first trigger.

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | WDTA0OVF2 | WDTA0OVF1 | WDTA0OVF0 | WDTA0WIE | WDTA0ERM | WDTA0WS1 | WDTA0WS0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 | — | 0 | R | Reserved |
| 6 | WDTA0OVF2 | 0 | R/W | Interval time mode<br>Please refer to Table 154.2.2.2 for details. |
| 5 | WDTA0OVF1 | 0 | R/W | Interval time mode<br>Please refer to Table 154.2.2.2 for details. |
| 4 | WDTA0OVF0 | 0 | R/W | Interval time mode<br>Please refer to Table 154.2.2.2 for details. |
| 3 | WDTA0WIE | 0 | R/W | 75% interrupt enable<br>0: 75% interrupt is disabled<br>1: 75% interrupt is enabled |
| 2 | WDTA0ERM | 1 | R/W | Error mode<br>0: NMI mode<br>1: Reset mode (Default) |
| 1 | WDTA0WS1 | 0 | R/W | Window Size selection<br>Please refer to Figure 154.2.2.1 and Table 154.2.2.1 for details. |
| 0 | WDTA0WS0 | 0 | R/W | Window Size selection<br>Please refer to Figure 154.2.2.1 and Table 154.2.2.1 for details. |

Note: 3 cycles are necessary with SASYNCRTφ before initial value is reflected on each bit after reset release. It must be designed in a way the register update timing and timer counter clear timing, which is the timing the register value is loaded, will not occur at the same time to avoid inconsistency.

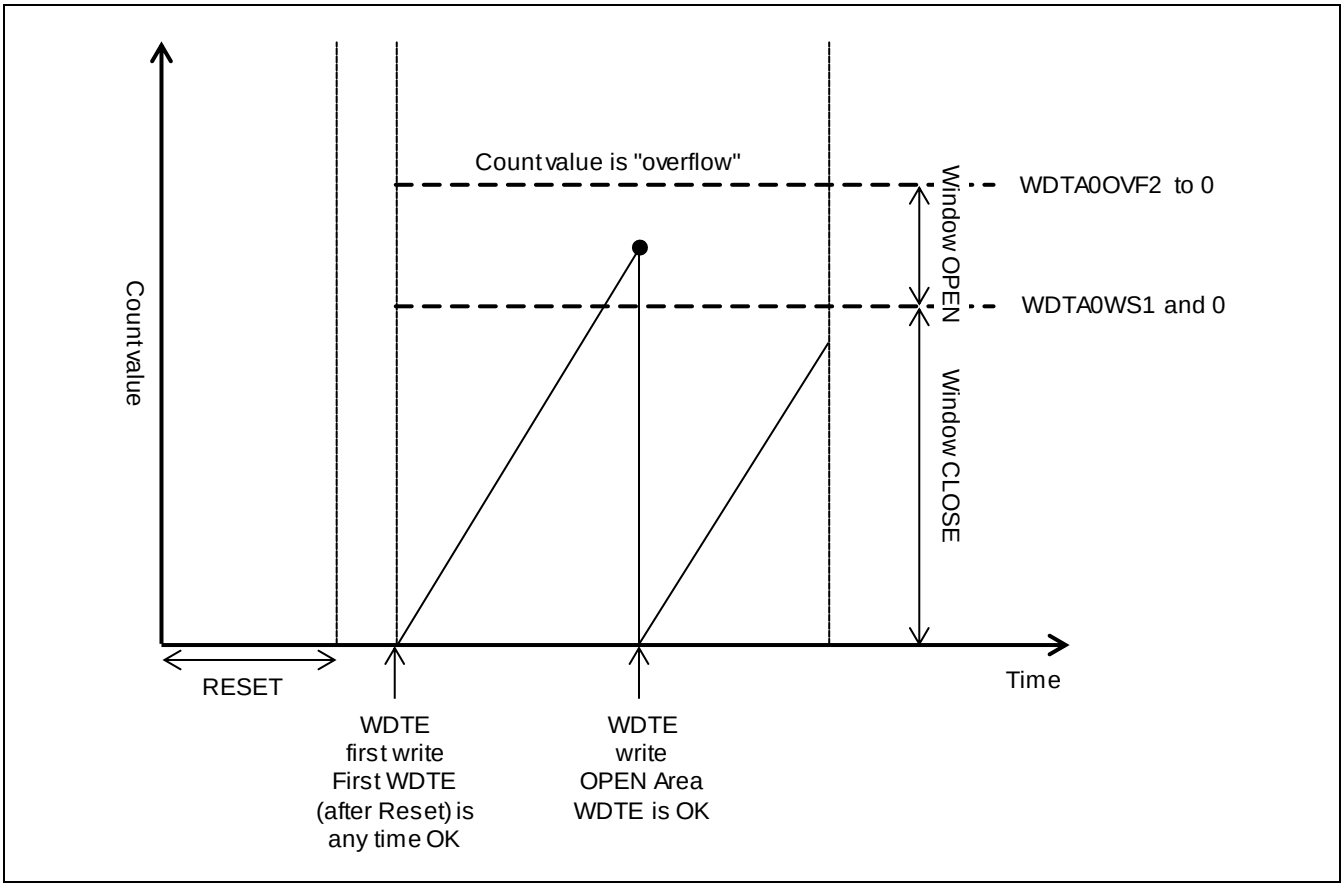



Figure 154.2.2.1 Window Size Selection

Table 154.2.2.1 **Window Size selection**

| WDTA0WS1 | WDTA0WS0 | Window Size selection |
|---|---|---|
| 0 | 0 | 25% |
| 0 | 1 | 50% |
| 1 | 0 | 75% |
| 1 | 1 | 100% |

Window Size selection is ration Window OPEN/ (Window OPEN + Window CLOSE).

Table 154.2.2.2 Interval Time Mode

| WDTA0OVF2 | WDTA0OVF1 | WDTA0OVF0 | Interval time |
|---|---|---|---|
| 0 | 0 | 0 | $2^9$/clk_wdt |
| 0 | 0 | 1 | $2^{10}$/clk_wdt |
| 0 | 1 | 0 | $2^{11}$/clk_wdt |
| 0 | 1 | 1 | $2^{12}$/clk_wdt |
| 1 | 0 | 0 | $2^{13}$/clk_wdt |
| 1 | 0 | 1 | $2^{14}$/clk_wdt |
| 1 | 1 | 0 | $2^{15}$/clk_wdt |
| 1 | 1 | 1 | $2^{16}$/clk_wdt |

Note: "clk_wdt" is watchdog timer operation clock which is input from RCLK. In case that input clock frequency is 32.8kHz and if the setting of overflow is set to clk_wdt/$2^9$, the overflow time is
32.8kHz / $2^9$ ➔ about 64.0Hz ➔ about 15.6msec

### 154.2.3 Operation

#### 154.2.3.1 Operation After Reset Timing

**(1) Software Trigger Start mode**

WWDT is reset by Power-On Reset For Software Trigger Start mode, WWDT is stopped with reset conditions after reset release.

**(2) Operation parameter after reset release**

After reset release, the operation parameter becomes the following set regardless of software trigger start mode.

- Overflow interval time: clk_wdt / $2^9$
- Window period size: 100%
- 75% Interruption disabled
- Software trigger mode

#### 154.2.3.2 Window watchdog function

Window open size is defined by WDTA0MD.WDTA0WS1-0, and writing the start trigger register outside the period results in error detection.

However, the window open size is 100% right after the reset release to enable software to make initial trigger without waiting the window to open.

Window at software trigger start mode

- Initial trigger of the counter can be done anytime.
- The triggering afterwards must be done during the window is open.

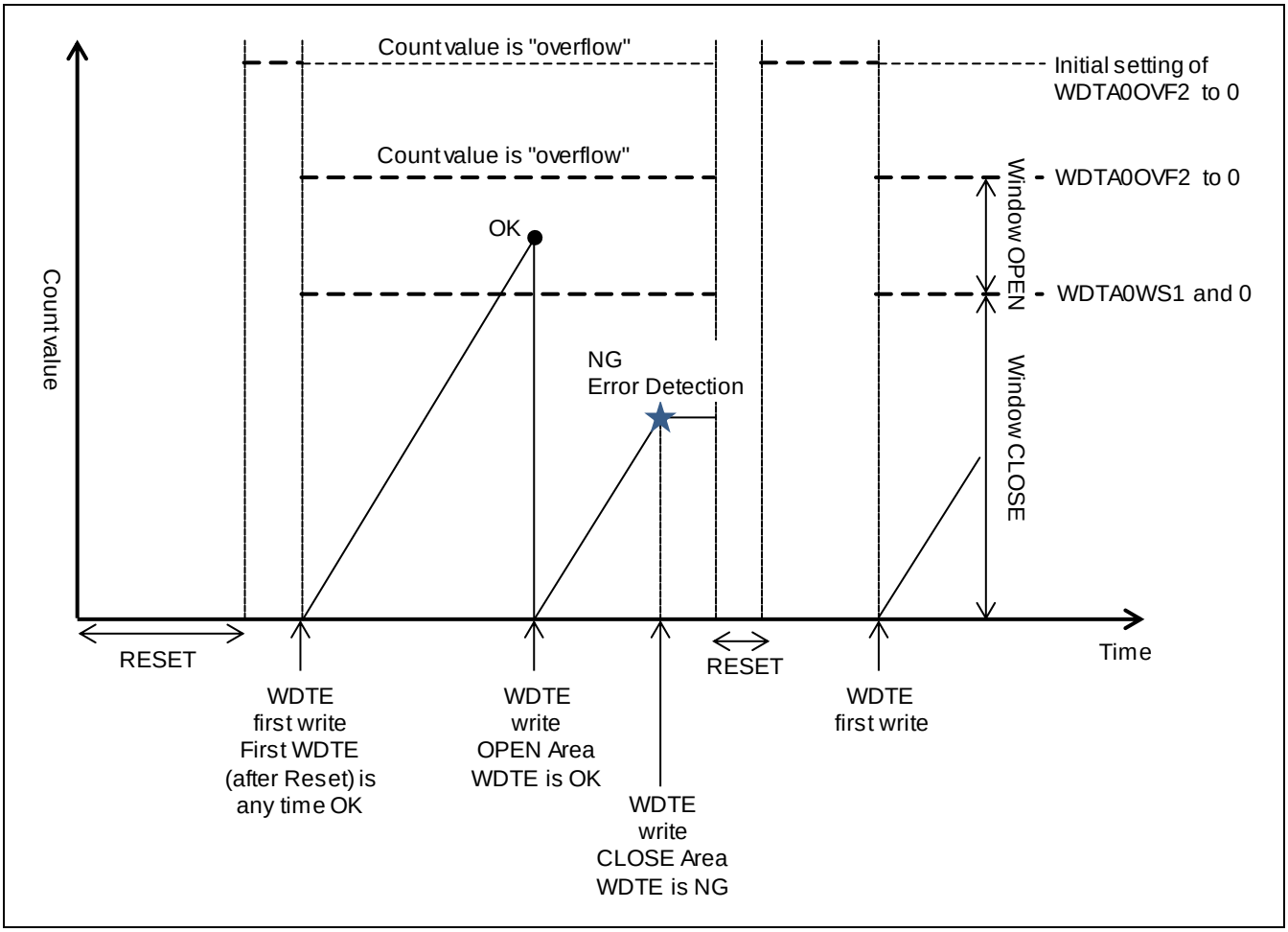



Figure 154.2.3.1 Window watchdog function

### 154.2.3.3 75% interrupt output

When the counter reaches 75% of the overflow setting defined by WDTA0MD.WDT0OVF2-0, 75% interrupt is generated with the pulse width of 1CLK of RCLK. This function is effective if WDTA0WIE bit is 1.

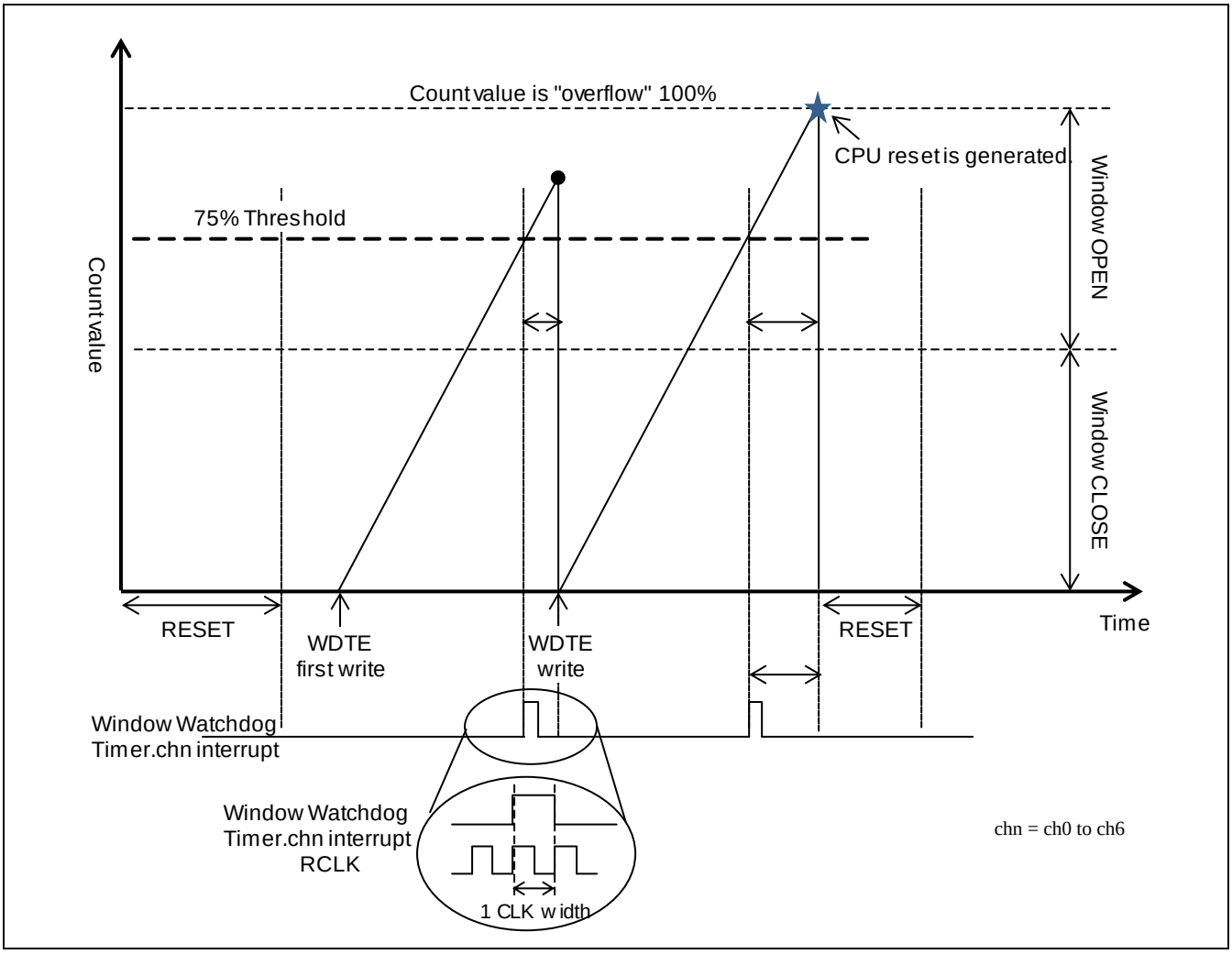



Figure 154.2.3.2 75% interrupt

### 154.2.3.4 NMI or Reset interrupt output

When this module detects error, either NMI or Reset is output with the pulse width of 1 RCLK. The selection of NMI or Reset is done by WDTA0MD.WDTA0ERM.

The counter stops upon error detection and can be restarted only by reset (Power-On Reset).

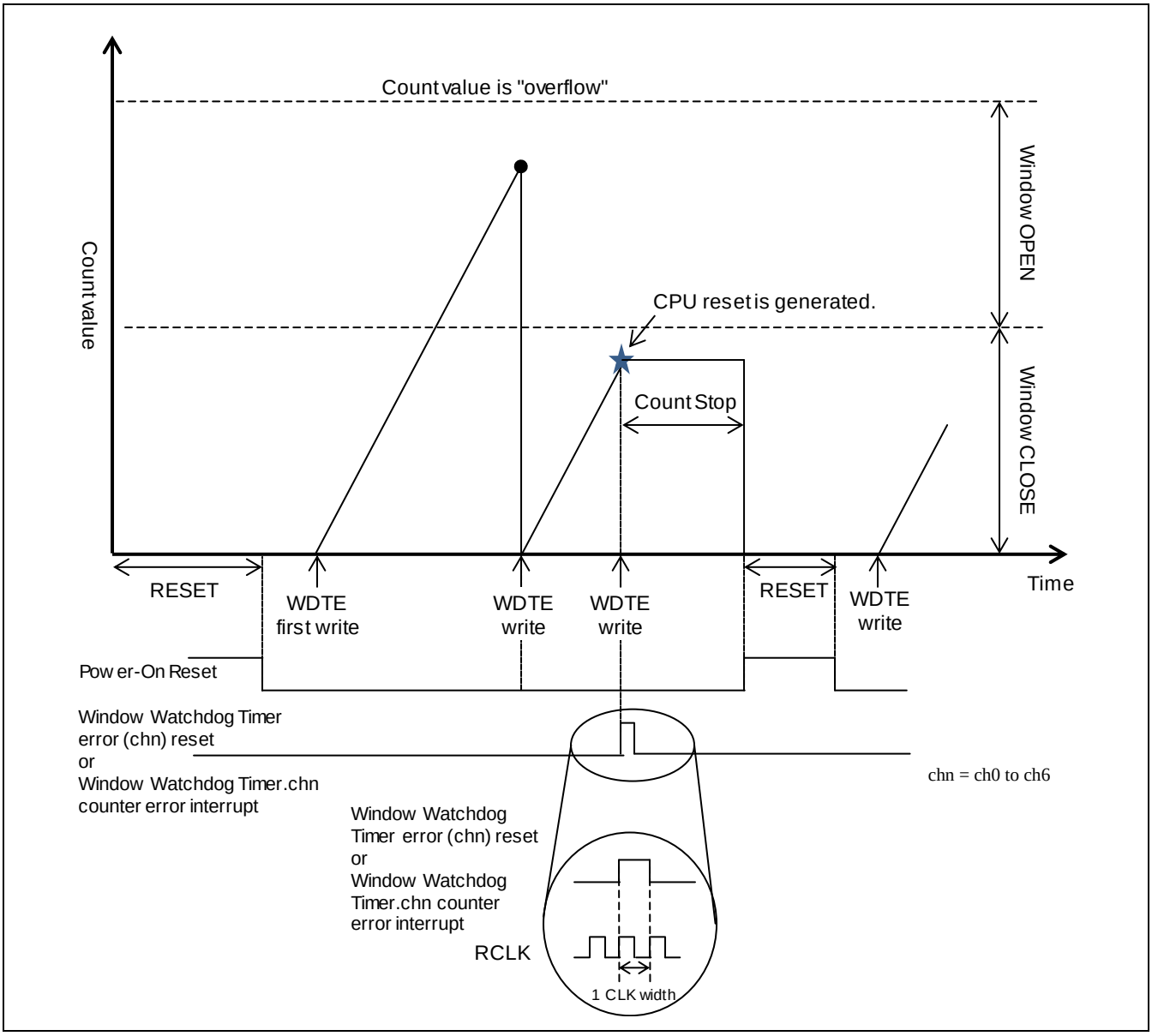



Figure 154.2.3.3 Detect error by NMI and Reset interrupt

### 154.2.3.5 Error Detection Condition

The conditions that this module regards as errors are described below.

**(1) Timer counter overflow**

An error is detected if timer counter reaches the overflow value defined by WDTA0MD.WDTA0OVF2-0.

**(2) Writing wrong trigger code on trigger register**

An error is detected if value other than H'AC is written on WDTA0WDTE register.

**(3) Writing trigger register when window is closed**

An error is detected if triggering is done while the window is closed.

Note: window size right after reset release is 100%.

**(4) Irregular update of WDTA0MD register**

This register must be updated only with the following conditions. An error is detected if violated.

— Updating before the first triggering is required.

— Updating more than once is only arrowed if the written value is same as the old value.

Following Table 154.2.3.1 describes irregular update of WDTA0MD.

**Table 154.2.3.1 Irregular update of WDTA0MD register**

| Condition | | | Error detection |
|---|---|---|---|
| **Update #** | **Write Before/After initial triggering** | **Write value** | |
| Initial update | Before | Same | — |
| | | New | — |
| | After | Same | — |
| | | New | Detect |
| Second and afterwards | Before | Same | — |
| | | New | Detect |
| | After | Same | — |
| | | New | Detect |

### 154.2.3.6 Setting Flow

H'AC must always be written as a trigger code on WDTA0WDTE.

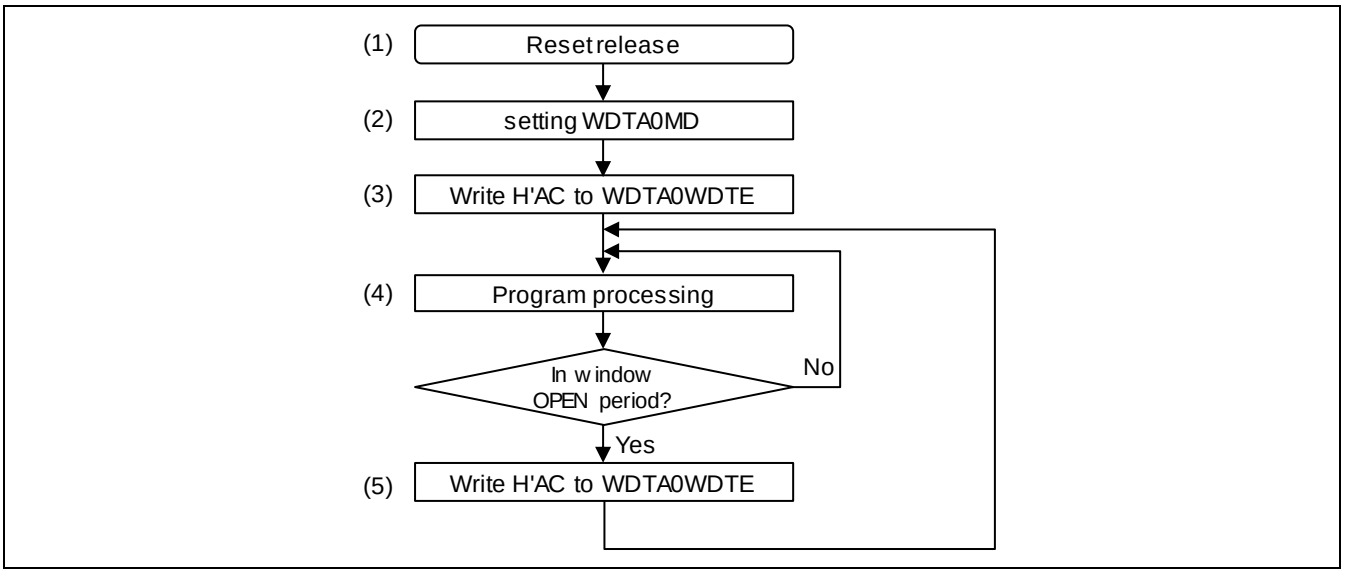



Figure 154.2.3.4 Setting flow WWDT module

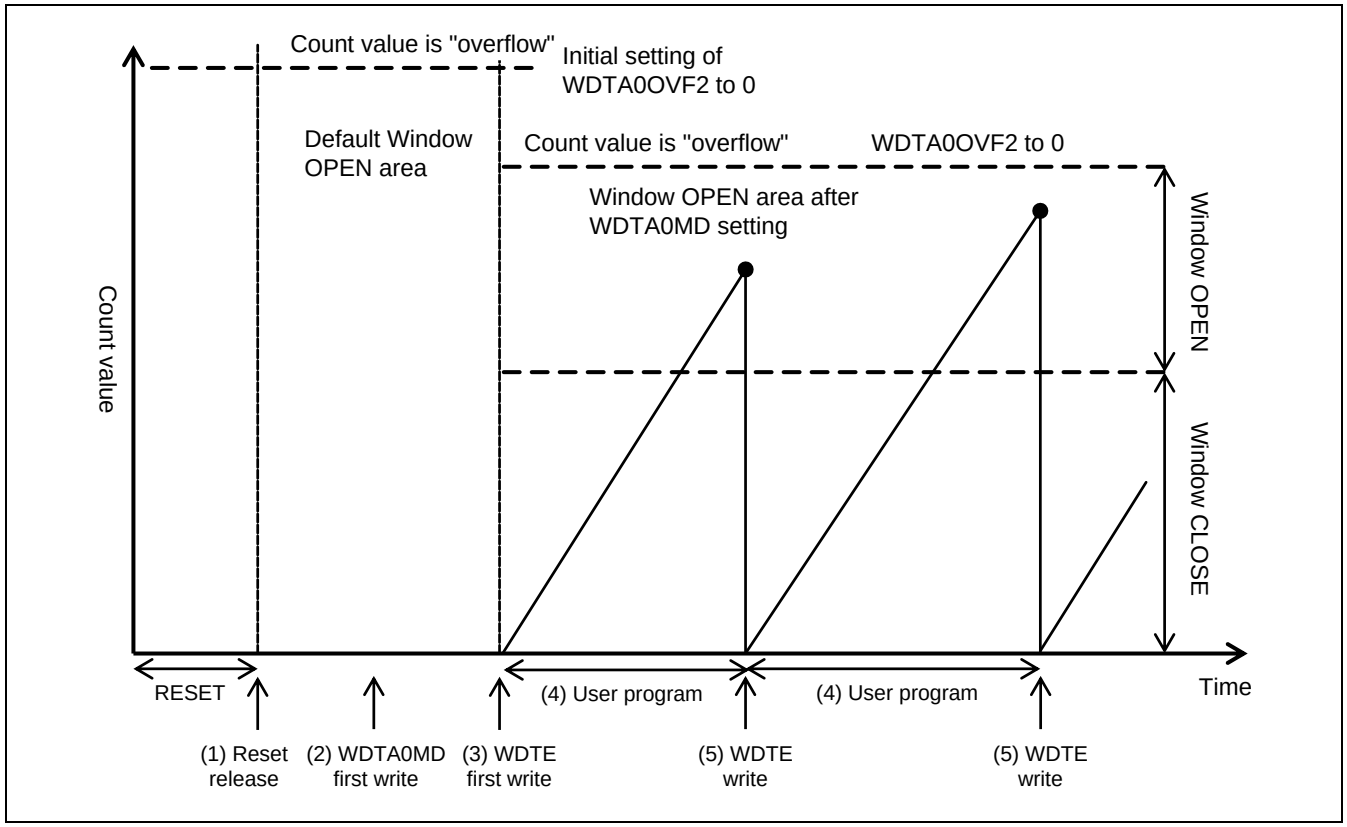



Figure 154.2.3.5 Steps setting flow

### 154.2.4 Usages Notes

#### 154.2.4.1 Timer counter start timing

The gap is caused from the completion of writing in the start trigger register to beginning the movement of the timer counter to operate with clock (RCLK) of the asynchronous system in the timer counter of this module.

The maximum of this gap is 3 RCLK cycle.

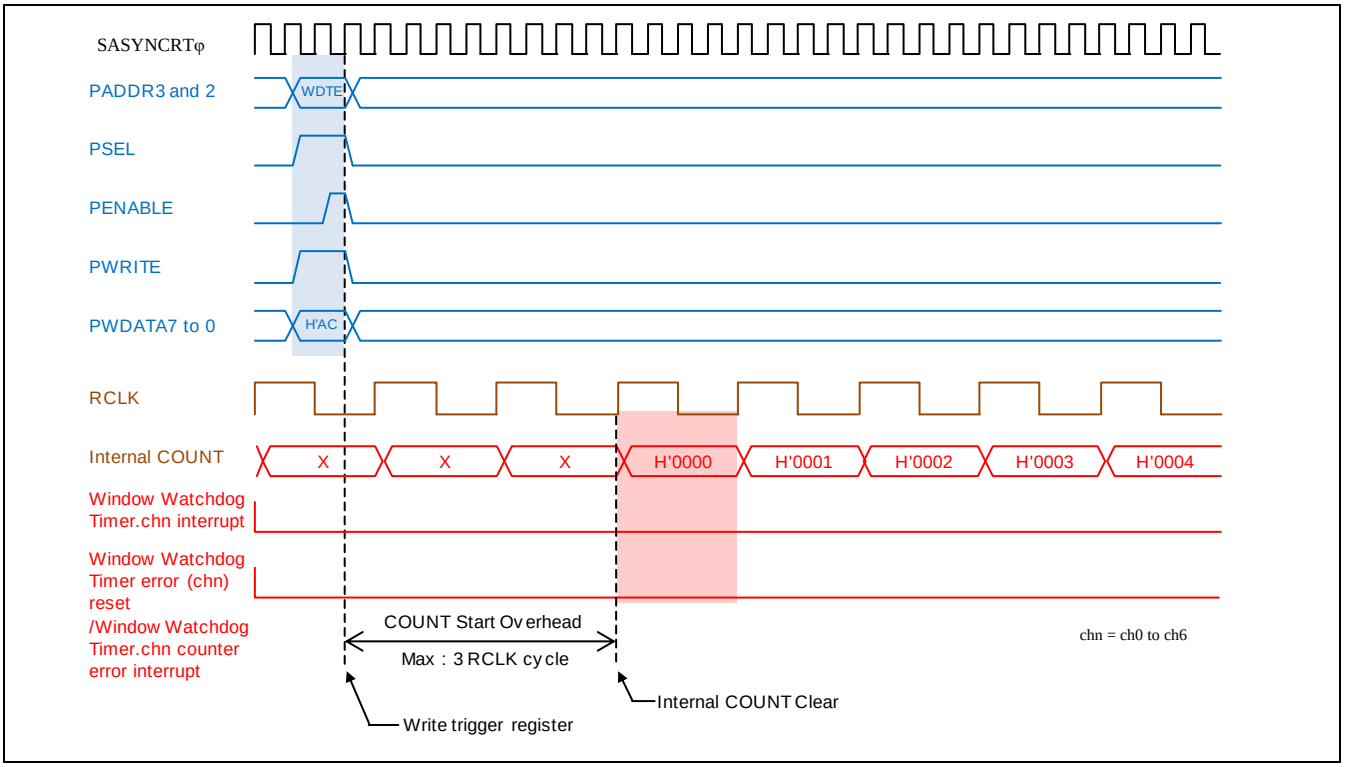



Figure 154.2.4.1 Timer counter start timing

### 154.2.4.2 75% interrupt output timing

In this module, after the period 75% of overflow setting value (WDTA0OVF), 75% interrupt output is output from the writing completion of the start trigger register.

But one RCLK cycle is necessary for output 75% interrupt after a timer counter reached 75%. Moreover, the gap is caused in the output timing of 75% interrupt for the gap of the timer counter by the asynchronous system.

Therefore, even when the window size is set to 75% (WDTA0WS = 10), the period when writing in the start trigger register is permitted is generated.

It is 3 RCLK cycle before the 75% interrupt start at the maximum for this period.

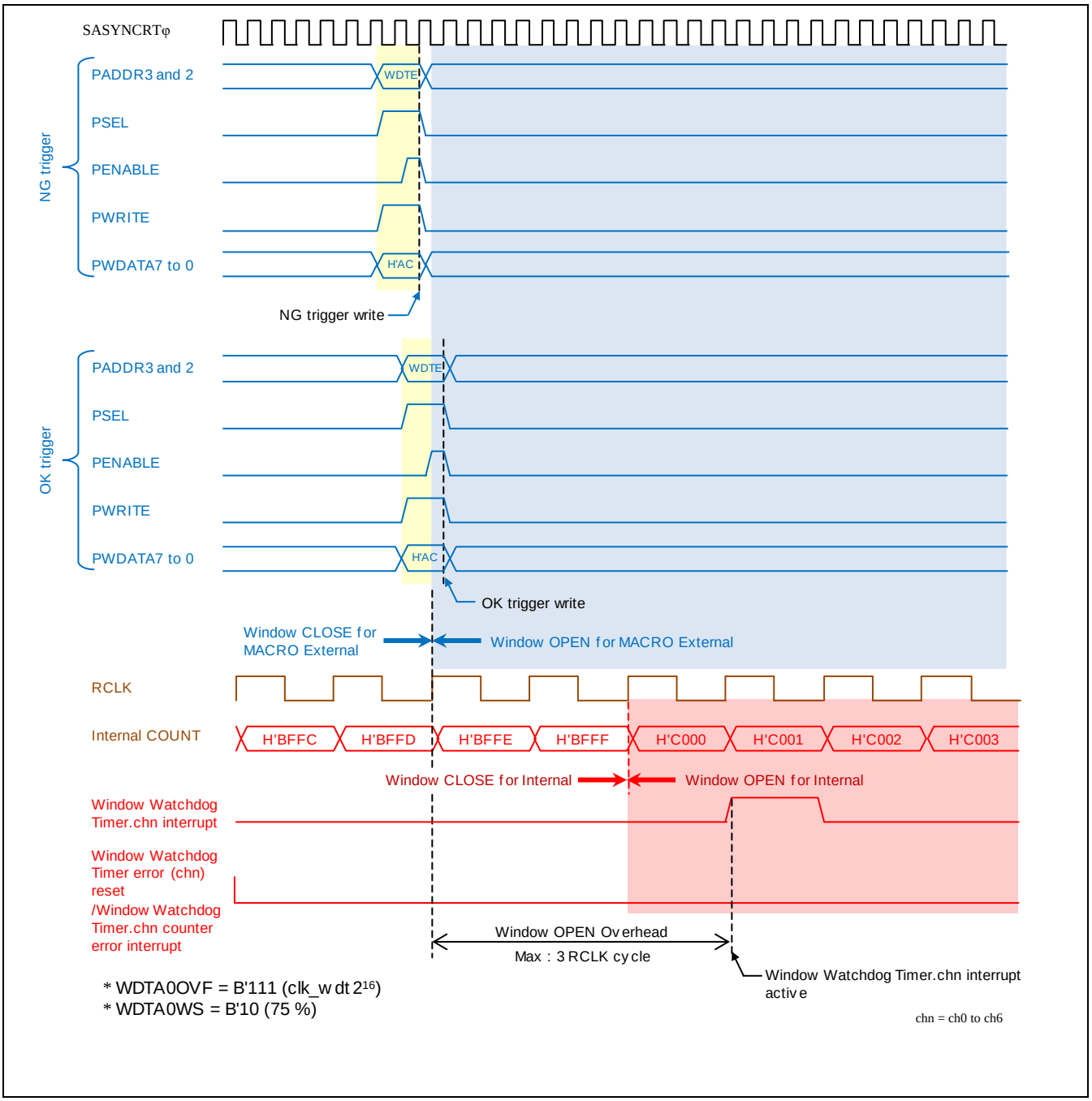



Figure 154.2.4.2 75% interrupt output timing

### 154.2.4.3 Reset / NMI output timing for Overflow

In this module, Reset/NMI output is output from the writing completion of the start trigger register after the period 100% of overflow setting value (WDTA0OVF), and the operation of this module stops.

But one RCLK cycle is necessary for output Reset / NSI after a timer counter reached the overflow. Moreover, the gap is caused in the output timing of Reset / NSI for the gap of the timer counter by the asynchronous system.

Therefore, writing in the start trigger register is disregarded because the overflow has been error detected internally before Reset / NMI is generated, and the period when Reset / NSI is output is generated.

It is 3 RCLK cycle before the Reset / NSI start at the maximum for this period.

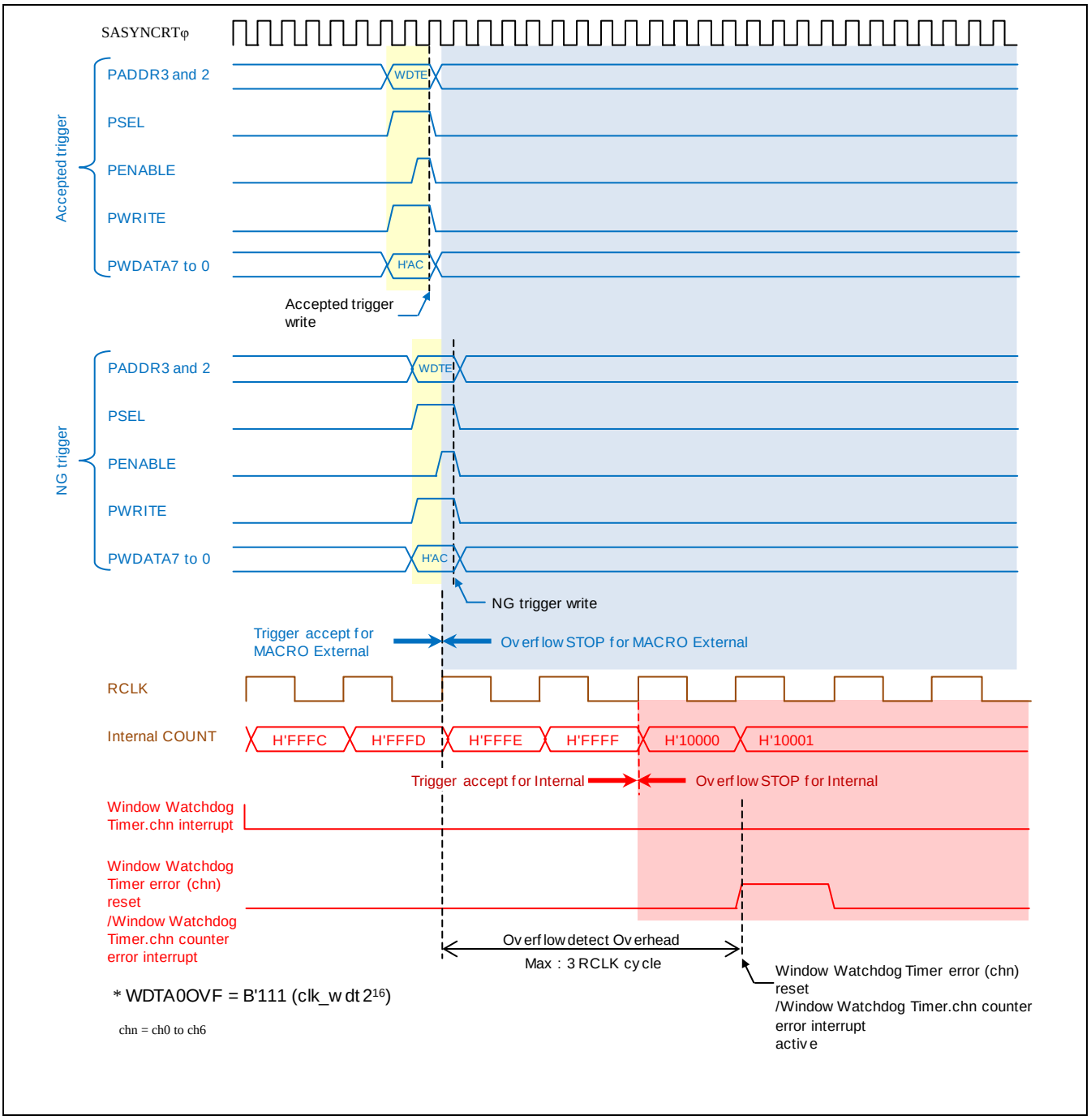



Figure 154.2.4.3 Reset / NMI output timing for Overflow

### 154.2.4.4 Reset / NMI output timing for error detection

In this module, Reset/NMI output is output after it error detects it of writing etc. of a wrong start trigger, and the operation of this module stops.

However, the register access, the timer counter operation, and the Reset / NMI output are late and the output timing of Reset / NMI is delayed because of the asynchronous system.

3 RCLK cycle at the maximum delays for register access to become the error detection after Reset / NMI are output.

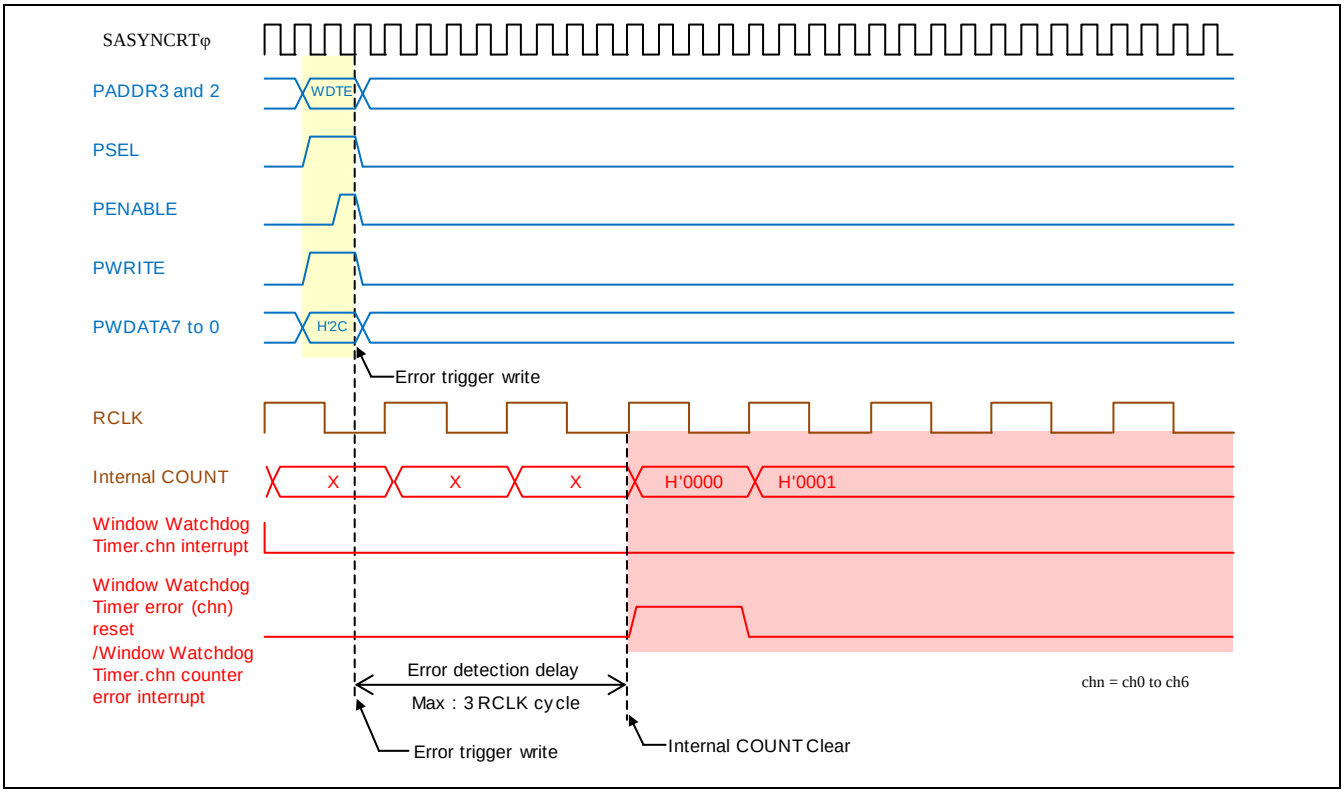



Figure 154.2.4.4 Reset / NMI output timing for error detection

#### 154.2.4.5 Notes on using emulator

When a break is issued to Cortex-A55 / Cortex-R52 when using the emulator, WWDT stop unlike R-Car Gen3.

### 154.2.5 Safety Mechanisms

#### 154.2.5.1 Power-On test

This module is protected by Power-On test.

# 154.3 System Watchdog Timer (SWDT)

## 154.3.1 Overview

This LSI includes the System Watchdog Timer (SWDT).
This LSI can be reset by the overflow of the counter when the value of the counter has not been updated because of a system runaway.
The System Watchdog timer is a single-channel timer that uses the OSCCLK as an input clock and can be used as a watchdog timer. OSCCLK is a clock which is generated by CPG.

### 154.3.1.1 Features

- One channel is provided.
- Can be used as a watchdog timer. A reset is generated when the counter overflows.

### 154.3.1.2 Block Diagram

Figure 154.3.1.1 shows a block diagram of the System Watchdog Timer.

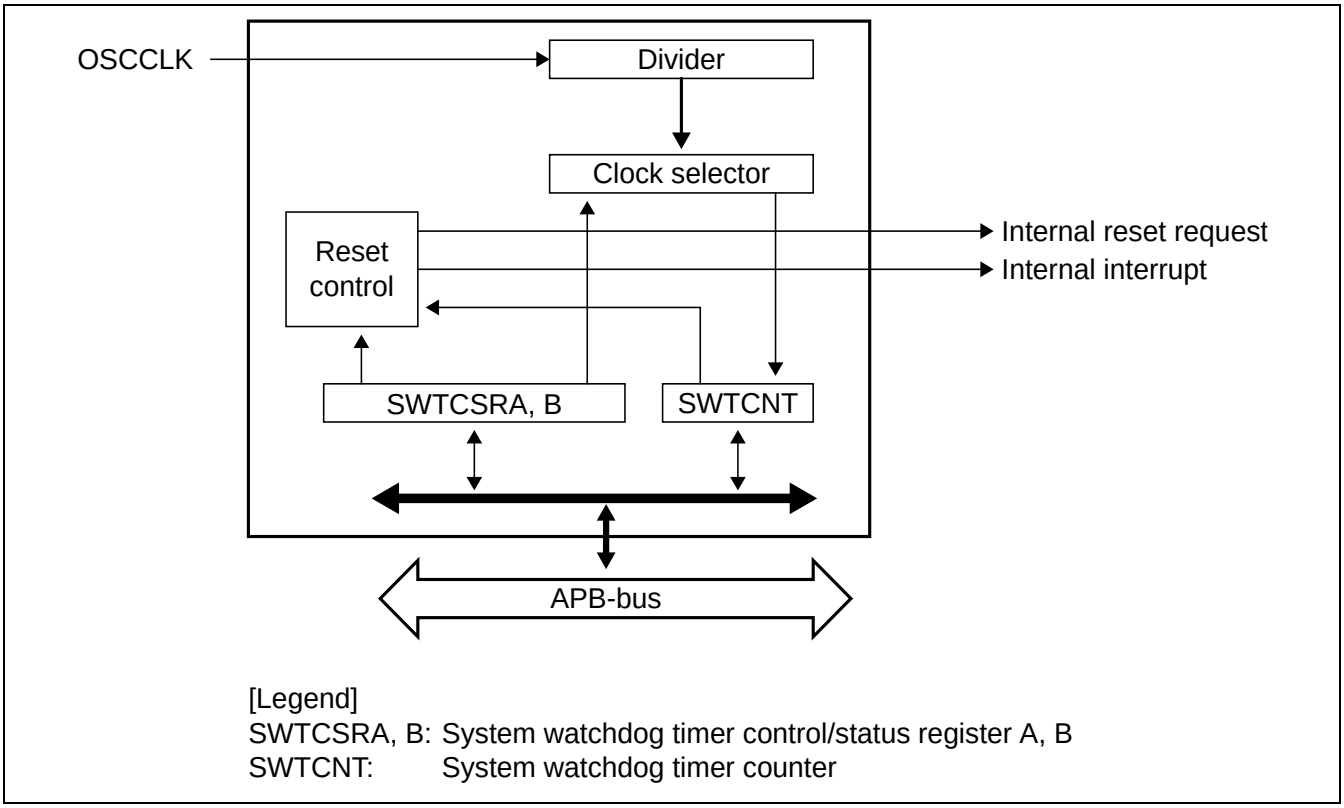



Figure 154.3.1.1 Block Diagram of System Watchdog Timer

### 154.3.1.3 Register Configuration

Table 154.3.1.1 shows the System Watchdog Timer register configuration. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined.

Table 154.3.1.2 shows the register state in each processing mode.

**Table 154.3.1.1 Register Configuration of System Watchdog Timer**

| Name | Abbreviation | R/W | Address | Access Size |
|---|---|---|---|---|
| System watchdog timer counter | SWTCNT | R/W | H'E603 0000 | 16/32*1 |
| System watchdog timer control/status register A | SWTCSRA | R/W*3 | H'E603 0004 | 8/32*2 |
| System watchdog timer control/status register B | SWTCSRB | R/W | H'E603 0008 | 8/32*2 |

Notes: 1. Write is performed in 32 bits and read in 16 bits. Operation cannot be guaranteed if the register is not written as a longword (32bits) and not read as a word (16bits).
2. Write is performed in 32 bits and read in 8 bits. Operation cannot be guaranteed if the register is not written as a longword (32bits) and not read as a byte (8bits).
3. Except for bit [4] that is R/WC0 register.

**Table 154.3.1.2 Register State of System Watchdog Timer in Each Processing Mode**

| Register Abbreviation | | Reset Caused by Other than Overflow | Reset Caused by Overflow* |
|---|---|---|---|
| Register | Bit | | |
| SWTCNT | All | Initialized | Initialized |
| SWTCSRA | WOVF | Initialized | Retained |
| | Other than WOVF | Initialized | Initialized |
| SWTCSRB | All | Initialized | Initialized |

Note: * System Watchdog Timer (SWDT) and Section 154.1. RCLK Watchdog Timer (RWDT) overflow reset.

### 154.3.2 Register Description

[Legend for Register Description]

Initial value: Register value after a reset.

—: Undefined value

R/W: Bit or field is readable and writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

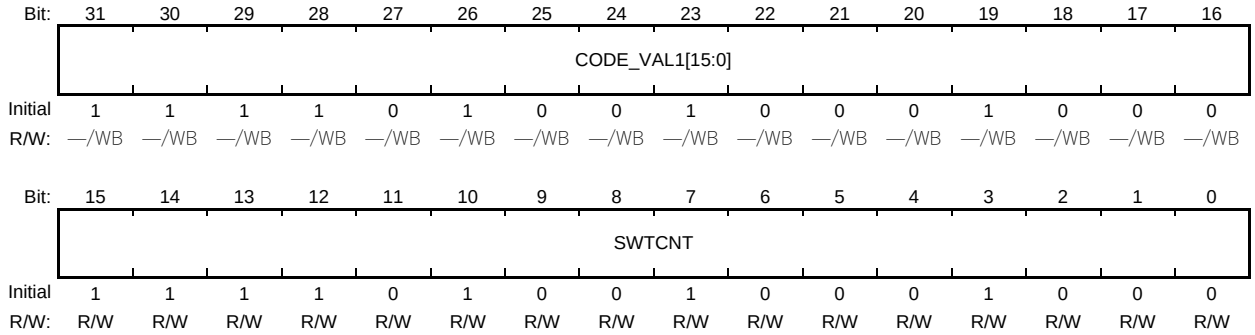
R: Read-only. The write value should always be initial value.

—/WB: Write-only. The read value is undefined.

#### 154.3.2.1 System Watchdog Timer Counter (SWTCNT)

SWTCNT is a 16-bit readable/writable register that increments on the selected clock. When an overflow occurs, it generates a reset. The SWTCNT counter is initialized to H'F488 by a power-on reset.

Use a long word access to write to the SWTCNT counter, with H'5A5A in the upper byte. Use a word access to read SWTCNT.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | CODE_VAL1 [15:0] | H’ F488 | —/WB | Code value 1<br>When writing this register, write H’5A5A to this bit field.<br>When reading, these bits are always read as same as SWTCNT [15:0]. |
| 15 to 0 | SWTCNT | H’F488 | R/W | Timer Counter Bits |

### 154.3.2.2 System Watchdog Timer Control/Status Register A (SWTCSRA)

SWTCSRA is an 8-bit readable/writable register composed of bits to select the clock used for the count, overflow flag, and enable bit.

Use a long word access to write to SWTCSRA, with H'A5 A5A5 in the upper byte. Use a byte access to read SWTCSRA.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CODE_VAL2[23:8] | | | | | | | | | | | | | | | |
| Initial | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| R/W: | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CODE_VAL2[7:0] | | | | | | | | TME | — | WRFLG | WOVF | WOVFE | CKS0[2:0] | | |
| Initial | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | —/WB | R/W | R | R | R/WC0 | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | CODE_VAL2 [23:0] | H'0F 0F0F | —/WB | Code value 2<br>When writing this register, write H'A5 A5A5 to this bit field.<br>When reading, these bits are always read as {SWTCSRA [7:0], SWTCSRA [7:0], SWTCSRA [7:0]}. |
| 7 | TME | 0 | R/W | Starts and stops timer operation.<br>0: Timer disabled: Count-up stops and SWTCNT value is retained<br>1: Timer enabled |
| 6 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5 | WRFLG | 0 | R | Write Status Flag<br>When this bit is 1, write access to SWTCNT is prohibited. If it isn't maintained, operation isn't secured. This bit indicates the period that the writing to SWTCNT is masked for synchronization after the writing to SWTCNT Confirm that this flag is 0 when SWTCNT is written to continuously. |
| 4 | WOVF | 0 | R/WC0 | Indicates that the SWTCNT has overflowed. This bit isn't initialized by a generated reset by an overflow of section 154.1. RCLK Watchdog Timer (RWDT) and System Watchdog Timer. Write 0 to this bit before using the System Watchdog Timer.<br>0: No overflow<br>1: SWTCNT has overflowed<br>Note: Only 0 can be written to clear the flag. |
| 3 | WOVFE | 1 | R/W | Overflow Interrupt Disable/Enable<br>0: Disables interrupts due to overflow<br>1: Enables interrupts due to overflow |

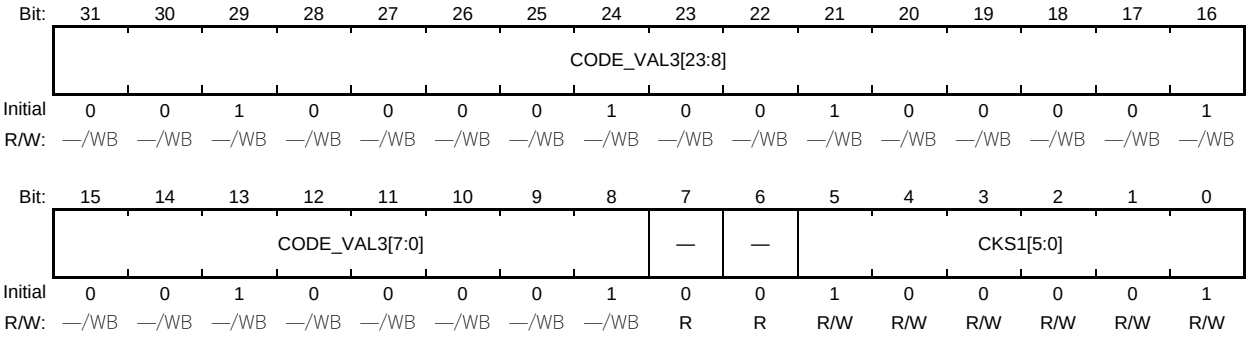
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 to 0 | CKS0[2:0] | 111 | R/W | RTC Clock Select<br>These bits select the clock to be used for the SWTCNT count from the eight types obtainable by dividing the OSCCLK. The overflow period that is shown inside the parenthesis in the table is the approximate value when the OSCCLK is 131.57 kHz.<br>000: OSCCLK (SWTCNT: H'0000 = 498 ms (SWTCNT: H'FBFC = 7.8 ms))<br>001: OSCCLK /4 (SWTCNT: H'0000 = 1.9 s (SWTCNT: H'FBFC = 31.2 ms))<br>010: OSCCLK /16 (SWTCNT: H'0000 = 7.9 s (SWTCNT: H'FBFC = 125.0 ms))<br>011: OSCCLK /32 (SWTCNT: H'0000 = 15.9 s (SWTCNT: H'FBFC = 250.0 ms))<br>100: OSCCLK /64 (SWTCNT: H'0000 = 31.8 s (SWTCNT: H'FBFC = 500.0 ms))<br>101: OSCCLK /128 (SWTCNT: H'0000 = 63.7 s (SWTCNT: H'FBFC = 1.0 s))<br>110: OSCCLK /1024 (SWTCNT: H'0000 = 510.0 s (SWTCNT: H'FBFC = 8.0 s))<br>111: OSCCLK select expanded mode<br>The clock cycle varies depending on the CKS1 bit in SWTCSRB. If SWTCSRB.CKS1[3:0] is H'0, division ratio is OSCCLK/4096 and values of overflow time is SWTCNT: H'0000 = 2040.2 s (SWTCNT: H'FBFC = 32.0 s). If SWTCSRB.CKS1[3:0] is other than H'0, values of overflow time is calculated with formula in chapter 154.3.3.2.<br>Note: These overflow periods are calculated from the rounded value of OSCCLK = 131.57 kHz. |

Note: If bits CKS0 [2:0] are modified when the System Watchdog Timer is operating, the up-count may not be performed correctly. Ensure that the bits CKS0 [2:0] are modified only when the System Watchdog Timer is not operating.

### 154.3.2.3 System Watchdog Timer Control/Status Register B (SWTCSRB)

SWTCSRB is an 8-bit readable/writable register composed of bits to select the clock used for the count.

Use a word access to write to SWTCSRB, with H'A5 A5A5 in the upper byte. Use a byte access to read SWTCSRB.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | CODE_ VAL3[23:0] | H'21 2121 | —/WB | Code value 3<br>When writing this register, write H'A5 A5A5 to this bit field.<br>When reading, these bits are always read as {SWTCSRB [7:0], SWTCSRB [7:0], SWTCSRB [7:0]}. |
| 7, 6 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | CKS1[5:0] | 100001 | R/W | OSCCLK Select for OSCCLK Select Expanded Mode<br>Selects the clock used for the SWTCNT count when the CKS0 bit in SWTCSRA is B'111.<br>Table 154.3.2.1 shows the time values (approximate value) used to cause an overflow from the counter H'0F13 (Overflow from counter H'FBFC). Assuming the overflow cycle in the table is calculated for the case when the OSCCLK is 131.57 kHz. For more values of overflow time, refer to chapter 154.3.3.2 |

Note: If the CKS1 [5] to CKS1 [0] bits are modified when the System Watchdog Timer is running, the up-count may not be performed correctly. Ensure that these bits are modified only when the System Watchdog Timer is not running.

Table 154.3.2.1 Example of Overflow time

| CKS1[3:0] \ CKS1[5:4] | B'00 | B'01 | B'10 | B'11 |
|---|---|---|---|---|
| H'0 | 1920.1 s (32.0 s) | | | |
| H'1 | 60.0 s<br>(1.0 s) | 5.0 min<br>(5.0 s) | 60.0 min<br>(1.0 min) | 300.0 min<br>(5.0 min) |
| H'2 | 120.0 s<br>(2.0 s) | 10.0 min<br>(10.0 s) | 120.0 min<br>(2.0 min) | 600.0 min<br>(10.0 min) |
| H'3 | 180.0 s<br>(3.0 s) | 15.0 min<br>(15.0 s) | 180.0 min<br>(3.0 min) | 900.0 min<br>(15.0 min) |
| H'4 | 240.0 s<br>(4.0 s) | 20.0 min<br>(20.0 s) | 240.0 min<br>(4.0 min) | 1200.0 min<br>(20.0 min) |
| H'5 | 5.0 min<br>(5.0 s) | 25.0 min<br>(25.0 s) | 300.0 min<br>(5.0 min) | 1500.0 min<br>(25.0 min) |
| H'6 | 6.0 min<br>(6.0 s) | 30.0 min<br>(30.0 s) | 360.0 min<br>(6.0 min) | 1800.1 min<br>(30.0 min) |
| H'7 | 7.0 min<br>(7.0 s) | 35.0 min<br>(35.0 s) | 420.0 min<br>(7.0 min) | Prohibited<br>(Prohibited) |
| H'8 | 8.0 min<br>(8.0 s) | 40.0 min<br>(40.0 s) | 480.0 min<br>(8.0 min) | Prohibited<br>(Prohibited) |
| H'9 | 9.0 min<br>(9.0 s) | 45.0 min<br>(45.0 s) | 540.0 min<br>(9.0 min) | Prohibited<br>(Prohibited) |
| H'A | 10.0 min<br>(10.0 s) | 50.0 min<br>(50.0 s) | 600.0 min<br>(10.0 min) | Prohibited<br>(Prohibited) |
| H'B | Prohibited<br>(Prohibited) | 55.0 min<br>(55.0 s) | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) |
| H'C | Prohibited<br>(Prohibited) | 60.0 min<br>(60.0 s) | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) |
| from H'D | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) | Prohibited<br>(Prohibited) |

## 154.3.3 Operation

### 154.3.3.1 Control of System Runaway

After a reset, System Watchdog Timer is disabled. When the counter overflow occurs after the counter starts, an internal reset is again generated. By this function, a reset can be automatically generated even when this LSI has caused a system runaway. While this LSI is operating correctly, write H'0000 (or the set value) to SWTCNT periodically so that SWTCNT does not overflow.

Take the following steps to use the System Watchdog Timer.

1. Clear the TME bit in SWTCSRA to 0 to temporarily stop counting.
   (Confirm that OSCCLK was input more than 2 cycles in case SWTCNT is not initialized)
2. Write H'0000 or the set value in SWTCNT
3. Clear the WOVF bit in SWTCSRA to 0.
4. Set the kind of count clock to the CKS0 [2:0] bits in SWTCSRA, and CKS1 [5:0] bits in SWTCSRB.
5. Confirm that SWTCSRA.WRFLG becomes 0.
   If WRFLG is 1, wait until it'll be 0.
6. Start the counting by setting the TME bit in SWTCSRA to 1.
   (TME and CKS0 can't be set at the same time. When setting that at the same time, behavior isn't guaranteed.)
7. While this LSI is operating correctly, write H'0000 (or the set value) to SWTCNT periodically so that SWTCNT does not overflow.
8. When SWTCNT overflows, Reset is generated because the System Watchdog Timer sets the WOVF flag in SWTCSRA to 1. At this time, SWTCNT, SWTCSRA (excluding the WOVF bit), and SWTCSRB are initialized.

Execute a power-on reset or a software reset at that time and initialize System Watchdog Timer.

When not initializing and using System Watchdog Timer, behavior isn't guaranteed.

### 154.3.3.2 Formula to calculate overflow time

This is the formula to calculate the overflow time for SWDT when choosing Expanded Mode.

Overflow time = 128 × T_OSCCLK × opset_val × (65536 – SWTCNT)

T_OSCCLK: is the period of OSCCLK clock.

SWTCNT: is value of SWTCNT in decade.

opset_val: refer to Table 154.3.3.1.

Table 154.3.3.1 Value of opset_val

| CKS1[5:0] | opset_val | CKS1[5:0] | opset_val | CKS1[5:0] | opset_val | CKS1[5:0] | opset_val |
|---|---|---|---|---|---|---|---|
| H'00 | 32 | H'10 | 32 | H'20 | 32 | H'30 | 32 |
| H'01 | 1 | H'11 | 5 | H'21 | 60 | H'31 | 300 |
| H'02 | 2 | H'12 | 10 | H'22 | 120 | H'32 | 600 |
| H'03 | 3 | H'13 | 15 | H'23 | 180 | H'33 | 900 |
| H'04 | 4 | H'14 | 20 | H'24 | 240 | H'34 | 1200 |
| H'05 | 5 | H'15 | 25 | H'25 | 300 | H'35 | 1500 |
| H'06 | 6 | H'16 | 30 | H'26 | 360 | H'36 | 1800 |
| H'07 | 7 | H'17 | 35 | H'27 | 420 | Other case | 1 |
| H'08 | 8 | H'18 | 40 | H'28 | 480 | | |
| H'09 | 9 | H'19 | 45 | H'29 | 540 | | |
| H'0A | 10 | H'1A | 50 | H'2A | 600 | | |
| Other case | 1 | H'1B | 55 | Other case | 1 | | |
| | | H'1C | 60 | | | | |
| | | Other case | 1 | | | | |

#### 154.3.3.3 Notes on Register Access

SWTCSRA: Bits TME, CKS0

SWTCSRB: Bits CKS1

After writing a value in this register, it is possible to read that value 2 cycles in counter input clock (OSCCLK) after writing has finished. And 2 cycles of OSCCLK need to be inputted until the set-up value is reflected.

SWTCNT: Bits 15 to 0

After writing a value in this register, it is possible to read that value 2 cycles in counter input clock (OSCCLK) after writing has finished. And 2 cycles of OSCCLK need to be inputted until the set-up value is reflected. Writing to this register, once started, is protected from further write operations until the writing operation has been completed. Read SWTCSRA. If WRFLG is 0, the next value can be written in SWTCNT.

The writing procedure to SWTCNT, SWTCSRA, and SWTCSRB differs from that of other registers with the purpose of preventing an unintended write. The procedure for writing to these registers is given below.

**Writing to SWTCNT, SWTCSRA, and SWTCSRB**

- These registers must be written by a long word transfer instruction. They cannot be written by a byte or word transfer instruction.
- When writing to SWTCNT, set the upper byte to H'5A5A and transfer the lower byte as the write data. When writing to SWTCSRA and SWTCSRB, set the upper byte to H'A5A5A5 and transfer the lower byte as the write data.

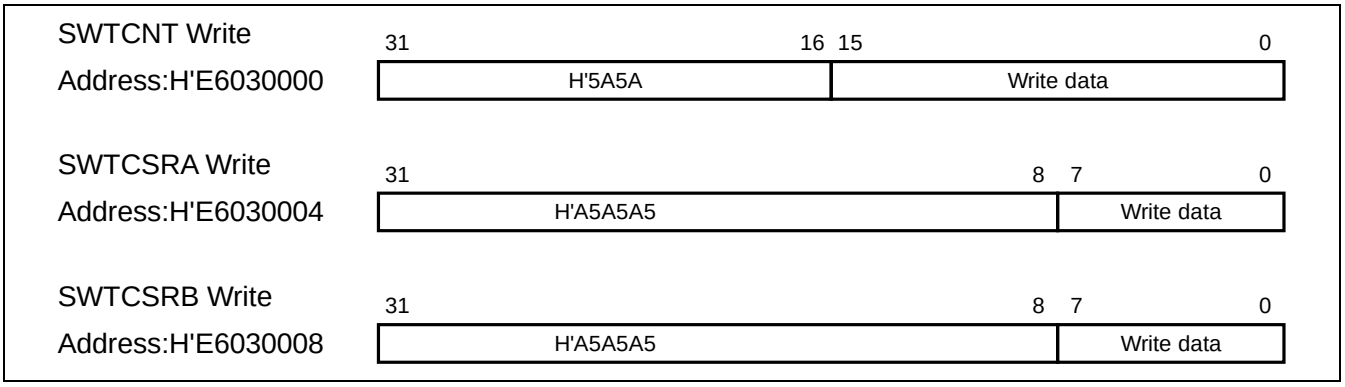



Figure 154.3.3.1 Writing to SWTCNT, SWTCSRA, and SWTCSRB

### 154.3.4 Usage Notes

No description on this section.

### 154.3.5 Safety Mechanisms

#### 154.3.5.1 Power-On test

This module is protected by Power-On test.

# 155. Timer Unit (TPU, CMT0, CMT1, TMU, SCMT, SUCMT, ...)

Timer Unit module is a module composed of TPU, CMT0, CMT1, TMU, SCMT, and SUCMT.

Refer to following section for the detailed function.

- TPU:155.1 16-Bit Timer Pulse Unit (TPU)
- CMT0:155.2 Compare Match Timer Type0 (CMT0)
- CMT1:155.3 Compare Match Timer Type1 (CMT1)
- TMU:155.4 Timer Unit (TMU)
- SCMT:155.5 System Timer (SCMT)
- SUCMT: 155.6 System Up-Time Clock (SUCMT)

# 155.1 16-Bit Timer Pulse Unit (TPU)

## 155.1.1 Overview

This LSI has a 16-bit timer pulse unit (TPU), which consists of four 16-bit timers.

Internal clock: SASYNCPERD4φ

### 155.1.1.1 Features

The TPU has the following features.

- Four timers are incorporated.
- Four pulse outputs are possible.
- Each timer of the TPU has four general registers (TGRAn, TGRBn, TGRCn, and TGRDn) for each timer. TGRAn enables an output compare setting. TGRBn, TGRCn and TGRDn in each timer can be used as timer-counter clearing registers. TGRCn and TGRDn can also be used as buffer registers.
- The following operations can be set for each timer.
  Waveform output on compare match: output of a 0 or 1 or toggling of output is selectable.
  Counter clearing operation: Counters can be cleared in response to a compare match.
  PWM mode allows the output of a PWM waveform with any duty cycle.
  The four outputs can be used to produce PWM output in up to four phases.
- A stepping motor control mode is available for timer 0.
- Buffer operation can be set for each timer.
- Automatic rewriting of values in output compare registers is possible.
- One interrupt request from TPU
  Enabling or disabling of compare match and counter overflows as interrupt sources can be set independently for each timer.
  In stepping motor control mode, enabling or disabling of state transitions, requests for data, and overflows of data as interrupt sources is possible for timer 0.

Table 155.1.1.1 shows the TPU functions.

**Table 155.1.1.1 TPU Functions**

| Item | | Timer 0 | Timer 1 | Timer 2 | Timer 3 |
|---|---|---|---|---|---|
| Counter clock | | Internal clock/1<br>Internal clock/4<br>Internal clock/16<br>Internal clock/64 | Internal clock/1<br>Internal clock/4<br>Internal clock/16<br>Internal clock/64 | Internal clock/1<br>Internal clock/4<br>Internal clock/16<br>Internal clock/64 | Internal clock/1<br>Internal clock/4<br>Internal clock/16<br>Internal clock/64 |
| General registers | | TGRA0<br>TGRB0 | TGRA1<br>TGRB1 | TGRA2<br>TGRB2 | TGRA3<br>TGRB3 |
| General and registers/ Buffer register | | TGRC0<br>TGRD0 | TGRC1<br>TGRD1 | TGRC2<br>TGRD2 | TGRC3<br>TGRD3 |
| Output pins | Normal operation | TPU0TO0 | TPU0TO1 | TPU0TO2 | TPU0TO3 |
| | Stepping motor control mode | TPU0TO0 to TPU0TO3 | None | None | None |
| Counter clearing | | Compare match with TGR | Compare match with TGR | Compare match with TGR | Compare match with TGR |
| Output in response to compare match | 0 | √ | √ | √ | √ |
| | 1 | √ | √ | √ | √ |
| | Toggling | √ | √ | √ | √ |
| PWM mode | | √ | √ | √ | √ |
| Buffer operation | | √ | √ | √ | √ |
| Interrupt | | 5 sources<br>Compare match/ overflow<br>6 more sources added in stepping motor control mode/<br>Transition to acceleration state<br>/transition to normal state<br>/transition to deceleration state<br>/transition to stop state<br>/data request<br>/data overflow | | | |

### 155.1.1.2 Block Diagram

Figure 155.1.1.1 is a block diagram of the TPU.

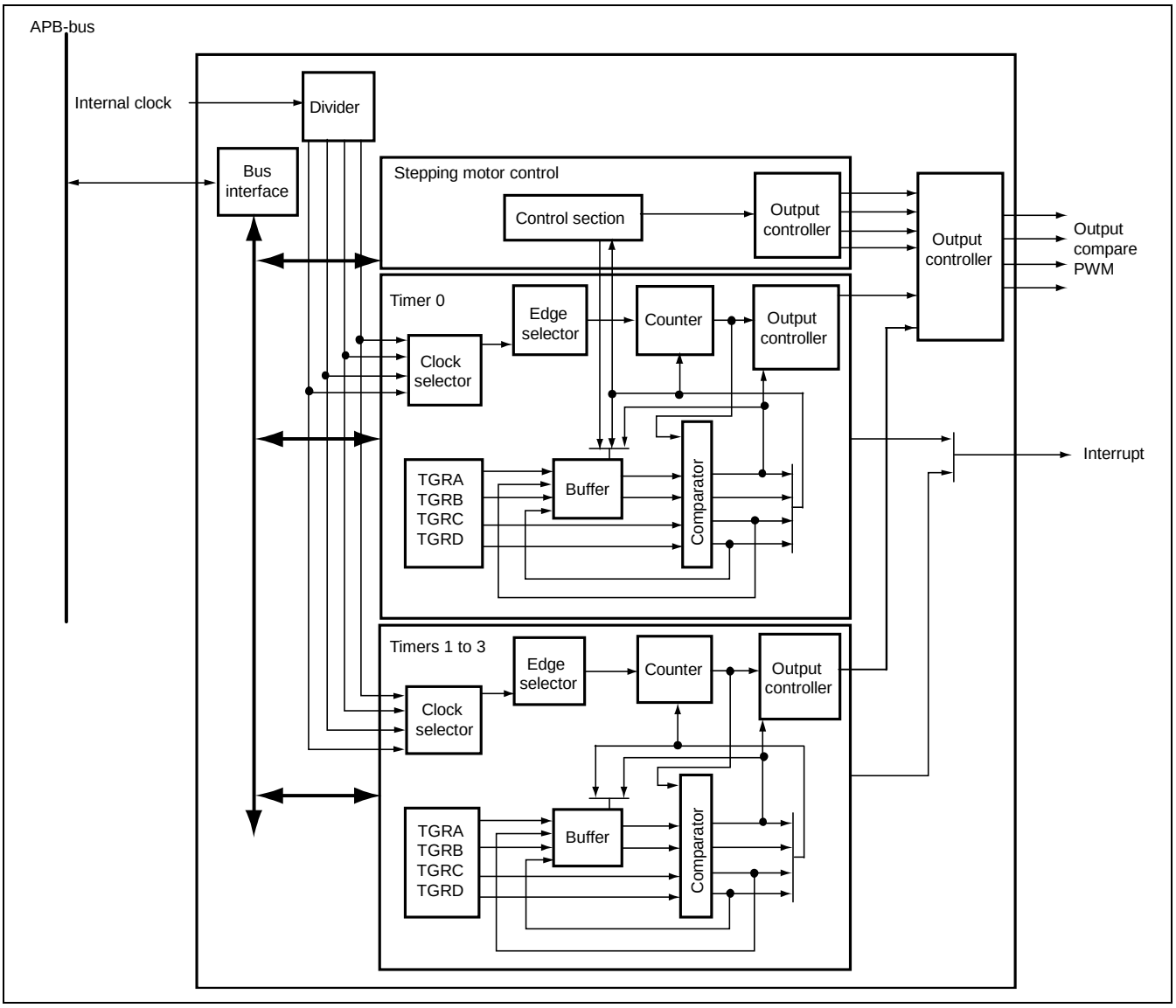



Figure 155.1.1.1 Block Diagram of the TPU

#### 155.1.1.3 External Pins

Table 155.1.1.2 shows the pin configuration of the TPU.

Table 155.1.1.2 Pin Configuration

| Module | Timer | Pin Name | Function | I/O | Description |
|---|---|---|---|---|---|
| TPU | 0 | TPU0TO0 | TPU output compare match 0 | Output | TGRA0 output compare output/ PWM output pin/pattern output pin 0 in stepping motor control mode |
| | 1 | TPU0TO1 | TPU output compare match 1 | Output | TGRA1 output compare output/ PWM output pin/pattern output pin 1 in stepping motor control mode |
| | 2 | TPU0TO2 | TPU output compare match 2 | Output | TGRA2 output compare output/ PWM output pin/pattern output pin 2 in stepping motor control mode |
| | 3 | TPU0TO3 | TPU output compare match 3 | Output | TGRA3 output compare output/ PWM output pin/pattern output pin 3 in stepping motor control mode |

#### 155.1.1.4 Register Configuration

Table 155.1.1.3 shows the TPU register configuration. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined. Access the register as a word (16 bits). Operation cannot be guaranteed if the register is not accessed as a word.

Table 155.1.1.4 shows the register states in each operating mode.

Table 155.1.1.3 Register Configuration

| Register Name | Abbreviation | R/W | Address | Access Size |
|---|---|---|---|---|
| Timer start register | TSTR | R/W | H'E6E8 0000 | 16 |
| Timer control register 0 | TCR0 | R/W | H'E6E8 0010 | 16 |
| Timer mode register 0 | TMDR0 | R/W | H'E6E8 0014 | 16 |
| Timer I/O control register 0 | TIOR0 | R/W | H'E6E8 0018 | 16 |
| Timer interrupt enable register 0 | TIER0 | R/W | H'E6E8 001C | 16 |
| Timer status register 0 | TSR0 | R/WC0 | H'E6E8 0020 | 16 |
| Timer counter 0 | TCNT0 | R/W | H'E6E8 0024 | 16 |
| Timer general register A0 | TGRA0 | R/W | H'E6E8 0028 | 16 |
| Timer general register B0 | TGRB0 | R/W | H'E6E8 002C | 16 |
| Timer general register C0 | TGRC0 | R/W | H'E6E8 0030 | 16 |
| Timer general register D0 | TGRD0 | R/W | H'E6E8 0034 | 16 |
| Timer control register 1 | TCR1 | R/W | H'E6E8 0050 | 16 |
| Timer mode register 1 | TMDR1 | R/W | H'E6E8 0054 | 16 |
| Timer I/O control register 1 | TIOR1 | R/W | H'E6E8 0058 | 16 |

| Register Name | Abbreviation | R/W | Address | Access Size |
|---|---|---|---|---|
| Timer interrupt enable register 1 | TIER1 | R/W | H'E6E8 005C | 16 |
| Timer status register 1 | TSR1 | R/W | H'E6E8 0060 | 16 |
| Timer counter 1 | TCNT1 | R/W | H'E6E8 0064 | 16 |
| Timer general register A1 | TGRA1 | R/W | H'E6E8 0068 | 16 |
| Timer general register B1 | TGRB1 | R/W | H'E6E8 006C | 16 |
| Timer general register C1 | TGRC1 | R/W | H'E6E8 0070 | 16 |
| Timer general register D1 | TGRD1 | R/W | H'E6E8 0074 | 16 |
| Timer control register 2 | TCR2 | R/W | H'E6E8 0090 | 16 |
| Timer mode register 2 | TMDR2 | R/W | H'E6E8 0094 | 16 |
| Timer I/O control register 2 | TIOR2 | R/W | H'E6E8 0098 | 16 |
| Timer interrupt enable register 2 | TIER2 | R/W | H'E6E8 009C | 16 |
| Timer status register 2 | TSR2 | R/W | H'E6E8 00A0 | 16 |
| Timer counter 2 | TCNT2 | R/W | H'E6E8 00A4 | 16 |
| Timer general register A2 | TGRA2 | R/W | H'E6E8 00A8 | 16 |
| Timer general register B2 | TGRB2 | R/W | H'E6E8 00AC | 16 |
| Timer general register C2 | TGRC2 | R/W | H'E6E8 00B0 | 16 |
| Timer general register D2 | TGRD2 | R/W | H'E6E8 00B4 | 16 |
| Timer control register 3 | TCR3 | R/W | H'E6E8 | 16 |

| Register Name | Abbreviation | R/W | Address | Access Size |
|---|---|---|---|---|
| | | | 00D0 | |
| Timer mode register 3 | TMDR3 | R/W | H'E6E8 00D4 | 16 |
| Timer I/O control register 3 | TIOR3 | R/W | H'E6E8 00D8 | 16 |
| Timer interrupt enable register 3 | TIER3 | R/W | H'E6E8 00DC | 16 |
| Timer status register 3 | TSR3 | R/W | H'E6E8 00E0 | 16 |
| Timer counter 3 | TCNT3 | R/W | H'E6E8 00E4 | 16 |
| Timer general register A3 | TGRA3 | R/W | H'E6E8 00E8 | 16 |
| Timer general register B3 | TGRB3 | R/W | H'E6E8 00EC | 16 |
| Timer general register C3 | TGRC3 | R/W | H'E6E8 00F0 | 16 |
| Timer general register D3 | TGRD3 | R/W | H'E6E8 00F4 | 16 |
| Motor control setting register | TMIR | R/W | H'E6E8 0100 | 16 |
| Motor deceleration (stop) transition register | TMRR | R/W | H'E6E8 0104 | 16 |
| Motor control status register | TMSR | R | H'E6E8 0108 | 16 |
| Motor operation pattern storing register 0 | TMMPR0 | R/W | H'E6E8 0110 | 16 |
| Motor operation pattern storing register 1 | TMMPR1 | R/W | H'E6E8 0114 | 16 |
| Motor stop pattern storing register 0 | TMSPR0 | R/W | H'E6E8 0118 | 16 |
| Motor stop pattern storing register 1 | TMSPR1 | R/W | H'E6E8 011C | 16 |
| Motor output pattern storing register | TMOPR | R/W | H'E6E8 0120 | 16 |

| Register Name | Abbreviation | R/W | Address | Access Size |
|---|---|---|---|---|
| Motor Acceleration Step Count Register | TMASR | R/W | H'E6E8 0130 | 16 |
| Motor normal the number of steps register | TMTSR | R/W | H'E6E8 0134 | 16 |
| Motor deceleration the number of steps register | TMRSR | R/W | H'E6E8 0138 | 16 |
| Motor control sequence counter register | TMSCR | R/W | H'E6E8 0140 | 16 |
| Motor control normal counter register | TMTCR | R | H'E6E8 0144 | 16 |

Table 155.1.1.4 Register States in Each Operating Mode

| Register Abbreviation | Power-on Reset/Software Reset *1 | Module Standby |
|---|---|---|
| TSTR | Initialized | Retained |
| TCR0 to TCR3 | Initialized | Retained |
| TMDR0 to TMDR3 | Initialized | Retained |
| TIOR0 to TIOR3 | Initialized | Retained |
| TIER0 to TIER3 | Initialized | Retained |
| TSR0 to TSR3 | Initialized | Retained |
| TCNT0 to TCNT3 | Initialized | Retained |
| TGRA0 to TGRA3 | Initialized | Retained |
| TGRB0 to TGRB3 | Initialized | Retained |
| TGRC0 to TGRC3 | Initialized | Retained |
| TGRD0 to TGRD3 | Initialized | Retained |
| TMIR | Initialized | Retained |
| TMRR | Initialized | Retained |
| TMSR | Initialized | Retained |
| TMMPR0<br>TMMPR1 | Initialized | Retained |
| TMSPR0<br>TMSPR1 | Initialized | Retained |
| TMOPR | Initialized | Retained |
| TMASR | Initialized | Retained |
| TMTSR | Initialized | Retained |
| TMRSR | Initialized | Retained |
| TMSCR | Initialized | Retained |
| TMTCR | Initialized | Retained |

Note: 1. Refer to 9. Module Standby, Software Reset and 14. Reset (RST).

## 155.1.2 Register Description

Registers in the TPU are allocated to and arranged in the address space of the internal bus.

Legend for Register Description

Initial value: Register value after a reset

—: Undefined value

R/W: Readable/writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

R: Read-only. When the bit is reserved, the write value should always be 0. Writing 1 to these bits can cause a malfunction of TPU.

—/W: Write-only. The read value is undefined.

### 155.1.2.1 Timer Control Register n (TCRn) (n = 0 to 3)

TCRn is provided for each of timers 0 to 3 and is used to control TCNT. TCRn is initialized to H'0000 by a reset.

Writing to TCRn is only allowed only when TCNTn is stopped.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | CCLR[2:0] | | | CKEG[1:0] | | TPSC[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 7 to 5 | CCLR[2:0] | 000 | R/W | Counter Clear<br>Specify the TCNTn clearing source.<br>000: TCNTn clearing disabled<br>001: TCNTn cleared in response to TGRAn compare match<br>010: TCNTn cleared in response to TGRBn compare match<br>011: Setting prohibited<br>100: TCNTn clearing disabled<br>101: TCNTn cleared in response to TGRCn compare match<br>110: TCNTn cleared in response to TGRDn compare match<br>111: Setting prohibited |
| 4, 3 | CKEG[1:0] | 00 | R/W | Clock Edge<br>Select the input clock edge for counting up.<br>Counting of both edges by the internal clock has the same effect as halving the cycle of the input clock (e.g., the rate of counting both edges of Internal clock/4 is same as the rate of counting rising edges of Internal clock/2). When the phase coefficient mode is selected, this setting is ignored.<br>00: Count rising edges<br>01: Count falling edges*<br>1X: Count both edges*<br>[Legend] X: Don't care<br>Note: * If the input clock is Internal clock/1, the timer does not operate with this setting. |
| 2 to 0 | TPSC[2:0] | 000 | R/W | Timer Prescaler<br>Select the clock for counting by TCNTn.<br>The clock source can be selected independently for each timer. Table 155.1.2.1 shows the clock sources that can be set for each timer. For more information on count clock selection, see Table 155.1.2.2. |

### 155.1.2.2 Timer Mode Register n (TMDRn) (n = 0 to 3)

TMDRn is provided for each of timers 0 to 3, and is used to specify the operating mode of the corresponding timer. TMDRn is initialized to H'0000 by a reset.

Writing to TMDRn is only allowed when TCNTn is stopped.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | BFWT | BFB | BFA | — | MD[2:0] | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 7 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 6 | BFWT | 0 | R/W | Buffer Write Timing<br>Specifies when to update TGRAn and TGRBn when TGRCn and TGRDn are in use as compare match buffers. When TGRCn and TGRDn are not in use as compare match buffer registers, this has no effect.<br>0: TGRAn and TGRBn are rewritten on a compare match with the counters.<br>1: TGRAn and TGRBn are rewritten on clearing of the counter. |
| 5 | BFB | 0 | R/W | Buffer Operation B<br>Specifies operation mode of TGRBn.<br>0: TGRBn is in normal operation.<br>1: TGRBn and TGRDn are used for buffered operation. |
| 4 | BFA | 0 | R/W | Buffer Operation A<br>Specifies operation mode of TGRAn.<br>0: TGRAn normal operation.<br>1: TGRAn and TGRCn are used for buffered operation. |
| 3 | — | 0 | R | Reserved<br>This bit is always read as 0 and cannot be modified. |
| 2 to 0 | MD[2:0] | 000 | R/W | Timer Operating Mode<br>Specifies the timer operating mode.<br>000: Normal operation<br>001: Setting prohibited<br>010: PWM mode<br>011 to 111: Setting prohibited |

### 155.1.2.3 Timer I/O Control Register n (TIORn) (n = 0 to 3)

TIORn is provided for each of timers 0 to 3, and is used to control the corresponding TPU0TO0 to TPU0TO3 pins. TIORn is initialized to H'0000 by a reset.

Writing to TIORn is only allowed when TCNTn is stopped. Note that the setting of TMDRn may affect TIORn.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | IOA[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 2 to 0 | IOA[2:0] | 000 | R/W | I/O Control<br>Bits IOA2 to IOA0 specify the functions of TGRAn and the TPU0TO0 to TPU0TO3 pins. For details, see Table 155.1.2.3. |

#### 155.1.2.4 Timer Interrupt Enable Register n (TIERn) (n = 0 to 3)

TIERn is provided for each of timers 0 to 3 and is used to enable or disable the sources of interrupt requests for the corresponding timer. TIERn is initialized to H'0000 by a reset. Bits 8 to 13 are only present in timer 0.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TMD OFE | TMD RFE | TMS 1ER | TMS 1ET | TMS 1EA | TMS 1ES | — | — | — | TC1EV | TG1ED | TG1EC | TG1EB | TG1EA |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15, 14 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 13 | TMDOFE | 0 | R/W | Motor Control Data Transfer Overflow Detection Enable<br>This bit enables or disables interrupt request in response to the state of the TMDOFS flag in the stepping motor control mode of timer 0.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>0: Interrupt request in response to the TMDOFS flag disabled<br>1: Interrupt request in response to the TMDOFS flag enabled |
| 12 | TMDRFE | 0 | R/W | Motor Control Data Transfer Request Detection Enable<br>This bit enables or disables interrupt request in response to the state of the TMDRFS flag in the stepping motor control mode of timer 0.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>0: Interrupt request in response to the TMDRFS flag disabled<br>1: Interrupt request in response to the TMDRFS flag enabled |
| 11 | TMS1ER | 0 | R/W | Motor Control Deceleration Transition Detection Enable<br>This bit enables or disables interrupt request in response to the state of the TMCFR flag in the stepping motor control mode of timer 0.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>0: Interrupt request in response to the TMCFR flag disabled<br>1: Interrupt request in response to the TMCFR flag enabled |
| 10 | TMS1ET | 0 | R/W | Motor Control Normal Transition Detection Enable<br>This bit enables or disables interrupt request in response to the state of the TMCFT flag in the stepping motor control mode of timer 0.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>0: Interrupt request in response to the TMCFT flag disabled<br>1: Interrupt request in response to the TMCFT flag enabled |
| 9 | TMS1EA | 0 | R/W | Motor Control Acceleration Transition Detection Enable<br>This bit enables or disables interrupt request in response to the state of the TMCFA flag in the stepping motor control mode of timer 0.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>0: Interrupt request in response to the TMCFA flag disabled<br>1: Interrupt request in response to the TMCFA flag enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8 | TMS1ES | 0 | R/W | Motor Control Stop Transition Detection Enable<br>This bit enables or disables interrupt request in response to the state of the TMCFS flag in the stepping motor control mode of timer 0,<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>0: Interrupt request in response to the TMCFS flag disabled<br>1: Interrupt request in response to the TMCFS flag enabled |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 4 | TC1EV | 0 | R/W | Overflow Interrupt Enable<br>This bit enables or disables the interrupt request in response to the state of the TCFV flag.<br>0: Interrupt request in response to the TCFV flag disabled<br>1: Interrupt request in response to the TCFV flag enabled |
| 3 | TG1ED | 0 | R/W | TGR Interrupt Enable D.<br>This bit enables or disables interrupt request in response to the state of the TGFD bit.<br>0: Interrupt request in response to the TGFD bit disabled<br>1: Interrupt request in response to the TGFD bit enabled |
| 2 | TG1EC | 0 | R/W | TGR Interrupt Enable C<br>This bit enables or disables interrupt request in response to the state of the TGFC bit.<br>0: Interrupt request in response to the TGFC bit disabled<br>1: Interrupt request in response to the TGFC bit enabled |
| 1 | TG1EB | 0 | R/W | TGR Interrupt Enable B<br>This bit enables or disables interrupt request in response to the state of the TGFB bit.<br>0: Interrupt request in response to the TGFB bit disabled<br>1: Interrupt request in response to the TGFB bit enabled |
| 0 | TG1EA | 0 | R/W | TGR Interrupt Enable A<br>This bit enables or disables interrupt request in response to the state of the TGFA bit.<br>0: Interrupt request in response to the TGFA bit disabled<br>1: Interrupt request in response to the TGFA bit enabled |

### 155.1.2.5 Timer Status Registers n (TSRn) (n = 0 to 3)

TSRn is provided for each of timers 0 to 3 and indicates the state of the corresponding timer. TSRn is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | TMD OFS | TMD RFS | TMCFR | TMCFT | TMCFA | TMCFS | — | — | — | TCFV | TGFD | TGFC | TGFB | TGFA |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/WC0 | R/WC0 | R/WC0 | R/WC0 | R/WC0 | R/WC0 | R | R | R | R/WC0 | R/WC0 | R/WC0 | R/WC0 | R/WC0 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15, 14 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 13 | TMDOFS | 0 | R/WC0 | Motor Control Data Transfer Overflow Detection<br>This flag indicates that a data transfer overflow has been detected in the stepping motor control mode of timer 0.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>[Clearing condition]<br>Writing 0 to the TMDOFS bit after reading the TMDOFS bit while TMDOFS is set to 1<br>[Setting condition]<br>When the DMA transfer overflow occurs |
| 12 | TMDRFS | 0 | R/WC0 | Motor Control Data Transfer Request Detection<br>This flag indicates that a data transfer request interrupt has been detected in the stepping motor control mode of timer 0. If DMA transfer is not in use, data are written to TGRD0 in response to this interrupt.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>[Clearing condition]<br>Writing 0 to the TMDRFS bit after reading the TMDRFS bit while TMDRFS is set to 1<br>[Setting condition]<br>When data transfer request occurs |
| 11 | TMCFR | 0 | R/WC0 | Motor Control Deceleration Transition Detection<br>This flag indicates that the current sequence state of timer 0 in the stepping motor control mode is the deceleration state.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>[Clearing condition]<br>Writing 0 to the TMCFR bit after reading TMCFR bit = 1<br>[Setting condition]<br>When the sequence state is shifted to the deceleration state |
| 10 | TMCFT | 0 | R/WC0 | Motor Control Normal Transition Detection<br>This flag indicates that the current sequence state of timer 0 in the stepping motor control mode is the normal state.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>[Clearing condition]<br>Writing 0 to the TMCFT bit after reading TMCFT bit = 1<br>[Setting condition]<br>When the sequence state is shifted to the normal state |

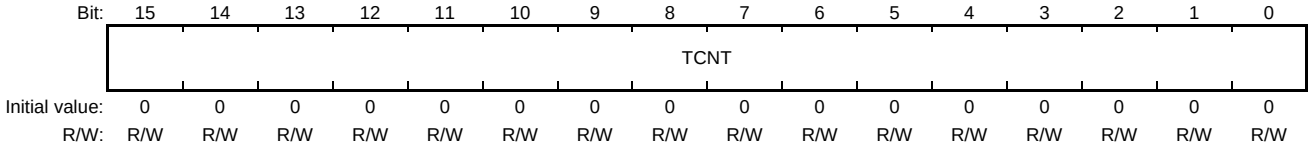
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | TMCFA | 0 | R/WC0 | Motor Control Acceleration Transition Detection<br>This flag indicates that the current sequence state of timer 0 in the stepping motor control mode is the acceleration state.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>[Clearing condition]<br>Writing 0 to the TMCFA bit after reading TMCFA bit = 1<br>[Setting condition]<br>When the sequence state is shifted to the acceleration state |
| 8 | TMCFS | 0 | R/WC0 | Motor Control Stop Transition Detection<br>This flag indicates that the current sequence state of timer 0 in the stepping motor control mode is the stop state.<br>This bit is reserved in timers 1 to 3, is always read as 0, and cannot be modified.<br>[Clearing condition]<br>Writing 0 to the TMCFS bit after reading TMCFS bit = 1<br>[Setting condition]<br>When the sequence state is shifted to the stop state |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 4 | TCFV | 0 | R/WC0 | Overflow<br>This flag indicates that TCNTn has overflowed.<br>[Clearing condition]<br>Writing 0 to the TCFV bit after reading TCFV bit = 1<br>[Setting condition]<br>When the value of TCNTn overflows (when its value changes from H'FFFF to H'0000) |
| 3 | TGFD | 0 | R/WC0 | Compare Flag D<br>This flag indicates that the values of TCNTn and TGRDn have matched (a compare match).<br>[Clearing condition]<br>Writing 0 to the TGFD bit after reading TGFD bit = 1<br>[Setting condition]<br>When the values of TCNTn and TGRDn have matched |
| 2 | TGFC | 0 | R/WC0 | Compare Flag C<br>This flag indicates that the values of TCNTn and TGRCn have matched (a compare match).<br>[Clearing condition]<br>Writing 0 to the TGFC bit after reading TGFC bit = 1<br>[Setting condition]<br>When the values of TCNTn and TGRCn have matched |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | TGFB | 0 | R/WC0 | Compare Flag B<br>This flag indicates that the values of TCNTn and TGRBn have matched (a compare match).<br>[Clearing condition]<br>Writing 0 to the TGFB bit after reading TGFB bit = 1<br>[Setting condition]<br>When the values of TCNTn and TGRBn have matched |
| 0 | TGFA | 0 | R/WC0 | Output Compare Flag A<br>This flag indicates that the values of TCNTn and TGRAn have matched (a compare match).<br>[Clearing condition]<br>Writing 0 to the TGFA bit after reading TGFA bit = 1<br>[Setting condition]<br>When the values of TCNTn and TGRAn are matched |

#### 155.1.2.6 Timer Counter n (TCNTn) (n = 0 to 3)

TCNTn is a 16-bit counter, and is provided for each of timers 0 to 3.

TCNTn is initialized to H'0000 by a reset.

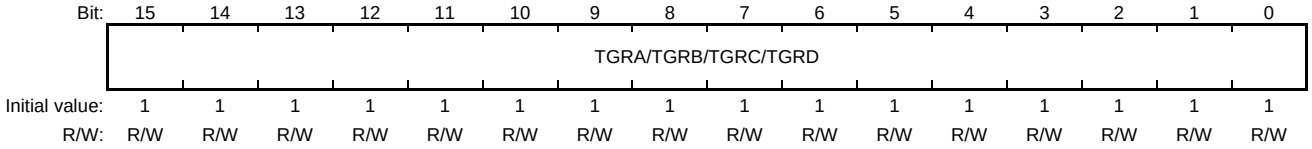



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | TCNT | All 0 | R/W | Timer Counter Bits |

### 155.1.2.7 Timer General Register A to D (TGRAn/TGRBn/TGRCn/TGRDn) (n = 0 to 3)

TGRAn, TGRBn, TGRCn and TGRDn are 16 bits general registers that are provided for each of timers 0 to 3. TGRCn and TGRDn can be designated for operation as buffer registers*. TGR is initialized to H'FFFF by a reset.

Note: * The combinations of TGR registers and buffer registers are TGRAn with TGRCn and TGRBn with TGRDn.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | TGRA/TGRB/ TGRC/TGRD | All 1 | R/W | Timer General Register Bits |

### 155.1.2.8 Timer Start Register (TSTR)

TSTR is used to start or stop the timer counter (TCNTn) of timers 0 to 3.

TSTR is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | TMST | CST3 | CST2 | CST1 | CST0 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 4 | TMST | 0 | R/W | Motor Control Sequence Start<br>Selects whether to start or stop the motor control sequence.<br>This bit is cleared to 0 when the timer has shifted to the stop state after motor control operation was started.<br>0: Stops motor control sequence<br>1: Starts motor control sequence |
| 3 | CST3 | 0 | R/W | Counter Start<br>Selects whether to start or stop the TCNT3 operation.<br>0: Stops the TCNT3 counting operation<br>1: Starts the TCNT3 counting operation |
| 2 | CST2 | 0 | R/W | Counter Start<br>Selects whether to start or stop the TCNT2 operation.<br>0: Stops the TCNT2 counting operation<br>1: Starts the TCNT2 counting operation |
| 1 | CST1 | 0 | R/W | Counter Start<br>Selects whether to start or stop the TCNT1 operation.<br>0: Stops the TCNT1 counting operation<br>1: Starts the TCNT1 counting operation |
| 0 | CST0 | 0 | R/W | Counter Start<br>Selects whether to start or stop the TCNT0 operation.<br>0: Stops the TCNT0 counting operation<br>1: Starts the TCNT0 counting operation |

### 155.1.2.9 Motor Control Setting Register (TMIR)

TMIR is used to configure operation in the motor control mode.

TMIR is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | TDMAE | MTRPAT DOWN | MTRPATKIND [1:0] | | MTRO N |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 4 | TDMAE | 0 | R/W | Selection of DMA Use<br>Selects whether to use DMA for step interval input (writing to TGRD0) in the motor control mode.<br>0: DMA is not used.<br>1: DMA is used.<br>Note: It is possible to write this register only Stepping-Motor Control Mode. Please fix this bit to 1'b0 other than Stepping-Motor Control Mode. Operation is not guaranteed if this bit is set to 1'b1 other than Stepping-Motor Control Mode. |
| 3 | MTRPAT DOWN | 0 | R/W | Operation and Stop Patterns Transition Ascending/Descending Select<br>Selects the direction of transitions within operation and stop patterns as ascending or descending in the motor control mode.<br>0: Ascending<br>1: Descending |
| 2, 1 | MTRPAT KIND[1:0\| | 00 | R/W | Operation and Stop Pattern Type Select<br>Selects the number of patterns in the cycles of stop and start patterns in the motor control mode.<br>00: Four<br>01: Eight<br>10: Setting prohibited<br>11: Setting prohibited |
| 0 | MTRON | 0 | R/W | TPU Mode/Stepping Motor Control Mode Select<br>0: TPU mode<br>1: Stepping motor control mode<br>Note: This bit also controls a selecting signal to switch TPU0TO0 to TPU0TO3 outputs. Set this bit to 1 after making initial settings for the pattern cycle. Do not modify this bit during the counter operation. |

### 155.1.2.10 Motor Deceleration (Stop) Transition Register (TMRR)

TMRR is used to place timer 0 in to the deceleration state from the normal state, in the motor control mode.

TMRR is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | REDUON0 | REDUON |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 1 | REDUON0 | 0 | R/W | This bit is used to force a transition of timer 0 to the deceleration (or stop) state from the normal state in motor control mode. Note that the state transition only proceeds when pattern 0 is the currently specified pattern. Setting this bit has no effect if timer 0 is not in the normal state. This bit is automatically cleared to 0 when the state of timer 0 shifts to the stop state.<br>0: No operation<br>1: Timer 0 is forcibly placed in the deceleration (or stop) state from the normal state. |
| 0 | REDUON | 0 | R/W | This bit is used to force a transition of timer 0 to the deceleration (or stop) state from the normal state in motor control mode. Setting this bit has no effect if timer 0 is not in the normal state. This bit is automatically cleared to 0 when the state of timer 0 shifts to the stop state.<br>0: No operation<br>1: Timer 0 is forcibly placed in the deceleration (or stop) state from the normal state. |

### 155.1.2.11 Motor Control Status Register (TMSR)

TMSR is a read-only register that indicates the sequence state in motor control mode.

TMSR is initialized to H'0001 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | SITR | SITT | SITA | SITS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 3 | SITR | 0 | R | This bit indicates the sequence state in motor control mode.<br>1: The sequence state is deceleration.<br>0: The sequence state is not deceleration. |
| 2 | SITT | 0 | R | This bit indicates the sequence state in motor control mode.<br>1: The sequence state is normal.<br>0: The sequence state is not normal. |
| 1 | SITA | 0 | R | This bit indicates the sequence state in motor control mode.<br>1: The sequence state is acceleration.<br>0: The sequence state is not acceleration. |
| 0 | SITS | 1 | R | This bit indicates the sequence state in motor control mode.<br>1: The sequence state is stop.<br>0: The sequence state is not stop. |

### 155.1.2.12 Motor Operation Pattern Storing Register 0 (TMMPR0)

TMMPR0 is used to set motor operation patterns [3] to [0] in motor control mode.

TMMPR0 is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MP33 | MP32 | MP31 | MP30 | MP23 | MP22 | MP21 | MP20 | MP13 | MP12 | MP11 | MP10 | MP03 | MP02 | MP01 | MP00 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | MP33 | 0 | R/W | Motor Operation Pattern [3] in Motor Control Mode |
| 14 | MP32 | 0 | R/W | MP33: TPU0TO3 output value in motor operation pattern [3] |
| 13 | MP31 | 0 | R/W | MP32: TPU0TO2 output value in motor operation pattern [3] |
| 12 | MP30 | 0 | R/W | MP31: TPU0TO1 output value in motor operation pattern [3] |
| | | | | MP30: TPU0TO0 output value in motor operation pattern [3] |
| 11 | MP23 | 0 | R/W | Motor Operation Pattern [2] in Motor Control Mode |
| 10 | MP22 | 0 | R/W | MP23: TPU0TO3 output value in motor operation pattern [2] |
| 9 | MP21 | 0 | R/W | MP22: TPU0TO2 output value in motor operation pattern [2] |
| 8 | MP20 | 0 | R/W | MP21: TPU0TO1 output value in motor operation pattern [2] |
| | | | | MP20: TPU0TO0 output value in motor operation pattern [2] |
| 7 | MP13 | 0 | R/W | Motor Operation Pattern [1] in Motor Control Mode |
| 6 | MP12 | 0 | R/W | MP13: TPU0TO3 output value in motor operation pattern [1] |
| 5 | MP11 | 0 | R/W | MP12: TPU0TO2 output value in motor operation pattern [1] |
| 4 | MP10 | 0 | R/W | MP11: TPU0TO1 output value in motor operation pattern [1] |
| | | | | MP10: TPU0TO0 output value in motor operation pattern [1] |
| 3 | MP03 | 0 | R/W | Motor Operation Pattern [0] in Motor Control Mode |
| 2 | MP02 | 0 | R/W | MP03: TPU0TO3 output value in motor operation pattern [0] |
| 1 | MP01 | 0 | R/W | MP02: TPU0TO2 output value in motor operation pattern [0] |
| 0 | MP00 | 0 | R/W | MP01: TPU0TO1 output value in motor operation pattern [0] |
| | | | | MP00: TPU0TO0 output value in motor operation pattern [0] |

### 155.1.2.13 Motor Operation Pattern Storing Register 1 (TMMPR1)

TMMPR1 is used to set motor operation pattern [7] to [4] in motor control mode.

TMMPR1 is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MP73 | MP72 | MP71 | MP70 | MP63 | MP62 | MP61 | MP60 | MP53 | MP52 | MP51 | MP50 | MP43 | MP42 | MP41 | MP40 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | MP73 | 0 | R/W | Motor Operation Pattern [7] in Motor Control Mode |
| 14 | MP72 | 0 | R/W | MP73: TPU0TO3 output value in motor operation pattern [7] |
| 13 | MP71 | 0 | R/W | MP72: TPU0TO2 output value in motor operation pattern [7] |
| 12 | MP70 | 0 | R/W | MP71: TPU0TO1 output value in motor operation pattern [7] |
| | | | | MP70: TPU0TO0 output value in motor operation pattern [7] |
| 11 | MP63 | 0 | R/W | Motor Operation Pattern [6] in Motor Control Mode |
| 10 | MP62 | 0 | R/W | MP63: TPU0TO3 output value in motor operation pattern [6] |
| 9 | MP61 | 0 | R/W | MP62: TPU0TO2 output value in motor operation pattern [6] |
| 8 | MP60 | 0 | R/W | MP61: TPU0TO1 output value in motor operation pattern [6] |
| | | | | MP60: TPU0TO0 output value in motor operation pattern [6] |
| 7 | MP53 | 0 | R/W | Motor Operation Pattern [5] in Motor Control Mode |
| 6 | MP52 | 0 | R/W | MP53: TPU0TO3 output value in motor operation pattern [5] |
| 5 | MP51 | 0 | R/W | MP52: TPU0TO2 output value in motor operation pattern [5] |
| 4 | MP50 | 0 | R/W | MP51: TPU0TO1 output value in motor operation pattern [5] |
| | | | | MP50: TPU0TO0 output value in motor operation pattern [5] |
| 3 | MP43 | 0 | R/W | Motor Operation Pattern [4] in Motor Control Mode |
| 2 | MP42 | 0 | R/W | MP43: TPU0TO3 output value in motor operation pattern [4] |
| 1 | MP41 | 0 | R/W | MP42: TPU0TO2 output value in motor operation pattern [4] |
| 0 | MP40 | 0 | R/W | MP41: TPU0TO1 output value in motor operation pattern [4] |
| | | | | MP40: TPU0TO0 output value in motor operation pattern [4] |

### 155.1.2.14 Motor Stop Pattern Storing Register 0 (TMSPR0)

TMSPR0 is used to set motor stop pattern (3) to (0) in motor control mode.

TMSPR0 is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SP33 | SP32 | SP31 | SP30 | SP23 | SP22 | SP21 | SP20 | SP13 | SP12 | SP11 | SP10 | SP03 | SP02 | SP01 | SP00 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | SP33 | 0 | R/W | Motor Stop Pattern (3) in Motor Control Mode |
| 14 | SP32 | 0 | R/W | SP33: TPU0TO3 output value in motor stop pattern (3) |
| 13 | SP31 | 0 | R/W | SP32: TPU0TO2 output value in motor stop pattern (3) |
| 12 | SP30 | 0 | R/W | SP31: TPU0TO1 output value in motor stop pattern (3) |
| | | | | SP30: TPU0TO0 output value in motor stop pattern (3) |
| 11 | SP23 | 0 | R/W | Motor Stop Pattern (2) in Motor Control Mode |
| 10 | SP22 | 0 | R/W | SP23: TPU0TO3 output value in motor stop pattern (2) |
| 9 | SP21 | 0 | R/W | SP22: TPU0TO2 output value in motor stop pattern (2) |
| 8 | SP20 | 0 | R/W | SP21: TPU0TO1 output value in motor stop pattern (2) |
| | | | | SP20: TPU0TO0 output value in motor stop pattern (2) |
| 7 | SP13 | 0 | R/W | Motor Stop Pattern (1) in Motor Control Mode |
| 6 | SP12 | 0 | R/W | SP13: TPU0TO3 output value in motor stop pattern (1) |
| 5 | SP11 | 0 | R/W | SP12: TPU0TO2 output value in motor stop pattern (1) |
| 4 | SP10 | 0 | R/W | SP11: TPU0TO1 output value in motor stop pattern (1) |
| | | | | SP10: TPU0TO0 output value in motor stop pattern (1) |
| 3 | SP03 | 0 | R/W | Motor Stop Pattern (0) in Motor Control Mode |
| 2 | SP02 | 0 | R/W | SP03: TPU0TO3 output value in motor stop pattern (0) |
| 1 | SP01 | 0 | R/W | SP02: TPU0TO2 output value in motor stop pattern (0) |
| 0 | SP00 | 0 | R/W | SP01: TPU0TO1 output value in motor stop pattern (0) |
| | | | | SP00: TPU0TO0 output value in motor stop pattern (0) |

### 155.1.2.15 Motor Stop Pattern Storing Register 1 (TMSPR1)

TMSPR1 is used to set motor stop pattern (7) to (4) in motor control mode.

TMSPR1 is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SP73 | SP72 | SP71 | SP70 | SP63 | SP62 | SP61 | SP60 | SP53 | SP52 | SP51 | SP50 | SP43 | SP42 | SP41 | SP40 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | SP73 | 0 | R/W | Motor Stop Pattern (7) in Motor Control Mode |
| 14 | SP72 | 0 | R/W | SP73: TPU0TO3 output value in motor stop pattern (7) |
| 13 | SP71 | 0 | R/W | SP72: TPU0TO2 output value in motor stop pattern (7) |
| 12 | SP70 | 0 | R/W | SP71: TPU0TO1 output value in motor stop pattern (7) |
| | | | | SP70: TPU0TO0 output value in motor stop pattern (7) |
| 11 | SP63 | 0 | R/W | Motor Stop Pattern (6) in Motor Control Mode |
| 10 | SP62 | 0 | R/W | SP63: TPU0TO3 output value in motor stop pattern (6) |
| 9 | SP61 | 0 | R/W | SP62: TPU0TO2 output value in motor stop pattern (6) |
| 8 | SP60 | 0 | R/W | SP61: TPU0TO1 output value in motor stop pattern (6) |
| | | | | SP60: TPU0TO0 output value in motor stop pattern (6) |
| 7 | SP53 | 0 | R/W | Motor Stop Pattern (5) in Motor Control Mode |
| 6 | SP52 | 0 | R/W | SP53: TPU0TO3 output value in motor stop pattern (5) |
| 5 | SP51 | 0 | R/W | SP52: TPU0TO2 output value in motor stop pattern (5) |
| 4 | SP50 | 0 | R/W | SP51: TPU0TO1 output value in motor stop pattern (5) |
| | | | | SP50: TPU0TO0 output value in motor stop pattern (5) |
| 3 | SP43 | 0 | R/W | Motor Stop Pattern (4) in Motor Control Mode |
| 2 | SP42 | 0 | R/W | SP43: TPU0TO3 output value in motor stop pattern (4) |
| 1 | SP41 | 0 | R/W | SP42: TPU0TO2 output value in motor stop pattern (4) |
| 0 | SP40 | 0 | R/W | SP41: TPU0TO1 output value in motor stop pattern (4) |
| | | | | SP40: TPU0TO0 output value in motor stop pattern (4) |

### 155.1.2.16 Motor Output Pattern Storing Register (TMOPR)

TMOPR is used to specify the motor output pattern in motor control mode. When read, it indicates the pattern currently being output. Writing to this register while the sequence is in the stop state leads to output of the pattern with the corresponding number. TMOPR is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | NOWPAT[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0 and cannot be modified. |
| 2 to 0 | NOWPAT [2:0] | 000 | R/W | In motor control mode,<br>Reading:<br>The pattern number currently being output is read.<br>Writing:<br>If the sequence is in the stop state, the pattern corresponding to the written pattern number is output. |

### 155.1.2.17 Motor Acceleration Step Count Register (TMASR)

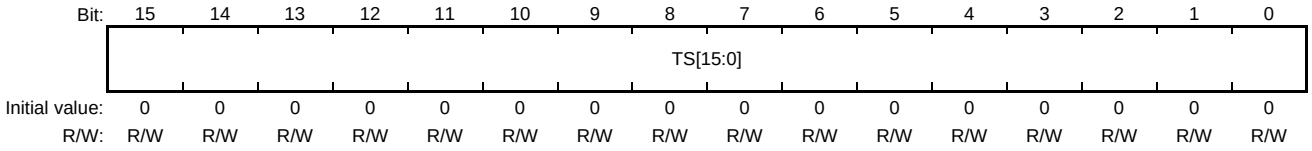
TMASR is used to set the number of steps in the acceleration state of motor control mode. TMASR is initialized to H'0000 by a reset.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AS[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | AS[15:0] | H'0000 | R/W | Specify the number of steps in the acceleration state of motor control mode.<br>When the setting is H'0000, the acceleration state is skipped. |

### 155.1.2.18 Motor Normal the Number of Steps Register (TMTSR)

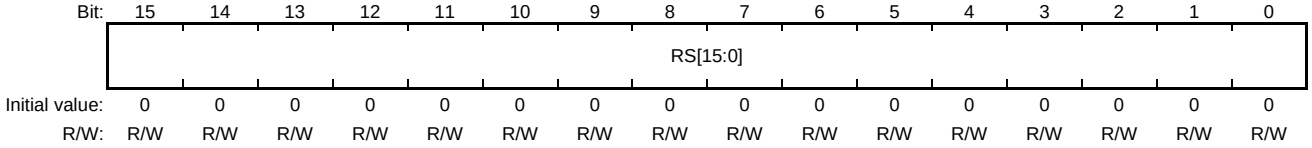
TMTSR is used to set the number of steps in the normal state of motor control mode. TMTSR is initialized to H'0000 by a reset.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | TS[15:0] | H'0000 | R/W | Specify the number of steps in the normal state of motor control mode.<br>When the setting is H'0000, the normal state is maintained until 1 is written to the REDUON0 bit or REDUON bit in TMRR. |

### 155.1.2.19 Motor Deceleration the Number of Steps Register (TMRSR)

TMRSR is used to set the number of steps in the deceleration state of motor control mode. TMRSR is initialized to H'0000 by a reset.

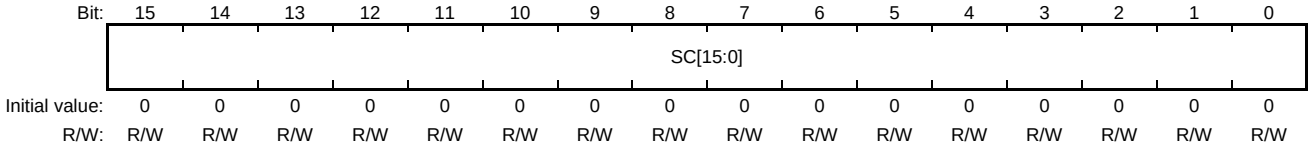



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | RS[15:0] | H'0000 | R/W | Specify the number of steps in the deceleration state of motor control mode.<br>When the setting is H'0000, the deceleration state is skipped. |

### 155.1.2.20 Motor Control Sequence Counter Register (TMSCR)

TMSCR indicates or controls the value of the sequence counter in motor control mode. Writing to this register is only allowed when the motor control mode is in use and the current state is the normal state.

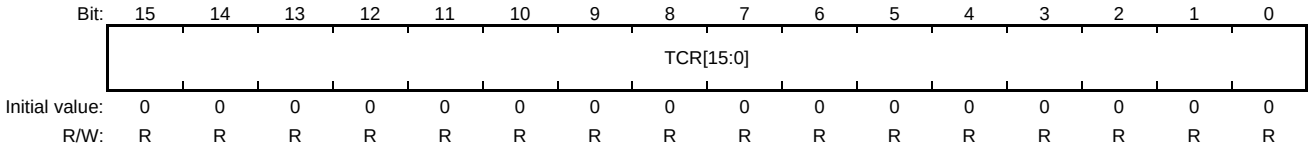
TMSCR is initialized to H'0000 by a reset.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | SC[15:0] | H'0000 | R/W | Value of Sequence Counter in Motor Control Mode<br>Reading:<br>The value of the sequence counter is read.<br>Writing:<br>Writing is only allowed in the normal state. The written value is directly stored in the sequence counter. |

### 155.1.2.21 Motor Control Normal Counter Register (TMTCR)

TMTCR is a read-only register that indicates the number of steps in the normal state of motor control mode. TMTCR is initialized to H'0000 by a reset.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 0 | TCR[15:0] | H'0000 | R | Indicate the number of steps in the normal state of motor control mode. |

Table 155.1.2.1 TPU Clock Sources

| | Internal Clock | | | |
|---|---|---|---|---|
| **Timer** | **Internal Clock /1** | **Internal Clock /4** | **Internal Clock /16** | **Internal Clock /64** |
| 0 | √ | √ | √ | √ |
| 1 | √ | √ | √ | √ |
| 2 | √ | √ | √ | √ |
| 3 | √ | √ | √ | √ |

[Legend]
√: Setting

Table 155.1.2.2 Count Clock Selection by the TPSC [2:0] Bits

| **Timer** | **TPSC[2]** | **TPSC[1]** | **TPSC[0]** | **Description** |
|---|---|---|---|---|
| 0 to 3 | 0 | 0 | 0 | Internal clock /1 (initial value) |
| | | | 1 | Internal clock /4 |
| | | 1 | 0 | Internal clock /16 |
| | | | 1 | Internal clock /64 |
| | 1 | * | * | Setting prohibited |

[Legend]
*: Don't care.

Table 155.1.2.3 Settings for Bits IOA [2:0], Initial States of Pins TPU0TO0 to TPU0TO3, and Results of Matching with TGRAn

| **Timer** | **IOA[2]** | **IOA[1]** | **IOA[0]** | **Description** | |
|---|---|---|---|---|---|
| 0 to 3 | 0 | 0 | 0 | Always output 0 (initial value) | |
| | | | 1 | Initial output from the TPU0TO0 to TPU0TO3 pins is 0 | Output 0 on compare match with TGRAn.* |
| | | 1 | 0 | | Output 1 on compare match with TGRAn. |
| | | | 1 | | Toggle output on compare match with TGRAn.* |
| | 1 | 0 | 0 | Always output 1 | |
| | | | 1 | Initial output from the TPU0TO0 to TPU0TO3 pins is 1 | Output 0 on compare match with TGRAn. |
| | | 1 | 0 | | Output 1 on compare match with TGRAn.* |
| | | | 1 | | Toggle output on compare match with TGRAn.* |

Note: * This setting is prohibited in PWM mode.

## 155.1.3 Operation

### 155.1.3.1 Overview

An overview of operation in the various modes is given below.

**(1) Ordinary Operation**

Each timer has TCNT and TGR registers. TCNTn counts up, and is capable of free-running operation and counting to define periods corresponding to register settings.

**(a) Buffered Operation**

When a compare match occurs, the buffer register value in the corresponding timer is transferred to TGR. Either a compare match or clearing of the counter can be selected as the trigger that defines the timing of updating from a buffer register.

**(b) PWM Mode**

In PWM mode, a PWM waveform is output. The duty cycle can be set by TGR. PWM waveforms with duty cycles in the range from 0 to 100% can be output in accord with the settings of TGRAn and TGRBn.

**(c) Stepping Motor Control Mode**

Pre-defined patterns can be output from TPU0TO3 to TPU0TO0 by placing timer 0 to stepping motor control mode.

### 155.1.3.2 Basic Functions

#### (1) Counter Operation

When a bit from among CST3 to CST0 in TSTR is set to 1, the TCNTn for the corresponding timer starts counting. A TCNTn can operate as a free-running counter, periodic counter, and so on.

#### (a) Example of Procedure for Setting up Counter Operations

Figure 155.1.3.1 shows an example of the procedure for setting up counter operation.

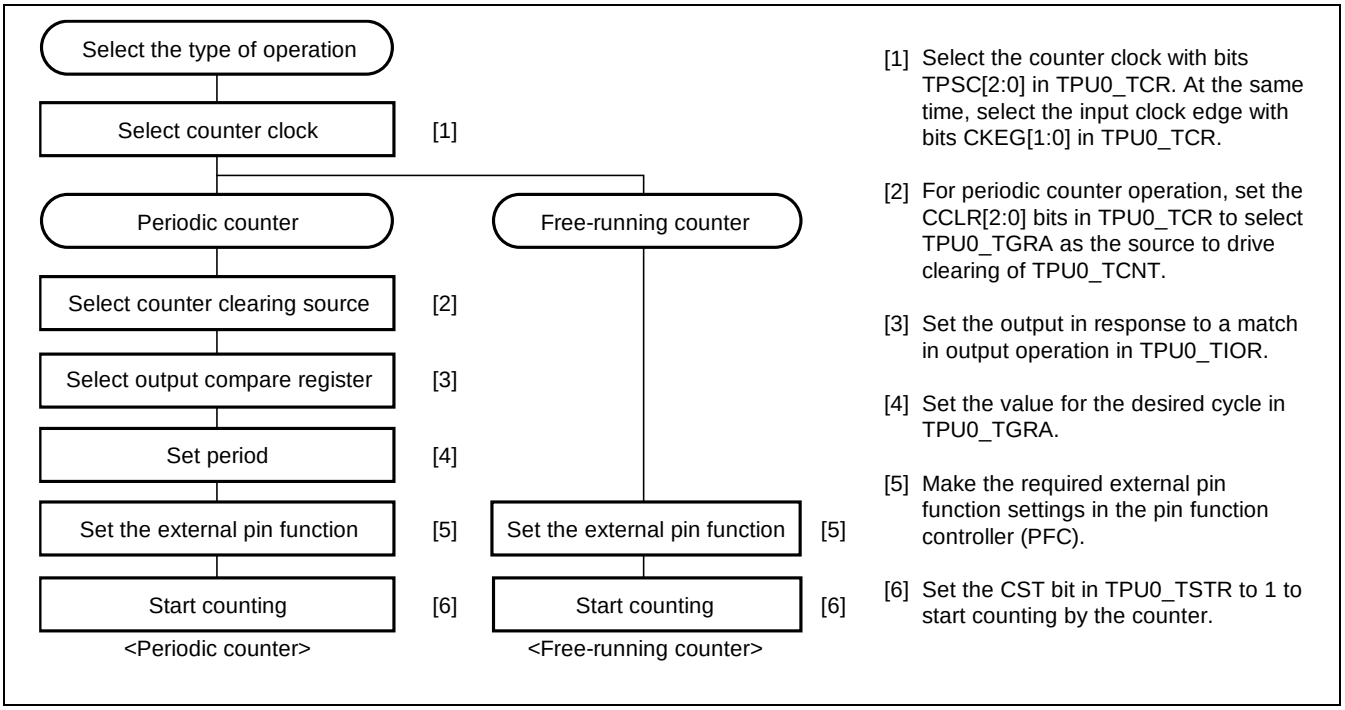



Figure 155.1.3.1 Example of Procedure for Setting up Counter Operation

**(b) Free-Running Counter Operation and Periodic Counter Operation**

Immediately after a reset, the TCNTn counters are all designated as free-running counters. When the relevant bit in TSTR is set to 1, the corresponding TCNTn starts counting up as a free-running counter. When TCNTn has overflowed (when its value changes from H'FFFF to H'0000), the TCFV bit in TSRn is set to 1. After it overflows, TCNTn starts counting up again from H'0000.

Figure 155.1.3.2 illustrates free-running counter operation.

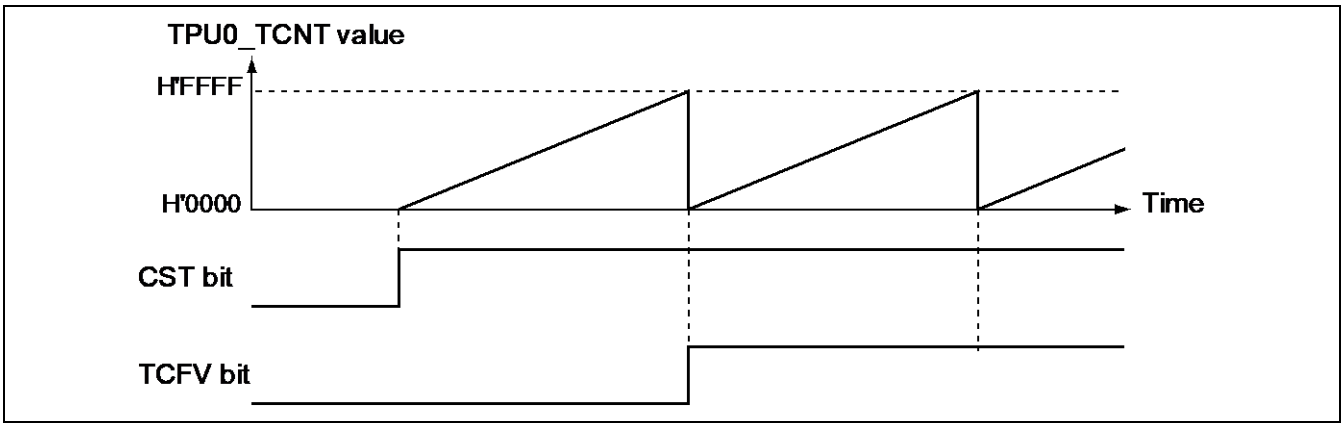



Figure 155.1.3.2 Free-Running Counter Operation

When a compare match is selected as the source to drive clearing of TCNTn, the TCNTn counter for the relevant timer operates as a periodic counter (counting a period defined by a general register). The TGR register for setting the period is designated as the output compare register, and counter clearing in response to a compare match is selected by means of bits CCLR[2:0] in TCRn. After the settings have been made, TCNTn starts counting as a periodic counter when the corresponding bit in TSTR is set to 1. When the counted value matches the value in the selected TGR, the TGF bit in TSRn is set to 1 and TCNTn is cleared to H'0000.

After a compare match, TCNTn starts counting up again from H'0000.

Figure 155.1.3.3 illustrates periodic counter operation.

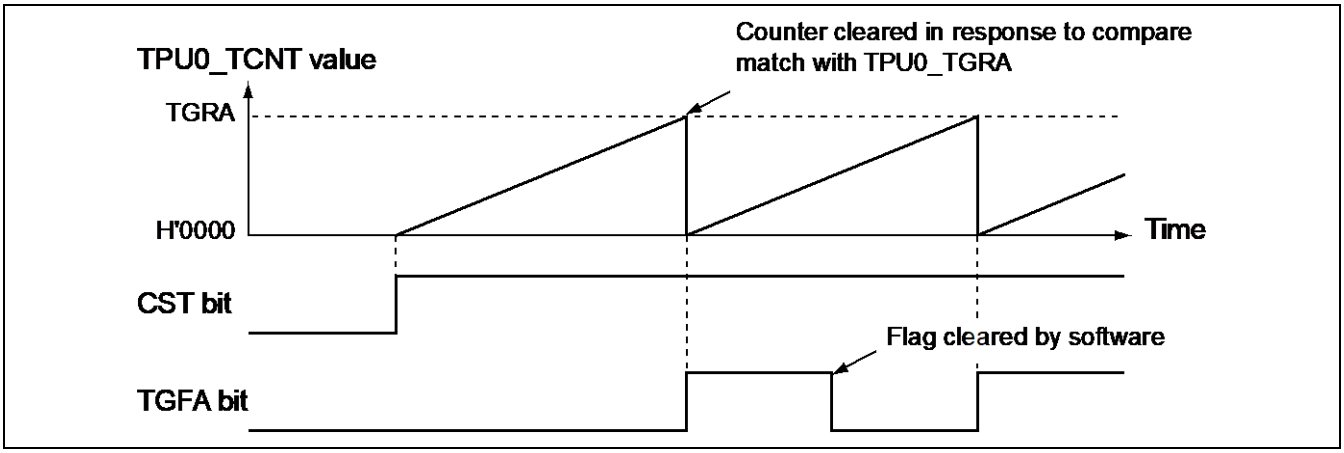



Figure 155.1.3.3 Periodic Counter Operation

### (2) Waveform Output by Compare Match

The TPU can be set for the output of logical 0 or 1, or toggling of the output on the output pin in response to a compare match with TGRAn.

#### (a) Example of Procedure for Setting up Waveform Output under Compare Match Control

Figure 155.1.3.4 shows an example of the procedure for setting up waveform output under compare match control.

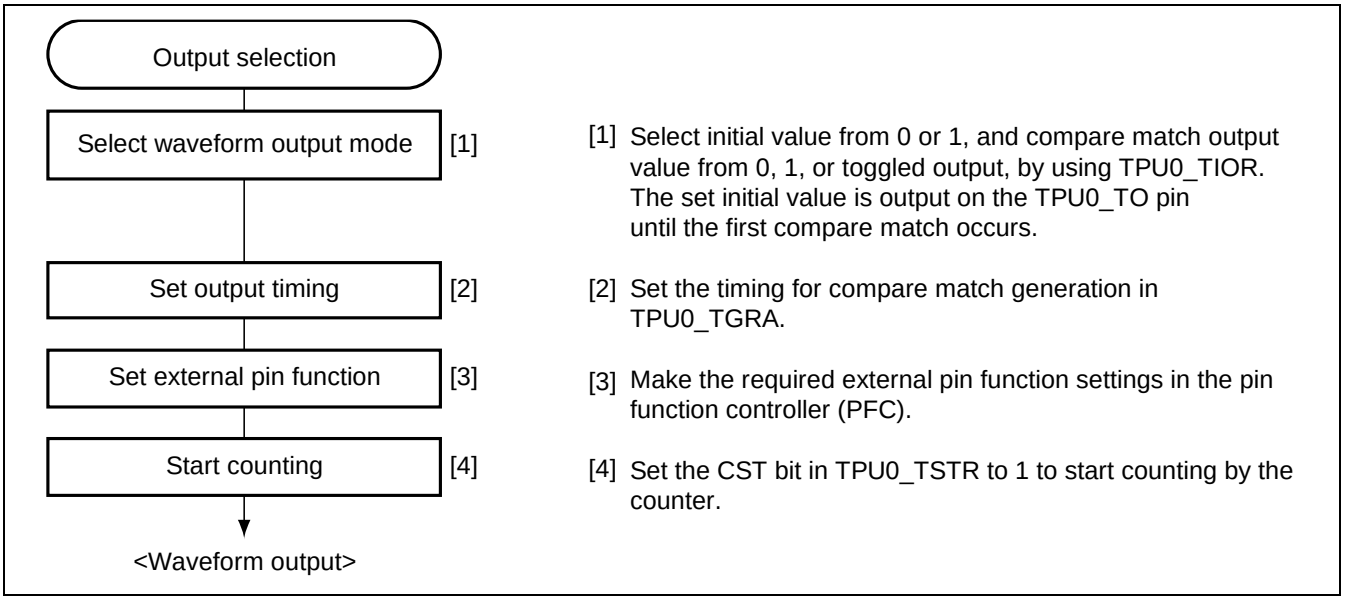



**Figure 155.1.3.4 Example of Procedure for Setting up Waveform Output under Compare Match Control**

#### (b) Examples of Waveform Output Operation

Figure 155.1.3.5 shows an example of the output of 0 or 1.

In this example, TCNTn has been designated as a free-running counter, and settings have been made so that 1 or 0 is output in response to compare match A. When the set level is the same as the level on the pin, the level on the pin does not change.

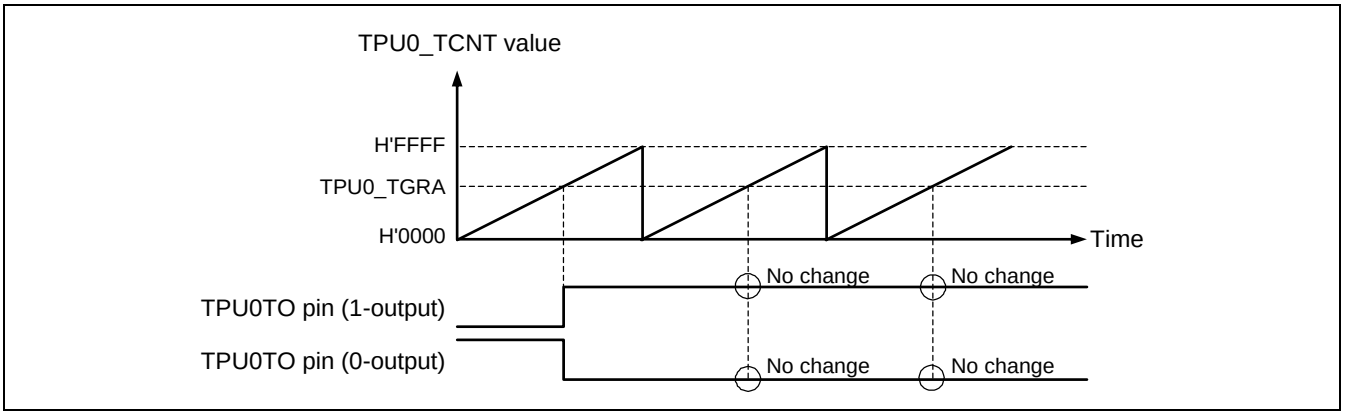



**Figure 155.1.3.5 Example of Output of 0 or 1**

Figure 155.1.3.7 shows an example of toggled output.

In this example, TCNTn has been designated as a periodic counter (with counter clearing handled by compare match B), and settings have been made so that the output is toggled in response to compare match A.

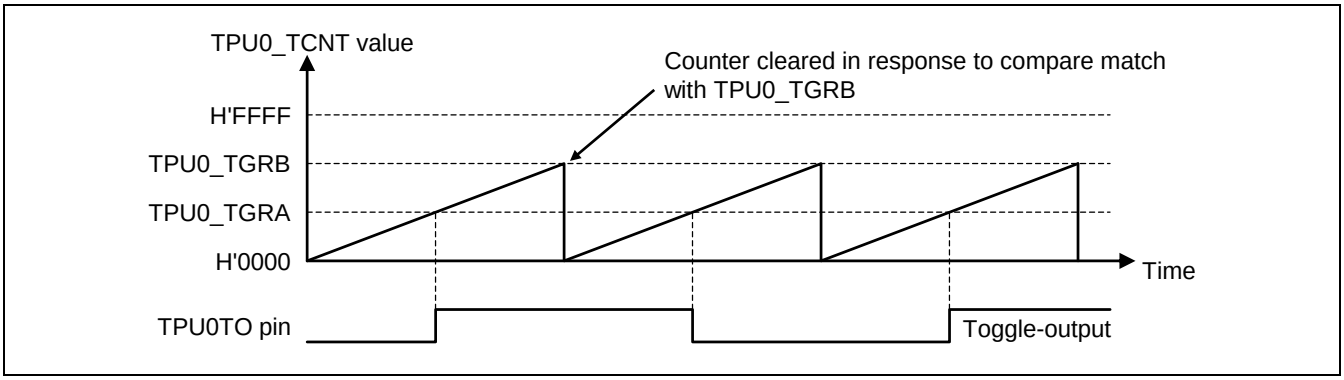



**Figure 155.1.3.6 Example of Toggled Output Operation**

### 155.1.3.3 Buffered Operation

TGRCn and TGRDn can be used as buffer registers.

Table 155.1.3.1 shows the register combinations used in buffered operation.

**Table 155.1.3.1 Register Combinations in Buffered Operation**

| Timer General Register | Buffer Register |
|---|---|
| TGRAn | TGRCn |
| TGRBn | TGRDn |

When a compare match occurs, the value in the buffer register for the corresponding timer is transferred to the timer general register. Either a compare match or clearing of the counter can be selected as the trigger that defines the timing of updating from a buffer register.

This operation is illustrated in Figure 155.1.3.7.

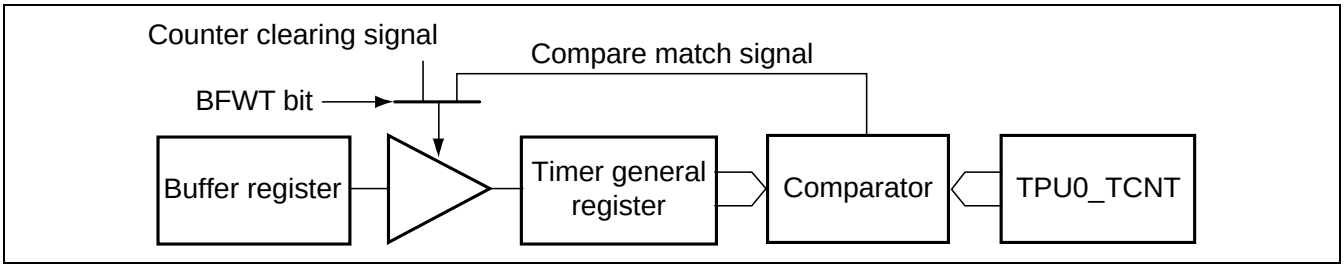



**Figure 155.1.3.7 Buffered Compare Match Operation**

**(1) Example of Procedure for Setting up Buffered Operation**

Figure 155.1.3.8 shows an example of the procedure for setting up buffered operation.

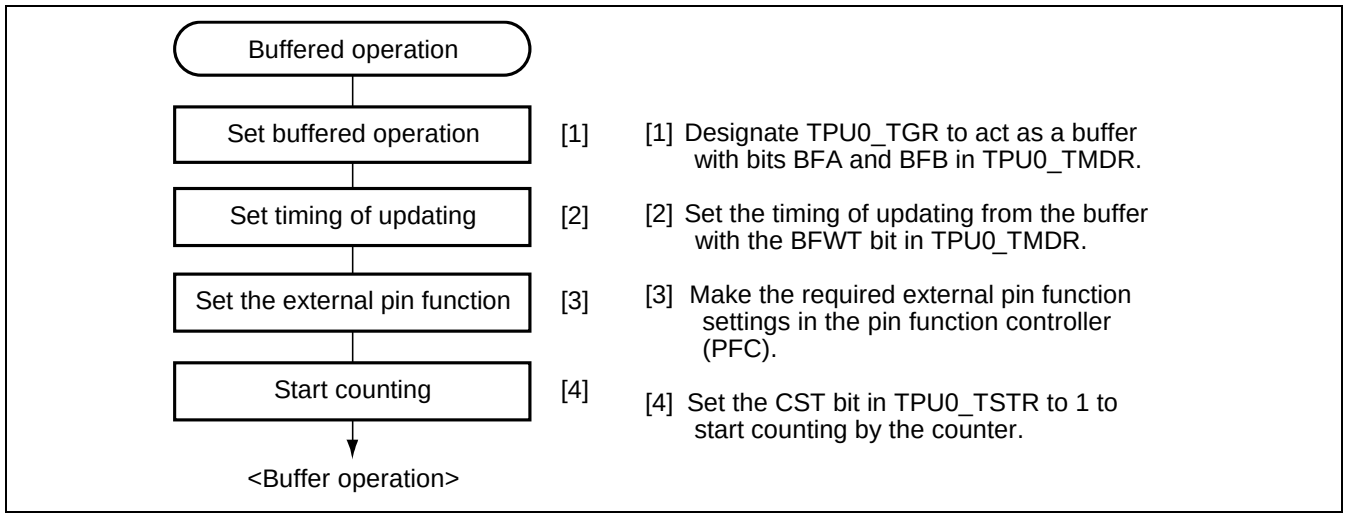



Figure 155.1.3.8 Example of Procedure for Setting up Buffered Operation

**(2) Examples of Buffered Operation**

Figure 155.1.3.9 shows an example of operation in which PWM mode has been designated for timer 0, and buffered operation has been designated for TGRAn and TGRCn. The settings used in this example are for clearing of TCNTn in response to compare match B, output of 1 (on the TPU0TO0 to 3 for the selected timer) on compare match A, output of 0 as the initial value by clearing the counter, and the timing for updating from the buffer register as clearing of the counter.

When compare match A occurs, the output changes. When counter clearing is generated by a match with TGRBn, the output changes and the value in buffer register TGRCn is simultaneously transferred to the timer general register TGRAn. This operation is repeated every time compare match A occurs.

For details on the PWM mode, see section 155.1.3.4, PWM Mode.

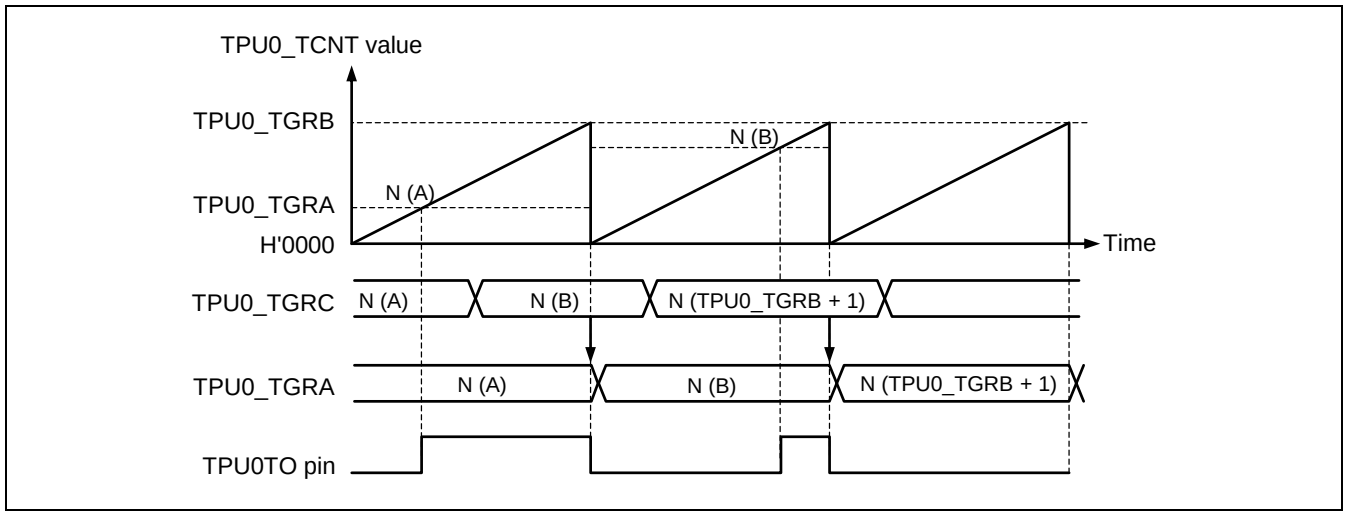



Figure 155.1.3.9 Example of Buffer Operation

#### 155.1.3.4 PWM Mode

In PWM mode, PWM waveforms are output from the output pins. The output of 0 or 1 can be selected as the response to a compare match with the given TGRAn.

Designating TGRBn compare match as the counter clearing source enables setting of the overall cycle in that register. All timers can be independently placed in PWM mode.

PWM output is generated from the output pins by using TGRAn and TGRBn to control the duty cycle and period, respectively. The initial output specified by TIORn is output on the output pin by counter clearing in response to a compare match with the period register. Be sure to set TIORn so that the initial output level is different from the compare match output. Selecting the same level or toggled output leads to no operation.

Conditions for 0% and 100% duty cycles are shown below.

- 0% duty cycle: The setting of the duty-cycle register (TGRAn) being equal to that of the period register plus one (TGRBn + 1)
- 100% duty cycle: The setting of the duty-cycle register (TGRAn) being 0

**(1) Example of Procedure for Setting up PWM Mode Operation**

Figure 155.1.3.10 shows an example of the procedure for setting up the PWM mode operation.

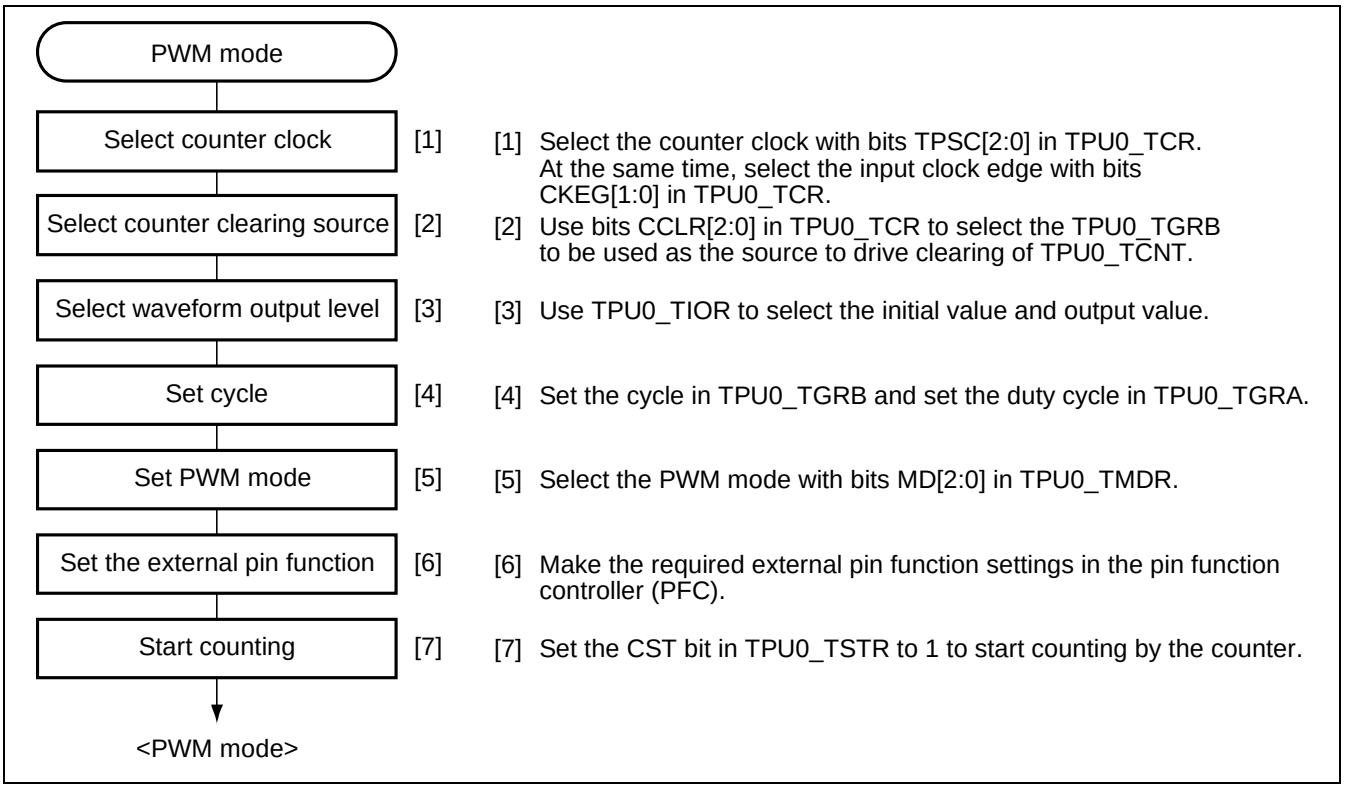



Figure 155.1.3.10 Example of Procedure for Setting up PWM Mode Operation

**(2) Examples of PWM Mode Operation**

Figure 155.1.3.11 shows an example of PWM mode operation.

In this example, compare match with TGRBn is set as the source to drive clearing of TCNTn, 0 is set as the initial output value for TGRAn, and 1 is set as the output value on a compare match with TGRAn.

In this case, the value set in TGRBn defines the cycle time (period), and the value set in TGRAn defines the duty cycle.

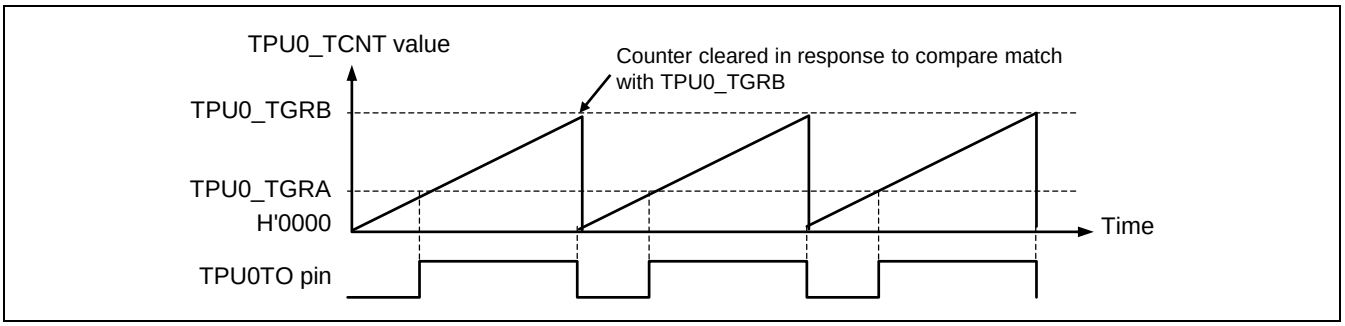



Figure 155.1.3.11 Example of PWM Mode Operation (1)

Figure 155.1.3.12 shows an example of PWM waveform output with 0% and 100% duty cycles in PWM mode.

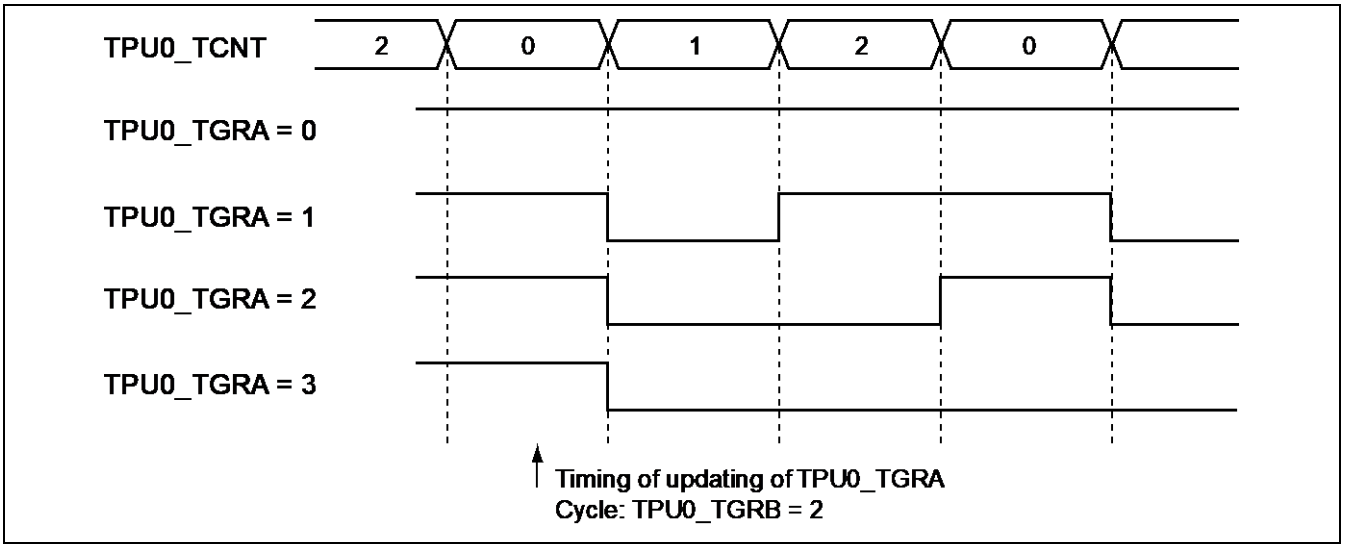



Figure 155.1.3.12 Example of PWM Mode Operation (2)

### 155.1.3.5 Stepping-Motor Control Mode

**(1) Features of Stepping-Motor Control Mode**

- This mode is for the output of signals for stepping-motor control on four pins and is only available for timer 0.
- States of the output pins change in response to compare matches with the counter of timer 0.
- By using the DMAC or an interrupt to rewrite the values of the compare-match registers, the intervals between transitions of the states of the output pins can be specified as desired.
- Specified patterns are output on the output pins.
- Four or eight patterns can be selected.
- The pattern-number counter is incremented or decremented by 1 on every compare match with the counter on timer 0. When the pattern-number counter overflows in normal mode, its value returns to 0. When the pattern-number counter underflows, its value becomes 3 (for a four-pattern cycle) or 7 (for an eight-pattern cycle).
- Patterns consist of stop patterns and operating patterns. Operating patterns are output while the counter is running, and stop patterns are output while the counter is stopped.
- While the counter is running, operation is in three states (the acceleration, deceleration, and normal states). Only for acceleration and deceleration, the interval between changes of pattern can be controlled.
- The acceleration or deceleration state can be skipped by writing H'0000 to TMASR and TMRSR.
- TMRR can be used to force a transition from the normal to the deceleration state.

**(2) Glossary**

Sequence State: The four states in the sequence, i.e. the acceleration, normal, deceleration, and stop states

Sequence Counter: Counter to control transitions to next sequence states
The sequence counter is decremented with every compare match with the counter of timer 0 and, when the value reaches 1, the control state will make the transition to the next state in the sequence on a further compare match with the counter of timer 0.

Auto Mode: Mode of operation when a value other than H'0000 has been written to TMTSR
The number of pattern transitions in the normal state corresponding to the setting of TMTSR, and this is followed by an automatic transition to the deceleration state (or, if TMRSR = H'0000, a transition to the stop state).

Manual Mode: Mode of operation when H'0000 has been written to TMTSR. In this case, the transition from the normal to the deceleration state (or the stop state if TMRSR = H'0000) is not made unless 1 is written to the REDUON0 and REDUON bits in TMRR.

Pulse Mode: Mode of operation when H'0000 has been written to TMASR and TMRSR. Operation is only in the normal state, so the output pattern is changed with a constant interval.

Step: The pattern is changed with every compare match with the counter of timer 0. Step refers to the interval of time between the change to one pattern and the change to the next.

Operation Pattern: The four-bit patterns for output from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the acceleration, normal, and deceleration states. 8 types of Operation Pattern are set by TMMPR0 and TMMPR1. “[ ]” in Figure 155.1.3.13, Figure 155.1.3.15, Figure 155.1.3.17, Figure 155.1.3.19 to Figure 155.1.3.22 and Table 155.1.3.5 means 8 types of Operation Pattern set by TMMPR0 and TMMPR1.

Stop Pattern: The four-bit patterns for output from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the stop state. 8 types of Operation Pattern are set by TMSPR0 and TMSPR1. “( )” in Figure 155.1.3.13, Figure 155.1.3.15, Figure 155.1.3.17, Figure 155.1.3.19 to Figure 155.1.3.22 and Table 155.1.3.5 means 8 types of Operation Pattern set by TMMPR0 and TMMPR1.

**(3) Setting up Basic Motor Control Operation**

An example of operation in stepping motor control is shown in Figure 155.1.3.13.

Stepping-motor control mode allows the output of patterns with a variable step interval.

"( )" in Figure 155.1.3.13 means Stop Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the stop state set by TMSPR0 and TMSPR1.

"[ ]" in Figure 155.1.3.13 means Operation Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the acceleration, normal, and deceleration states set by TMMPR0 and TMMPR1.

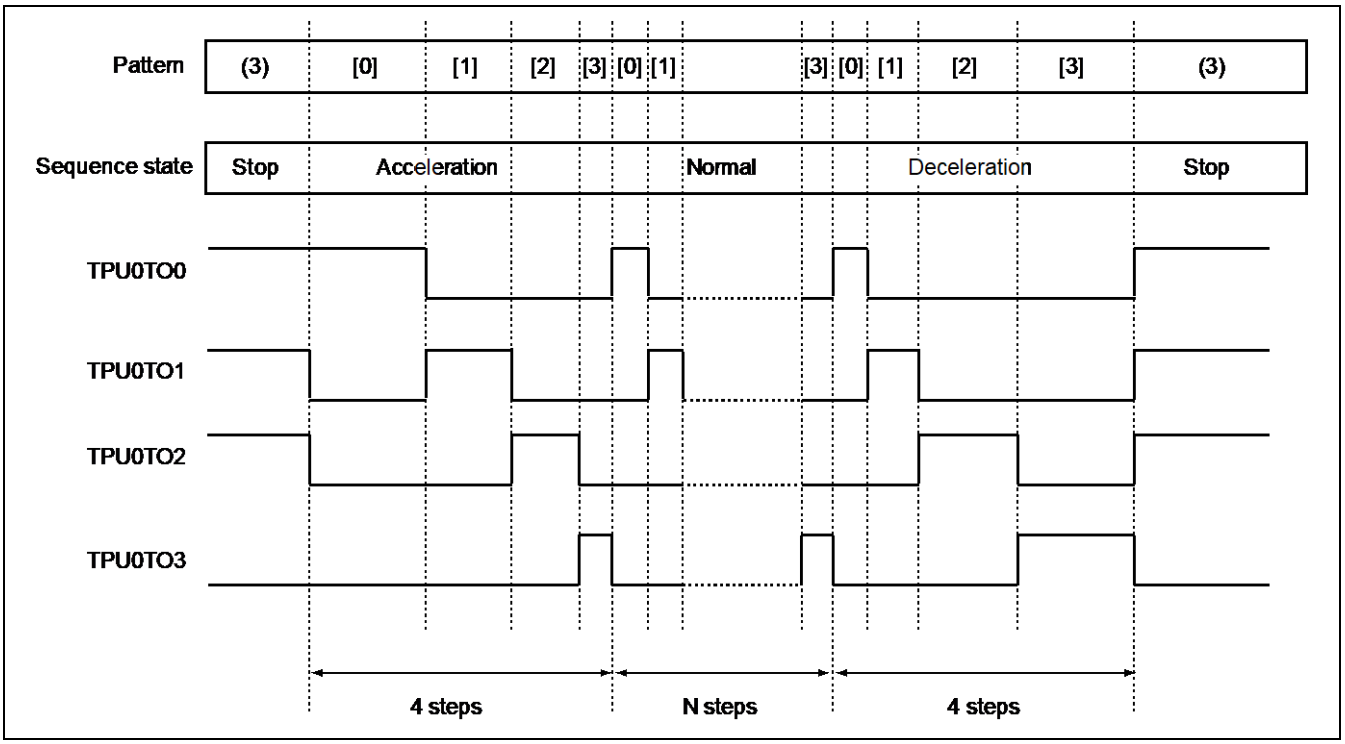



Figure 155.1.3.13 Example of Stepping-Motor Control

**(a) Compare Match Settings for Timer 0**

For stepping motor control, set TCR0 and TMDR0 as follows:

Table 155.1.3.2 Compare Match Settings for Timer 0

| Register | Setting | Description |
|---|---|---|
| TCR0 | B'0000 0000 0100 0xxx | CCLR[2:0] bits = B'010<br>CKEG[1:0] bits = B'00<br>TPSC[2:0] bits = B'xxx (optional) |
| TMDR0 | B'0000 0000 0110 0000 | BFWT bit = 1<br>BFB bit = 1<br>BFA bit = 0<br>MD[2:0] bits = B'000 |

[Legend]
x: Don't care.

With the above settings, TGRD0 is used as a buffer register for TGRB0, and the contents of the general register are rewritten when the timer counter is cleared. The step interval can be changed by using DMA transfer or interrupt control to write a desired value to TGRD0.

The TPSC [2:0] bits in TCR0 are for a timer prescaler setting. The step interval is calculated from the following formula.

**[Calculation example]**

With Internal clock at 99MHz, B'011 as the value of TPSC[2:0] in TCR0, H'1425 as the value of TGRD0, the result is 15ns × 64 × 5157 (H'1425) = 4950720ns ≅ 4.95ms. That is, the pattern will change after approximately 4.95ms.

**(b) Sequence State and the Number of Steps**

In stepping-motor control mode, operation is in three states (acceleration, normal, and deceleration) while the counter is running. In the acceleration and deceleration states, the step interval can be changed. Accordingly, the value of TGRD0 must be changed the same number of times as the patterns are changed.

In the normal state, the step interval cannot be changed. So, TGRD0 is only set once for a period in the normal state.

The numbers of steps in the acceleration, normal, and deceleration states are set in TMASR, TMTSR, and TMRSR, respectively. If H'0000 is written to TMASR or TMRSR, the corresponding state is skipped. Furthermore, the manual mode can be selected by writing H'0000 to TMTSR. In this mode, the transition from normal mode can only be initiated by changing the value in TMRR.

Table 155.1.3.3 gives descriptions of the sequence states and Figure 155.1.3.14 shows the transitions between sequence states.

Table 155.1.3.3 Description of Sequence States

| | | Description | |
|---|---|---|---|
| **Start/Stop** | **State** | **Step Interval** | **Definition in TGRD0** |
| Start | Acceleration | Variable | The number of steps specified for this state |
| | Normal | Fixed | Only once |
| | Deceleration | Variable | The number of steps specified for this state |
| Stop | Stop | — | — |

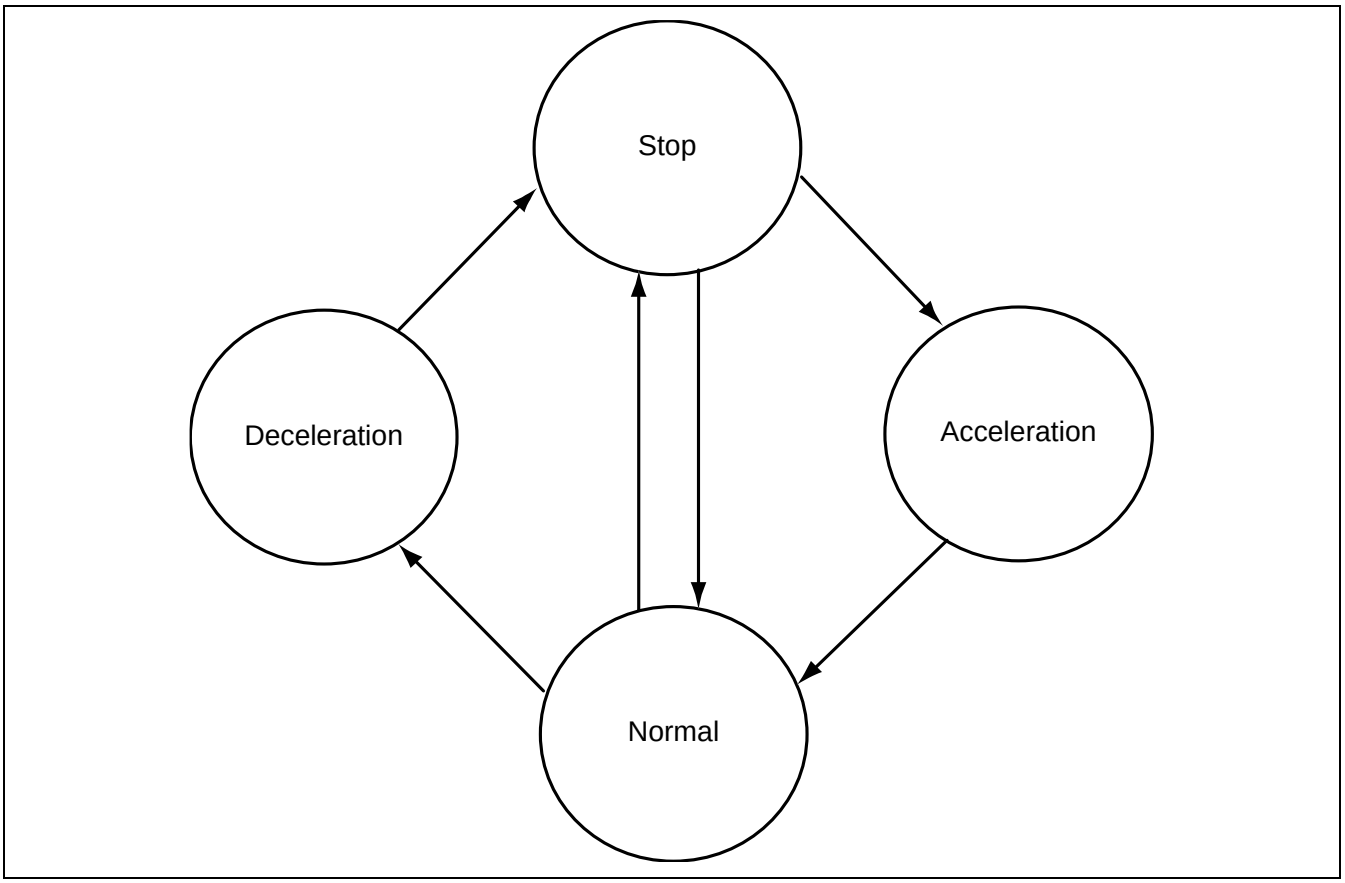



Figure 155.1.3.14 Sequence State Transition

**(c) Setting of the Step Interval**

Direct memory access or interrupt-driven transfer can be used to set the step interval

- DMA transfer

Setting the TDMAE bit in TMIR to 1 selects the use of DMA transfer to modify the step interval.

The number of requests for DMA transfer is given by TMASR + 1 (to transfer the step interval for the normal state) + TMRSR. The step intervals must be defined in sequence from the address indicated by the DMA source address register. Figure 155.1.3.15 shows an example of timing in operation with DMA transfer requests used to set the step intervals. "( )" in Figure 155.1.3.15 means Stop Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the stop state set by TMSPR0 and TMSPR1. "[ ]" in Figure 155.1.3.15 means Operation Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the acceleration, normal, and deceleration states set by TMMPR0 and TMMPR1. Figure 155.1.3.16 shows the corresponding definitions of step intervals in the address map.

Table 155.1.3.4 lists an example of the DMA settings.

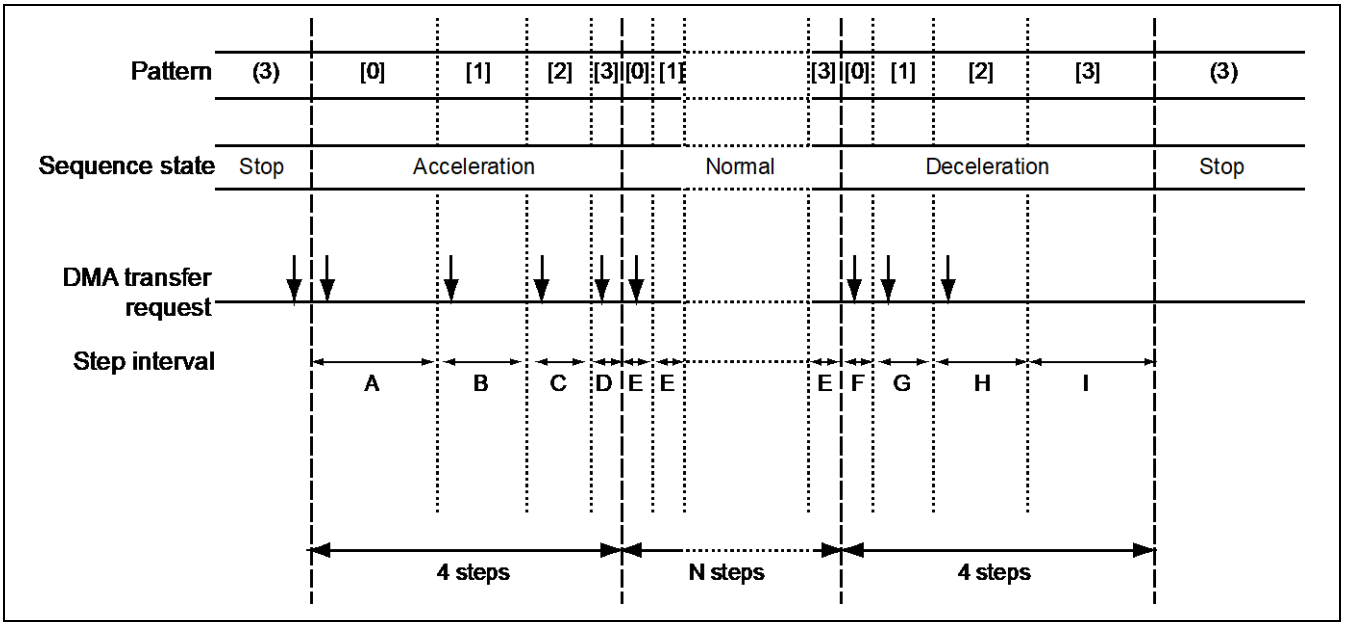



Figure 155.1.3.15 Operation with DMA Transfer Requests Used to Adjust the Step Interval

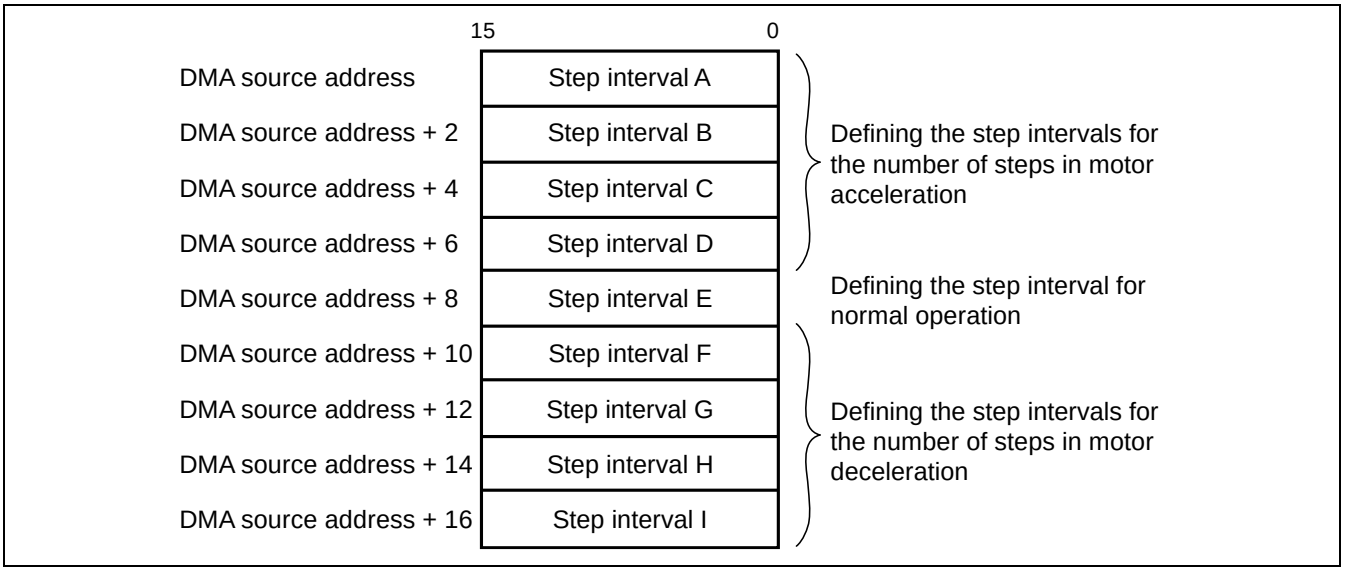



Figure 155.1.3.16 Address Map of Step Interval Definitions for the Example of Operation

Table 155.1.3.4 DMA Settings

| DMA Register | Setting | Description |
|---|---|---|
| DMA source address register | H'XXXX XXXX | Address where the table of step intervals starts |
| DMA destination address register | H'E6E8 0034 | Address of TGRD0 |
| DMA transfer count register | H'0000 0000 | (Setting for the maximum number of transfer operations) * |
| DMA transfer count register B | H'XXXX XXXX | The number of DMA transfer requests is set in the higher- and lower-order 16 bits. (for Figure 155.1.3.15, H'0009 0009) |
| DMA channel control register | H'E6E8 80*C | Reload mode:<br>Use source address registers (SARs) as the source for reloading.<br>Fixed destination address.<br>Incrementation of source addresses.<br>The DMA extension resource selector is used.<br>DMA transfer is in word (two-byte) units.<br>The other bits are in the default settings. |
| DMA extension resource selector | H'17 in the bit field for the selected DMA channel | Select TPU |
| DMA operation register | H'0001 | Permits DMA transfer on all channels.<br>The other bits are in the default settings |

Note: * Transfer is stopped when the value of the DMA transfer counter reaches 0, so change the value of the DMA transfer counter back to 1 after TPU-related operations are finished.

- Interrupt-driven transfer

Setting the TMDRFE bit in TIER0 to 1 allows the use of interrupt-driven transfer to change the step interval.

TMASR−1 + 1 (to transfer the step interval in the normal state) + TMRSR gives the number of times TGRD0 must be written to by interrupt-driven transfer.

This method is unlike DMA transfer in that an interrupt is not generated when the initial transfer is required. The initial value should thus be written to TGRD0 before counter operation starts.

Once interrupt generation starts, the interrupt handler writes the step intervals to TGRD0 and writes 0 to the TMDRFS bit after reading it as 1.

Figure 155.1.3.17 shows an example of timing in operation with interrupt-driven transfer used to set the step intervals. “( )” in Figure 155.1.3.17 means Stop Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the stop state set by TMSPR0 and TMSPR1. “[ ]” in Figure 155.1.3.17 means Operation Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the acceleration, normal, and deceleration states set by TMMPR0 and TMMPR1.

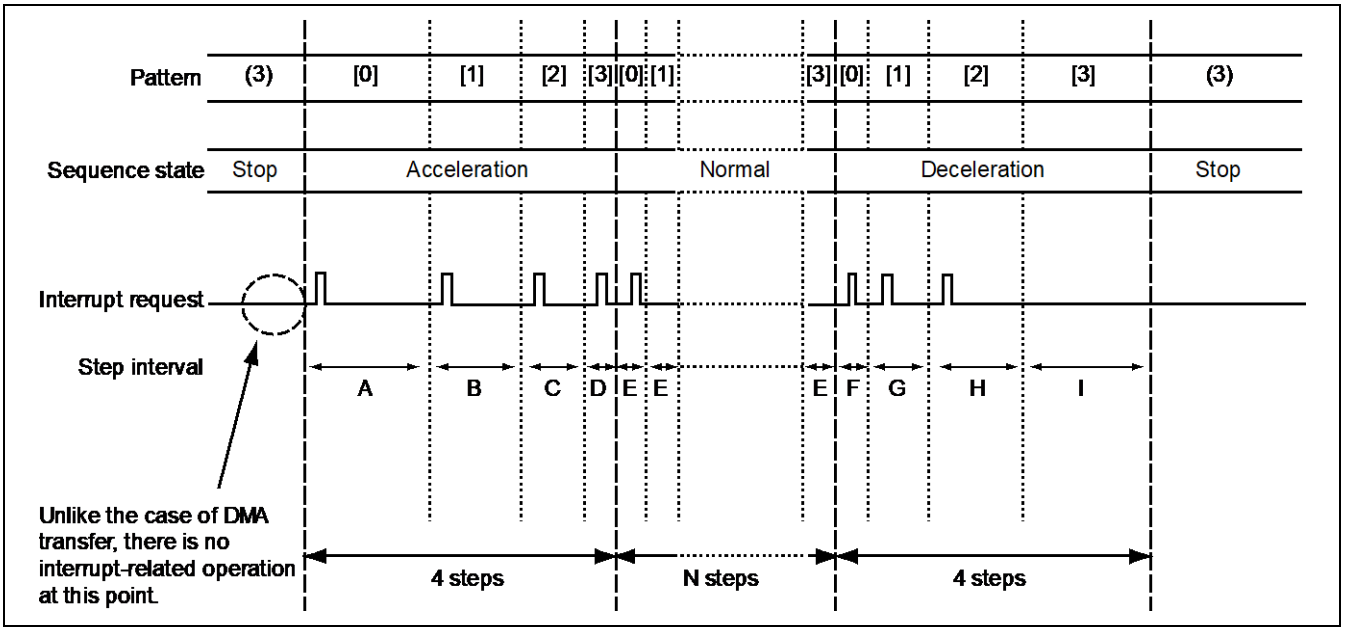



Figure 155.1.3.17 Interrupt-Driven Adjustment of the Step Interval

The step-intervals in the example of operation shown in Figure 155.1.3.17 are defined in the following stages.

| | |
|---|---|
| Write step interval A to TGRD0 before starting the counter.<br>Write step interval B to TGRD0 in response to the first interrupt.<br>Write step interval C to TGRD0 in response to the second interrupt.<br>Write step interval D to TGRD0 in response to the third interrupt. | Defining the step intervals for the number of steps in motor acceleration |
| Write step interval E to TGRD0 in response to the forth interrupt. | Defining the step interval for normal operation |
| Write step interval F to TGRD0 in response to the fifth interrupt.<br>Write step interval G to TGRD0 in response to the sixth interrupt.<br>Write step interval H to TGRD0 in response to the seventh interrupt.<br>Write step interval I to TGRD0 in response to the eighth interrupt. | Defining the step intervals for the number of steps in motor deceleration |

**(d) Pattern Setting**

In stepping-motor control mode, each four-bit value for output on pins TPU0TO0 to TPU0TO3 is regarded as a single pattern. The patterns are in a sequence, and a compare match with the timer counter triggers each transition to the next pattern.

The following rules apply to pattern transfer.

- Either a four- or an eight-pattern cycle may be used. The number is determined by the MTRPATKIND bit in TMIR0.
- Each pattern is indicated by a pattern number. The current pattern number is either incremented or decremented on each compare match with the timer counter. Incrementation or decrementation of the pattern numbers is determined by the MTRPATDOWN bit in TMIR0.
- The pattern-number counter returns to 0 when it overflows. When it underflows, the pattern number becomes three (for a four-pattern cycle) or seven (for an eight-pattern cycle).
- Operating patterns are defined as the output patterns in the acceleration, normal, and deceleration states, and stop patterns are defined as the output patterns in the stop state. In transitions from an operating pattern to a stop pattern, the pattern number remains the same. In transitions from a stop pattern to an operating pattern, the current pattern number changes to that of the next pattern.

The operating patterns are set in TMMPR0 and TMMPR1, and the stop patterns are set in TMSPR0 and TMSPR1.

Table 155.1.3.5 lists the order of patterns for different settings of the MTRPATDOWN and MTRPATKIND bits in TMIR.

**Table 155.1.3.5 Settings of the MTRPATDOWN and MTRPATKIND Bits in TMIR and Orders of Patterns**

| TMIR | | Example of Pattern Order |
|---|---|---|
| MTRPATDOWN Bit | MTRPATKIND Bits | (Operating Pattern [ ], Stop Pattern ( )) |
| 0 | B'00 | [0] → [1] → [2] → [3] → [0] → [1] → [2] → [3] → (3) → [0] |
| 1 | | [0] → [3] → [2] → [1] → [0] → [3] → [2] → [1] → (1) → [0] |
| 0 | B'01 | [0] → [1] → [2] → [3] → [4] → [5] → [6] → [7] → (7) → [0] |
| 1 | | [0] → [7] → [6] → [5] → [4] → [3] → [2] → [1] → (1) → [0] |

**(e) Specifying the Output Pattern in the Stop State**

Specifying a pattern number in the NOWPAT [2:0] bits of TMOPR while output is in the stop state leads to output of the corresponding pattern on pins TPU0TO3 to TPU0TO0. The initial value is 0.

**(f) Selecting the Stepping-Motor Control Mode**

Stepping-motor control by the TPU circuit is selected by setting the MTRON bit in TMIR to 1. The output on pins TPU0TO3 to TPU0TO0 is switched to the output corresponding to stepping-motor control.

Switching should proceed while the motor is stopped.

The stop pattern specified as pattern 0 in TMOPR is output after switching. Therefore, verify the value in TMOPR or write the desired pattern to TMOPR.

**(g) Starting Stepping-Motor Control**

Stepping-motor control is started by setting the TMST bit in TSTR to 1. This bit is automatically cleared (to 0) in the stop state after motor-control operations are over.

**(h) Example of the Procedure for Setting up Stepping-Motor Control**

Figure 155.1.3.18 is a flowchart of stepping-motor control.

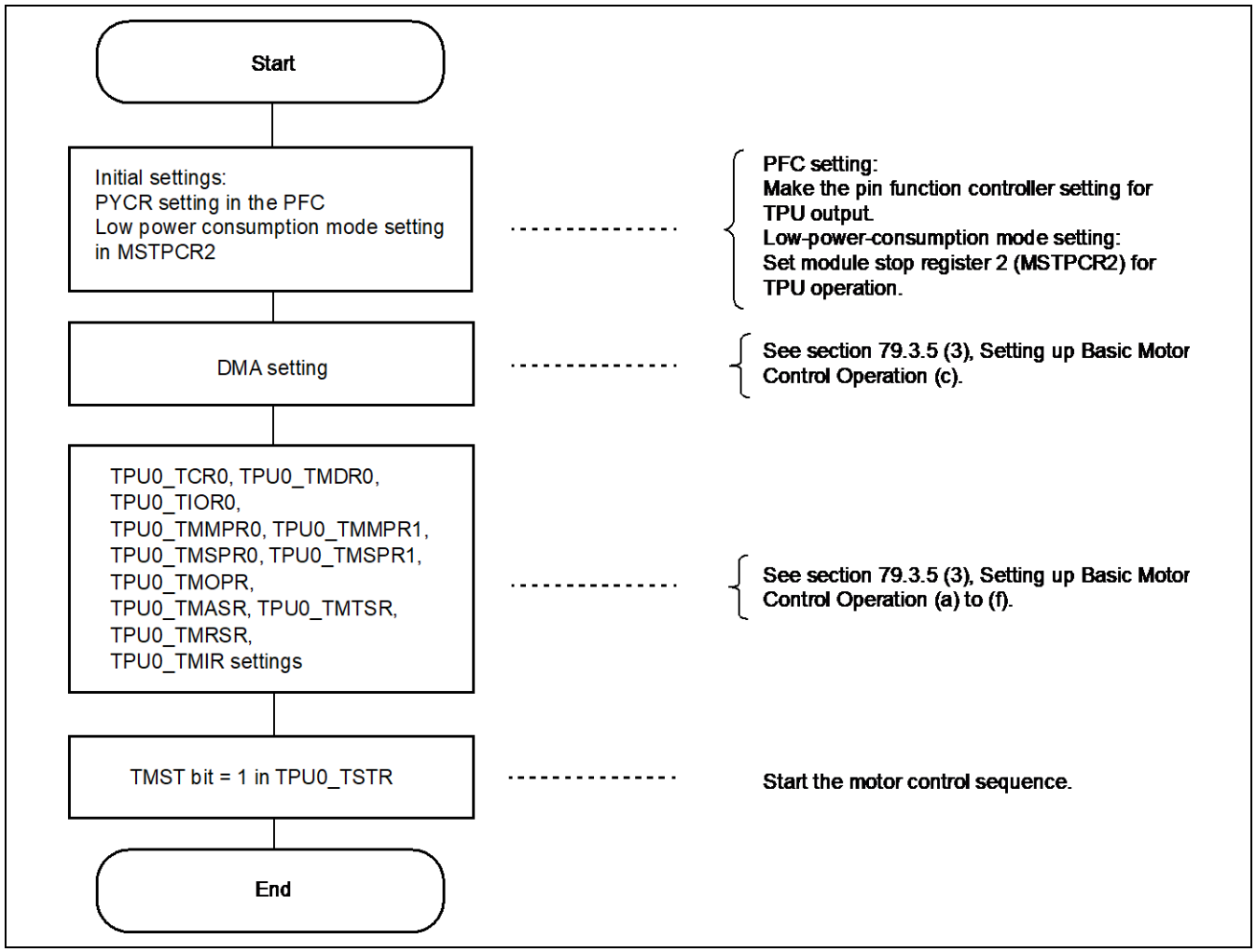



Figure 155.1.3.18 Flowchart

**(4) Changing the Number of Steps in the Normal State.**

The number of steps in the normal state is set in TMTSR. The operations for changing the number of steps are described under (a) and (b) below.

In manual mode (TMTSR = H'0000), the operation under (a) is the only way of switching from the normal state.

**(a) Changes by TMRR**

Writing 1 to the REDUON0 bit or the REDUON bit in TMRR initiates the transition to the deceleration state (or, if TMRSR = H'0000, to the stop state).

Table 155.1.3.6 describes how operation in response to the writing of 1 to the REDUON0 bit varies with the operating state. Table 155.1.3.7 describes how operation in response to the writing of 1 to the REDUON bit varies with the operating state.

Table 155.1.3.6 Timing of Writing 1 to the REDUON0 Bit and Operation

| Timing of Writing 1 to the REDUON0 Bit | Operation |
|---|---|
| Acceleration | Operation continues normally until the end of acceleration.<br>Control then enters the normal state. |
| Normal | Control remains in the normal state until the pattern number becomes [0]. The transition to the deceleration state (or the stop state if TMRSR is H'0000) follows the end of this step in the normal state. |
| Deceleration | Writing 1 to the bit has no effect on deceleration. |
| Stop | TMRR is cleared to H'0000 after control has made the transition from the normal or deceleration state to the stop state.<br>When 1 is written to the REDUON0 bit after control is in the stop state, the sequence in the next round of motor control continues until the pattern number in the normal state becomes [0]. The transition to the deceleration state (or the stop state if TMRSR is H'0000) is made at the end of this step in the normal state. |

Table 155.1.3.7 Timing of Writing 1 to the REDUON Bit and Operation

| Timing to Write 1 in REDUON Bit | Operation |
|---|---|
| Acceleration | Operation continues normally until the end of acceleration. |
| Normal | The transition to the deceleration state (or the stop state if TMRSR is H'0000) is made after a single step in the normal state. |
| Deceleration | Writing 1 to the bit has no effect on deceleration. |
| Stop | TMRR is cleared to H'0000 after control has made the transition from the normal or deceleration state to the stop state.<br>When 1 is written to the REDUON0 bit after control is in the stop state, the sequence in the next round of motor control continues until it enters the normal state. The transition to the deceleration state (or the stop state if TMRSR is H'0000) is made at the end of a single step in the normal state. |

**(b) Changing the Number of Steps by Changing the Value in TMSCR**

In auto mode (i.e. when TMTSR ≠ H'0000), TMSCR indicates the remaining number of steps in the current state.

When the control sequence is in the normal state, the number of steps can be changed by writing the desired number of steps. Writing is ineffective in states other than the normal state.

Do not write to this register when control is in manual mode (TMTSR = H'0000).

**(5) Acquiring the Number of Steps in the Normal State**

In the stop state, TMTCR can be read to acquire the number of steps in the normal state of the immediately preceding sequence.

**(6) Pulse Mode**

Writing H'0000 to TMASR and TMRSR selects operation in the normal state alone, with a constant step interval. That is, operation starts with the step interval defined for the normal mode and the interval does not subsequently change. This is referred to as pulse mode operation.

Examples of pulse-mode operation are described below.

**(a) Single-Pulse Shifts in Pulse Mode and Auto Mode**

Table 155.1.3.8 gives an example of register settings for single-pulse steps in auto mode.

Figure 155.1.3.19 is a timing chart for single-pulse shifts in auto mode. “( )” in Figure 155.1.3.19 means Stop Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the stop state set by TMSPR0 and TMSPR1. “[ ]” in Figure 155.1.3.19 means Operation Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the normal states set by TMMPR0 and TMMPR1.

Table 155.1.3.8 Register Settings to Select One Pulse as the Number of Steps in Auto Mode

| Register Setting Value | Contents |
|---|---|
| TMASR: H'0000 | No acceleration state |
| TMTSR: H'0001 | Only one step in the normal state |
| TMRSR: H'0000 | No deceleration state |

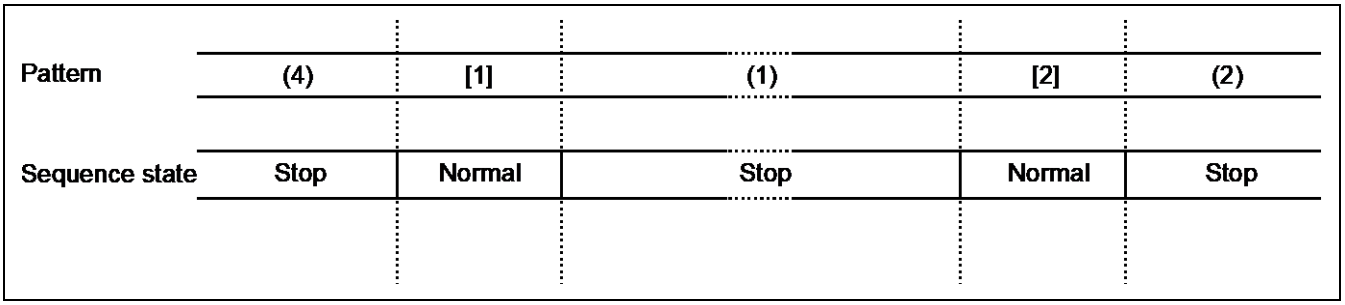



Figure 155.1.3.19 Timing of Single-Pulse Shifts in Auto Mode

**(b) Single-Pulse Shifts in Pulse Mode and Manual Mode**

In manual mode, setting the REDUON0 or REDUON bit in TMRR to 1 while the sequence is in the normal state enables a single pulse shift.

Table 155.1.3.9 gives an example of register settings for single-pulse steps in manual mode.

Figure 155.1.3.20 is a timing chart for single-pulse shifts in manual mode. "( )" in Figure 155.1.3.20 means Stop Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the stop state set by TMSPR0 and TMSPR1. "[ ]" in Figure 155.1.3.20 means Operation Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the normal states set by TMMPR0 and TMMPR1.

Table 155.1.3.9 Register Settings to Select One Pulse as the Number of Steps in Manual Mode

| Register Setting Value | Contents |
|---|---|
| TMASR: H'0000 | No acceleration state |
| TMTSR: H'0000 | Looping in the normal state |
| TMRSR: H'0000 | No deceleration state |

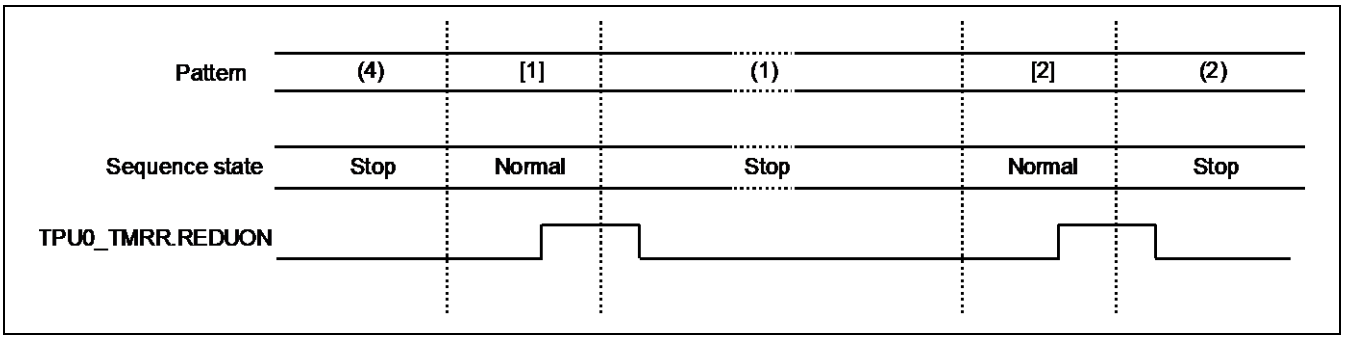



Figure 155.1.3.20 Timing of Single-Pulse Shifts in Manual Mode

**(7) Interrupts**

Timer 0 has six added interrupt conditions for use with stepping-motor control mode.

The interrupt conditions are described below.

**(a) Detection of Overflows of Data Transferred for Motor Control**

Setting the TMDOFE bit in TIER0 to 1 selects the detection of overflows of data transferred for motor control (motor control data transfer overflows). Here, overflow refers to the condition where buffer-related operation, i.e. the transfer of the data from TGRD0, is delayed; that is, when the operation TGRD0→ TGRB0 is delayed. To cancel this interrupt, write 0 to the TMDOFS bit of TSR0 after having read it as 1.

Figure 155.1.3.21 shows an example of DMA transfer where an overflow in data transfer for motor control occurs and detection of overflows has been enabled. "( )" in Figure 155.1.3.21 means Stop Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the stop state set by TMSPR0 and TMSPR1. "[ ]" in Figure 155.1.3.21 means Operation Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the acceleration, normal, and deceleration states set by TMMPR0 and TMMPR1.

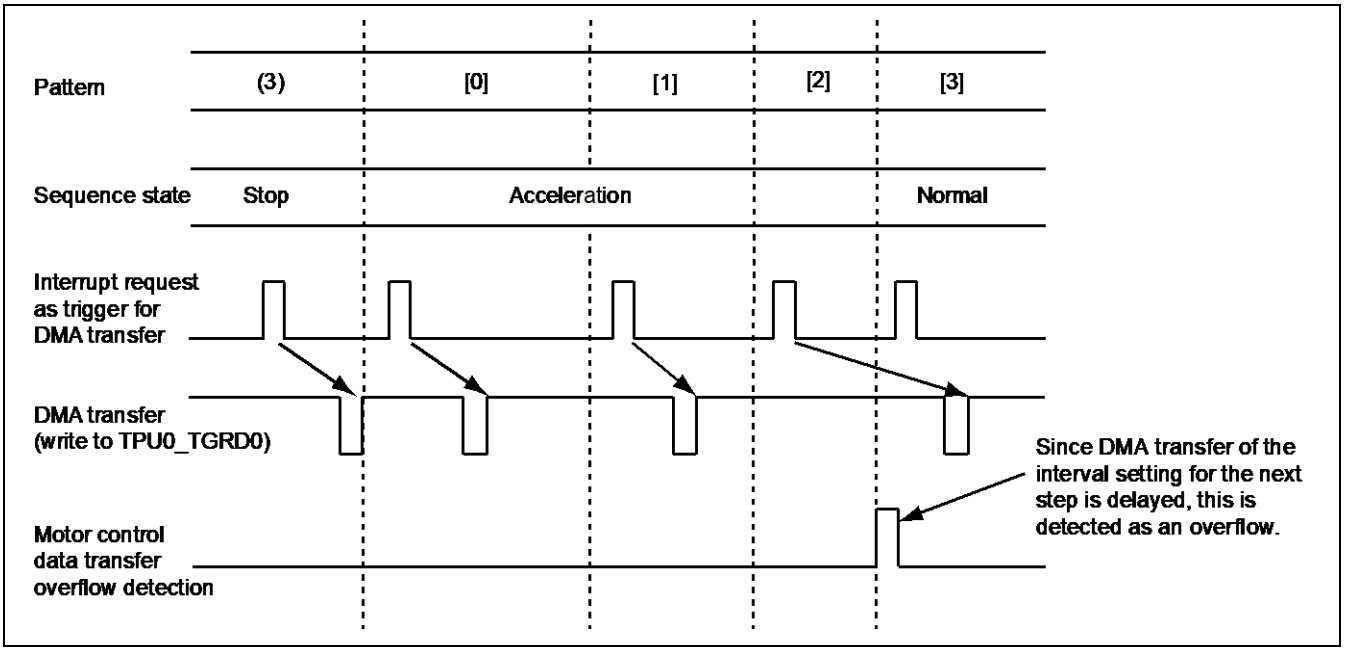



Figure 155.1.3.21 DMA Transfer with Detection of Motor Control Data Transfer Overflows Enabled

Since this interrupt is for use in the debugging of defective operation, normal operation will not be restored even if it is detected.

**(b) Detection of Requirement for Transfer of Motor control Data**

Detection of requirements for the transfer of motor control data (detect motor control data transfer request) is enabled by setting the TMDRFE bit of TIER0. This interrupt is generated when the next pattern interval is to be changed from the current interval. On detection of this interrupt, the handler should write the setting for the next interval to TGRD0 and then write 0 to the TMDRFS bit of TSR0 after having read it as 1.

This interrupt is not available if DMA transfer has been selected (the TDMAE bit in TMIR = 1).

See Figure 155.1.3.17 for an example of usage of this interrupt.

**(c) Detecting the Transition to Deceleration**

Setting the TMS1ER bit in TIER0 to 1 enables detection of the transition to deceleration. The handler should clear the TMCFR flag of TSR0 by writing 0 to the bit after having read it as 1.

**(d) Detecting the Transition to the Normal State**

Setting the TMS1ET bit in TIER0 to 1 enables detection of the transition to the normal state. The handler should clear the TMCFT flag of TSR0 by writing 0 to the bit after having read it as 1.

**(e) Detecting the Transition to Acceleration**

Setting the TMS1EA bit in TIER0 to 1 enables detection of the transition to acceleration. The handler should clear the TMCFA flag of TSR0 by writing 0 to the bit after having read it as 1.

**(f) Detecting the Transition to the Stop State**

Setting the TMS1ES bit in TIER0 to 1 enables detection of the transition to the stop state. The handler should clear the TMCFS flag of TSR0 by writing 0 to the bit after having read it as 1.

**(8) Checking the State of Operation**

Read TMSR and TMSCR to check the current state of operations. TMSCR indicates the value resulting from decrementation (on every compare match with the timer counter) from the values in the number of steps register (TMASR, TMTSR, or TMRSR) during sequence transitions.

Figure 155.1.3.22 shows how the values in TMSR and TMSCR vary through the four states. "( )" in Figure 155.1.3.22 means Stop Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the stop state set by TMSPR0 and TMSPR1. "[ ]" in Figure 155.1.3.22 means Operation Pattern outputted from TPU0TO0, TPU0TO1, TPU0TO2, and TPU0TO3 in the acceleration, normal, and deceleration states set by TMMPR0 and TMMPR1. As is shown in the figure, the current step within the overall sequence of state transitions can be estimated by, for example, reading TMSR = H'0008 and TMSCR = H'0003 in the interval indicated in the figure below.

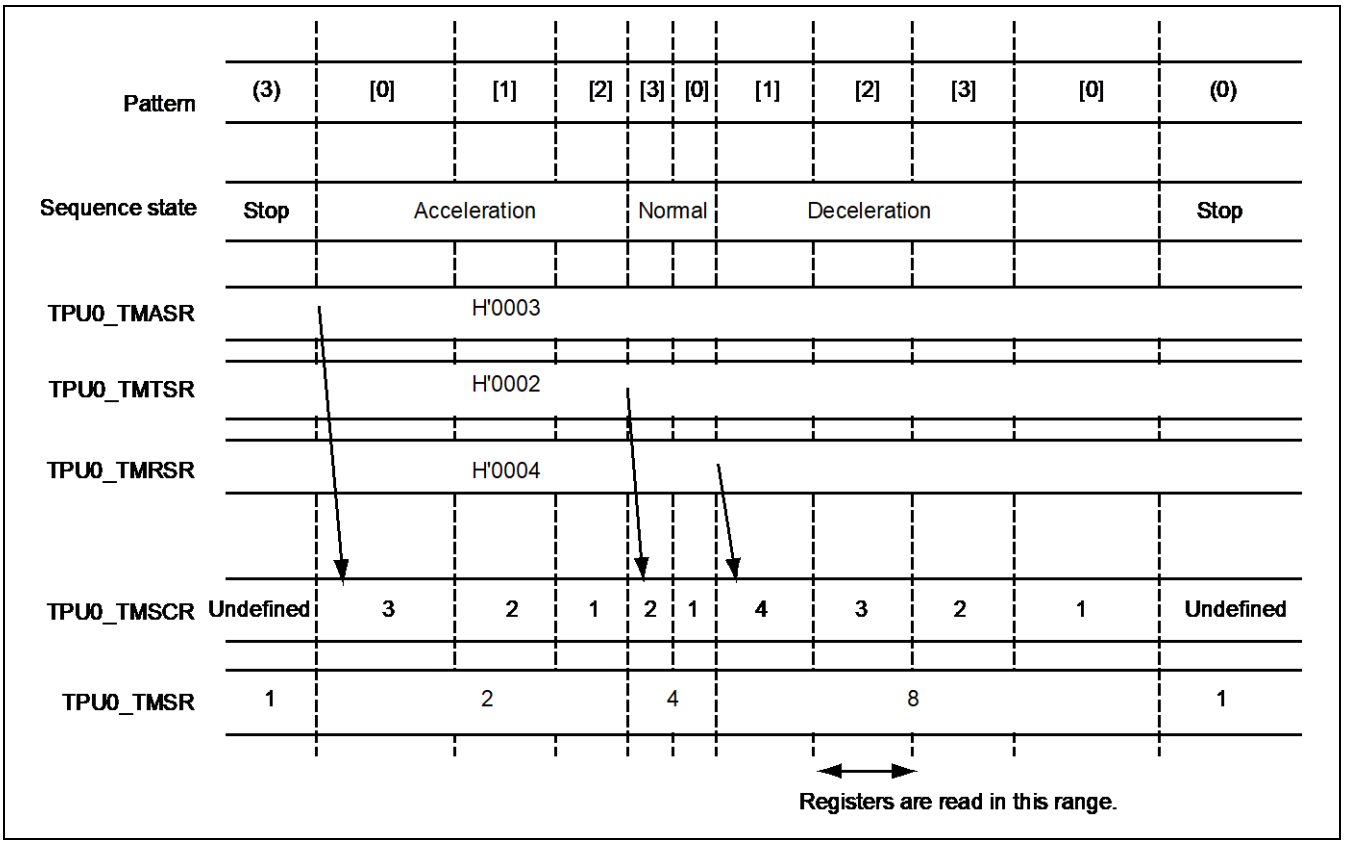



Figure 155.1.3.22 Values of TMSR and TMSCR

### 155.1.4 Usage Notes

No description on this section.

### 155.1.5 Safety Mechanisms

This module is treated as non-safety-related function in case of the products supporting functional safety. So, there is no special spec Requirement for Safety mechanisms.

# 155.2 Compare Match Timer Type0 (CMT0)

## 155.2.1 Overview

Compare Match Timer Type0 (CMT0) is a 32-bit compare match timer of two channel.

### 155.2.1.1 Features

- Two channels
- 16 bits/32 bits can be selected as counter size (bit-width).
- Provided with 32-bit constant registers and 32-bit up counters that can be written or read at any time.
- Following four clocks can be selected as counter clocks:
  — RCLK: 1/1, 1/8, 1/32, and 1/128
- One-shot operation or free-running operation is selectable.
- Compare match or overflow can be selected as interrupt source.
- Counter operation can be enabled or disabled at the time of debugging of CPU core using the debugging mode operation selector.

### 155.2.1.2 Block Diagram

Figure 155.2.1.1 shows a block diagram of the CMT0.

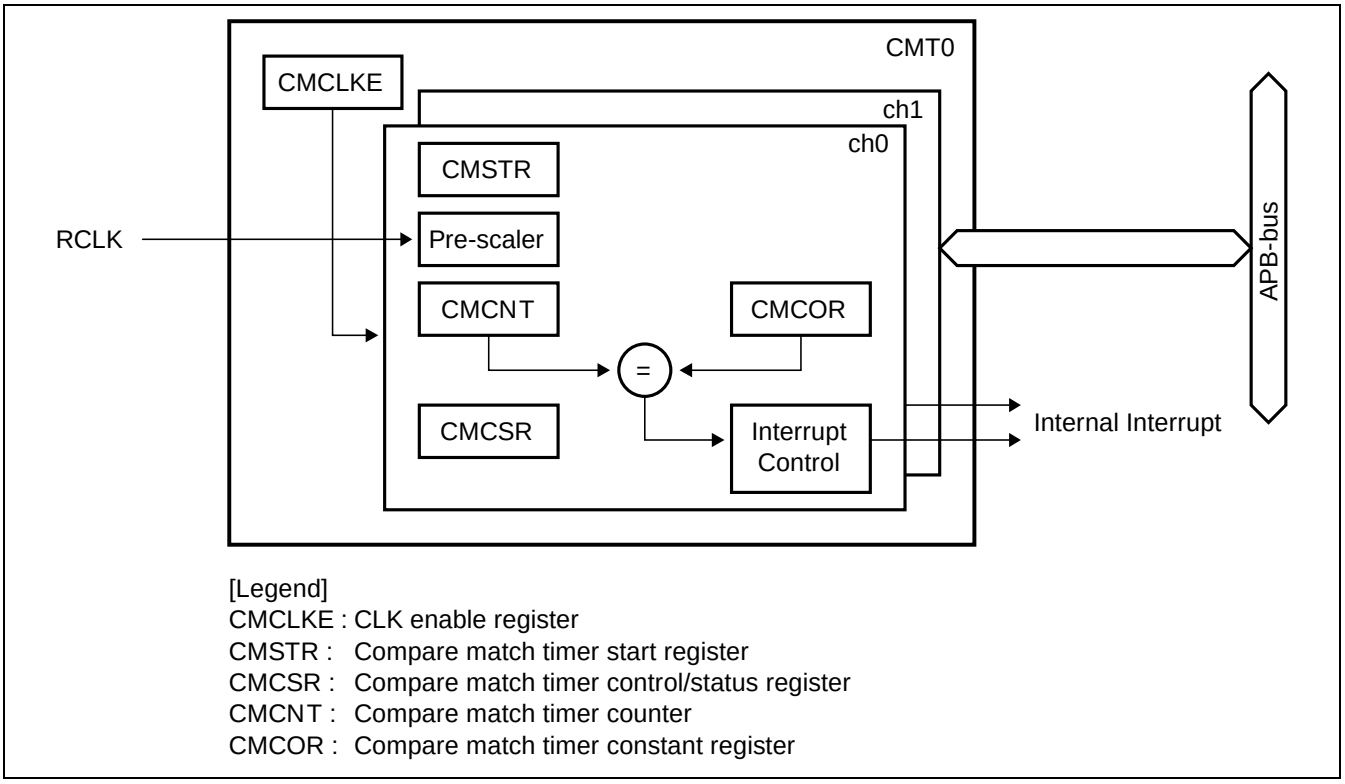



Figure 155.2.1.1 Block Diagram of CMT0

### 155.2.1.3 Register Configuration

Table 155.2.1.1 shows the CMT0 register configuration. Table 155.2.1.2 shows the register states in each operating mode. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined. Access the register as a longword (32 bits). Operation cannot be guaranteed if the register is not accessed as a longword.

Table 155.2.1.1 Register Configuration

| Register Name | Abbreviation | R/W | Address | Access Size | R-Car V4M |
|---|---|---|---|---|---|
| CLK enable register | CMCLKE | R/W | H'E60F 1000 | 32 | √ |
| Compare match timer start register 0 | CMSTR0 | R/W | H'E60F 0500 | 32 | √ |
| Compare match timer control/status register 0 | CMCSR0 | R/W* | H'E60F 0510 | 32 | √ |
| Compare match timer counter 0 | CMCNT0 | R/W | H'E60F 0514 | 32 | √ |
| Compare match timer constant register 0 | CMCOR0 | R/W | H'E60F 0518 | 32 | √ |
| Compare match timer start register 1 | CMSTR1 | R/W | H'E60F 0600 | 32 | √ |
| Compare match timer control/status register 1 | CMCSR1 | R/W* | H'E60F 0610 | 32 | √ |
| Compare match timer counter 1 | CMCNT1 | R/W | H'E60F 0614 | 32 | √ |
| Compare match timer constant register 1 | CMCOR1 | R/W | H'E60F 0618 | 32 | √ |

Note: * Except for bit [15:14] that are R/WC0 register.

Table 155.2.1.2 Register States in Each Operating Mode

| Register Abbreviation | Power-on Reset/Software Reset | Module Standby |
|---|---|---|
| CMCLKE | Initialized | Retained |
| CMSTR0 | Initialized | Retained |
| CMCSR0 | Initialized | Retained |
| CMCNT0 | Initialized | Retained |
| CMCOR0 | Initialized | Retained |
| CMSTR1 | Initialized | Retained |
| CMCSR1 | Initialized | Retained |
| CMCNT1 | Initialized | Retained |
| CMCOR1 | Initialized | Retained |

### 155.2.2 Register Description

[Legend for Register Description]

Initial value: Register value after a reset.

—: Undefined value
R/W: Bit or field is readable and writable. The written value can be read.
R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.
R: Read-only. The write value should always be initial value.
All access to register is made in longword units.

#### 155.2.2.1 CLK Enable Register (CMCLKE)

CMCLKE is a 32-bit register, which specifies clock supply to each channel. When there are unused channels, set 0 to this register, for stop supplying clock to the channel. It is prohibited to stop clock supply, while counter is working.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | Ch1 clke | Ch0 clke | — | — | — | — | — |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 7 | — | All 1 | R | Reserved<br>These bits are always read as 1. The write value should always be 1. |
| 6 | Ch1clke | 1 | R/W | 0: Clock isn't supplied to ch1.<br>1: Clock is supplied to ch1. |
| 5 | Ch0clke | 1 | R/W | 0: Clock isn't supplied to ch0.<br>1: Clock is supplied to ch0. |
| 4 to 0 | — | All 1 | R | Reserved<br>These bits are always read as 1. The write value should always be 1. |

### 155.2.2.2 Compare Match Timer Start Registers 0 and 1 (CMSTR0, CMSTR1)

CMSTRn (n = 0 and 1) is a 32-bit register which specifies the operation of compare match timer counters (CMCNTn).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | STR0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | STR0 | 1 | R/W | Count Start<br>These bits specify start/halt of compare match timer counter (CMCNTn).<br>0: CMCNTn halts<br>1: CMCNTn starts counting |

### 155.2.2.3 Compare Match Timer Control/Status Registers 0 and 1 (CMCSR0, CMCSR1)

CMCSRn (n = 0 and 1) is a 32-bit register that indicates the occurrence of compare matches, enables interrupts, and sets the counter input clocks.

Do not change bits other than the CMF and OVF bits, while compare match timer counter (CMCNTn) is under counting.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMF | OVF | WRFLG | — | — | — | CMS | CMM | — | — | CMR[1:0] | | DBGIVD | CKS[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R/WC0 | R/WC0 | R | R | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | CMF | 0 | R/WC0 | Compare Match Flag<br>This flag indicates whether values of the compare match timer counter (CMCNTn) and compare match timer constant register (CMCORn) have matched or not.<br>Software cannot write 1 to the bit. When one-shot is selected for the counter operation, counting resumes by clearing this bit.<br>0: CMCNTn and CMCORn values have not matched.<br>[Clearing condition]<br>Write 0 to CMF<br>1: CMCNTn and CMCORn values have matched.<br>Note: Only 0 can be written to clear the flag. |
| 14 | OVF | 0 | R/WC0 | Overflow Flag<br>This flag indicates whether the compare match timer counter (McCants) has overflowed or not. Software cannot write 1 to this bit.<br>0: CMCNTn has not overflowed.<br>[Clearing condition]<br>Write 0 to OVF<br>1: CMCNTn has overflowed.<br>Note: Only 0 can be written to clear the flag. |
| 13 | WRFLG | 0 | R | Write State Flag<br>Write access to CMCNTn is prohibited, while this this bit is 1. Further behavior is not guaranteed, if data is written while this bit is 1.<br>This bit indicates CMCNTn is in synchronization period for setting previously written data.<br>Confirm that this flag is 0, before writing to CMCNTn. |
| 12 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | CMS | 0 | R/W | Compare Match Timer Counter Size<br>Specify whether the compare match timer counter (CMCNTn) is used as a 16-bit counter or a 32-bit counter.<br>This bit specifies the valid size of compare match timer constant register (CMCORn).<br>0: Operates as a 32-bit counter.<br>1: Operates as a 16-bit counter.<br>Note: Refer to section 155.2.3.5, CMT0 Register Access, for the note regarding writing to or reading from this bit. |
| 8 | CMM | 1 | R/W | Compare Match Mode<br>Specify counter operation mode.<br>0: One-shot operation<br>1: Free-running operation<br>Note: Refer to section 155.2.3.5, CMT0 Register Access, for the note regarding writing to or reading from this bit. |
| 7, 6 | — | All 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5, 4 | CMR[1:0] | 00 | R/W | Compare Match Request<br>These bits enable or disable an internal interrupt request in a compare match.<br>00: Disables an internal interrupt request.<br>01: Setting prohibited<br>10: Enables an internal interrupt request.<br>11: Setting prohibited<br>Note: Refer to section 155.2.3.5, CMT0 Register Access, for the note regarding writing to or reading from this bit. |
| 3 | DBGIVD | 1 | R/W | Debug Mode Operation Select<br>Sets the counter operation in debug mode.<br>0: Stops the counter operation in debug mode.<br>1: Enables the counter operation even in debug mode.<br>Note: Refer to section 155.2.3.5, CMT0 Register Access, for the note regarding writing to or reading from this bit. |
| 2 to 0 | CKS[2:0] | 111 | R/W | Clock Select<br>These bits select the input clock to CMCNTn. When the count start bit (STR0) in CMSTRn is set to 1, CMCNTn begins incrementing with the clock selected by these bits.<br>000: Setting prohibited<br>001: Setting prohibited<br>010: Setting prohibited<br>011: Setting prohibited<br>100: RCLK/8<br>101: RCLK/32<br>110: RCLK/128<br>111: RCLK/1<br>Note: Refer to section 155.2.3.5, CMT0 Register Access, for the note regarding writing to or reading from this bit. |

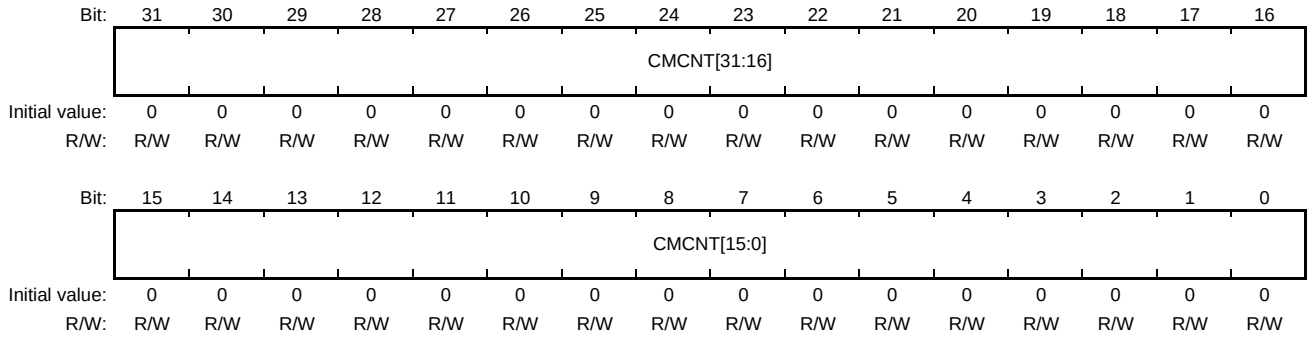
#### 155.2.2.4 Compare Match Timer Counters 0 and 1 (CMCNT0, CMCNT1)

CMCNTn (n = 0 and 1) is a 32-bit register that is used as an up-counter.

To specify counter operation, set compare match timer control/status register n (CMCSRn), before starting operation of corresponding channel.

When the 16-bit counter operation is selected by the CMS bit, bits 31 to 16 of this register become invalid. When data is written to this register in 16-bit mode, write H'0000 to the upper 16 bits.

When CMCNTn is read during the counter operation, the read value may be wrong because different clock is used between counter and bus-interface. For exact value, read this register continuously, until same values are read from this register.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMCNT [31:0] | H'0000 0000 | R/W | Compare match timer counter bit31 to 0<br>Note: Refer to section 155.2.3.5, CMT0 Register Access, for the note regarding writing to or reading from this bit. |

### 155.2.2.5 Compare Match Timer Constant Registers 0 and 1 (CMCOR0, CMCOR1)

CMCORn (n = 0 and 1) is a 32-bit register that sets the compare match period with CMCNTn.

When the 16-bit counter operation is selected by the CMS bit in CMCSRn, bits 15 to 0 of this register become valid. Write H'0000 to upper 16bits in 16-bit counter operation.

An overflow is detected when CMCNTn is cleared to 0 and this register is H'FFFF FFFF.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMCOR[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMCOR[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMCOR [31:0] | H'FFFF FFFF | R/W | Compare match timer constant register bit31 to 0<br>Note: Refer to section 155.2.3.5, CMT0 Register Access, for the note regarding writing to or reading from this bit. |

## 155.2.3 Operation

### 155.2.3.1 Counter Operation

The CMT0 starts the operation of the counter by writing 1 to the STR0 bit in CMSTRn after each register has been set. Complete all of the settings before starting the operation. Do not change the register settings other than clearing flag bits, during the compare match timer n (CMCNTn) is under operation.

The counter operates in one of two ways.

- One-Shot Operation

  One-shot operation is selected by setting the CMM bit in CMCSRn to 0. When the value in CMCNTn matches the value in CMCORn, the value in CMCNTn is cleared to H'0000 0000 and the CMF bit in CMCSRn is set to 1. Counting by CMCNTn stops after CMCNTn has been cleared.

  Counting by CMCNTn will be resumed after CMCSRn.CMF bit has been cleared.

  To detect an overflow interrupt, set the value in CMCORn to H'FFFF FFFF. When the value in CMCNTn matches the value in CMCORn, CMCNTn is cleared to H'0000 0000 and the CMF and OVF bits in CMCSRn are set to 1.

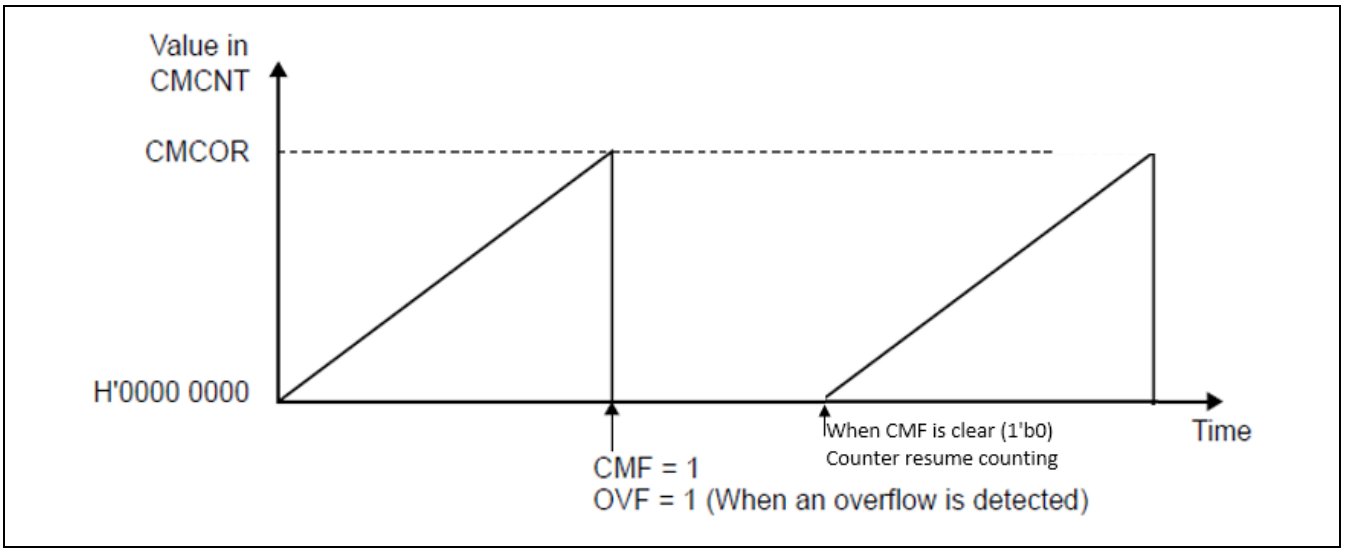



Figure 155.2.3.1 Counter Operation (One-Shot Operation)

- Free-Running Operation

  Free-running operation is selected by setting the CMM bit in CMCSRn to 1. When the value in CMCNTn matches the value in CMCORn, CMCNTn is cleared to H'0000 0000 and the CMF bit in CMCSRn is set to 1. CMCNTn resumes counting-up after it has been cleared.

  To detect an overflow interrupt, set CMCORn to H'FFFF FFFF. When the values in CMCNTn and CMCORn match, CMCNTn is cleared to H'0000 0000 and the CMF and OVF bits in CMCSRn are set to 1.

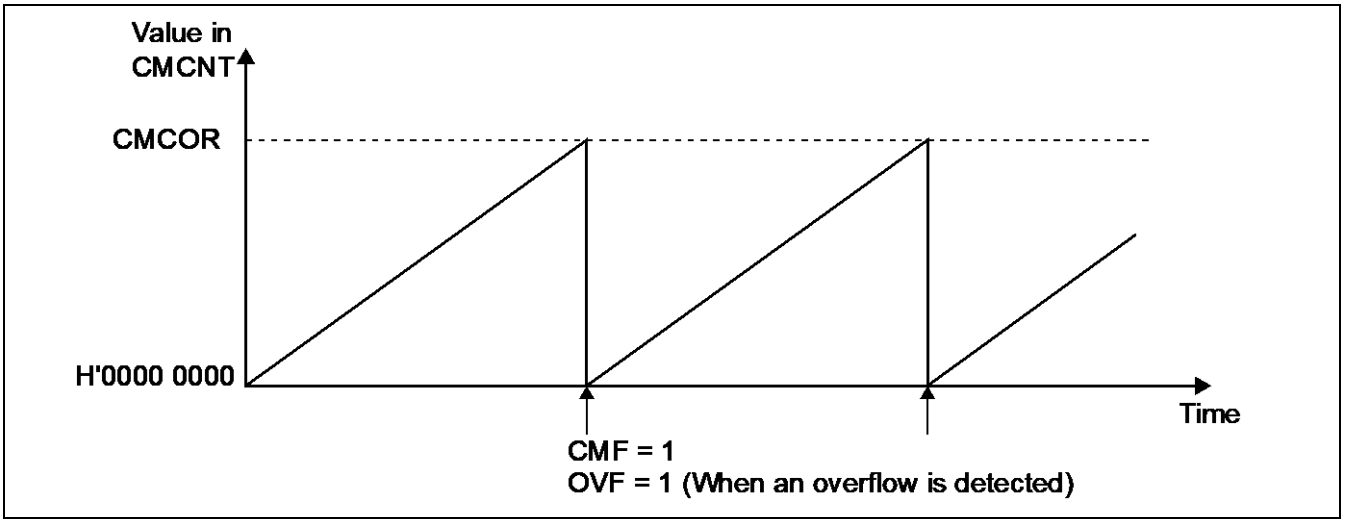



Figure 155.2.3.2 Counter Operation (Free-Running Operation)

#### 155.2.3.2 Counter Size

In this module, the size of the counter can be selected from 16, or 32 bits. This is selected by the CMS bit in CMCSRn.

When the 16-bit size is selected, H'0000 should be used as upper 16 bits of write data to CMCORn. To detect an overflow interrupt, CMCORn must be set to H'0000 FFFF.

#### 155.2.3.3 Timing for Counting by CMCNTn

In this module, the clock for the counter can be selected from among the following:

- RCLK: 1/1, 1/8, 1/32, and 1/128

The clock for the counter is selected by the CKS bits in CMCSRn. CMCNTn is incremented at the rising edge of the selected clock.

#### 155.2.3.4 Internal Interrupt Request to CPU

By CMR bits in CMCSRn, internal interrupt request to the CPU at a compare match can be asserted.

To clear an internal interrupt request to the CPU in Free-Running Operation, the CMF bit should be set to 0. Set the CMF bit to 0 in the handling routine for the CMT0 interrupt.

To clear an internal interrupt request to the CPU with resuming count-up in One-Shot Operation, the CMF bit should be set to 0 while keeping 1 in STR0 bit in CMSTRn. Set the CMF bit to 0 while keeping 1 in STR0 bit in the handling routine for the CMT0 interrupt.

To clear an internal interrupt request to the CPU without resuming count-up in One-Shot Operation, the CMF bit should be set to 0 after setting STR0 bit in CMSTRn to 0. Set the CMF bit to 0 after setting STR0 bit to 0 in the handling routine for the CMT0 interrupt.

#### 155.2.3.5 CMT0 Register Access

After writing to following registers, written data can be read 2 cycles in counter input clock (RCLK) after writing has finished. And it takes 2 cycles in counter input clock (RCLK), for reflecting written data to counter behavior.

CMCSRn: Bits CKS, CMM, CMS, CMR, DBGIVD

CMCORn: Bits 31 to 0

CMSTRn: Bit STR0

After writing to following registers, written data can be read 2 cycles in counter input clock (RCLK) after writing has finished. And it takes 2 cycles in counter input clock (RCLK), for reflecting written data to counter behavior.

And for following registers, write access is prohibited, while previously written data is under writing. Do not perform next write access, while CMCSRn.WRFLG is 1.

CMCNT*: Bits 31 to 0

### 155.2.3.6 Compare Match Flag Set/Clear Timing

The CMF bit in CMCSRn is set to 1 by the compare match signal generated when CMCORn and CMCNTn match. The compare match signal is generated upon the final state of the match (timing at which the CMCNTn value is updated to 0). Consequently, after CMCORn and CMCNTn match, a compare match signal will not be generated until a CMCNTn counter clock is input. Figure 155.2.3.3 shows the set timing of the CMF bit.

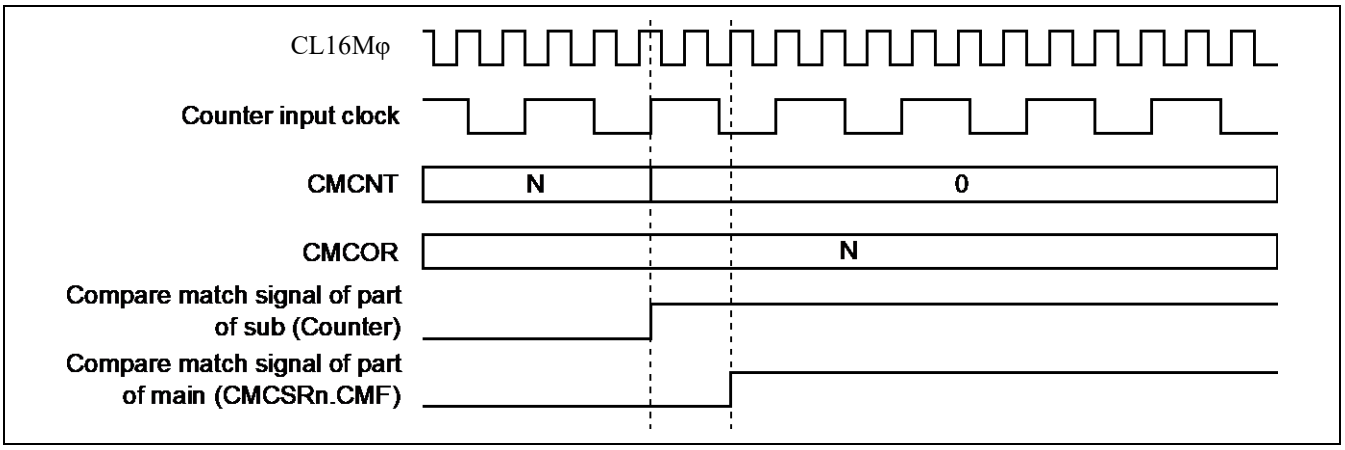



Figure 155.2.3.3 CMF Set Timing

Set the CMF bit as 0. The CMF flag is cleared immediately.

### 155.2.3.7 CMT0 Usage

Take the following steps to use the CMT0.

1. Clear the STR0 bit in CMSTRn to 0 to temporarily stop counting.
   (Confirm that RCLK was input more than 2 cycles.)
2. Write H'0000 0000 on CMCNTn.
3. Set counter size, compare match mode, type of counter clock and interrupt request and clear the bit of OVF and CMF in CMCSRn.
4. Set the value on CMCORn.
5. Confirm that CMCSRn.WRFLG is 0.
   If WRFLG is 1, wait until it'll be 0.
6. Start the counting by setting the STR0 bit in CMSTRn to 1.
   (Refer to section 155.2.3.5, CMT0 Register Access)

### 155.2.3.8 Module Stop/ CMCLKE Setting

When counter is not used, clock of this module can be stopped by CMCLKE register. Confirm that, following conditions are satisfied, before stopping counter’s clock.

- CMSTRn.STR0 bit is 0
- CMCSRn.WRFLG is 0
- 3 or more cycles have passed in RCLK, since the last Register access

### 155.2.4 Usage Notes

No description on this section.

### 155.2.5 Safety Mechanisms

#### 155.2.5.1 Power-On test

This module is protected by Power-On test.

# 155.3 Compare Match Timer Type1 (CMT1/2/3)

## 155.3.1 Overview

This section is written to introduce about CMT1/2/3, total include 24 channels, CMT2/3 are exactly same as CMT1, CMT1's usage are introduced here in, refer to 155.3.1.3 for CMT2/3 register address.

### 155.3.1.1 Features

- Eight channels
- 16 bits/32 bits/48 bits can be selected as counter size (bit-width).
- 48-bit constant registers and 48-bit up counters that can be written or read at any time.
- For channel 0 to 4, following eleven clocks can be selected as counter clock.
  - CPEXϕ: 1/8, 1/32, and 1/128
  - RCLK: 1/1, 1/8, 1/32, and 1/128
  - Pseudo 32 kHz: 1/1, 1/8, 1/32, and 1/128
- For channel 5 to 7, following four clocks can be selected as counter clock.
  - RCLK: 1/1,1/8, 1/32, and 1/128
- One-shot operation or free-running operations are selectable.
- Compare match can be used as interrupt source.
- Support Module standby mode.
- RCLK-synchronous counter start/stop mode for channel 0
- Counter operation can be enabled or disabled at the time of debugging of CPU core using the debugging mode operation selector.

### 155.3.1.2 Block Diagram

Figure 155.3.1.1 shows a block diagram of the CMT1.

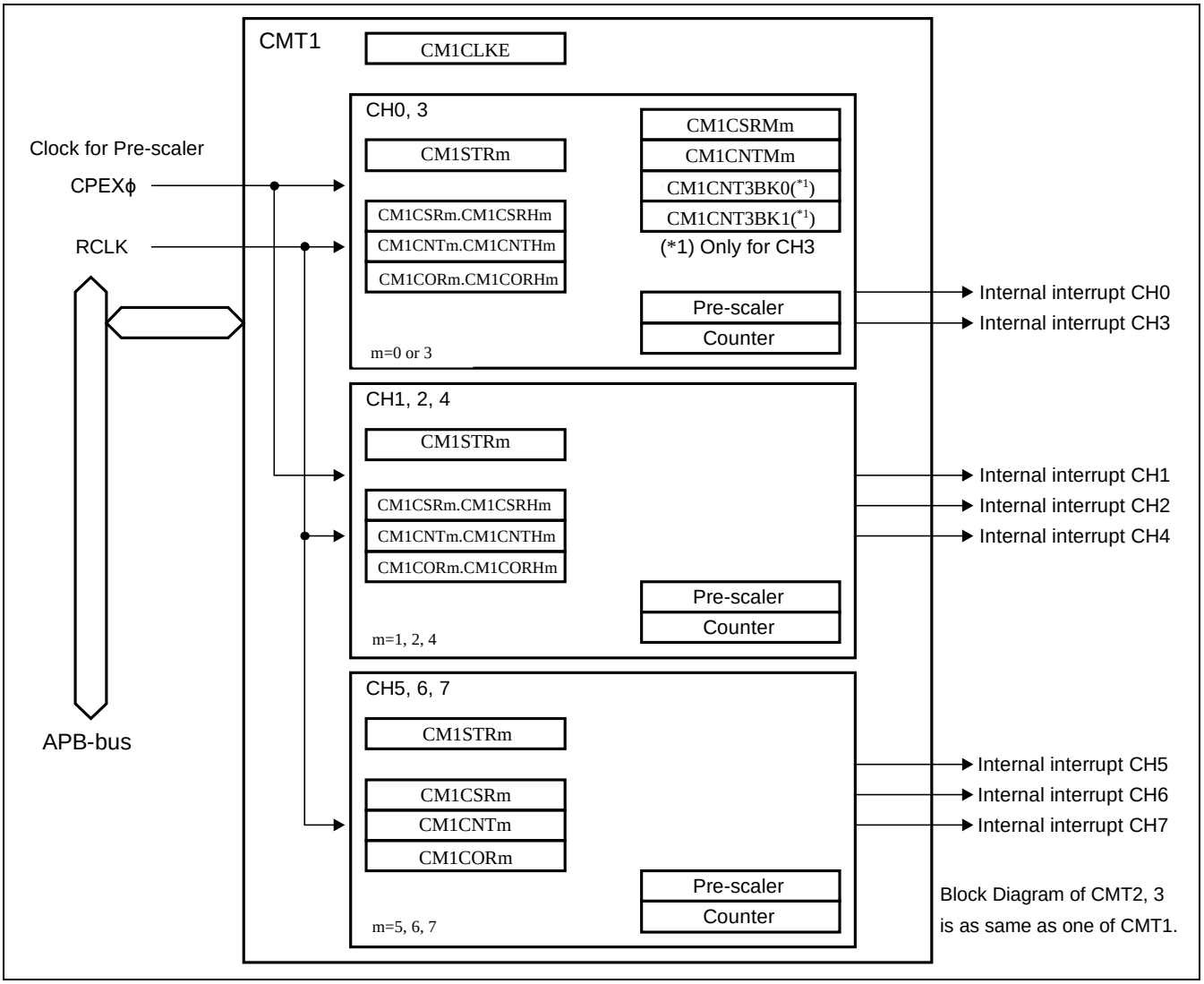



**Figure 155.3.1.1 Block Diagram of CMT1/2/3**

#### 155.3.1.3 Register Configuration

Table 155.3.1.1 shows the CMT1/2/3 register configuration.Table 155.3.1.2 shows the register states in each operating mode. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined. Access the register as a longword (32 bits). Operation cannot be guaranteed if the register is not accessed as a longword.

Table 155.3.1.1 Register Configuration

| Register Name (n=1~3) | CMT1/2/3 | Abbreviation (n=1~3) | R/W | Address | Access Size |
|---|---|---|---|---|---|
| CLK enable register n | CMT1 | CMnCLKE | R/W | H'E613 1000 | 32 |
| | CMT2 | | | H'E614 1000 | |
| | CMT3 | | | H'E614 9000 | |
| Compare match timer n start register 0 | CMT1 | CMnSTR0 | R/W | H'E613 0000 | 32 |
| | CMT2 | | | H'E614 0000 | |
| | CMT3 | | | H'E614 8000 | |
| Compare match timer n control/status register 0 | CMT1 | CMnCSR0 | R/W*1 | H'E613 0010 | 32 |
| | CMT2 | | | H'E614 0010 | |
| | CMT3 | | | H'E614 8010 | |
| Compare match timer n counter 0 | CMT1 | CMnCNT0 | R/W | H'E613 0014 | 32 |
| | CMT2 | | | H'E614 0014 | |
| | CMT3 | | | H'E614 8014 | |
| Compare match timer n constant register 0 | CMT1 | CMnCOR0 | R/W | H'E613 0018 | 32 |
| | CMT2 | | | H'E614 0018 | |
| | CMT3 | | | H'E614 8018 | |
| Compare match timer n control/status register H0 | CMT1 | CMnCSRH0 | R/W | H'E613 0020 | 32 |
| | CMT2 | | | H'E614 0020 | |
| | CMT3 | | | H'E614 8020 | |
| Compare match timer n counter H0 | CMT1 | CMnCNTH0 | R/W | H'E613 0024 | 32 |
| | CMT2 | | | H'E614 0024 | |
| | CMT3 | | | H'E614 8024 | |
| Compare match timer n constant register H0 | CMT1 | CMnCORH0 | R/W | H'E613 0028 | 32 |
| | CMT2 | | | H'E614 0028 | |
| | CMT3 | | | H'E614 8028 | |
| Compare match timer n match control/status register 0 | CMT1 | CMnCSRM0 | R/W | H'E613 0040 | 32 |
| | CMT2 | | | H'E614 0040 | |
| | CMT3 | | | H'E614 8040 | |
| Compare match timer n match counter 0 | CMT1 | CMnCNTM0 | R | H'E613 0044 | 32 |
| | CMT2 | | | H'E614 0044 | |
| | CMT3 | | | H'E614 8044 | |
| Compare match timer n start register 1 | CMT1 | CMnSTR1 | R/W | H'E613 0100 | 32 |
| | CMT2 | | | H'E614 0100 | |
| | CMT3 | | | H'E614 8100 | |
| Compare match timer n control/status register 1 | CMT1 | CMnCSR1 | R/W*2 | H'E613 0110 | 32 |
| | CMT2 | | | H'E614 0110 | |
| | CMT3 | | | H'E614 8110 | |

| Register Name (n=1~3) | CMT1/2/3 | Abbreviation (n=1~3) | R/W | Address | Access Size |
|---|---|---|---|---|---|
| Compare match timer n counter 1 | CMT1 | CMnCNT1 | R/W | H'E613 0114 | 32 |
| | CMT2 | | | H'E614 0114 | |
| | CMT3 | | | H'E614 8114 | |
| Compare match timer n constant register 1 | CMT1 | CMnCOR1 | R/W | H'E613 0118 | 32 |
| | CMT2 | | | H'E614 0118 | |
| | CMT3 | | | H'E614 8118 | |
| Compare match timer n control/status register H1 | CMT1 | CMnCSRH1 | R/W | H'E613 0120 | 32 |
| | CMT2 | | | H'E614 0120 | |
| | CMT3 | | | H'E614 8120 | |
| Compare match timer n counter H1 | CMT1 | CMnCNTH1 | R/W | H'E613 0124 | 32 |
| | CMT2 | | | H'E614 0124 | |
| | CMT3 | | | H'E614 8124 | |
| Compare match timer n constant register H1 | CMT1 | CMnCORH1 | R/W | H'E613 0128 | 32 |
| | CMT2 | | | H'E614 0128 | |
| | CMT3 | | | H'E614 8128 | |
| Compare match timer n start register 2 | CMT1 | CMnSTR2 | R/W | H'E613 0200 | 32 |
| | CMT2 | | | H'E614 0200 | |
| | CMT3 | | | H'E614 8200 | |
| Compare match timer n control/status register 2 | CMT1 | CMnCSR2 | R/W*2 | H'E613 0210 | 32 |
| | CMT2 | | | H'E614 0210 | |
| | CMT3 | | | H'E614 8210 | |
| Compare match timer n counter 2 | CMT1 | CMnCNT2 | R/W | H'E613 0214 | 32 |
| | CMT2 | | | H'E614 0214 | |
| | CMT3 | | | H'E614 8214 | |
| Compare match timer n constant register 2 | CMT1 | CMnCOR2 | R/W | H'E613 0218 | 32 |
| | CMT2 | | | H'E614 0218 | |
| | CMT3 | | | H'E614 8218 | |
| Compare match timer n control/status register H2 | CMT1 | CMnCSRH2 | R/W | H'E613 0220 | 32 |
| | CMT2 | | | H'E614 0220 | |
| | CMT3 | | | H'E614 8220 | |
| Compare match timer n counter H2 | CMT1 | CMnCNTH2 | R/W | H'E613 0224 | 32 |
| | CMT2 | | | H'E614 0224 | |
| | CMT3 | | | H'E614 8224 | |
| Compare match timer n constant register H2 | CMT1 | CMnCORH2 | R/W | H'E613 0228 | 32 |
| | CMT2 | | | H'E614 0228 | |
| | CMT3 | | | H'E614 8228 | |
| Compare match timer n start register 3 | CMT1 | CMnSTR3 | R/W | H'E613 0300 | 32 |
| | CMT2 | | | H'E614 0300 | |
| | CMT3 | | | H'E614 8300 | |

| Register Name (n=1~3) | CMT1/2/3 | Abbreviation (n=1~3) | R/W | Address | Access Size |
|---|---|---|---|---|---|
| Compare match timer n control/status register 3 | CMT1 | CMnCSR3 | R/W*2 | H'E613 0310 | 32 |
| | CMT2 | | | H'E614 0310 | |
| | CMT3 | | | H'E614 8310 | |
| Compare match timer n counter 3 | CMT1 | CMnCNT3 | R/W | H'E613 0314 | 32 |
| | CMT2 | | | H'E614 0314 | |
| | CMT3 | | | H'E614 8314 | |
| Compare match timer n constant register 3 | CMT1 | CMnCOR3 | R/W | H'E613 0318 | 32 |
| | CMT2 | | | H'E614 0318 | |
| | CMT3 | | | H'E614 8318 | |
| Compare match timer n control/status register H3 | CMT1 | CMnCSRH3 | R/W | H'E613 0320 | 32 |
| | CMT2 | | | H'E614 0320 | |
| | CMT3 | | | H'E614 8320 | |
| Compare match timer n counter H3 | CMT1 | CMnCNTH3 | R/W | H'E613 0324 | 32 |
| | CMT2 | | | H'E614 0324 | |
| | CMT3 | | | H'E614 8324 | |
| Compare match timer n constant register H3 | CMT1 | CMnCORH3 | R/W | H'E613 0328 | 32 |
| | CMT2 | | | H'E614 0328 | |
| | CMT3 | | | H'E614 8328 | |
| Compare match timer n counter 3 backup 0 | CMT1 | CMnCNT3BK0 | R | H'E613 0330 | 32 |
| | CMT2 | | | H'E614 0330 | |
| | CMT3 | | | H'E614 8330 | |
| Compare match timer n counter 3 backup 1 | CMT1 | CMnCNT3BK1 | R | H'E613 0334 | 32 |
| | CMT2 | | | H'E614 0334 | |
| | CMT3 | | | H'E614 8334 | |
| Compare match timer n match control/status register 3 | CMT1 | CMnCSRM3 | R/W | H'E613 0340 | 32 |
| | CMT2 | | | H'E614 0340 | |
| | CMT3 | | | H'E614 8340 | |
| Compare match timer n match counter 3 | CMT1 | CMnCNTM3 | R | H'E613 0344 | 32 |
| | CMT2 | | | H'E614 0344 | |
| | CMT3 | | | H'E614 8344 | |
| Compare match timer n start register 4 | CMT1 | CMnSTR4 | R/W | H'E613 0400 | 32 |
| | CMT2 | | | H'E614 0400 | |
| | CMT3 | | | H'E614 8400 | |
| Compare match timer n control/status register 4 | CMT1 | CMnCSR4 | R/W*2 | H'E613 0410 | 32 |
| | CMT2 | | | H'E614 0410 | |
| | CMT3 | | | H'E614 8410 | |
| Compare match timer n counter 4 | CMT1 | CMnCNT4 | R/W | H'E613 0414 | 32 |
| | CMT2 | | | H'E614 0414 | |
| | CMT3 | | | H'E614 8414 | |

| Register Name (n=1~3) | CMT1/2/3 | Abbreviation (n=1~3) | R/W | Address | Access Size |
|---|---|---|---|---|---|
| Compare match timer n constant register 4 | CMT1 | CMnCOR4 | R/W | H'E613 0418 | 32 |
| | CMT2 | | | H'E614 0418 | |
| | CMT3 | | | H'E614 8418 | |
| Compare match timer n control/status register H4 | CMT1 | CMnCSRH4 | R/W | H'E613 0420 | 32 |
| | CMT2 | | | H'E614 0420 | |
| | CMT3 | | | H'E614 8420 | |
| Compare match timer n counter H4 | CMT1 | CMnCNTH4 | R/W | H'E613 0424 | 32 |
| | CMT2 | | | H'E614 0424 | |
| | CMT3 | | | H'E614 8424 | |
| Compare match timer n constant register H4 | CMT1 | CMnCORH4 | R/W | H'E613 0428 | 32 |
| | CMT2 | | | H'E614 0428 | |
| | CMT3 | | | H'E614 8428 | |
| Compare match timer n start register 5 | CMT1 | CMnSTR5 | R/W | H'E613 0500 | 32 |
| | CMT2 | | | H'E614 0500 | |
| | CMT3 | | | H'E614 8500 | |
| Compare match timer n control/status register 5 | CMT1 | CMnCSR5 | R/W*2 | H'E613 0510 | 32 |
| | CMT2 | | | H'E614 0510 | |
| | CMT3 | | | H'E614 8510 | |
| Compare match timer n counter 5 | CMT1 | CMnCNT5 | R/W | H'E613 0514 | 32 |
| | CMT2 | | | H'E614 0514 | |
| | CMT3 | | | H'E614 8514 | |
| Compare match timer n constant register 5 | CMT1 | CMnCOR5 | R/W | H'E613 0518 | 32 |
| | CMT2 | | | H'E614 0518 | |
| | CMT3 | | | H'E614 8518 | |
| Compare match timer n start register 6 | CMT1 | CMnSTR6 | R/W | H'E613 0600 | 32 |
| | CMT2 | | | H'E614 0600 | |
| | CMT3 | | | H'E614 8600 | |
| Compare match timer n control/status register 6 | CMT1 | CMnCSR6 | R/W*2 | H'E613 0610 | 32 |
| | CMT2 | | | H'E614 0610 | |
| | CMT3 | | | H'E614 8610 | |
| Compare match timer n counter 6 | CMT1 | CMnCNT6 | R/W | H'E613 0614 | 32 |
| | CMT2 | | | H'E614 0614 | |
| | CMT3 | | | H'E614 8614 | |
| Compare match timer n constant register 6 | CMT1 | CMnCOR6 | R/W | H'E613 0618 | 32 |
| | CMT2 | | | H'E614 0618 | |
| | CMT3 | | | H'E614 8618 | |
| Compare match timer n start register 7 | CMT1 | CMnSTR7 | R/W | H'E613 0700 | 32 |
| | CMT2 | | | H'E614 0700 | |
| | CMT3 | | | H'E614 8700 | |

| Register Name (n=1~3) | CMT1/2/3 | Abbreviation (n=1~3) | R/W | Address | Access Size |
|---|---|---|---|---|---|
| Compare match timer n control/status register 7 | CMT1 | CMnCSR7 | R/W*2 | H'E613 0710 | 32 |
| | CMT2 | | | H'E614 0710 | |
| | CMT3 | | | H'E614 8710 | |
| Compare match timer n counter 7 | CMT1 | CMnCNT7 | R/W | H'E613 0714 | 32 |
| | CMT2 | | | H'E614 0714 | |
| | CMT3 | | | H'E614 8714 | |
| Compare match timer n constant register 7 | CMT1 | CMnCOR7 | R/W | H'E613 0718 | 32 |
| | CMT2 | | | H'E614 0718 | |
| | CMT3 | | | H'E614 8718 | |

Notes: 1. Except for bit [15:14] / [12:11] that are R/WC0 register.
2. Except for bit [15:14] that are R/WC0 register.

Table 155.3.1.2 Register States in Each Operating Mode

| Register Abbreviation (n=1~3) | Power-on Reset/Software Reset | Module Standby |
|---|---|---|
| CMnCLKE | Initialized | Retained |
| CMnSTR0 | Initialized | Retained |
| CMnCSR0 | Initialized | Retained |
| CMnCNT0 | Initialized | Retained |
| CMnCOR0 | Initialized | Retained |
| CMnCSRH0 | Initialized | Retained |
| CMnCNTH0 | Initialized | Retained |
| CMnCORH0 | Initialized | Retained |
| CMnCSRM0 | Initialized | Retained |
| CMnCNTM0 | Initialized | Retained |
| CMnSTR1 | Initialized | Retained |
| CMnCSR1 | Initialized | Retained |
| CMnCNT1 | Initialized | Retained |
| CMnCOR1 | Initialized | Retained |
| CMnCSRH1 | Initialized | Retained |
| CMnCNTH1 | Initialized | Retained |
| CMnCORH1 | Initialized | Retained |
| CMnSTR2 | Initialized | Retained |
| CMnCSR2 | Initialized | Retained |
| CMnCNT2 | Initialized | Retained |
| CMnCOR2 | Initialized | Retained |
| CMnCSRH2 | Initialized | Retained |
| CMnCNTH2 | Initialized | Retained |
| CMnCORH2 | Initialized | Retained |
| CMnSTR3 | Initialized | Retained |
| CMnCSR3 | Initialized | Retained |
| CMnCNT3 | Initialized | Retained |
| CMnCOR3 | Initialized | Retained |

| Register Abbreviation (n=1~3) | Power-on Reset/Software Reset | Module Standby |
|---|---|---|
| CMnCSRH3 | Initialized | Retained |
| CMnCNTH3 | Initialized | Retained |
| CMnCORH3 | Initialized | Retained |
| CMnCNT3BK0 | Initialized | Retained |
| CMnCNT3BK1 | Initialized | Retained |
| CMnCSRM3 | Initialized | Retained |
| CMnCNTM3 | Initialized | Retained |
| CMnSTR4 | Initialized | Retained |
| CMnCSR4 | Initialized | Retained |
| CMnCNT4 | Initialized | Retained |
| CMnCOR4 | Initialized | Retained |
| CMnCSRH4 | Initialized | Retained |
| CMnCNTH4 | Initialized | Retained |
| CMnCORH4 | Initialized | Retained |
| CMnSTR5 | Initialized | Retained |
| CMnCSR5 | Initialized | Retained |
| CMnCNT5 | Initialized | Retained |
| CMnCOR5 | Initialized | Retained |
| CMnSTR6 | Initialized | Retained |
| CMnCSR6 | Initialized | Retained |
| CMnCNT6 | Initialized | Retained |
| CMnCOR6 | Initialized | Retained |
| CMnSTR7 | Initialized | Retained |
| CMnCSR7 | Initialized | Retained |
| CMnCNT7 | Initialized | Retained |
| CMnCOR7 | Initialized | Retained |

### 155.3.2 Register Description

[Legend for Register Description]

Initial value: Register value after a reset.

—: Undefined value

R/W: Bit or field is readable and writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

R: Read-only. The write value should always be initial value.

All access to register is made in longword units.

#### 155.3.2.1 CLK Enable Register n (CMnCLKE) (n = 1 to 3)

CMnCLKE is a 32bits register, which specify clock supply to each channel. When there are unused channels, set ‘0’ as this register, to stop supplying clock to the channel. It is prohibited to stop clock supply, while counter is working.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Ch7 clke | Ch6 clke | Ch5 clke | Ch4 clke | Ch3 clke | Ch2 clke | Ch1 clke | Ch0 clke |
| Initial | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | — | All 1 | R | Reserved<br>These bits are always read as 1. The write value should always be 1. |
| 7 | Ch7clke | 1 | R/W | 0: Clock isn't supplied to ch7<br>1: Clock is supplied to ch7 |
| 6 | Ch6clke | 1 | R/W | 0: Clock isn't supplied to ch6<br>1: Clock is supplied to ch6 |
| 5 | Ch5clke | 1 | R/W | 0: Clock isn't supplied to ch5<br>1: Clock is supplied to ch5 |
| 4 | Ch4clke | 1 | R/W | 0: Clock isn't supplied to ch4<br>1: Clock is supplied to ch4 |
| 3 | Ch3clke | 1 | R/W | 0: Clock isn't supplied to ch3<br>1: Clock is supplied to ch3 |
| 2 | Ch2clke | 1 | R/W | 0: Clock isn't supplied to ch2<br>1: Clock is supplied to ch2 |
| 1 | Ch1clke | 1 | R/W | 0: Clock isn't supplied to ch1<br>1: Clock is supplied to ch1 |
| 0 | Ch0clke | 1 | R/W | 0: Clock isn't supplied to ch0<br>1: Clock is supplied to ch0 |

### 155.3.2.2 Compare Match Timer n Start Register m (CMnSTRm) (n = 1 to 3) (m = 0 to 7)

CMnSTRm (n = 1 to 3) (m = 0 to 7) is a 32-bit register which specify the operation of compare match timer counter (CMnCNTm). Refer to section 155.3.3.5, Register Access, for register value update timing.

**(1) CMnSTR0 (n = 1 to 3)**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | STR0RS | — | — | — | — | — | — | — | STR0 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 8 | STR0RS | 0 | R/W | RCLK-Synchronous Counter Start/Stop Mode Select<br>0: Normal operation<br>Channel 0 starts or stops counting, immediately after data is written to STR0.<br>1: RCLK-synchronous counter start/stop mode<br>Channel 0 starts or stops counting on detecting an RCLK rising edge, after data is written to STR0. |
| 7 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | STR0 | 0 | R/W | Count Start 0<br>These bits specify start/halt of compare match timer counter 0 (CMnCNT0).<br>0: CMnCNT0 halts<br>1: CMnCNT0 start counting |

### (2) CMnSTRm (n = 1 to 3) (m = 1 to 4)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | STR0 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | STR0 | 0 | R/W | Count Start 0<br>These bits specify start/halt of compare match timer counter (CMnCNTm).<br>0: CMnCNTm halts<br>1: CMnCNTm starts counting |

### (3) CMnSTRm (n = 1 to 3) (m = 5 to 7)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | STR0 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | STR0 | 1 | R/W | Count Start 0<br>These bits specify start/halt of compare match timer counter (CMnCNTm).<br>0: CMnCNTm halts<br>1: CMnCNTm starts counting |

### 155.3.2.3 Compare Match Timer Control/Status n Register m (CMnCSRm) (n = 1 to 3) (m = 0 to 7)

CMnCSRm (n = 1 to 3) (m = 0 to 7) is a 32-bit register that indicates the occurrence of compare match, enable interrupt and set the counter input clock.

Do not change bits other than the CMF and OVF bits, while compare match timer counter (CMnCNTm) is under counting.

Note: * Refer to section 155.3.3.5, Register Access, for the note regarding writing to or reading form the following bits.

- CMnCSR0.CH0SSIE
- CMnCSRm.CMS
- CMnCSRm.CMM
- CMnCSRm.CMR [1:0]
- CMnCSRm.DBGIVD
- CMnCSRm.CKS [2:0]

**(1) CMnCSR0 (n = 1 to 3)**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMF | OVF | WRFLG | CH0 STTF | CH0 STPF | CH0 SSIE | CMS | CMM | — | — | CMR[1:0] | | DBG IVD | CKS[2:0] | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R/WC0 | R/WC0 | R | R/WC0 | R/WC0 | R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | CMF | 0 | R/WC0 | Compare Match Flag<br>This flag indicates whether values of the compare match timer counter (CMnCNT0) and compare match timer constant register (CMnCOR0) have matched or not.<br>Software cannot write 1 to the bit. When one-shot is selected for the counter operation, counting resumes by clearing this bit.<br>0: CMnCNT0 and CMnCOR0 values have not matched<br>[Clearing conditions]<br>• Write 0 to this bit<br>1: CMnCNT0 and CMnCOR0 values have matched<br>Note: Only 0 can be written to clear the flag. |
| 14 | OVF | 0 | R/WC0 | Overflow Flag<br>This flag indicates whether the compare match timer counter (CMnCNT0) has overflowed or not. Software cannot write 1 to this bit.<br>0: CMnCNT0 has not overflowed<br>[Clearing conditions]<br>• Write 0 to this bit<br>1: CMnCNT0 has overflowed<br>Note: Only 0 can be written to clear the flag. |
| 13 | WRFLG | 0 | R | Write State Flag<br>Write access to CMnCNT0 is prohibited, while this bit is 1. Further behavior is not guaranteed, if data is written while this bit is 1.<br>This bit indicates CMnCNT0 is in synchronization period for setting previously written data.<br>Confirm that this flag is 0, before writing to CMnCNT0. |
| 12 | CH0STTF | 0 | R/WC0 | Channel 0 Start Flag<br>When RCLK-synchronous channel 0 counter start/stop mode is selected, this flag indicates whether the counter in channel 0 started on detecting an RCLK rising edge after 1 was written to the STR0 bit in CMnSTR0.<br>0: Channel 0 counter has not started.<br>[Clearing condition]<br>• Write 0 to CH0STTF.<br>1: Channel 0 counter has started.<br>Note: Only 0 can be written to clear the flag. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 11 | CH0STPF | 0 | R/WC0 | Channel 0 Stop Flag<br>When RCLK-synchronous channel 0 counter start/stop mode is selected, this flag indicates whether the counter in channel 0 stopped on detecting an RCLK rising edge after 1 was written to the STR0 bit in CMnSTR0.<br>0: Channel 0 counter has not stopped.<br>[Clearing condition]<br>• Write 0 to CH0STPF.<br>1: Channel 0 counter has stopped.<br>Note: Only 0 can be written to clear the flag. |
| 10 | CH0SSIE | 0 | R/W | Channel 0 Start/Stop Interrupt Enable<br>When RCLK-synchronous channel 0 counter start/stop mode is selected, this bit enables or disables an interrupt due to the start or stop of the counter in channel 0.<br>0: Disables an interrupt due to start or stop of channel 0 counter.<br>1: Enables an interrupt due to start or stop of channel 0 counter. |
| 9 | CMS | 0 | R/W | Compare Match Timer Counter Size<br>This bit and CMnCSRH0.CMSH select whether the compare match timer counter 0 (CMnCNTH0 [15:0], and CMnCNT0 [31:0]) is used as a 16-bit counter, a 32-bit counter, or a 48-bit counter.<br>This bit also specifies valid size of the compare match timer constant register 0 (CMnCORH0 [15:0], and CMnCOR0 [31:0]).<br>CMnCSRH0[9].CMSH, and CMnCSR0[9].CMS:<br>1x: Operates as a 48-bit counter<br>00: Operates as a 32-bit counter<br>01: Operates as a 16-bit counter |
| 8 | CMM | 1 | R/W | Compare Match Mode<br>Specify operation mode of the counter.<br>0: One-shot operation<br>1: Free-running operation |
| 7 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 5, 4 | CMR[1:0] | 00 | R/W | Compare Match Request<br>These bits enable or disable internal interrupt request in a compare match.<br>00: Disables internal interrupt request<br>01: Setting prohibited<br>10: Enables an internal interrupt request<br>11: Setting prohibited |
| 3 | DBGIVD | 1 | R/W | Debug Mode Operation Select<br>Sets the counter operation in debugging mode.<br>0: Stops the counter operation in debugging mode.<br>1: Continues the counter operation even in debugging mode. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 to 0 | CKS[2:0] | 111 | R/W | Clock Select<br>These bits and CMnCSRH0.CKSH select the clock input to CMnCNT0. When the count start bit (STR0) for the corresponding channel is set to 1, CMnCNT0 begins incrementing with the clock selected by these bits.<br>CMnCSRH0.CKSH[0]+CMnCSR0.CKS[2:0]:<br>x000: CPEXϕ/8<br>x001: CPEXϕ/32<br>x010: CPEXϕ/128<br>x011: Setting prohibited<br>0100: RCLK/8<br>0101: RCLK/32<br>0110: RCLK/128<br>0111: RCLK/1<br>1100: Pseudo 32 kHz/8<br>1101: Pseudo 32 kHz/32<br>1110: Pseudo 32 kHz/128<br>1111: Pseudo 32 kHz/1 |

### (2) CMnCSRm (n = 1 to 3) (m = 1 to 4)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMF | OVF | WRFLG | — | — | — | CMS | CMM | — | — | CMR[1:0] | | DBGIVD | CKS[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R/WC0 | R/WC0 | R | R | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | CMF | 0 | R/WC0 | Compare Match Flag<br>This flag indicates whether values of the compare match timer counter (CMnCNTm) and compare match timer constant register (CMnCORm) have matched or not.<br>Software cannot write 1 to the bit. When one-shot is selected for the counter operation, counting resumes by clearing this bit.<br>0: CMnCNTm and CMnCORm values have not matched<br>[Clearing conditions]<br>• Write 0 to this bit<br>1: CMnCNTm and CMnCORm values have matched<br>Note: Only 0 can be written to clear the flag. |
| 14 | OVF | 0 | R/WC0 | Overflow Flag<br>This flag indicates whether the compare match timer counter (CMnCNTm) has overflowed or not. Software cannot write 1 to this bit.<br>0: CMnCNTm has not overflowed<br>[Clearing conditions]<br>• Write 0 to this bit<br>1: CMnCNTm has overflowed<br>Note: Only 0 can be written to clear the flag. |
| 13 | WRFLG | 0 | R | Write State Flag<br>Write access to CMnCNTm is prohibited, while this this bit is 1. Further behavior is not guaranteed, if data is written while this bit is 1.<br>This bit indicates CMnCNTm is in synchronization period for setting previously written data.<br>Confirm that this flag is 0, before writing to CMnCNTm. |
| 12 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | CMS | 0 | R/W | Compare Match Timer Counter Size<br>This bit and CMnCSRHm.CMSH specify whether the compare match timer counter (CMnCNTHm[15:0] and CMnCNTm[31:0]) is used as a 16-bit counter, a 32-bit counter, or a 48-bit counter.<br>This setting becomes the valid size for the compare match timer constant register (CMnCORHm[15:0], CMnCORm[31:0]).<br>CMnCSRHm[9].CMSH+CMnCSRm[9].CMS:<br>1x: Operates as a 48-bit counter<br>00: Operates as a 32-bit counter<br>01: Operates as a 16-bit counter |
| 8 | CMM | 1 | R/W | Compare Match Mode<br>Specify counter operation mode.<br>0: One-shot operation<br>1: Free-running operation |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7, 6 | — | All 0 | R | Reserved<br>These bits are read as 0. The write value should always be 0. |
| 5, 4 | CMR[1:0] | 00 | R/W | Compare Match Request<br>These bits enable or disable internal interrupt request in a compare match.<br>00: Disables internal interrupt request<br>01: Setting prohibited<br>10: Enables an internal interrupt request<br>11: Setting prohibited |
| 3 | DBGIVD | 1 | R/W | Debug Mode Operation Select<br>Sets the counter operation in debugging mode.<br>0: Stops the counter operation in debugging mode.<br>1: Continues the counter operation even in debugging mode. |
| 2 to 0 | CKS[2:0] | 111 | R/W | Clock Select<br>These bits and CMnCSRHm.CKSH specify the input clock to CMnCNTm. When the count start bit (STR0) for the corresponding channel is set to 1, CMnCNTm begins incrementing with the clock selected by these bits.<br>CMnCSRHm.CKSH[0], CMnCSRm.CKS[2:0]:<br>x000: CPEXϕ/8<br>x001: CPEXϕ/32<br>x010: CPEXϕ/128<br>x011: Setting prohibited<br>0100: RCLK/8<br>0101: RCLK/32<br>0110: RCLK/128<br>0111: RCLK/1<br>1100: Pseudo 32 kHz/8<br>1101: Pseudo 32 kHz/32<br>1110: Pseudo 32 kHz/128<br>1111: Pseudo 32 kHz/1 |

### (3) CMnCSRm (n = 1 to 3) (m = 5 to 7)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMF | OVF | WRFLG | — | — | — | CMS | CMM | — | — | CMR[1:0] | | DBGIVD | CKS[2:0] | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R/WC0 | R/WC0 | R | R | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | CMF | 0 | R/WC0 | Compare Match Flag<br>This flag indicates whether values of the compare match timer counter (CMnCNTm) and compare match timer constant register (CMnCORm) have matched or not.<br>Software cannot write 1 to the bit. When one-shot is selected for the counter operation, counting resumes by clearing this bit.<br>0: CMnCNTm and CMnCORm values have not matched<br>[Clearing conditions]<br>• Write 0 to this bit<br>1: CMnCNTm and CMnCORm values have matched<br>Note: Only 0 can be written to clear the flag. |
| 14 | OVF | 0 | R/WC0 | Overflow Flag<br>This flag indicates whether the compare match timer counter (CMnCNTm) has overflowed or not. Software cannot write 1 to this bit.<br>0: CMnCNTm has not overflowed<br>[Clearing conditions]<br>• Write 0 to this bit<br>1: CMnCNTm has overflowed<br>Note: Only 0 can be written to clear the flag. |
| 13 | WRFLG | 0 | R | Write State Flag<br>Write access to CMnCNTm is prohibited, while this this bit is 1. Further behavior is not guaranteed, if data is written while this bit is 1.<br>This bit indicates CMnCNTm is in synchronization period for setting previously written data.<br>Confirm that this flag is 0, before writing to CMnCNTm. |
| 12 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | CMS | 0 | R/W | Compare Match Timer Counter Size<br>Selects whether the compare match timer counter (CMnCNTm) is used as a 16-bit counter or a 32-bit counter.<br>This setting becomes the valid size for the compare match timer constant register (CMnCORm).<br>0: Operates as a 32-bit counter.<br>1: Operates as a 16-bit counter. |
| 8 | CMM | 1 | R/W | Compare Match Mode<br>Specify counter operation mode.<br>0: One-shot operation<br>1: Free-running operation |
| 7, 6 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5, 4 | CMR[1:0] | 00 | R/W | Compare Match Request<br>These bits enable or disable internal interrupt request in a compare match.<br>00: Disables internal interrupt request<br>01: Setting prohibited<br>10: Enables an internal interrupt request<br>11: Setting prohibited |
| 3 | DBGIVD | 1 | R/W | Debug Mode Operation Select<br>Sets the counter operation in debugging mode.<br>0: Stops the counter operation in debugging mode.<br>1: Continues the counter operation even in debugging mode. |
| 2 to 0 | CKS[2:0] | 111 | R/W | Clock Select<br>These bits select the clock input to CMnCNTm. When the count start bit (STR0) in CMnSTRm is set to 1, CMnCNTm begins incrementing with the clock selected by these bits.<br>000: Setting prohibited<br>001: Setting prohibited<br>010: Setting prohibited<br>011: Setting prohibited<br>100: RCLK/8<br>101: RCLK/32<br>110: RCLK/128<br>111: RCLK/1 |

### 155.3.2.4 Compare Match Timer n Control/Status Register H m (CMnCSRHm) (n = 1 to 3) (m = 0 to 4)

CMnCSRHm is a 32-bit register which specify the counter size and input clocks.

Note: * For write to or read from CMnCSRHm.CMSH and CMnCSRHm.CKSH, refer to section 155.3.3.5, Register Access.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | CMSH | — | — | — | — | — | — | — | — | CKSH |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | CMSH | 0 | R/W | Compare Match Timer Counter Size<br>This bit and CMnCSRm.CMS specify whether the compare match timer counter (CMnCNTHm[15:0] and CMnCNTm[31:0]) is used as a 16-bit counter, a 32-bit counter, or a 48-bit counter.<br>This bit also specify valid size for the compare match timer constant register (CMnCORHm [15:0], CMnCORm [31:0]).<br>CMnCSRHm.CMSH[9], CMnCSRm.CMS[9]:<br>1x: Operates as a 48-bit counter<br>00: Operates as a 32-bit counter<br>01: Operates as a 16-bit counter |
| 8 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | CKSH | 0 | R/W | Clock Select<br>This bit and CMnCSRm.CKS [2:0] select the clock input to CMnCNTm. When the count start bit (STR0) for the corresponding channel is set to 1, CMnCNTm begins incrementing with the clock selected by these bits.<br>CMnCSRHm.CKSH[0], CMnCSRm.CKS[2:0]:<br>x000: CPEXϕ/8<br>x001: CPEXϕ/32<br>x010: CPEXϕ/128<br>x011: Setting prohibited<br>0100: RCLK/8<br>0101: RCLK/32<br>0110: RCLK/128<br>0111: RCLK/1<br>1100: Pseudo 32kHz/8<br>1110: Pseudo 32kHz/32<br>1101: Pseudo 32kHz/128<br>1111: Pseudo 32kHz/1 |

### 155.3.2.5 Compare Match Timer n Counter m (CMnCNTm) (n = 1 to 3) (m = 0 to 7)

CMnCNTm (n = 1 to 3) (m = 0 to 7) is a 32-bit register which is used as an up-counter of each channel.

To specify counter operation, set compare match timer control/status register n (CMnCSRm), before starting operation of corresponding channel. When the 16-bit counter operation is selected by the CMnCSRHm.CMSH and CMnCSRm.CMS bits, bits 31 to 16 of this register is invalid. When data is written to this register in 16-bit mode, write H'0000 to the upper 16 bit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMCNT[31:16] | | | | | | | | | | | | | | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMCNT[15:0] | | | | | | | | | | | | | | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMCNT [31:0] | All 0 | R/W | Compare match timer counter bit31 to 0<br>Note: For access to this register, refer to section 155.3.3.5, Register Access. |

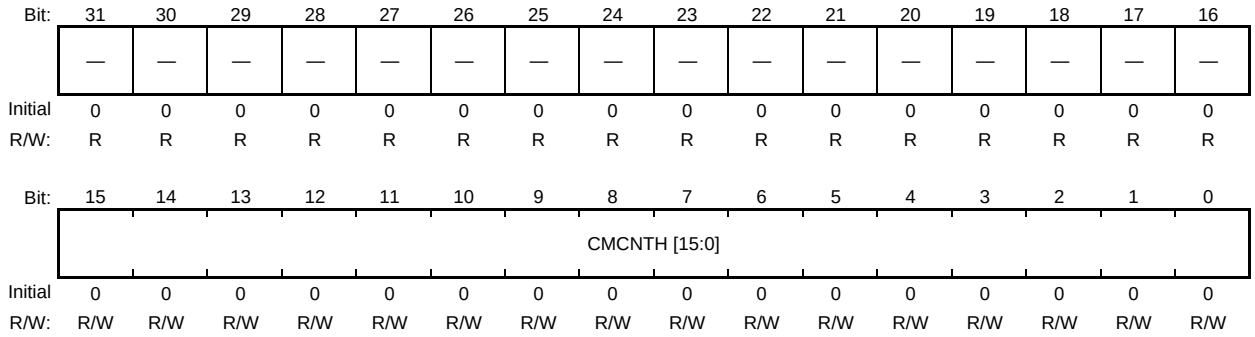
### 155.3.2.6 Compare Match Timer n Counter H m (CMnCNTHm) (n = 1 to 3) (m = 0 to 4)

CMnCNTHm (n = 1 to 3) (m = 0 to 4) is a 32-bit register which is used as an up-counter for channel0 to 4.

To specify counter operation, set compare match timer control/status register n (CMnCSRm/CMnCSRHm), before starting operation of corresponding channel. When the 48-bit counter operation is selected by the CMnCSRHm.CMSH and CMnCSRm.CMS bits, bits 15 to 0 of this register become valid as the upper 16-bit of 48-bit counter. The value written to this register is reflected to counter behavior, when lower data is written to CMnCNTm. Write the upper data to this register first. The value is not reflected to counter's behavior, if data is not written to CMnCNTm later.

When reading the value of 48-bit counter, upper 16-bit can be read out by reading bits 15 to 0 of this register. At the same time of the reading, the value of 48-bit counter is stored to the read-buffer. After reading CMnCNTHm, lower 32-bit can be read out by reading CMnCNTm.

Note: If CMnCNTm is read out before reading CMnCNTHm, the read out value of 48-bit counter may not be correct.



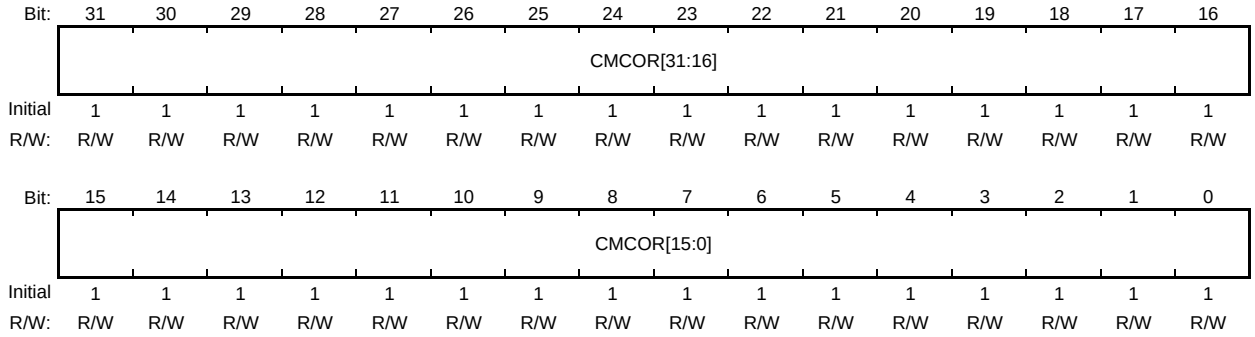

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 0 | CMCNTH [15:0] | All 0 | R/W | Compare match timer counter H bit15 to 0<br>Note: For access to this register, refer to section 155.3.3.5, Register Access. |

### 155.3.2.7 Compare Match Timer n Constant Register m (CMnCORm) (n = 1 to 3) (m = 0 to 7)

CMnCORm is a 32-bit register which specify the compare match period of CMnCNTm for each channel.

When the 16-bit counter operation is selected by the CMnCSRHm.CMSH and CMnCSRm.CMS bits, bits 31 to 16 of this register is invalid. When data is written to this register in 16-bit operation mode, write H'0000 as upper 16bit.

An overflow is detected when CMnCNTm is cleared to 0 and this register is H'FFFF FFFF (when the 16-bit counter operation it is H’0000 FFFF).



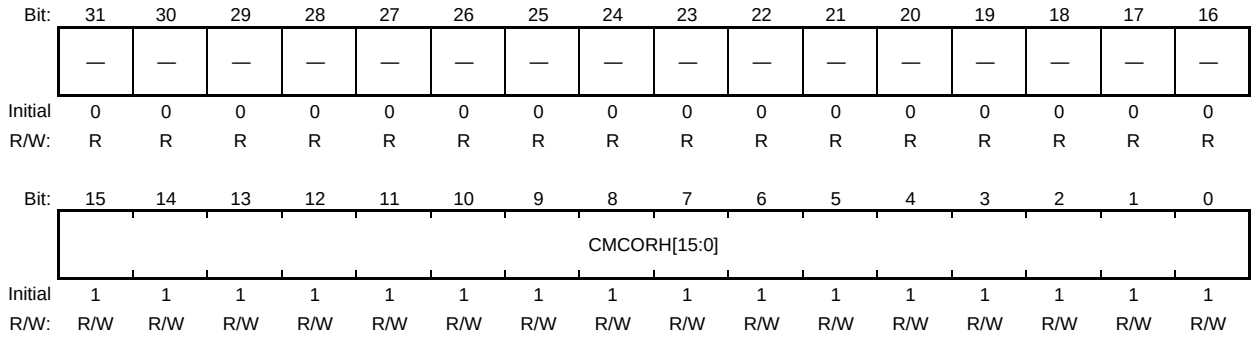

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMCOR [31:0] | All 1 | R/W | Compare match timer constant register bit31 to 0<br>Note: For access to this register, refer to section 155.3.3.5, Register Access. |

### 155.3.2.8 Compare Match Timer n Constant Registers H m (CMnCORHm) (n = 1 to 3) (m = 0 to 4)

CMnCORHm is a 32-bit register which specify the compare match period with CMnCNTHm for channel0 to 4.

When the 48-bit counter operation is selected by the CMnCSRHm.CMSH and CMnCSRm.CMS bits, bits 15 to 0 of this register become valid as the upper 16-bit of 48-bit counter.

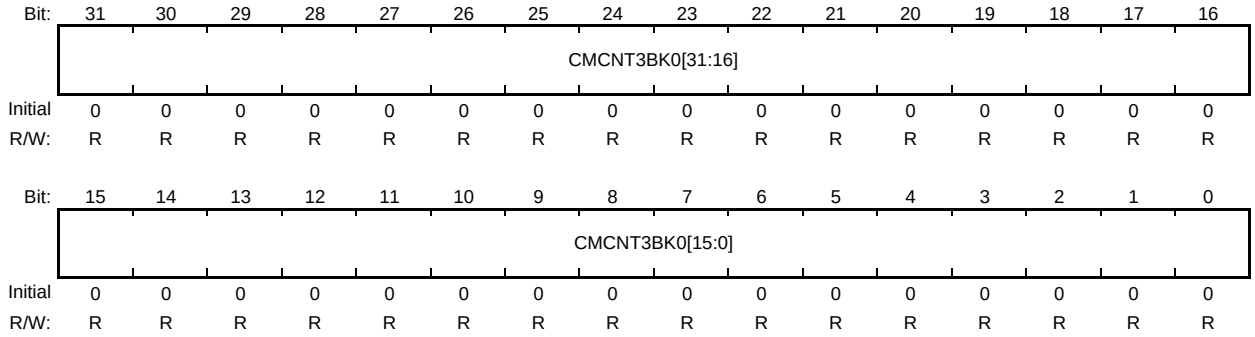
An overflow is detected when CMnCNTHm is cleared to 0 and this register is H'FFFF.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 0 | CMCORH [15:0] | All 1 | R/W | Compare match timer constant register H bit15 to 0<br>Note: For access to this register, refer to section 155.3.3.5, Register Access. |

### 155.3.2.9 Compare Match Timer n Counter 3 Backup 0 (CMnCNT3BK0) (n = 1 to 3)

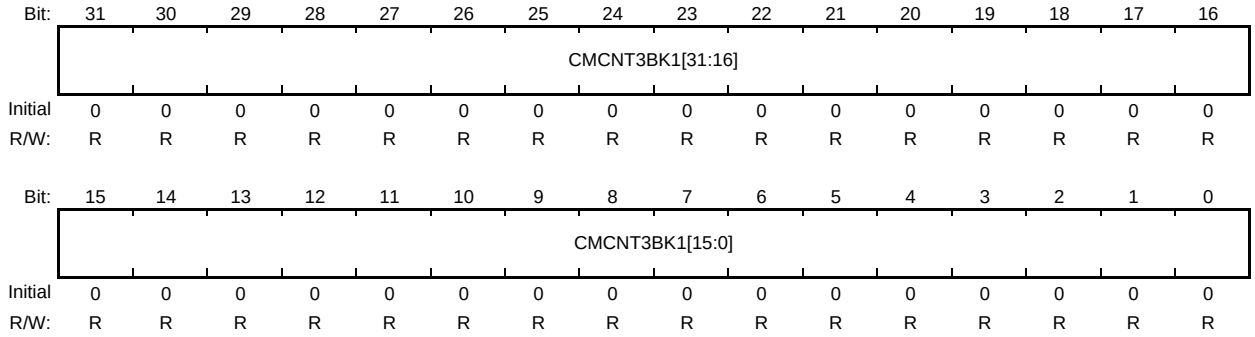
CMnCNT3BK0 is a 32-bit register which stores a copy of the CMnCNT3 value immediately after the counter in channel 0 stops in RCLK-synchronous channel 0 counter start/stop mode.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMCNT3BK0[31:0] | All 0 | R | Compare match timer counter 3 backup 0 bit31 to 0 |

### 155.3.2.10 Compare Match Timer n Counter 3 Backup 1 (CMnCNT3BK1) (n = 1 to 3)

CMnCNT3BK1 is a 32-bit register which stores a copy of the CMnCNT3 value immediately after the counter in channel 0 starts in RCLK-synchronous channel 0 counter start/stop mode.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMCNT3BK1[31:0] | All 0 | R | Compare match timer counter 3 backup 1 bit31 to 0 |

### 155.3.2.11 Compare Match Timer n Match Control/Status Register m (CMnCSRMm) (n = 1 to 3) (m = 0 and 3)

CMnCSRMm is a 32-bit register which resets to compare match timer match counter, and sets the counter start/halt. This register is only present in channels 0 and 3.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WRFLG | — | — | — | — | — | — | — | — | — | — | — | — | — | CMPCLR | CMPSTART |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 | WRFLG | 0 | R | Write state flag.<br>When this Bit is 1, CMnCNTMm cannot be cleared.<br>After CMnCNTMm is cleared, this bit become 1 for a while (*), for indicating synchronization period.<br>Needs to confirm this flag is 0 or wait the period enough when continuously clearing.<br>Note: * In the maximum, "Counter input clock 6 cycle "<br>• Counter input clock<br>CMnCSRm.CKS2 = 0: CPEXϕ<br>CMnCSRm.CKS2 = 1: RCLK |
| 14 to 2 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 1 | CMPCLR | 0 | R/W | Counter Clear<br>[When writing]<br>0: No operation<br>1: Clears counter<br>When CMPCLR and CMPSTART are written 1 at the same time, the counter starts after clearing the counter.<br>[When reading]<br>This bit is always read as 0. |
| 0 | CMPSTART | 0 | R/W | Count Start<br>This bit specifies start/halt of the compare match timer match counter (CMnCNTMm) of each channel.<br>0: CMnCNTMm Halts<br>1: CMnCNTMm Starts counting |

### 155.3.2.12 Compare Match Timer n Match Counter m (CMnCNTMm) (n = 1 to 3) (m = 0 and 3)

CMnCNTMm is a 32-bit register which is used as an up-counter. This register is only present in channels 0 and 3.

A counter operation is set by the compare match timer match control/status register (CMnCSRMm). After the counter CMnCNTm starts, this counter increases whenever at a compare match.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMCNTM[31:16] | | | | | | | | | | | | | | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMCNTM[15:0] | | | | | | | | | | | | | | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMCNTM [31:0] | All 0 | R | Compare match timer match counter bit31 to 0 |

## 155.3.3 Operation

### 155.3.3.1 Counter Operation

The CMT1 starts the operation of the counter by writing H'1 to the STR0 bit in CMnSTRm after each register has been set. Complete all of the settings before starting the operation. Do not change the register settings other than clearing flag bits, while the compare match timer (CMnCNTm) is under operation.

The counter operates in one of two ways.

- One-Shot Operation

  One-shot operation is selected by setting the CMM bit in CMnCSRm to H'0. When the value in CMnCNTm matches the value in CMnCORm, the value in CMnCNTm is cleared to H'0000 0000 and the CMF bit in CMnCSRm is set to H'1. Counting by CMnCNTm stops after CMnCNTm has been cleared.

  Counting by CMnCNTm will be resumed after CMnCSRm.CMF bit has been cleared.

  To detect an overflow interrupt, set the value in CMnCORm to H'FFFF FFFF. When the value in CMnCNTm matches the value in CMnCORm, CMnCNTm is cleared to H'0000 0000 and the CMF and OVF bits in CMnCSRm are set to H'1.

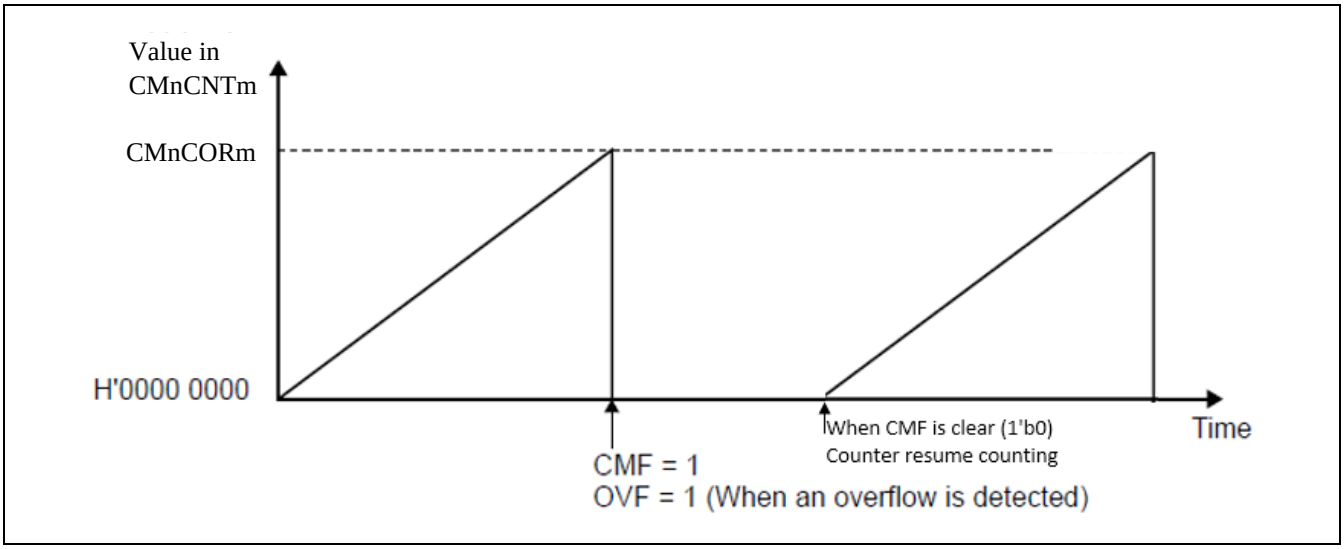



**Figure 155.3.3.1 Counter Operation (One-Shot Operation)**

- Free-Running Operation

  Free-running operation is selected by setting the CMM bit in CMnCSRm to H'1. When the value in CMnCNTm matches the value in CMnCORm, CMnCNTm is cleared to H'0000 0000 and the CMF bit in CMnCSRm is set to H'1. CMnCNTm resumes counting-up after it has been cleared.

  To detect an overflow interrupt, set CMnCORm to H'FFFF FFFF. When the values in CMnCNTm and CMnCORm match, CMnCNTm is cleared to H'0000 0000 and the CMF and OVF bits in CMnCSRm are set to H'1.

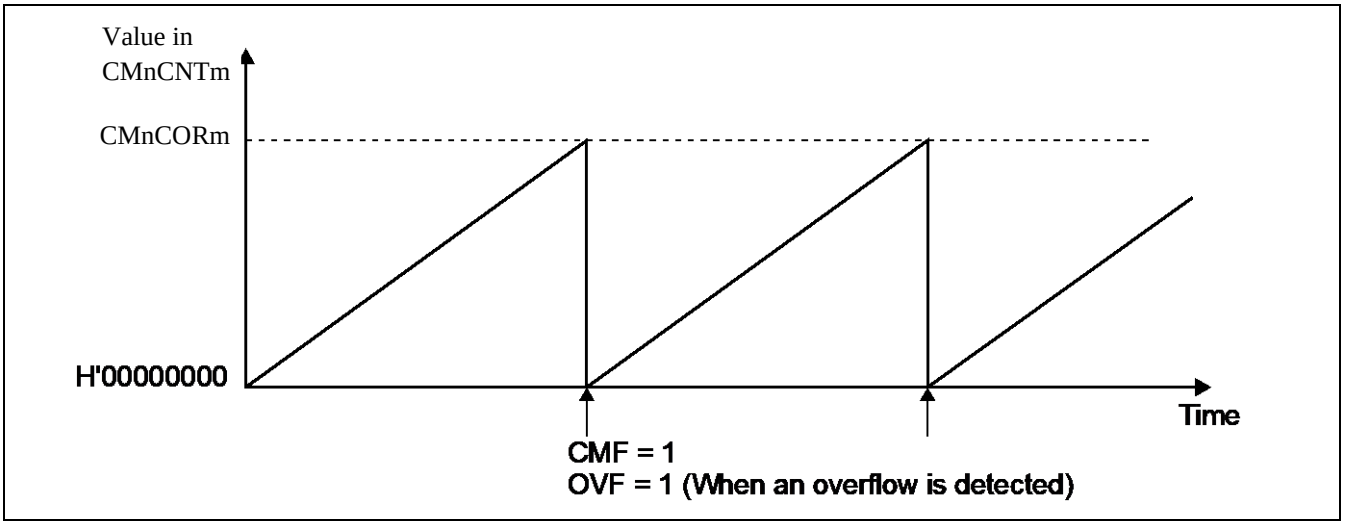



**Figure 155.3.3.2 Counter Operation (Free-Running Operation)**

#### 155.3.3.2 Counter Size

In this module, the size of the counter can be selected from 16, 32, or 48 bits. This is selected by the CMS bit in CMnCSRm and CMSH bit in CMnCSRHm.

When the 16-bit/32-bit size is selected, upper 32 bits/16 bits of CMnCORm is ignored. To detect an overflow interrupt, the value 1 must be set to valid bits of CMnCORm.

(for example, H'0000 0000 FFFF, H'0000 FFFF FFFF)

#### 155.3.3.3 Timing for Counting by CMnCNTm

In this module, the clock for the counter can be selected from among the following:

| | | |
|---|---|---|
| Channels 0 to 4: | CPEXϕ clock: | 1/8, 1/32, and 1/128 |
| | RCLK: | 1/1, 1/8, 1/32, and 1/128 |
| | Pseudo 32-kHz: | 1/1, 1/8, 1/32 and 1/128 |
| Channel 5 to 7: | RCLK: | 1/1, 1/8, 1/32, and 1/128 |

For details of CPEXϕ and RCLK, refer to section 8, Clock Pulse Generator (CPG).

Pseudo 32-kHz is a clock, which is generated inside this module. Refer to section 155.3.3.8, Pseudo 32-kHz Counter.

The clock for the counter is selected by the CKS bits in CMnCSRm and CKSH bit in CMnCSRHm. CMnCNTm is incremented at the rising edge of the selected clock.

#### 155.3.3.4 Internal Interrupt Request to CPU

By CMR bits in CMnCSRm, internal interrupt request to the CPU at a compare match can be asserted.

To clear an internal interrupt request to the CPU in Free-Running Operation, the CMF bit should be set to H'0. Set the CMF bit to H'0 in the handling routine for the CMTn interrupt.

To clear an internal interrupt request to the CPU with resuming count-up in One-Shot Operation, the CMF bit should be set to H'0 while keeping H'1 in STR0 bit in CMnSTRm. Set the CMF bit to H'0 while keeping H'1 in STR0 bit in the handling routine for the CMTn interrupt.

To clear an internal interrupt request to the CPU without resuming count-up in One-Shot Operation, the CMF bit should be set to H'0 after setting STR0 bit in CMnSTRm to H'0. Set the CMF bit to H'0 after setting STR0 bit to H'0 in the handling routine for the CMTn interrupt.

#### 155.3.3.5 Register Access

After writing to following registers, written data can be read 2 cycles in counter input clock (RCLK) after writing has finished. And it takes 2 cycles in counter input clock (RCLK), for reflecting written data to counter behavior.

- CMnCSRm: Bits CH0SSIE(ch0), CMS, CMM, CMR[1:0], DBGIVD, CKS[2:0]
- CMnCORm, CMnCORHm
- CMnSTRm: Bit STR0RS, STR0
- CMnCSRMm: Bit CMPCLR, CMPSTART

After writing to following registers, written data can be read 2 cycles in counter input clock (RCLK or CPEXϕ) after writing has finished. And it takes 2 cycles in counter input clock (RCLK or CPEXϕ), for reflecting written data to counter behavior. After writing data to this register, next write access is prohibited, until CMnCSRm.WRFLG become 0.

- CMnCNTm, CMnCNTHm

### 155.3.3.6 Compare Match Flag Set/Clear Timing

The CMF bit in CMnCSRm is set to 1 by the compare match signal generated when CMnCORm and CMnCNTm match. The compare match signal is generated upon the final state of the match (timing at which the CMnCNTm value is updated to H'0000). Consequently, after CMnCORm and CMnCNTm match, a compare match signal will not be generated until a CMnCNTm counter clock is input. Figure 155.3.3.3 shows the set timing of the CMF bit.

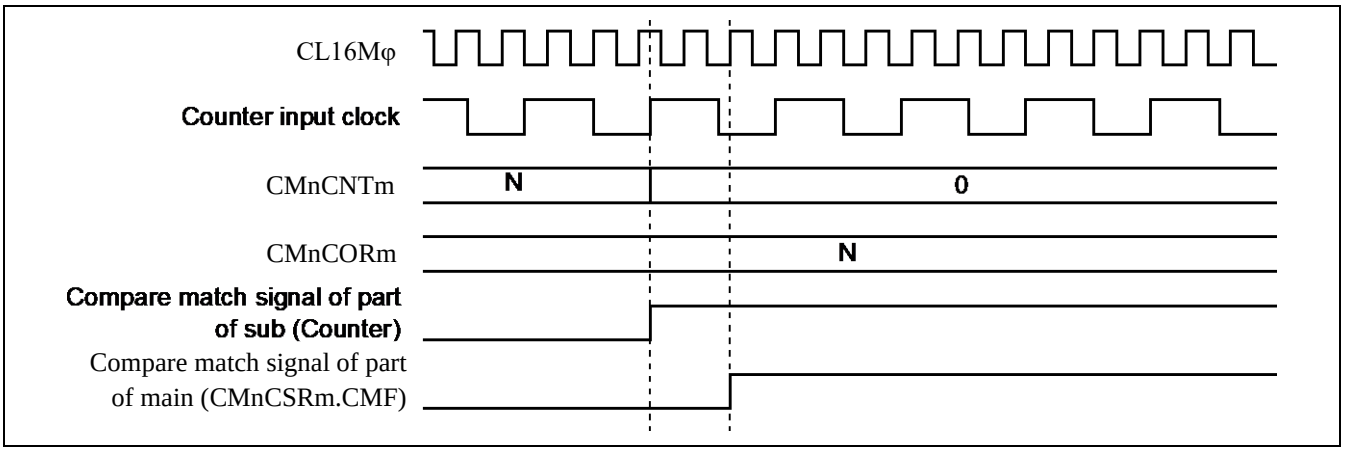



**Figure 155.3.3.3 CMF Set Timing**

Set the CMF bit to 0. The CMF flag is cleared immediately.

### 155.3.3.7 RCLK-Synchronous Counter Start/Stop Mode in Channel 0

The CMT1 provides the RCLK-synchronous counter start/stop mode in channel 0, in which channels 0 and 3 can be used together for time measurement.

Figure 155.3.3.4 shows a flow chart for preparation.

Figure 155.3.3.5 shows a flow chart for starting measurement.

Figure 155.3.3.6 shows a flow chart for stopping measurement.

Figure 155.3.3.7 shows a timing chart when measurement starts.

Figure 155.3.3.8 shows a timing chart when measurement stops.

Figure 155.3.3.9 shows how to calculate the time measured by the counters.

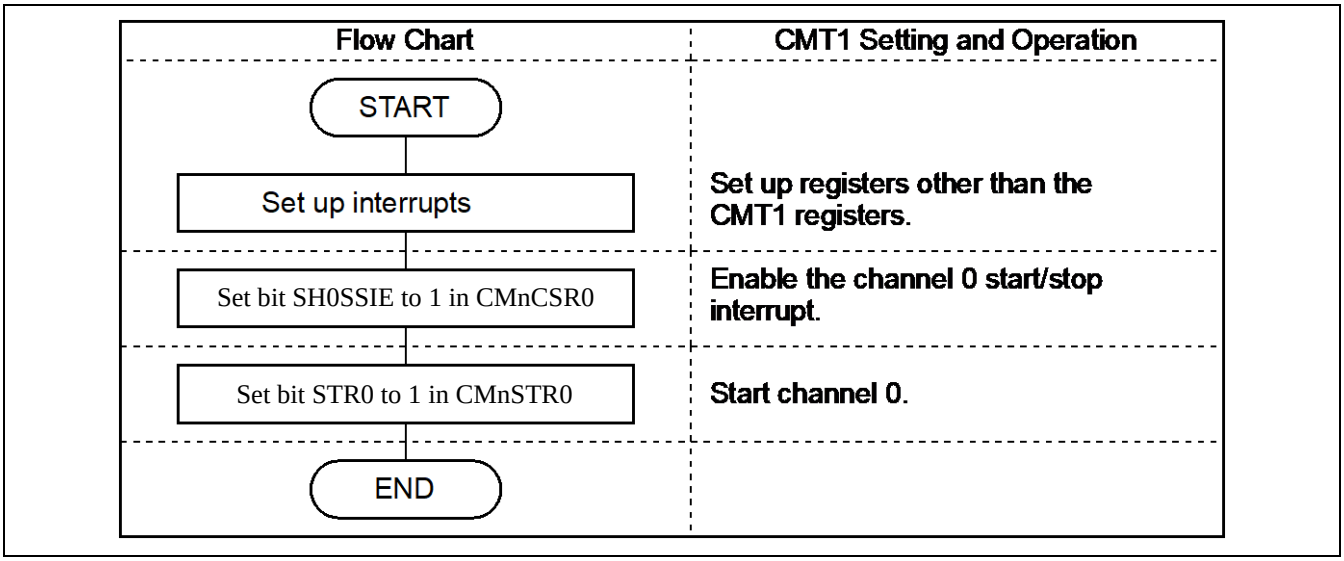



Figure 155.3.3.4 Flow Chart for Preparation

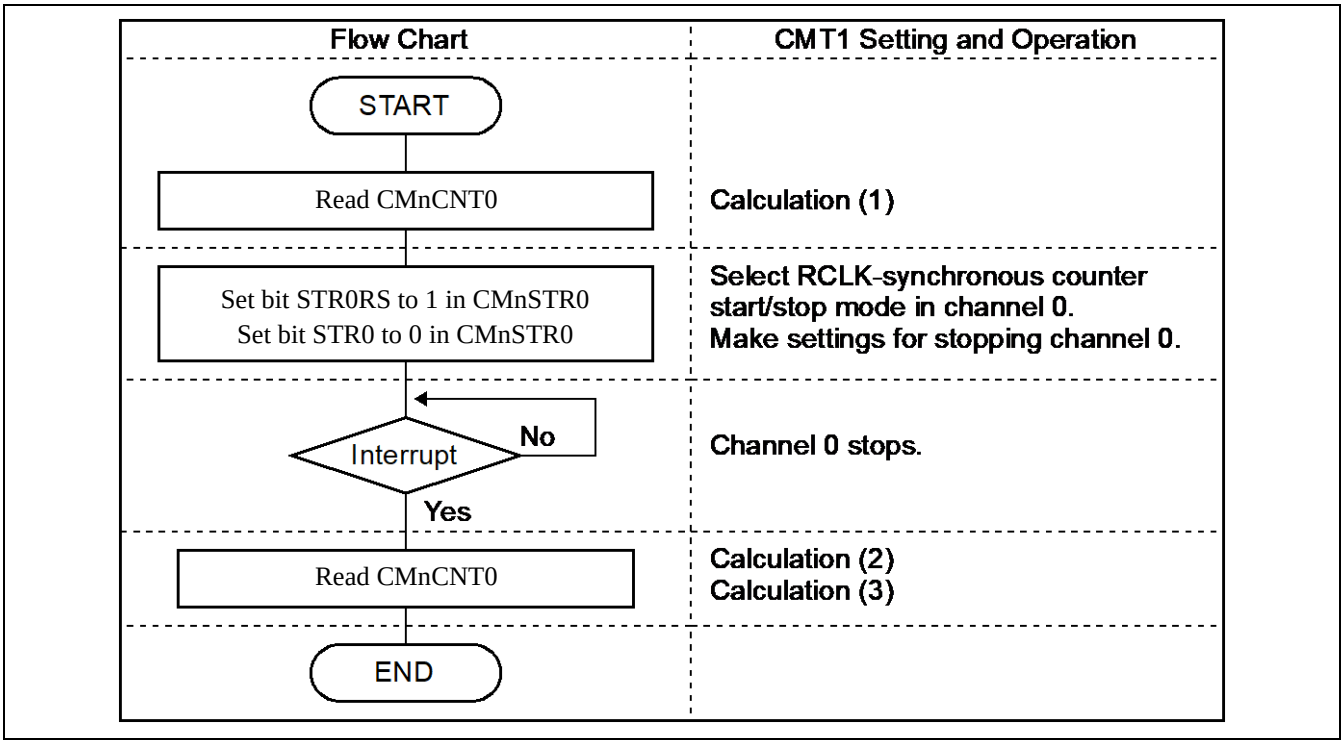



Figure 155.3.3.5 Flow Chart for Starting Measurement

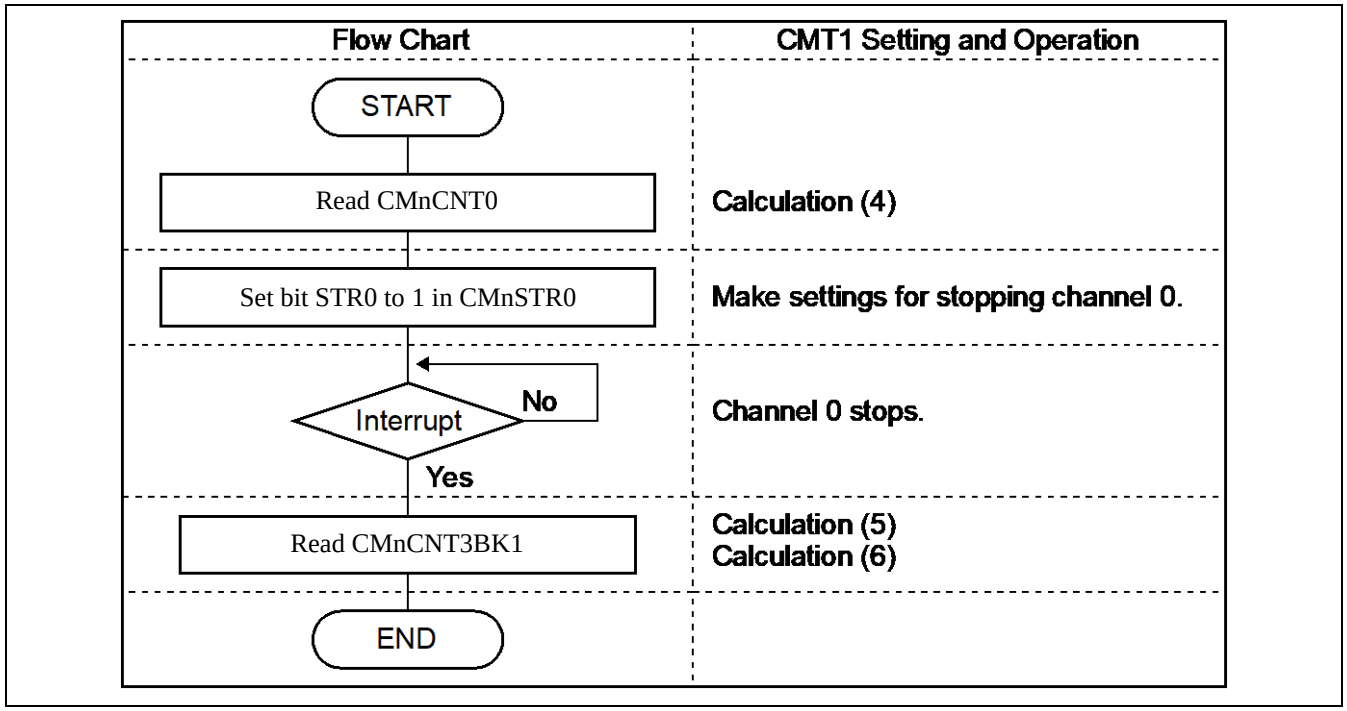



**Figure 155.3.3.6 Flow Chart for Stopping Measurement**

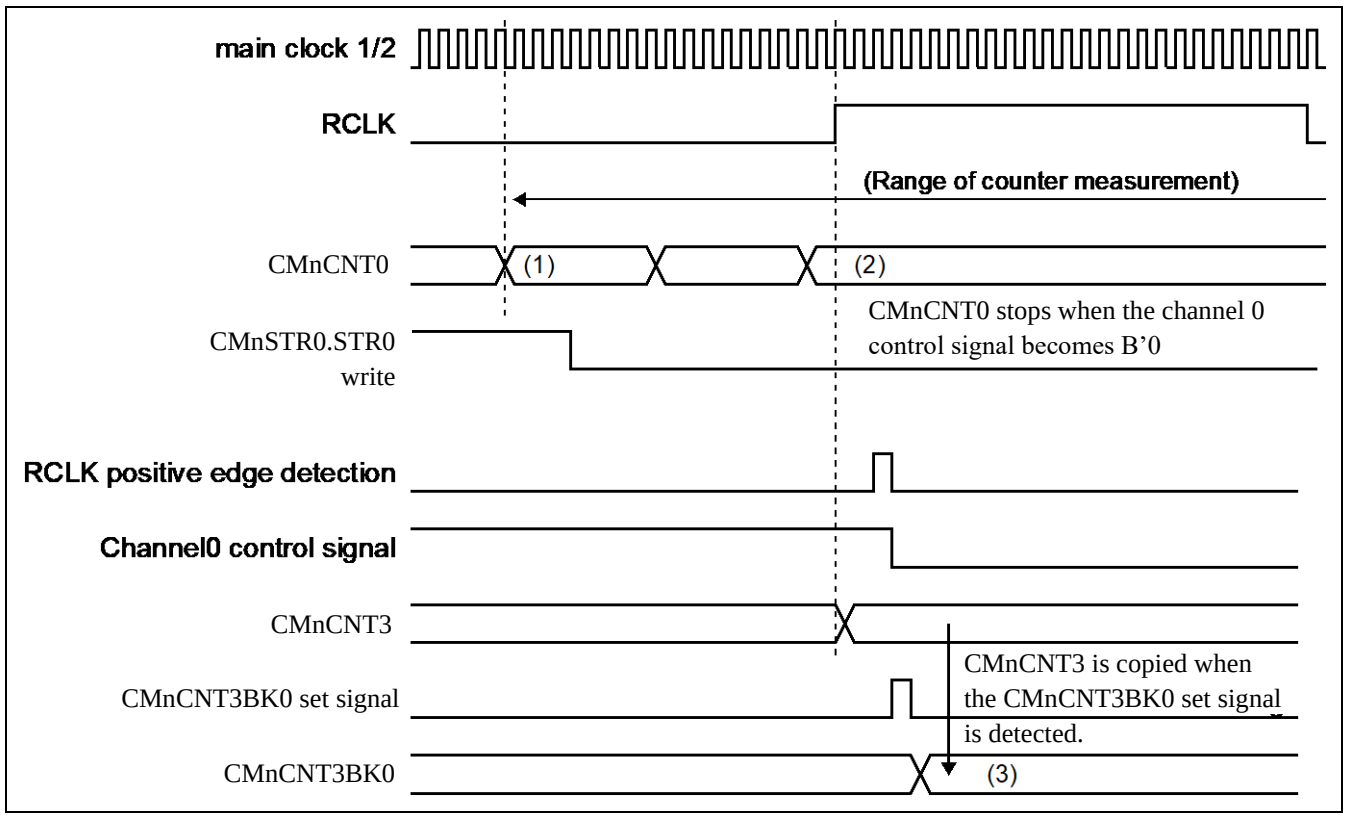



**Figure 155.3.3.7 Timing Chart When Measurement Starts**

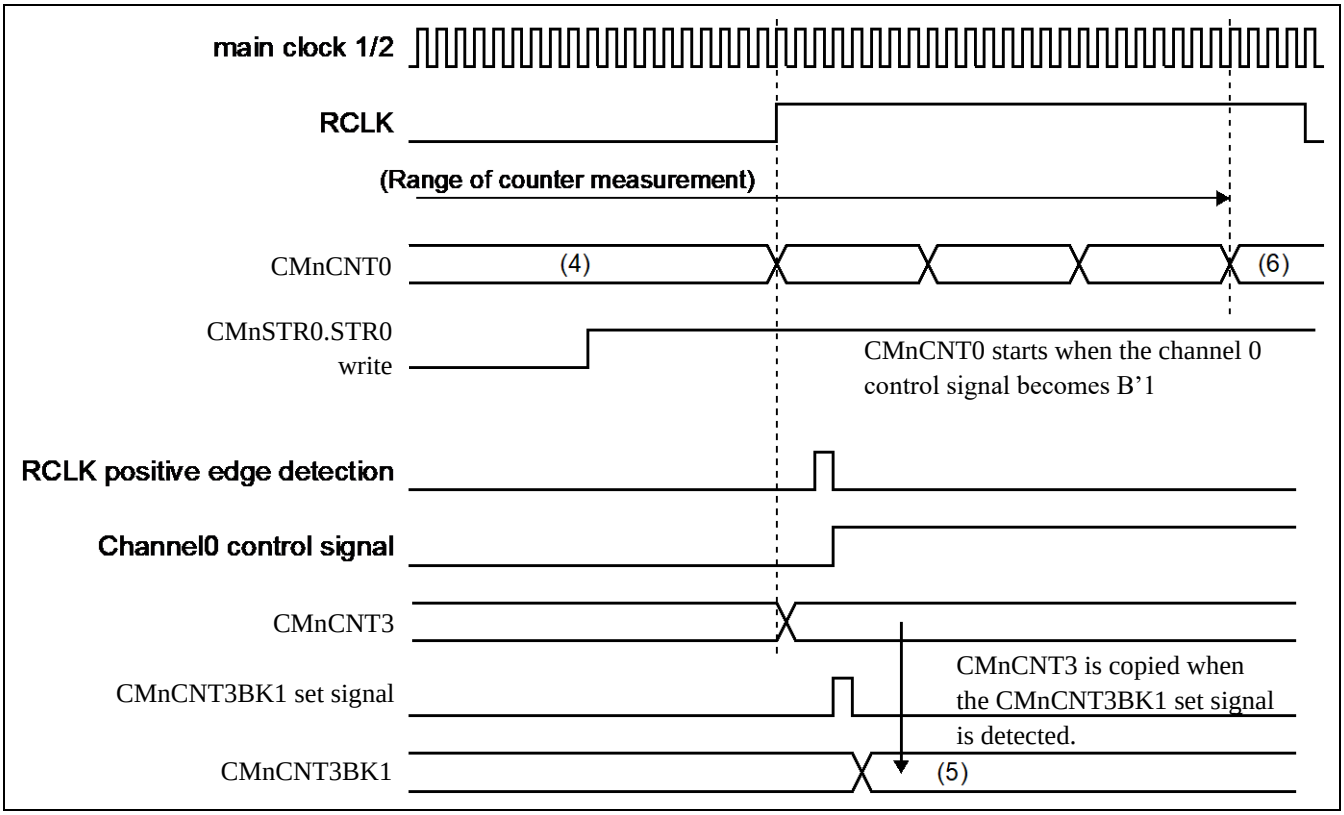



**Figure 155.3.3.8 Timing Chart When Measurement Stops**

(((2) - (1)) + ((6) - (4) - 1)) × Tapc + ((5) - (3)) × Tr
Tapc : Cycle of main clock 1/2
Tr : Cycle of RCLK

**Figure 155.3.3.9 Calculation of Time Measured by Counters**

#### 155.3.3.8 Pseudo 32-kHz Counter

Pseudo 32-kHz clock is created inside this module, based on RCLK by Pseudo 32-kHz Counter. By removing 3 edge of RCLK every 128 cycle, Pseudo 32-kHz clock is generated. By this edge removing, frequency of this clock becomes 128/131 of RCLK.

#### 155.3.3.9 CMT1 Usage

Take the following steps to use the CMT1.

1. Clear the STR0 bit in CMnSTRm to 0 to temporarily stop counting.
   (Confirm that Counter input clock was input more than 2 cycles.)
2. Write H'0000 0000 on CMnCNTm.
3. Set counter size, compare match mode, kind of count clock and interrupt request and clear the bit of OVF and CMF in CMnCSRm.
4. Set the value on CMnCORm.
5. Confirm that CMnCSRm.WRFLG is 0.
   If WRFLG is 1, wait until it'll be 0.
6. Start the counting by setting the STR0 bit in CMnSTRm to 1 (refer to section 155.3.3.5, Register Access).

#### 155.3.3.10 Module Stop/ CMnCLKE Setting

When counter is not used, clock of this module can be stopped by CMnCLKE register. Confirm that, following conditions are satisfied, before stopping counter’s clock.

- CMnSTRm.STR0 bit is 0
- CMnCSRm.WRFLG is 0
- 3 or more cycles have passed in Counter input clock since the last Register access

### 155.3.4 Usage Notes

No description on this section.

### 155.3.5 Safety Mechanisms

#### 155.3.5.1 Power-On test

This module is protected by Power-On test.

# 155.4 Timer Unit (TMU)

## 155.4.1 Overview

This LSI includes a 32-bit timer unit (TMU) with 15 channels (channels 0 to 14).
Channels 0 to 2, 3 to 5, 6 to 8, 9 to 11 and 12 to 14 are grouped into timers 0, 1, 2, 3, 4 respectively.
For channels 0 to 2, SASYNCRTΦ is provided as a base clock. For channels 3 to 14, SASYNCPERD2Φ is provided.

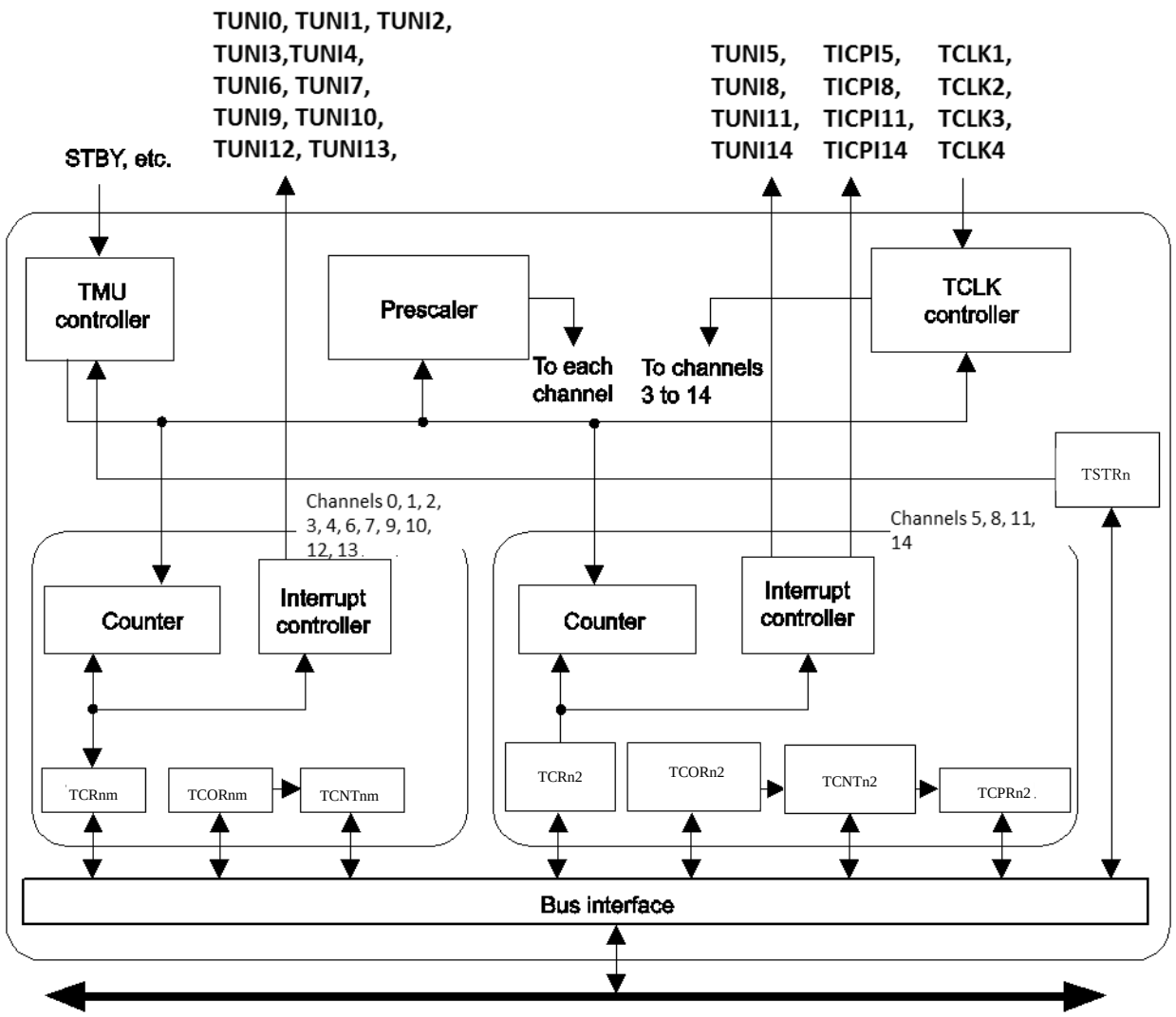
### 155.4.1.1 Features

TMU has the following features:

- Auto-reload type 32-bit down counter is provided for each channel.
- Input capture function provided: Channels 5, 8, 11 and 14.
- Selection of rising edge or falling edge as external clock input edge when external clock is selected or input capture function is used: Channels 3 to 14
- 32-bit timer constant register for auto-reload use, readable/writable at any time, and 32-bit down-counter are provided for each channel.
- Selection of five counter input clocks: Channels 0 to 2.
  Five peripheral clocks (SASYNCRTΦ/4, SASYNCRTΦ/16, SASYNCRTΦ/64, SASYNCRTΦ/256, and SASYNCRTΦ/1024).
- Selection of six counter input clocks: Channels 3 to 5.
  External clock (TCLK1), and five peripheral clocks (SASYNCPERD2Φ/4, SASYNCPERD2Φ/16, SASYNCPERD2Φ/64, SASYNCPERD2Φ/256, and SASYNCPERD2Φ/1024).
- Selection of six counter input clocks: Channels 6 to 8.
- External clock (TCLK2), and five peripheral clocks (SASYNCPERD2Φ/4, SASYNCPERD2Φ/16, SASYNCPERD2Φ/64, SASYNCPERD2Φ/256, and SASYNCPERD2Φ/1024).
- Selection of six counter input clocks: Channels 9 to 11.
  External clock (TCLK3), and five peripheral clocks (SASYNCPERD2Φ/4, SASYNCPERD2Φ/16, SASYNCPERD2Φ/64, SASYNCPERD2Φ/256, and SASYNCPERD2Φ/1024).
- Selection of six counter input clocks: Channels 12 to 14.
  External clock (TCLK4), and five peripheral clocks (SASYNCPERD2Φ/4, SASYNCPERD2Φ/16, SASYNCPERD2Φ/64, SASYNCPERD2Φ/256, and SASYNCPERD2Φ/1024).
- Two interrupt sources.
  One underflow source (each channel) and one input capture source (channels 5, 8, 11 and 14).

### 155.4.1.2 Block Diagram

Figure 155.4.1.1 shows block diagrams of TMU.




[Legend]
TSTRn: Timer start register n
TCORnm: Timer constant register n m
TCNTnm: Timer counter n m
TCRnm: Timer control register n m
TCPRnm: Input capture register n m (channels 5, 8, 11 and 14 only)

**Figure 155.4.1.1 Block Diagram of TMU**

### 155.4.1.3 External Pins

Table 155.4.1.1 shows the pin configuration of the TMU.

Table 155.4.1.1 Pin Configuration

| Name | Abbreviation | I/O | Function | R-Car V4M |
|---|---|---|---|---|
| Clock input 1 | TCLK1 | Input | External clock input pin for channels 3 to 5.<br>Input capture control input pin for channel 5. | √ |
| Clock input 2 | TCLK2 | Input | External clock input pin for channels 6 to 8.<br>Input capture control input pin for channel 8. | √ |
| Clock input 3 | TCLK3 | Input | External clock input pin for channels 9 to 11.<br>Input capture control input pin for channel 11. | √ |
| Clock input 4 | TCLK4 | Input | External clock input pin for channels 12 to 14.<br>Input capture control input pin for channel 14. | √ |

#### 155.4.1.4 Register Configuration

The TMU has the following registers. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined.

Table 155.4.1.2 Register Configurations

| Channel | Name | Abbreviation | R/W | Address | Size | Power-on Reset/ Software Reset | Module Standby |
|---|---|---|---|---|---|---|---|
| Common to 0 to 2 | Timer start register 0 | TSTR0 | R/W | H'E61E 0004 | 8 | H'00 | Retained |
| 0 | Timer constant register 0 0 | TCOR00 | R/W | H'E61E 0008 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 0 0 | TCNT00 | R/W | H'E61E 000C | 32 | H'FFFF FFFF | Retained |
| | Timer control register 0 0 | TCR00 | R/W | H'E61E 0010 | 16 | H'0000 | Retained |
| 1 | Timer constant register 0 1 | TCOR01 | R/W | H'E61E 0014 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 0 1 | TCNT01 | R/W | H'E61E 0018 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 0 1 | TCR01 | R/W | H'E61E 001C | 16 | H'0000 | Retained |
| 2 | Timer constant register 0 2 | TCOR02 | R/W | H'E61E 0020 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 0 2 | TCNT02 | R/W | H'E61E 0024 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 0 2 | TCR02 | R/W | H'E61E 0028 | 16 | H'0000 | Retained |
| Common to 3 to 5 | Timer start register 1 | TSTR1 | R/W | H'E6FC 0004 | 8 | H'00 | Retained |
| 3 | Timer constant register 1 0 | TCOR10 | R/W | H'E6FC 0008 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 1 0 | TCNT10 | R/W | H'E6FC 000C | 32 | H'FFFF FFFF | Retained |
| | Timer control register 1 0 | TCR10 | R/W | H'E6FC 0010 | 16 | H'0000 | Retained |
| 4 | Timer constant register 1 1 | TCOR11 | R/W | H'E6FC 0014 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 1 1 | TCNT11 | R/W | H'E6FC 0018 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 1 1 | TCR11 | R/W | H'E6FC 001C | 16 | H'0000 | Retained |
| 5 | Timer constant register 1 2 | TCOR12 | R/W | H'E6FC 0020 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 1 2 | TCNT12 | R/W | H'E6FC 0024 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 1 2 | TCR12 | R/W | H'E6FC 0028 | 16 | H'0000 | Retained |
| | Input capture register 1 2 | TCPR12 | R | H'E6FC 002C | 32 | Undefined | Retained |

| Channel | Name | Abbreviation | R/W | Address | Size | Power-on Reset/ Software Reset | Module Standby |
|---|---|---|---|---|---|---|---|
| Common to 6 to 8 | Timer start register 2 | TSTR2 | R/W | H'E6FD 0004 | 8 | H'00 | Retained |
| 6 | Timer constant register 2 0 | TCOR20 | R/W | H'E6FD 0008 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 2 0 | TCNT20 | R/W | H'E6FD 000C | 32 | H'FFFF FFFF | Retained |
| | Timer control register 2 0 | TCR20 | R/W | H'E6FD 0010 | 16 | H'0000 | Retained |
| 7 | Timer constant register 2 1 | TCOR21 | R/W | H'E6FD 0014 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 2 1 | TCNT21 | R/W | H'E6FD 0018 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 2 1 | TCR21 | R/W | H'E6FD 001C | 16 | H'0000 | Retained |
| 8 | Timer constant register 2 2 | TCOR22 | R/W | H'E6FD 0020 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 2 2 | TCNT22 | R/W | H'E6FD 0024 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 2 2 | TCR22 | R/W | H'E6FD 0028 | 16 | H'0000 | Retained |
| | Input capture register 2 2 | TCPR22 | R | H'E6FD 002C | 32 | Undefined | Retained |
| Common to 9 to 11 | Timer start register 3 | TSTR3 | R/W | H'E6FE 0004 | 8 | H'00 | Retained |
| 9 | Timer constant register 3 0 | TCOR30 | R/W | H'E6FE 0008 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 3 0 | TCNT30 | R/W | H'E6FE 000C | 32 | H'FFFF FFFF | Retained |
| | Timer control register 3 0 | TCR30 | R/W | H'E6FE 0010 | 16 | H'0000 | Retained |
| 10 | Timer constant register 3 1 | TCOR31 | R/W | H'E6FE 0014 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 3 1 | TCNT31 | R/W | H'E6FE 0018 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 3 1 | TCR31 | R/W | H'E6FE 001C | 16 | H'0000 | Retained |
| 11 | Timer constant register 3 2 | TCOR32 | R/W | H'E6FE 0020 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 3 2 | TCNT32 | R/W | H'E6FE 0024 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 3 2 | TCR32 | R/W | H'E6FE 0028 | 16 | H'0000 | Retained |
| | Input capture register 3 2 | TCPR32 | R | H'E6FE 002C | 32 | Undefined | Retained |
| Common to 12 to 14 | Timer start register 4 | TSTR4 | R/W | H'FFC0 0004 | 8 | H'00 | Retained |
| 12 | Timer constant register 4 0 | TCOR40 | R/W | H'FFC0 0008 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 4 0 | TCNT40 | R/W | H'FFC0 000C | 32 | H'FFFF FFFF | Retained |
| | Timer control register 4 0 | TCR40 | R/W | H'FFC0 0010 | 16 | H'0000 | Retained |
| 13 | Timer constant register 4 1 | TCOR41 | R/W | H'FFC0 0014 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 4 1 | TCNT41 | R/W | H'FFC0 0018 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 4 1 | TCR41 | R/W | H'FFC0 001C | 16 | H'0000 | Retained |
| 14 | Timer constant register 4 2 | TCOR42 | R/W | H'FFC0 0020 | 32 | H'FFFF FFFF | Retained |
| | Timer counter 4 2 | TCNT42 | R/W | H'FFC0 0024 | 32 | H'FFFF FFFF | Retained |
| | Timer control register 4 2 | TCR42 | R/W | H'FFC0 0028 | 16 | H'0000 | Retained |
| | Input capture register 4 2 | TCPR42 | R | H'FFC0 002C | 32 | Undefined | Retained |

### 155.4.2 Register Description

Registers in the TMU are allocated to and arranged in the address space of the internal bus.

Legend for Register Description

Initial value: Register value after a reset

—: Undefined value

R/W: Readable/writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

R: Read-only. When the bit is reserved, the write value should always be 0. Writing 1 to these bits can cause a malfunction of TMU.

—/W: Write-only. The read value is undefined.

#### 155.4.2.1 Timer Start Registers (TSTRn) (n = 0 to 4)

TSTRn are 8-bit readable/writable registers that select whether to run or halt the TCNTnm.

- (TSTR0)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | STR02 | STR01 | STR00 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | STR02 | 0 | R/W | Counter Start 02<br>Selects whether to run or halt TCNT02.<br>0: TCNT02 count halted<br>1: TCNT02 counts |
| 1 | STR01 | 0 | R/W | Counter Start 01<br>Selects whether to run or halt TCNT01.<br>0: TCNT01 count halted<br>1: TCNT01 counts |
| 0 | STR00 | 0 | R/W | Counter Start 00<br>Selects whether to run or halt TCNT00.<br>0: TCNT00 count halted<br>1: TCNT00 counts |

- (TSTR1)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | STR12 | STR11 | STR10 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | STR12 | 0 | R/W | Counter Start 12<br>Selects whether to run or halt TCNT12.<br>0: TCNT12 count halted<br>1: TCNT12 counts |
| 1 | STR11 | 0 | R/W | Counter Start 11<br>Selects whether to run or halt TCNT11.<br>0: TCNT11 count halted<br>1: TCNT11 counts |
| 0 | STR10 | 0 | R/W | Counter Start 10<br>Selects whether to run or halt TCNT10.<br>0: TCNT10 count halted<br>1: TCNT10 counts |

- (TSTR2)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | STR22 | STR21 | STR20 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | STR22 | 0 | R/W | Counter Start 22<br>Selects whether to run or halt TCNT22.<br>0: TCNT22 count halted<br>1: TCNT22 counts |
| 1 | STR21 | 0 | R/W | Counter Start 21<br>Selects whether to run or halt TCNT21.<br>0: TCNT21 count halted<br>1: TCNT21 counts |
| 0 | STR20 | 0 | R/W | Counter Start 20<br>Selects whether to run or halt TCNT20.<br>0: TCNT20 count halted<br>1: TCNT20 counts |

- (TSTR3)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | STR32 | STR31 | STR30 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | STR32 | 0 | R/W | Counter Start 32<br>Selects whether to run or halt TCNT32.<br>0: TCNT32 count halted<br>1: TCNT32 counts |
| 1 | STR31 | 0 | R/W | Counter Start 31<br>Selects whether to run or halt TCNT31.<br>0: TCNT31 count halted<br>1: TCNT31 counts |
| 0 | STR30 | 0 | R/W | Counter Start 30<br>Selects whether to run or halt TCNT30.<br>0: TCNT30 count halted<br>1: TCNT30 counts |

- (TSTR4)

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | STR42 | STR41 | STR40 |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 2 | STR42 | 0 | R/W | Counter Start 42<br>Selects whether to run or halt TCNT42.<br>0: TCNT42 count halted<br>1: TCNT42 counts |
| 1 | STR41 | 0 | R/W | Counter Start 41<br>Selects whether to run or halt TCNT41.<br>0: TCNT41 count halted<br>1: TCNT41 counts |
| 0 | STR40 | 0 | R/W | Counter Start 40<br>Selects whether to run or halt TCNT40.<br>0: TCNT40 count halted<br>1: TCNT40 counts |

### 155.4.2.2 Timer Constant Register n m (TCORnm) (n = 0 to 4) (m = 0 to 2)

TCORnm are 32-bit readable/writable registers. After underflow has been generated according to the result of the TCNTnm countdown, the value of TCORnm is set to TCNTnm and TCNTnm continues countdown from the value.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCORnm(n = 0 to 4)(m = 0 to 2)[31:16] | | | | | | | | | | | | | | | |
| Initial | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCORnm(n = 0 to 4)(m = 0 to 2)[15:0] | | | | | | | | | | | | | | | |
| Initial | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

### 155.4.2.3 Timer Counter n m (TCNTnm) (n = 0 to 4) (m = 0 to 2)

TCNTnm are 32-bit readable/writable registers that count down upon the input clock selected using the bits TPSC2 to TPSC0 in TCRnm.

When a TCNTnm countdown results in an underflow, the UNF in TCRnm of corresponding channel is set. At the same time, the value of TCORnm is set to TCNTnm and TCNTnm continues countdown from that value.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCNTnm(n = 0 to 4)(m = 0 to 2)[31:16] | | | | | | | | | | | | | | | |
| Initial | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCNTnm(n = 0 to 4)(m = 0 to 2)[15:0] | | | | | | | | | | | | | | | |
| Initial | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

### 155.4.2.4 Timer Control Register n m (TCRnm) (n = 0 to 4) (m = 0 to 2)

TCRnm are 16-bit readable/writable registers that select a count clock and edge when an external clock is selected, and control an interrupt generation when the flag that indicates the generation of a TCNTnm is set to 1.

TCRn2 of channels 5, 8, 11 and 14 control the input capture function and generation of an interrupt during the input capture.

- Bit assgiment for non input capture function
  TCR00, TCR01, TCR02, TCR10, TCR11, TCR20, TCR21, TCR30, TCR31 TCR40, TCR41

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | UNF | — | — | UNIE | CKEG | | TPSC | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

- Bit assgiment for input capture function
  TCR12, TCR22, TCR32, TCR42

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | ICPF | UNF | ICPE | | UNIE | CKEG | | TPSC | | |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | ICPF*1 | 0 | R/W*3 | Input Capture Interrupt Flag<br>Status flag, provided in channels 5 and 8, 11, 14 only, which indicates the occurrence of input capture.<br>0: No input capture has occurred<br>[Clearing condition]<br>When 0 is written to ICPF<br>1: Input capture has occurred<br>[Setting condition]<br>When an input capture occurs*2 |
| 8 | UNF | 0 | R/W | Underflow Flag<br>Status flag which indicates the occurrence of a TCNTnm underflow.<br>0: TCNTnm has not underflowed<br>[Clearing condition]<br>When 0 is written to UNF<br>1: TCNTnm has underflowed<br>[Setting condition]<br>When TCNTnm underflows*2 |
| 7, 6 | ICPE*1 | 00 | R/W*3 | Input Capture Control<br>A function of channels 5, 8, 11, 14: determines whether the input capture function can be used, and when used, whether or not to enable interrupts.<br>Use the CKEG bits to designate use of either the rising or falling edge of the TCLK pin to set the values of TCNT12, TCNT22, TCNT32 and TCNT42 to TCPR12, TCPR22, TCPR32 and TCPR42 respectively.<br>Only when the ICPF bits in TCR12, TCR22, TCR32 and TCR42 are 0, the values of TCNT12, TCNT22, TCNT32 and TCNT42 are set to TCPR12, TCPR22, TCPR32 and TCPR42. When the ICPF bit is set to 1, neither TCPR12, TCPR22, TCPR32 and TCPR42 is set even when input capture is generated.<br>00: Input capture function is not used.<br>01: Reserved (setting prohibited).<br>10: Input capture function is used. Interrupt due to input capture (TICPI5, TICPI8, TICPI11 and TICPI14) is not enabled.<br>11: Input capture function is used. Interrupt due to input capture (TICPI5, TICPI8, TICPI11 and TICPI14) is enabled. |
| 5 | UNIE | 0 | R/W | Underflow Interrupt Control<br>Controls enabling of interrupt generation when the status flag (UNF) indicating TCNTnm underflow has been set to 1.<br>0: Interrupt due to underflow (TUNI) is not enabled<br>1: Interrupt due to underflow (TUNI) is enabled |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4, 3 | CKEG | 00 | R/W | Clock Edge<br>Select an input edge of the external clock when the external clock is selected, or when the input capture function is used.<br>00: Count/capture register set on rising edge<br>01: Count/capture register set on falling edge<br>Other: Count/capture register set on both rising and falling edge |
| 2 to 0 | TPSC | 000 | R/W | Timer Pre-scaler 2 to 0<br>Select the TCNTnm count clock.<br>000: Count on (input-clock)/4<br>001: Count on (input-clock)/16<br>010: Count on (input-clock)/64<br>011: Count on (input-clock)/256<br>100: Count on (input-clock)/1024<br>101: Setting prohibited<br>110: Setting prohibited<br>111: Count on external clock<br>Channels 3 to 5 TCLK1, channels 6 to 8 TCLK2,<br>channels 9 to 11 TCLK3, channels 12 to 14 TCLK4<br>Not usable in channels 0, 1, 2.<br>Note: Input-clock is SASYNCRTΦ for channel 0/1/2,<br>SASYNCPERD2Φ for channel 3/4/5/6/7/8/9/10/11/12/13/14. |

Notes: 1. This bit is reserved in channels 0, 1, 2, 3, 4, 6, 7, 9, 10, 12 and 13 (initial value is 0 and read-only).
2. Writing 1 does not change the value.
3.These bits of channels 5, 8, 11 and 14 are R/W, other channels are R.

### 155.4.2.5 Input Capture Register n m (TCPRnm) (n = 1 to 4) (m = 2)

TCPR12, TCPR22, TCPR32 and TCPR42 are read-only 32-bit registers used for the input capture function provided only in channels 5, 8, 11 and 14. The ICPE and CKEG bits in TCR12, TC22, TCR32 and TCR42 control the input capture function. When an input capture occurs, the value of TCNT12 is copied to TCPR12, the value of TCNT22 is copied to TCPR22, the value of TCNT32 is copied to TCPR32, the value of TCNT42 is copied to TCPR42. The values of TCNT12, TCNT22, TCNT32 and TCNT42 are set in TCPR12, TCPR22, TCPR32 and TCPR42, respectively, only when the ICPF bits in TCR12, TCR22, TCR32 and TCR42 are 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCPRnm[31:16] | | | | | | | | | | | | | | | |
| Initial | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TCPRnm[15:0] | | | | | | | | | | | | | | | |
| Initial | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

## 155.4.3 Operation

Each channel has a 32-bit timer counter (TCNTnm) and 32-bit timer constant register (TCORnm). TCNTnm counts down. The auto-reload function enables synchronized counting and external-event counting. Channels 5, 8, 11 and 14 have the input capture function.

### 155.4.3.1 Counter Operation

When the bits STR00 to STR42 in TSTR0 to TSTR4 are set to 1, TCNTnm of corresponding channel starts counting. When TCNTnm underflows, the UNF flag of corresponding TCRnm is set. In this case, if the UNIE bit in TCRnm is set to 1, an interrupt request is sent to the CPU. Also, the value is copied from TCORnm to TCNTnm and the down-count operation is continued (Auto reload function).

**(1) Procedure for setting count operation**

Figure 155.4.3.1 shows an example of the procedure for setting the count operation.

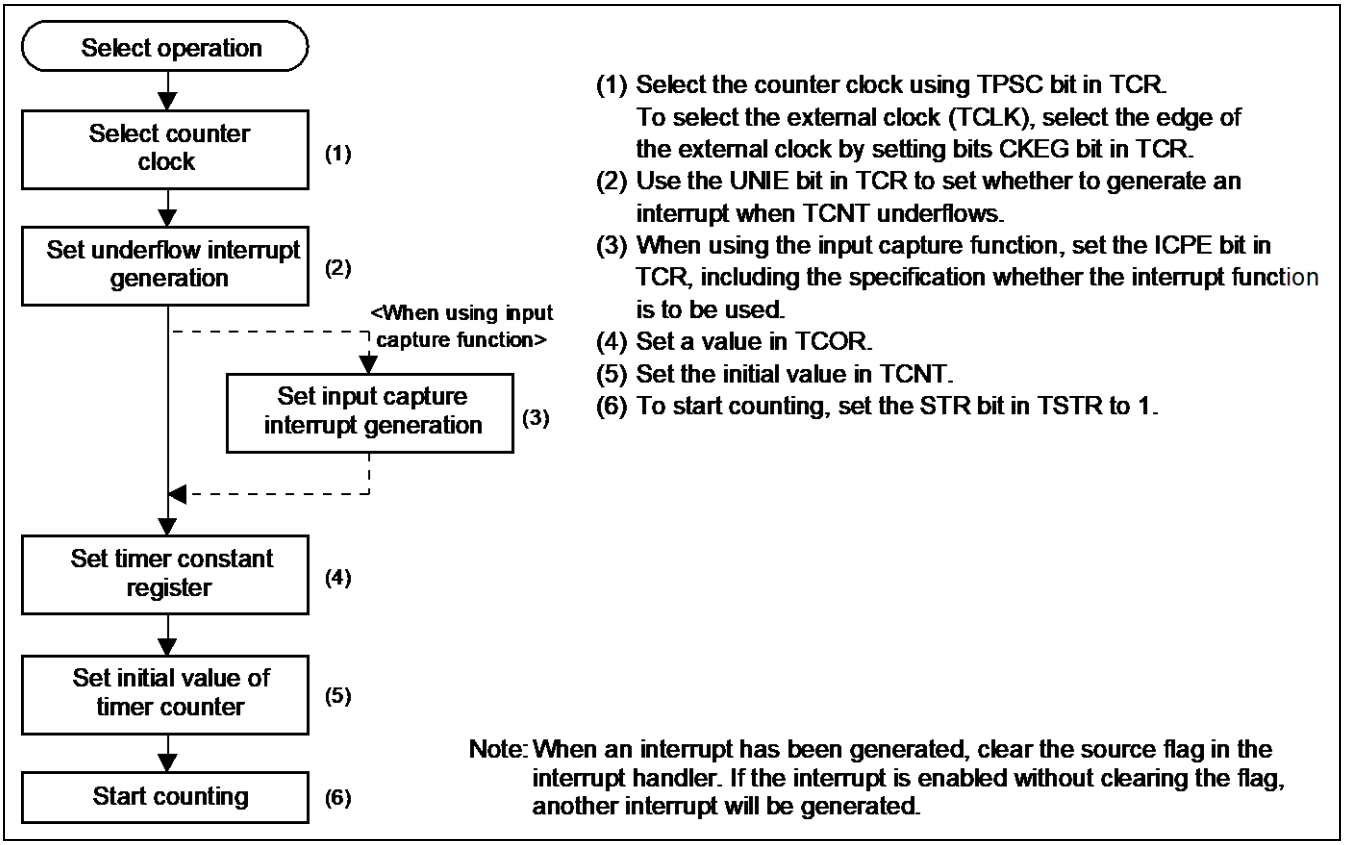



Figure 155.4.3.1 Procedure for Setting Count Operation

**(2) Auto-reload count operation**

Figure 155.4.3.2 shows the TCNTnm auto-reload operation.

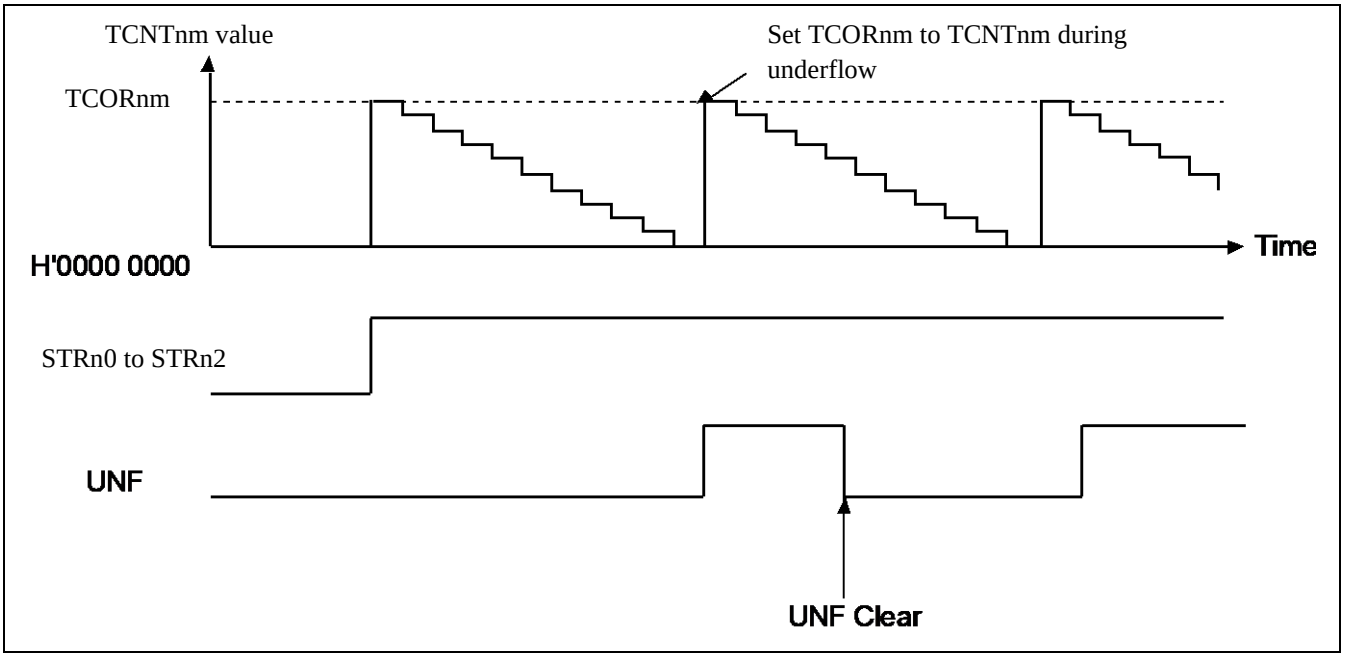



Figure 155.4.3.2 TCNTnm Auto-Reload Operation

**(3) TCNTnm count timing**

- Internal clock operation

  Five clocks (SASYNCRTΦ/4, SASYNCRTΦ/16, SASYNCRTΦ/64, SASYNCRTΦ/256, SASYNCRTΦ/1024) and five clocks (SASYNCPERD2Φ/4, SASYNCPERD2Φ/16, SASYNCPERD2Φ/64, SASYNCPERD2Φ/256, SASYNCPERD2Φ/1024) that are created by dividing peripheral clocks are selected as count clocks by setting bits TPSC2 to TPSC0 in TCRnm.

  Figure 155.4.3.3 show the timing.

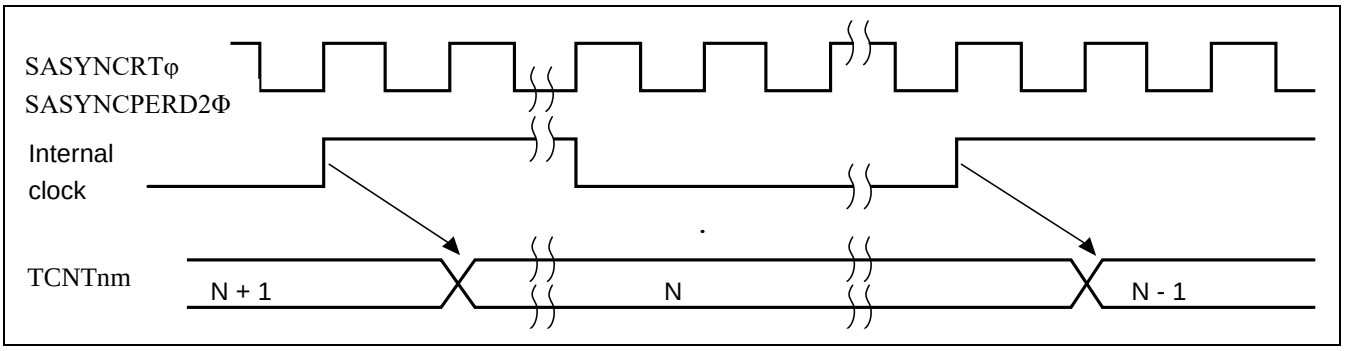



Figure 155.4.3.3 Count Timing when Internal Clock is Operating

- External clock operation
  Set the bits TPSC2 to TPSC0 in TCRnm to select the external clock pin (TCLK) as the timer clock. Use the bits CKEG1 and CKEG0 in TCRnm to select the detection edge. Rise, fall, or both can be selected.
  Figure 155.4.3.4 show the timing for both-edge detection.

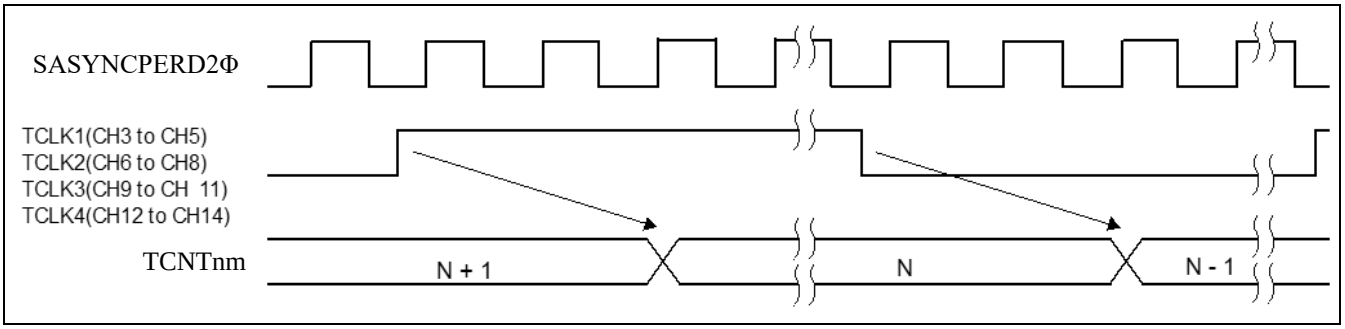



**Figure 155.4.3.4 Count Timing when External Clock is Operating**

### 155.4.3.2 Input Capture Function

Channels 5, 8, 11 and 14 have the input capture function. When using the input capture function, follow the procedure shown below.

1. Set the timer operating clock as an internal clock with bits TPSC2 to TPSC0 in TCRn2.
2. Set use of the input capture function and whether to generate an interrupt on using it with bits ICPE1 and ICPE0 in TCRn2.
3. Specify either rising edge or falling edge of the TCLK1 (channel 5), TCLK2 (channel 8) , TCLK3 (channel 11) and TCLK4 (channel 14) pin to be used to set the value of TCNTn2 to TCPR12, TCPR22, TCPR32 and TCPR42 with bits CKEG1 and CKEG0 in TCRn2.

Only when an input capture is occurred and the ICPF bits in TCR12, TCR22, TCR32 and TCR42 are 0, the values of TCNT12, TCNT22, TCNT32 and TCNT42 are set in TCPR5, TCPR8, TCPR11 and TCPR14, respectively.

Figure 155.4.3.5 shows the operating timing when the input capture function is used (the rising edge of TCLK1/2/3/4 is used).

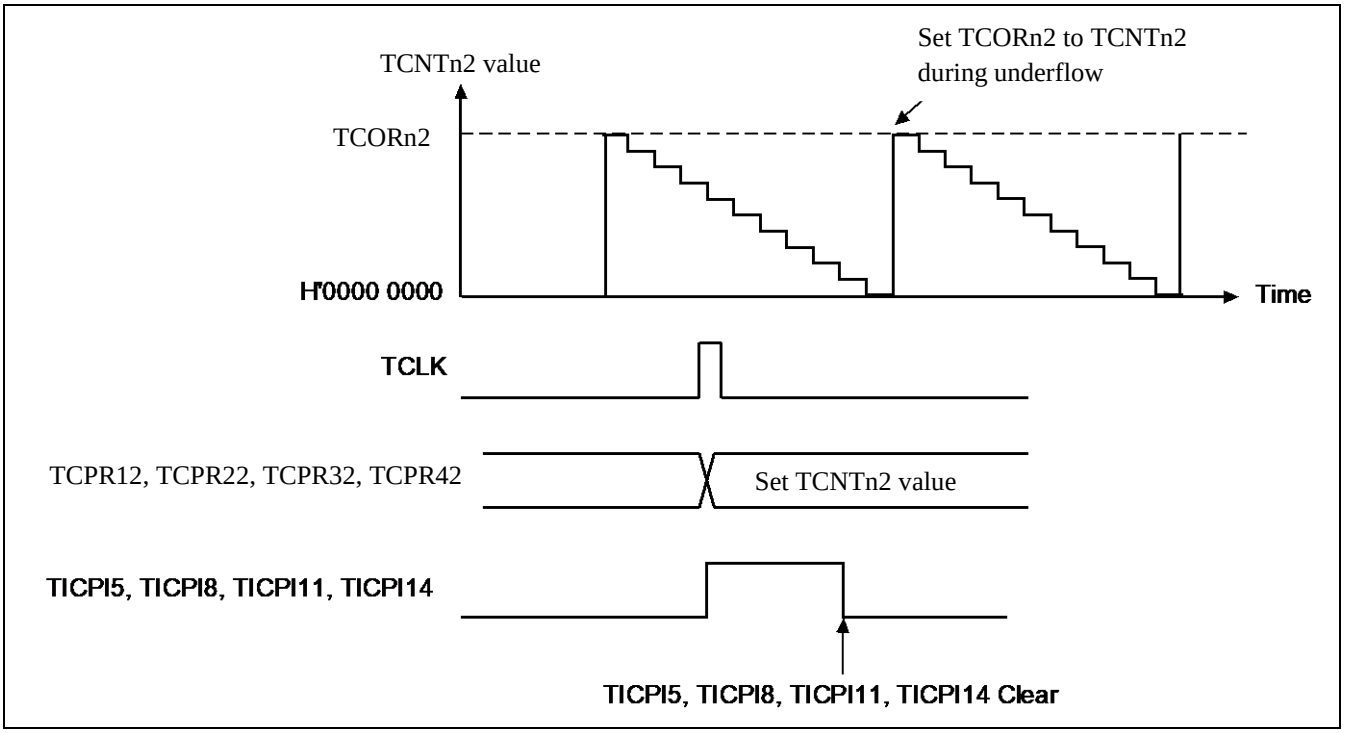



Figure 155.4.3.5 Operating Timing when Using Input Capture Function

### 155.4.3.3 Interrupt

The TMU interrupt sources are underflow interrupt or input capture interrupt when the input capture function is used. The underflow interrupt is generated at each channel. The input capture interrupt is generated at channels 5, 8, 11, 14 only.

An underflow interrupt request is generated (for each channel) when both the UNF bit and the interrupt enable bit for that channel are set to 1.

When the input capture function is used and the input capture request is generated, an interrupt request is generated if the ICPF bit in TCR12 or TCR22 or TCR32 or TCR42 is 1 and the input capture control bits ICPE1 and ICPE0 in TCR12 or TCR22 or TCR32 or TCR42 are 11.

Table 155.4.3.1 shows the TMU interrupt sources.

Table 155.4.3.1 TMU Interrupt Sources

| Channel | Interrupt Sources | Description |
|---|---|---|
| 0 | TUNI0 | Underflow interrupt 0 |
| 1 | TUNI1 | Underflow interrupt 1 |
| 2 | TUNI2 | Underflow interrupt 2 |
| 3 | TUNI3 | Underflow interrupt 3 |
| 4 | TUNI4 | Underflow interrupt 4 |
| 5 | TUNI5 | Underflow interrupt 5 |
| | TICPI5 | Input capture interrupt 5 |
| 6 | TUNI6 | Underflow interrupt 6 |
| 7 | TUNI7 | Underflow interrupt 7 |
| 8 | TUNI8 | Underflow interrupt 8 |
| | TICPI8 | Input capture interrupt 8 |

| Channel | Interrupt Sources | Description |
|---|---|---|
| 9 | TUNI9 | Underflow interrupt 9 |
| 10 | TUNI10 | Underflow interrupt 10 |
| 11 | TUNI11 | Underflow interrupt 11 |
| | TICPI11 | Input capture interrupt 11 |
| 12 | TUNI12 | Underflow interrupt 12 |
| 13 | TUNI13 | Underflow interrupt 13 |
| 14 | TUNI14 | Underflow interrupt 14 |
| | TICPI14 | Input capture interrupt 14 |

## 155.4.4 Usage Notes

### 155.4.4.1 Writing to Registers

When writing to the TMU registers, clear the start bits (STR42 to STR00) of the corresponding TSTR channel to stop the timer counting.

Writing to TSTR and clearing bits UNF and ICPF in TCRnm can be executed during counting. To clear flags UNF and ICPF during counting, do not change the values of bits other than those to be cleared.

### 155.4.4.2 Reading TCNTnm Register

Reading from TCNTnm is performed synchronously with the timer count operation. When timer counting and register read processing are performed simultaneously, the value before TCNTnm counting down is read.

### 155.4.4.3 External Clock Frequency

The frequency of external clock (TCLK1/2/3/4) for each channel should be SASYNCPERD2Φ/4 or less.

### 155.4.5 Safety Mechanisms

#### 155.4.5.1 Power-On test

This module is protected by Power-On test.

# 155.5 System Timer (SCMT)

## 155.5.1 Overview

System Timer is a 32-bit compare match timer of one channel.

### 155.5.1.1 Features

- One channel is implemented.
- 16 bits/32 bits can be selected.
- Provided with 32-bit constant registers and 32-bit up-counters that can be written or read at any time.
- Allows selection among four counter input clocks
  — OSCCLK: 1/1, 1/8, 1/32, and 1/128
- One-shot operation or free-running operation is selectable.
- Compare match or overflow can be selected as interrupt source.
- Counter operation can be enabled or disabled at the time of debugging of CPU core using the debugging mode operation selector.

### 155.5.1.2 Block Diagram

Figure 155.5.1.1 shows a block diagram of the System Timer.

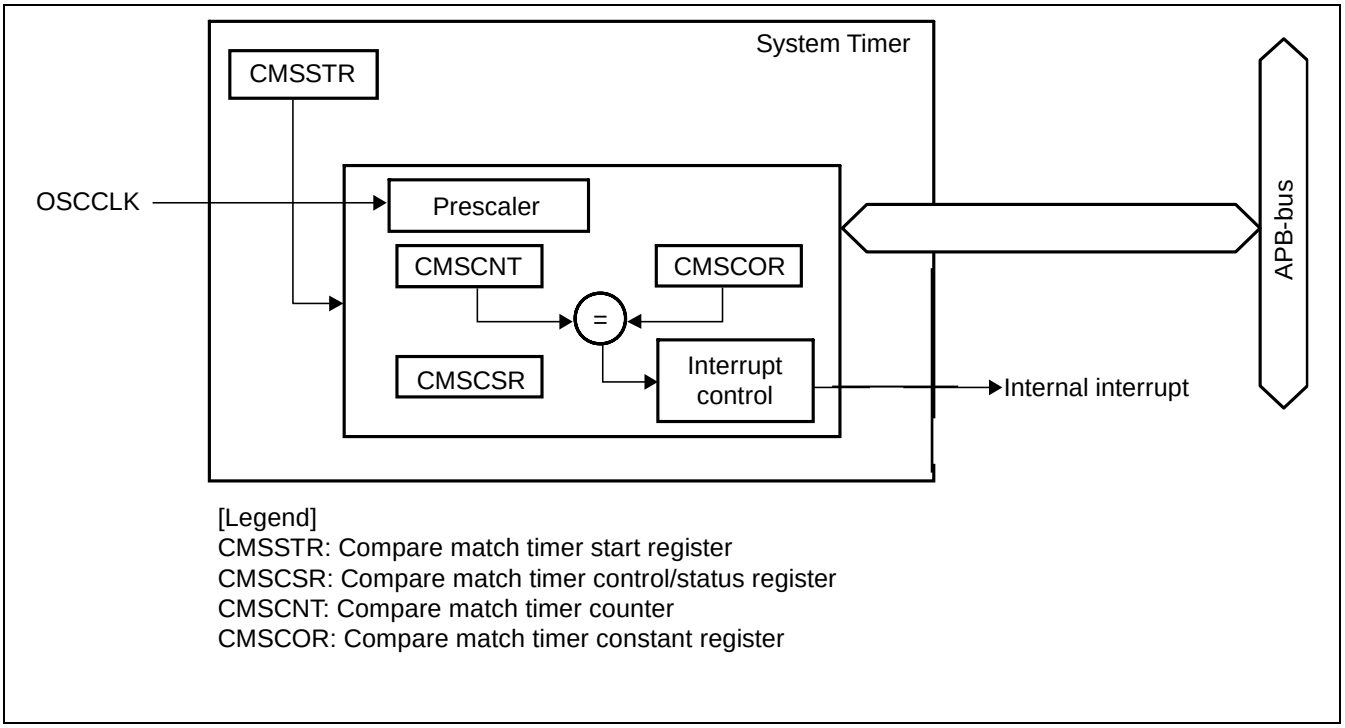



Figure 155.5.1.1 Block Diagram of System Timer

### 155.5.1.3 Register Configuration

Table 155.5.1.1 shows the System Timer register configuration. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined. Access CMSSTR and CMSCSR as a word (16 bits). Operation cannot be guaranteed if CMSSTR and CMSCSR are not accessed as a word. Access CMSCNT and CMSCOR as a longword (32 bits). Operation cannot be guaranteed if CMSCNT and CMSCOR are not accessed as a longword.

Table 155.5.1.1 Register Configurations

| Register Name | Abbreviation | R/W | Address | Access Size | Initial value after Power-On Reset |
|---|---|---|---|---|---|
| Compare match timer start register | CMSSTR | R/W | H'E604 0000 | 16 | H’0000 |
| Compare match timer control/status register | CMSCSR | R/W*1 | H'E604 0040 | 16 | H'012F |
| Compare match timer counter | CMSCNT | R/W | H'E604 0044 | 32 | H’0000 0000 |
| Compare match timer constant register | CMSCOR | R/W | H'E604 0048 | 32 | H'0144 3FCF |

Note: 1. Except for bit [15:14] that are R/WC0 register.

Table 155.5.1.2 shows the register states in each operating mode.

**Table 155.5.1.2 Register States in Each Operating Mode**

| Register Abbreviation | Power-on Reset/Software Reset *1 |
|---|---|
| CMSSTR | Initialized |
| CMSCSR | Initialized |
| CMSCNT | Initialized |
| CMSCOR | Initialized |

Note: 1. For details, refer to 'General Part Section 14. Reset (RST) and '9. Module Standby, Software Reset.

### 155.5.2 Register Description

[Legend for Register Description]

Initial value: Register value after a reset.

—: Undefined value

R/W: Bit or field is readable and writable. The written value can be read.

R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.

R: Read-only. The write value should always be initial value.

#### 155.5.2.1 Compare Match Timer Start Register (CMSSTR)

CMSSTR is a 16-bit register which specify whether the compare match timer counter (CMSCNT) is operated or halted.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | STR5 | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 6 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 | STR5 | 0 | R/W | Count Start<br>Specify halt/counting of compare match timer counter (CMSCNT).<br>0: CMSCNT Halts.<br>1: CMSCNT starts counting. |
| 4 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 155.5.2.2 Compare Match Timer Control/Status Register (CMSCSR)

CMSCSR is a 16-bit register that indicates the occurrence of compare matches, enables interrupts, and sets the counter input clocks.

Do not change bits other than the CMF and OVF bits, while the compare match timer counter (CMSCNT) is under operating.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMF | OVF | WRFLG | — | — | — | CMS | CMM | — | — | CMR[1:0] | | DBGIVD | CKS[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R/WC | R/WC | R | R | R | R | R/W | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | CMF | 0 | R/WC0 | Compare Match Flag<br>This flag indicates whether the values of the compare match timer counter (CMSCNT) and the compare match timer constant register (CMSCOR) have matched or not.<br>Software cannot write 1 to the bit. When one-shot is selected for the counter operation, counting resumes by clearing this bit.<br>0: CMSCNT and CMSCOR values have not matched.<br>[Clearing condition]<br>• Write 0 to CMF<br>1: CMSCNT and CMSCOR values have matched.<br>* Only 0 can be written, for flag clearing. |
| 14 | OVF | 0 | R/WC0 | Overflow Flag<br>This flag indicates whether the compare match timer counter (CMSCNT) has overflowed or not. Software cannot write 1 to this bit.<br>0: CMSCNT has not overflowed.<br>[Clearing condition]<br>• Write 0 to OVF<br>1: CMSCNT has overflowed.<br>* Only 0 can be written, for flag clearing. |
| 13 | WRFLG | 0 | R | Write State Flag<br>When this bit is 1, write access to CMSCNT is prohibited. If it isn't maintained, operation isn't secured. This bit indicates the period that the writing to CMSCNT is masked for synchronization after the writing to CMSCNT. Confirm that this flag is 0 when CMSCNT is written to continuously. |
| 12 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | CMS | 0 | R/W | Compare Match Timer Counter Size<br>This bit selects whether the compare match timer counter (CMSCNT) is used as a 16-bit counter or 32-bit counter.<br>This register setting becomes the valid bit-size for the compare match timer constant register (CMSCOR).<br>0: Operates as a 32-bit counter.<br>1: Operates as a 16-bit counter.<br>Note: When Write or Read in this bit, refer to 155.5.3.5 Register Access. |
| 8 | CMM | 1 | R/W | Compare Match Mode<br>Specify operation mode of the counter.<br>0: One-shot operation<br>1: Free-running operation<br>Note: When Write or Read in this bit, refer to 155.5.3.5 Register Access. |
| 7 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | — | 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5, 4 | CMR[1:0] | 10 | R/W | Compare Match Request<br>These bits enable or disable an internal interrupt request at a compare match.<br>00: Disables an internal interrupt request.<br>01: Setting prohibited<br>10: Enables an internal interrupt request.<br>11: Setting prohibited |
| 3 | DBGIVD | 1 | R/W | Debug Mode Operation Select<br>This bit sets counter operation when in debug mode.<br>0: Stops counter operation when in debug mode.<br>1: Enables counter operation even when in debug mode.<br>Note: When Write or Read in this bit, refer to 155.5.3.5 Register Access. |
| 2 to 0 | CKS[2:0] | 111 | R/W | Clock Select<br>These bits select the clock input to CMSCNT. When the count start bit (STR5) in CMSSTR is set to 1, CMSCNT begins incrementing with the clock selected by these bits.<br>000: Setting prohibited<br>001: Setting prohibited<br>010: Setting prohibited<br>011: Setting prohibited<br>100: clock/8<br>101: clock/32<br>110: clock/128<br>111: clock/1<br>Note: When Write or Read in this bit, refer to 155.5.3.5 Register Access. |

### 155.5.2.3 Compare Match Timer Counter (CMSCNT)

CMSCNT is a 32-bit register which is used as an up-counter.

To specify counter operation, set compare match timer control/status register (CMSCSR), before starting operation.

When the 16-bit counter operation is selected by the CMS bit, bits 31 to 16 of this register is invalid. When data is written in 16-bit mode, write H'0000 as upper 16-bits data.

When CMSCNT is read during the counter operation, the read value may be wrong because of an asynchronous clock between counter and bus-interface. For exact value, read this register continuously, until same values are read from this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMSCNT[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMSCNT[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMSCNT [31:0] | All 0 | R/W | Compare match timer counter bit 31 to 0<br>Note: When Write or Read in this bit, refer to 155.5.3.5 Register Access. |

### 155.5.2.4 Compare Match Timer Constant Register (CMSCOR)

CMSCOR is a 32-bit register which specify the compare match period with the compare match timer counter (CMSCNT).

The initial value of this register is H'0144 3FCF.

When the 16-bit counter operation is selected by the CMS bit in the compare match timer control/status register (CMSCSR), bits 15 to 0 of this register become valid. When data is written to this register in 16-bit mode, write H'0000 as upper 16-bits data.

An overflow is detected when CMSCNT is cleared to 0 and this register is set to H'FFFF FFFF.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMSCOR[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMSCOR[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMSCOR [31:0] | H'0144 3FCF | R/W | Compare match timer constant register bit31 to 0<br>Note: When Write or Read in this bit, refer to 155.5.3.5 Register Access. |

## 155.5.3 Operation

### 155.5.3.1 Counter Operation

System Timer starts the operation of the counter by writing 1 to the STR5 bit in CMSSTR after each register has been set. Complete all of the settings before starting the operation. Do not change the register settings other than by clearing flag bits during the compare match timer (CMSCNT) operation.

The counter operates in one of two ways.

- One-Shot Operation

  One-shot operation is selected by setting the CMM bit in CMSCSR to 0. When the value in CMSCNT matches the value in CMSCOR, the value in CMSCNT is cleared to H'0000 0000, and the CMF bit in CMSCSR is set to 1. Counting by CMSCNT stops after CMSCNT has been cleared.

  Counting by CMSCNT will be resumed after CMSCSR.CMF bit has been cleared.

  To detect an overflow interrupt, set the value in CMSCOR to H'FFFF FFFF. When the value in CMSCNT matches the value in CMSCOR, CMSCNT is cleared to H'0000 0000 and the CMF and OVF bits in CMSCSR are set to 1

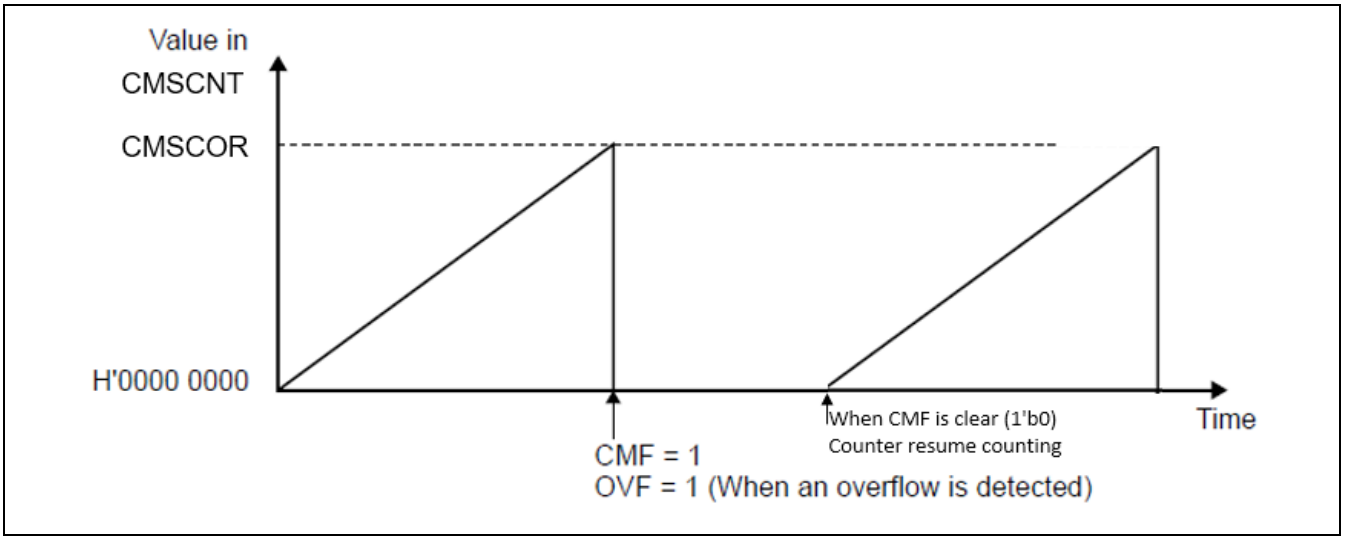



Figure 155.5.3.1 Counter Operation (One-Shot Operation)

- Free-Running Operation

  Free-running operation is selected by setting the CMM bit in CMSCSR to 1. When the value in CMSCNT matches the value in CMSCOR, CMSCNT is cleared to H'0000 0000 and the CMF bit in CMSCSR is set to 1. CMSCNT resumes counting-up after it has been cleared.

  To detect an overflow interrupt, set CMSCOR to H'FFFF FFFF. When the values in CMSCNT and CMSCOR match, CMSCNT is cleared to H'0000 0000 and the CMF and OVF bits in CMSCSR are set to 1.

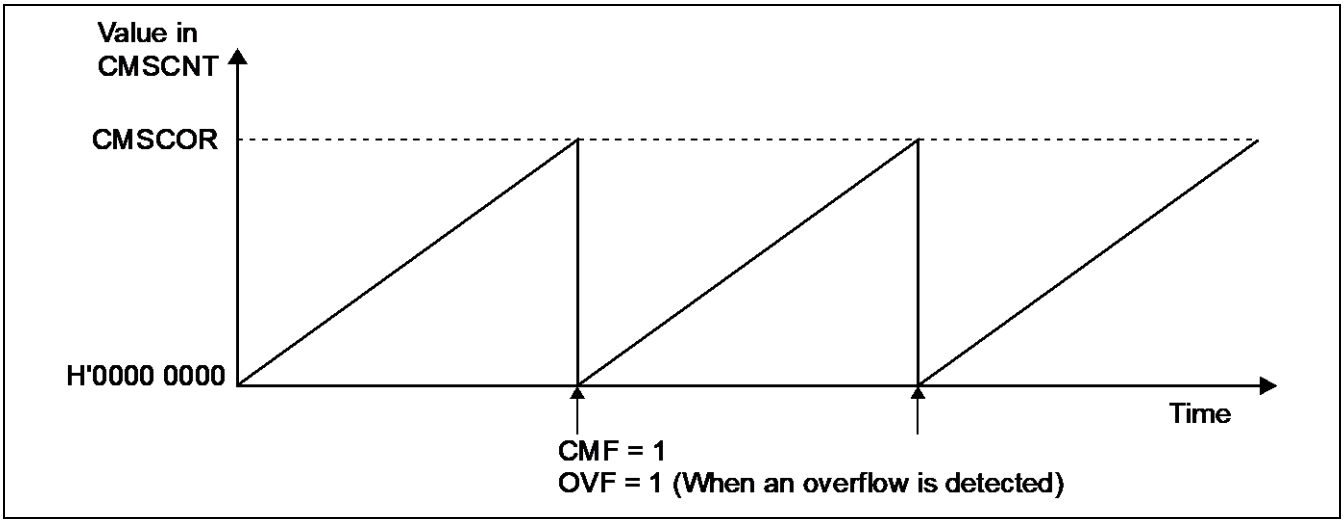



Figure 155.5.3.2 Counter Operation (Free-Running Operation)

#### 155.5.3.2 Counter Size

In this module, the size of the counter can be selected from 16 or 32 bits. This is selected by the CMS bit in CMSCSR.

When the 16-bit size is selected, use H'0000 for upper 16-bit of write data for CMSCOR. To detect an overflow interrupt, CMSCOR must be set to H'0000 FFFF.

#### 155.5.3.3 Timing for Counting by CMSCNT

In this module, the clock for the CMSCNT counter can be selected from following clocks:

- OSCCLK: 1/1, 1/8, 1/32, and 1/128

The clock for the counter is selected by the CKS bits in CMSCSR. CMSCNT is incremented at the rising edge of the selected clock.

#### 155.5.3.4 Internal Interrupt Request to CPU

The setting of the CMR bits in CMSCSR selects the sending of an internal interrupt request to the CPU at a compare match.

To clear an internal interrupt request to the CPU in Free-Running Operation, the CMF bit should be set to b'0. Set the CMF bit to b'0 in the handling routine for the System Timer interrupt.

To clear an internal interrupt request to the CPU with resuming count-up in One-Shot Operation, the CMF bit should be set to b'0 while keeping b'1 in STR5 bit in CMSSTR. Set the CMF bit to b'0 while keeping b'1 in STR5 bit in the handling routine for the System Timer interrupt.

To clear an internal interrupt request to the CPU without resuming count-up in One-Shot Operation, the CMF bit should be set to b'0 after setting STR5 bit in CMSSTR to b'0. Set the CMF bit to b'0 after setting STR5 bit to b'0 in the handling routine for the System Timer interrupt.

#### 155.5.3.5 Register Access

After writing to following registers, written data can be read 2 cycles in counter input clock (OSCCLK) after writing has finished. And it takes 2 cycles in counter input clock (OSCCLK), for reflecting written data to counter behavior.

CMSCSR: Bits CKS, CMM, CMS, CMR, DBGIVD
CMSCOR: Bits 31 to 0
CMSSTR: Bit STR5

After writing to following registers, written data can be read 2 cycles in counter input clock (OSCCLK) after writing has finished. And it takes 2 cycles in counter input clock (OSCCLK), for reflecting written data to counter behavior.

And for following registers, write access is prohibited, while previously written data is under writing. Do not perform next write access, while CMSCSR. WRFLG is 1.

CMSCNT: Bits 31 to 0

### 155.5.3.6 Compare Match Flag Set/Clear Timing

The CMF bit in CMSCSR is set to 1 by the compare match signal generated when CMSCOR and CMSCNT match. The compare match signal is generated upon the final state of the match (timing at which the CMSCNT value is updated to H'0000 0000). Consequently, after CMSCOR and CMSCNT match, a compare match signal will not be generated until a CMSCNT counter clock is input. Figure 155.5.3.3 shows the set timing of the CMF bit.

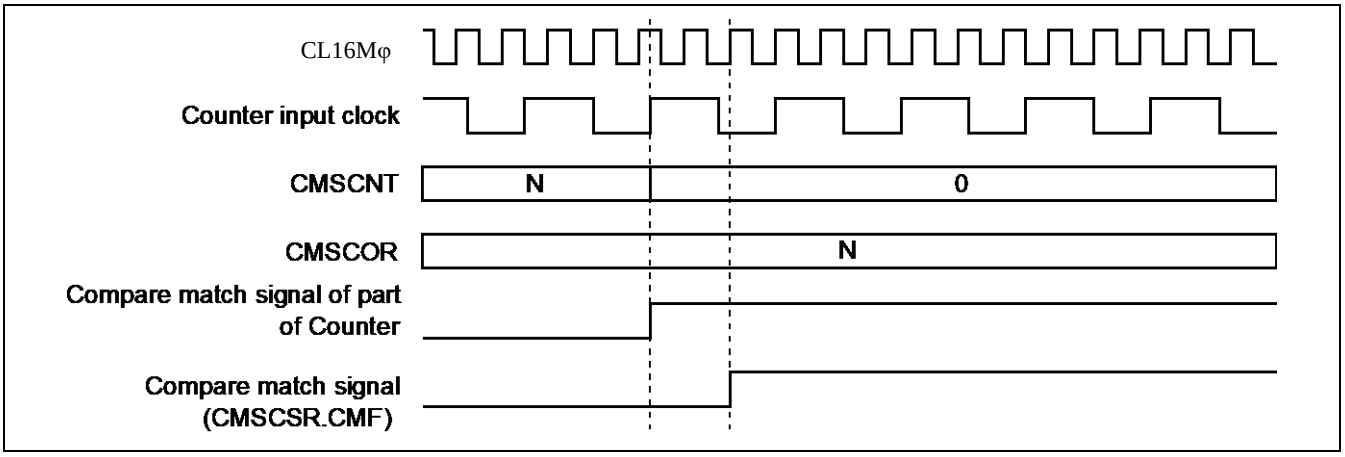



Figure 155.5.3.3 CMF Set Timing

Set the CMF bit as 0. The CMF flag is cleared immediately.

### 155.5.3.7 System Timer Usage

Take the following steps to use the System Timer.

1. Clear the STR5 bit in CMSSTR to 0 to temporarily stop counting.
   (Confirm that OSCCLK was input more than 2 cycles.)
2. Write H'0000 0000 on CMSCNT.
3. Set counter size, compare match mode, kind of count clock and interrupt request and clear the bit of OVF and CMF in CMSCSR.
4. Set the value on CMSCOR.
5. Confirm that CMSCSR. WRFLG becomes 0.
   If WRFLG is 1, wait until it'll be 0.
6. Start the counting by setting the STR5 bit in CMSSTR to 1.
   (Refer to 155.5.3.5 Register Access)

### 155.5.4 Usage Notes

No description on this section.

### 155.5.5 Safety Mechanisms

#### 155.5.5.1 Power-On test

This module is protected by Power-On test.

# 155.6 System Up-Time Clock (SUCMT)

## 155.6.1 Overview

System Up-Time Clock is an internal 32-bit compare match timer of one channel.

### 155.6.1.1 Features

- One channel is installed.
- 16 bits/32 bits can be selected as counter size (bit-width).
- Provided with 32-bit constant registers and 32-bit up-counters that can be written or read at any time.
- Following four clocks can be selected as counter clocks.
  — RCLK: 1/1, 1/8, 1/32, and 1/128
- One-shot operation or free-running operation is selectable.
- Compare match or overflow can be selected as interrupt source.
- Module standby mode can be set.
- Counter operation can be enabled or disabled at the time of debugging of CPU core using the debugging mode operation selector.

### 155.6.1.2 Block Diagram

Figure 155.6.1.1 shows a block diagram of the System Up-Time Clock.

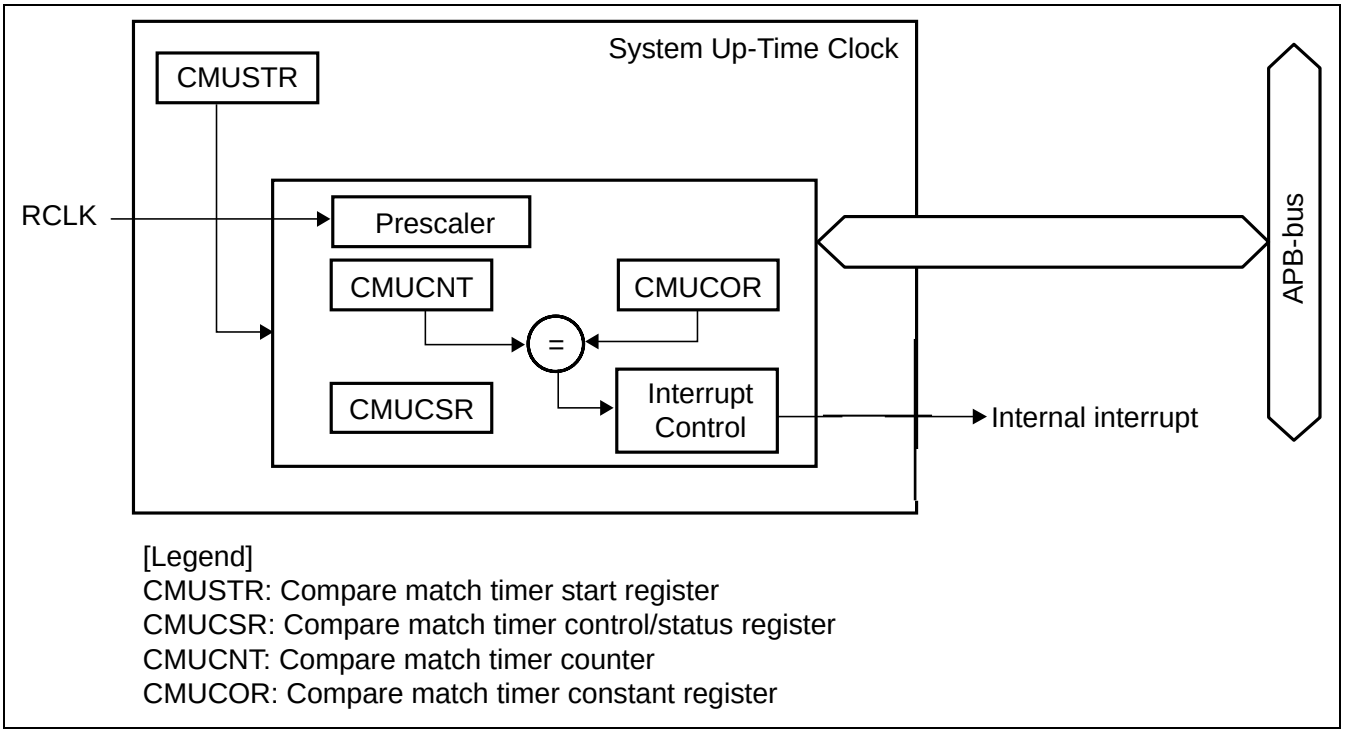



Figure 155.6.1.1 Block Diagram of System Up-Time Clock

### 155.6.1.3 Register Configuration

Table 155.6.1.1 shows the System Up-Time Clock register configuration. Do not write to any addresses other than listed below. Operations cannot be guaranteed if writing is attempted. Values read from addresses other than those listed below are undefined. Access CMUSTR and CMUCSR as a word (16 bits). Operation cannot be guaranteed if CMUSTR and CMUCSR are not accessed as a word. Access CMUCNT and CMUCOR as a longword (32 bits). Operation cannot be guaranteed if CMUCNT and CMUCOR are not accessed as a longword.

Table 155.6.1.1 Register Configurations

| Register Name | Abbreviation | R/W | Address | Access Size | Initial value after Power-On Reset |
|---|---|---|---|---|---|
| Compare match timer start register | CMUSTR | R/W | H'E61D 0000 | 16 | H’0000 |
| Compare match timer control/status register | CMUCSR | R/W*1 | H'E61D 0040 | 16 | H'010F |
| Compare match timer counter | CMUCNT | R/W | H'E61D 0044 | 32 | H’0000 0000 |
| Compare match timer constant register | CMUCOR | R/W | H'E61D 0048 | 32 | H'FFFF FFFF |

Note: 1. Except for bit [15:14] that are R/WC0 register.

Table 155.6.1.2 shows the register states in each operating mode.

**Table 155.6.1.2 Register States in Each Operating Mode**

| Register Abbreviation | Power-on Reset/Software Reset *1 | Module Standby |
|---|---|---|
| CMUSTR | Initialized | Retained |
| CMUCSR | Initialized | Retained |
| CMUCNT | Initialized | Retained |
| CMUCOR | Initialized | Retained |

Note: 1. For details, refer to ‘General Part Section 14. Reset (RST) and ‘9. Module Standby, Software Reset

### 155.6.2 Register Description

[Legend for Register Description]

Initial value: Register value after a reset.
—: Undefined value
R/W: Bit or field is readable and writable. The written value can be read.
R/WC0: Readable/writable. Writing 0 initializes the bit. Writing 1 is ignored.
R: Read-only. The write value should always be initial value.

#### 155.6.2.1 Compare Match Timer Start Register (CMUSTR)

CMUSTR is a 16-bit register that selects whether the compare match timer counter (CMUCNT) is operated or halted.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | STR5 | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 6 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 5 | STR5 | 0 | R/W | Count Start<br>These bits specify start/halt of compare match timer counter (CMUCNT).<br>0: CMUCNT halts.<br>1: CMUCNT starts counting. |
| 4 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### 155.6.2.2 Compare Match Timer Control/Status Register (CMUCSR)

CMUCSR is a 16-bit register which indicates the occurrence of compare matches, enables interrupts, and sets the counter input clocks.

Do not change bits other than the CMF and OVF bits, while the compare match timer counter (CMUCNT) is under operation.

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMF | OVF | WRFLG | WERR | — | — | CMS | CMM | — | WER | CMR[1:0] | | DBGIVD | CKS[2:0] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| R/W: | R/WC0 | R/WC0 | R | R | R | R | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 | CMF | 0 | R/WC0 | Compare Match Flag<br>This flag indicates whether the values of the compare match timer counter (CMUCNT) and the compare match timer constant register (CMUCOR) have matched or not.<br>Software cannot write 1 to the bit. When one-shot is selected for the counter operation, counting resumes by clearing this bit.<br>0: CMUCNT and CMUCOR values have not matched.<br>[Clearing condition]<br>• Write 0 to CMF<br>1: CMUCNT and CMUCOR values have matched. |
| 14 | OVF | 0 | R/WC0 | Overflow Flag<br>This flag indicates whether the compare match timer counter (CMUCNT) has overflowed or not. Software cannot write 1 to this bit.<br>0: CMUCNT has not overflowed.<br>[Clearing condition]<br>• Write 0 to OVF<br>1: CMUCNT has overflowed. |
| 13 | WRFLG | 0 | R | Write State Flag<br>When this bit is 1, write access to CMUCNT is prohibited. If it isn't maintained, operation isn't secured. This bit indicates the period that the writing to CMUCNT is masked for synchronization after the writing to CMUCNT. Confirm that this flag is 0 when CMUCNT is written to continuously. |
| 12 | WERR | 0 | R | Write Access Error Flag<br>This flag indicates whether or not the write access from the public domain. Software cannot write 1 to this bit.<br>0: Public domain has not Write access<br>[Clearing condition]<br>• Read register<br>1: Public domain has Write access |
| 11 to 10 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 9 | CMS | 0 | R/W | Compare Match Timer Counter Size<br>This bit selects whether the compare match timer counter (CMUCNT) is used as a 16-bit counter or 32-bit counter.<br>This register setting becomes the valid bit-size for the compare match timer constant register (CMUCOR).<br>0: Operates as a 32-bit counter.<br>1: Operates as a 16-bit counter.<br>Note: When Write or Read in this bit, refer to 155.6.3.5 Register Access. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 8 | CMM | 1 | R/W | Compare Match Mode<br>This bit selects one-shot operation or free-running operation of the counter.<br>0: One-shot operation<br>1: Free-running operation<br>Note: When Write or Read in this bit, refer to 155.6.3.5 Register Access. |
| 7 | — | 0 | R | Reserved<br>This bit is always read as 0. The write value should always be 0. |
| 6 | WER | 0 | R/W | Write Access Error Request<br>This bit Write Access Error interrupt request at the public domain.<br>0: Disables an internal interrupt request.<br>1: Enables an internal interrupt request.<br>Note: When Write or Read in this bit, refer to 155.6.3.5 Register Access. |
| 5, 4 | CMR[1:0] | 00 | R/W | Compare Match Request<br>These bits enable or disable an internal interrupt request at a compare match.<br>00: Disables an internal interrupt request.<br>01: Setting prohibited<br>10: Enables an internal interrupt request.<br>11: Setting prohibited<br>Note1: When various standby modes are canceled by the System Up-Time Clock, these bits should be set to B'10.<br>Note2: When Write or Read in this bit, refer to 155.6.3.5 Register Access. |
| 3 | DBGIVD | 1 | R/W | Debug Mode Operation Select<br>This bit sets counter operation when in debug mode.<br>0: Stops counter operation when in debug mode.<br>1: Enables counter operation even when in debug mode.<br>Note: When Write or Read in this bit, refer to 155.6.3.5 Register Access. |
| 2 to 0 | CKS[2:0] | 111 | R/W | Clock Select<br>These bits select the clock input to CMUCNT. When the count start bit (STR5) in CMUSTR is set to 1, CMUCNT begins incrementing with the clock selected by these bits.<br>000: Setting prohibited<br>001: Setting prohibited<br>010: Setting prohibited<br>011: Setting prohibited<br>100: RCLK/8<br>101: RCLK/32<br>110: RCLK/128<br>111: RCLK/1<br>Note: When Write or Read in this bit, refer to 155.6.3.5 Register Access. |

Note: * Only 0 can be written, for flag clearing.

### 155.6.2.3 Compare Match Timer Counter (CMUCNT)

CMUCNT is a 32-bit register which is used as an up-counter.

To specify counter operation, set compare match timer control/status register n (CMUCSR), before starting operation of corresponding channel.

When the 16-bit counter operation is selected by the CMS bit, bits 31 to 16 of this register become invalid. When data is written to this register in 16-bit mode, write H'0000 as upper 16bits

When CMUCNT is read during counter operation, the read value may be wrong because different clock is used between counter and bus-interface. For exact value, read this register continuously, until same values are read from this register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMUCNT[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMUCNT[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMUCNT [31:0] | All 0 | R/W | Compare match timer counter bit31 to 0<br>Note: When Write or Read in this bit, refer to 155.6.3.5 Register Access. |

### 155.6.2.4 Compare Match Timer Constant Register (CMUCOR)

CMUCOR is a 32-bit register which specify the compare match period with the compare match timer counter (CMUCNT).

When the 16-bit counter operation is selected by the CMS bit in the compare match timer control/status register (CMUCSR), bits 15 to 0 of this register become valid. Write H'0000 to upper 16bits in 16-bit counter operation.

An overflow is detected when CMUCNT is cleared to 0 and this register is set to H'FFFF FFFF.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMUCOR[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMUCOR[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CMUCOR [31:0] | All 1 | R/W | Compare match timer counter bit31 to 0<br>Note: When Write or Read in this bit, refer to 155.6.3.5 Register Access. |

## 155.6.3 Operation

### 155.6.3.1 Counter Operation

System Up-Time Clock starts the operation of the counter by writing 1 to the STR5 bit in CMUSTR after each register has been set. Complete all of the settings before starting the operation. Do not change the register settings other than by clearing flag bits during the compare match timer (CMUCNT) operation.

The counter operates in one of two ways.

- One-Shot Operation

  One-shot operation is selected by setting the CMM bit in CMUCSR to H'0. When the value in CMUCNT matches the value in CMUCOR, the value in CMUCNT is cleared to H'0000 0000, and the CMF bit in CMUCSR is set to 1. Counting by CMUCNT stops after CMUCNT has been cleared.

  Counting by CMUCNT will be resumed after CMUCSR.CMF bit has been cleared.

  To detect an overflow interrupt, set the value in CMUCOR to H'FFFF FFFF. When the value in CMUCNT matches the value in CMUCOR, CMUCNT is cleared to H'0000 0000 and the CMF and OVF bits in CMUCSR are set to 1

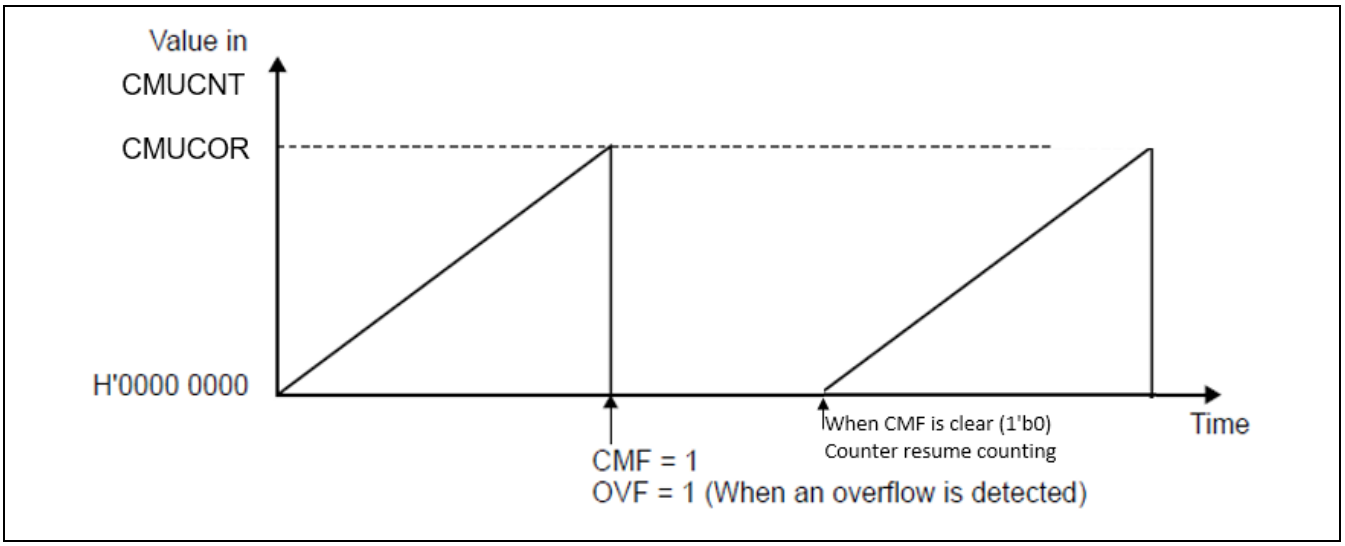



Figure 155.6.3.1 Counter Operation (One-Shot Operation)

- Free-Running Operation

  Free-running operation is selected by setting the CMM bit in CMUCSR to 1. When the value in CMUCNT matches the value in CMUCOR, CMUCNT is cleared to H'0000 0000 and the CMF bit in CMUCSR is set to 1. CMUCNT resumes counting-up after it has been cleared.

  To detect an overflow interrupt, set CMUCOR to H'FFFF FFFF. When the values in CMUCNT and CMUCOR match, CMUCNT is cleared to H'0000 0000 and the CMF and OVF bits in CMUCSR are set to 1.

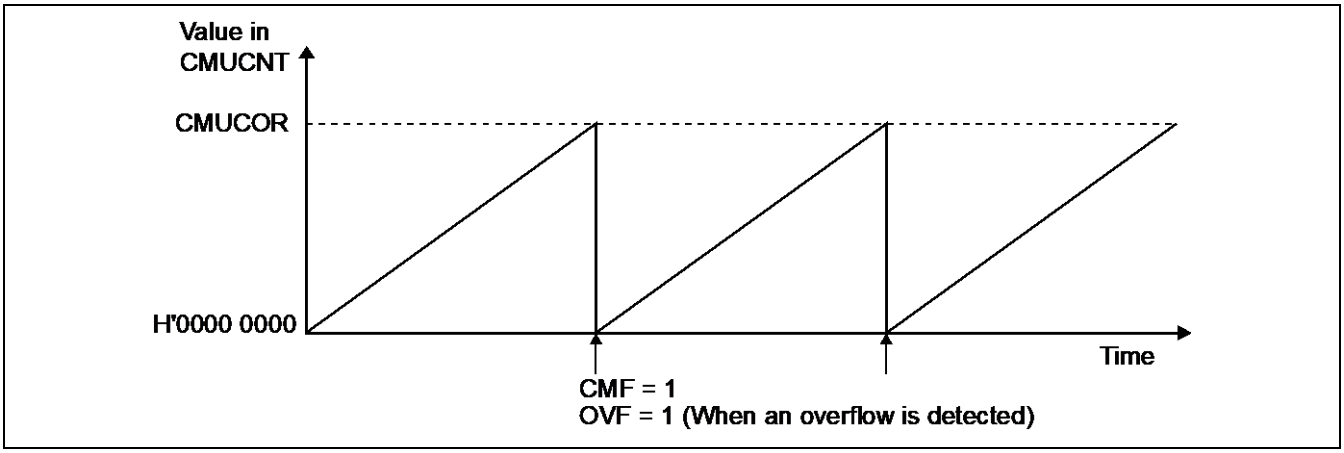



Figure 155.6.3.2 Counter Operation (Free-Running Operation)

### 155.6.3.2 Counter Size

In this module, the size of the counter can be selected from either 16, or 32 bits. This is selected by the CMS bit in CMUCSR.

When the 16-bit size is selected, H'0000 should be used as upper 16bit of write data to CMUCOR. To detect an overflow interrupt, CMUCOR must be set to H'0000 FFFF.

### 155.6.3.3 Timing for Counting by CMUCNT

In this module, the clock for the CMUCNT counter can be selected from following clocks:

- RCLK: 1/1, 1/8, 1/32, and 1/128

The clock for the counter is selected by the CKS bits in CMUCSR. CMUCNT is incremented at the rising edge of the selected clock.

### 155.6.3.4 Internal Interrupt Request to CPU

By CMR bits in CMUCSR, interrupt request to CPU can be asserted, at a compare match.

To clear an internal interrupt request to the CPU in Free-Running Operation, the CMF bit should be set to 0. Set the CMF bit to 0 in the handling routine for the System Up-Time Clock interrupt.

To clear an internal interrupt request to the CPU with resuming count-up in One-Shot Operation, the CMF bit should be set to 0 while keeping 1 in STR5 bit in CMUSTR. Set the CMF bit to 0 while keeping 1 in STR5 bit in the handling routine for the System Up-Time Clock interrupt.

To clear an internal interrupt request to the CPU without resuming count-up in One-Shot Operation, the CMF bit should be set to 0 after setting STR5 bit in CMUSTR to 0. Set the CMF bit to 0 after setting STR5 bit to 0 in the handling routine for the System Up-Time Clock interrupt.

### 155.6.3.5 Register Access

After writing to following registers, written data can be read 2 cycles in counter input clock (RCLK) after writing has finished. And it takes 2 cycles in counter input clock (RCLK), for reflecting written data to counter behavior.

CMUCSR: Bits CKS, CMM, CMS, WER, CMR, DBGIVD
CMUCOR: Bits 31 to 0
CMUSTR: Bit STR5

After writing to following registers, written data can be read 2 cycles in counter input clock (RCLK) after writing has finished. And it takes 2 cycles in counter input clock (RCLK), for reflecting written data to counter behavior.

And for following registers, write access is prohibited, while previously written data is under writing. Do not perform next write access, while CMUCSR.WRFLG is 1.

CMUCNT: Bits 31 to 0

### 155.6.3.6 Compare Match Flag Set/Clear Timing

The CMF bit in CMUCSR is set to 1 by the compare match signal generated when CMUCOR and CMUCNT match. The compare match signal is generated upon the final state of the match (timing at which the CMUCNT value is updated to H'0). Consequently, after CMUCOR and CMUCNT match, a compare match signal will not be generated until a CMUCNT counter clock is input. Figure 155.6.3.3 shows the set timing of the CMF bit.

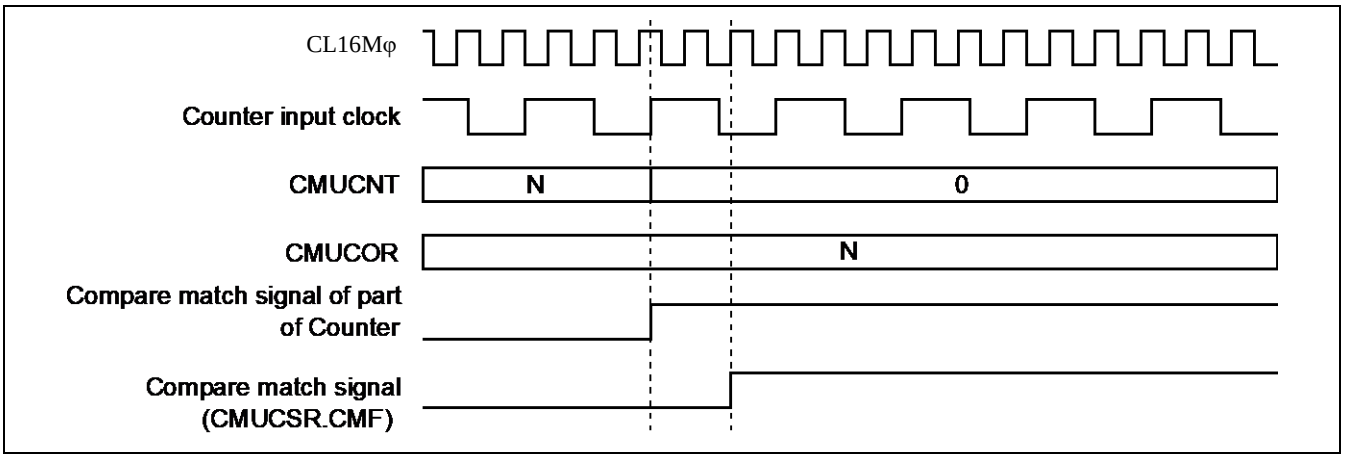



Figure 155.6.3.3 CMF Set Timing

Set the CMF bit to 0. The CMF flag is cleared immediately.

#### 155.6.3.7 System Up-Tim Clock Usage

Take the following steps to use the System Up-Time Clock.

1. Clear the STR5 bit in CMUSTR to 0 to temporarily stop counting.
   (Confirm that RCLK was input more than 2 cycles.)
2. Write H'0000 0000 on CMUCNT.
3. Set counter size, compare match mode, kind of count clock and interrupt request and clear the bit of OVF and CMF in CMUCSR.
4. Set the value on CMUCOR.
5. Confirm that CMUCSR.WRFLG becomes 0.
   If WRFLG is 1, wait until it'll be 0.
6. Start the counting by setting the STR5 bit in CMUSTR to 1.
   (Refer to 155.6.3.5 Register Access)

### 155.6.4 Usage Notes

No description on this section.

### 155.6.5 Safety Mechanisms

This module is treated as non-safety-related function in case of the products supporting functional safety. So, there is no special spec Requirement for Safety mechanisms.

# 168. Boundary Scan

## 168.1 Overview

Example Boundary Scan is a test method defined by IEEE Std 1149.1.
It is used for testing interconnections between devices mounted on printed circuits boards.
GEN4 complies with the IEEE Std 1149.1-2001.

Boundary Scan generic issues (e.g. explanation of each instruction, state transition diagram of TAP controller, etc.) are not described in this document.

## 168.2 JTAG interface

Communication with a Boundary Scan host test system is executed via the JTAG interface, as described in the following table.

**Table 168.1 JTAG interface**

| Port | Signal | Recommendation if unused |
|---|---|---|
| DCUTDI_LPDI | Data input | Open |
| DCUTDO_LPDO | Data output | Open |
| DCUTCK_LPDCLK | Clock | Open |
| DCUTMS | Mode | Open |
| DCUTRST#_LPDRST# | Reset | Open |

## 168.3 Entering Boundary Scan mode

Boundary Scan mode is entered by setting BSMODE = 1 (VDDQ18).
Note that the sequence of setting BSMODE is different depending on the value of MD25 pin. For details, refer to Figure 168.1 and Figure 168.2. After entering Boundary Scan Mode, keep BSMODE high until the power is off.

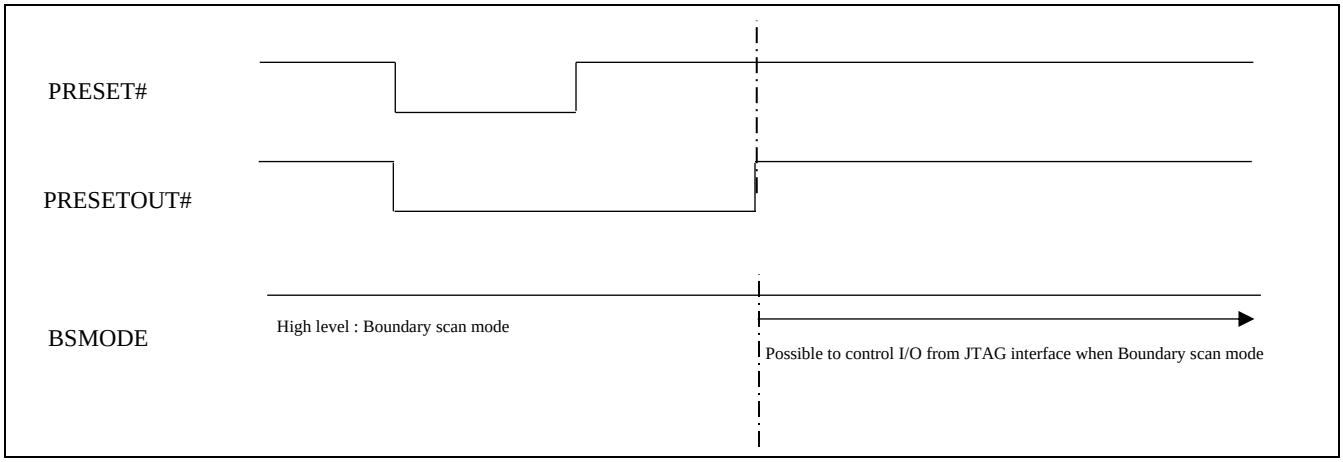



**Figure 168.1** Setting of BSMODE signal (MD25 = 0)

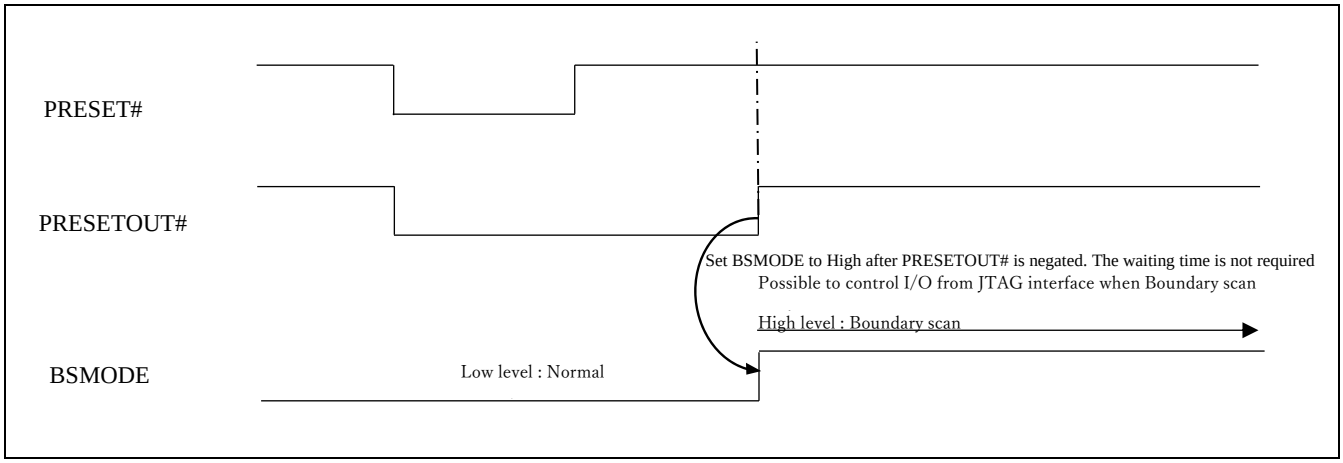



**Figure 168.2** Setting of BSMODE signal (MD25 = 1)

As the normal boot operation will be executed in case the BSMODE is selected, it is recommended to stop boot rom execution by one of the following methods:
1) Execute BSCAN mode only with an empty boot flash device.
2) Select SCIF boot mode without providing a boot code. Boot rom execution will stop at the "Please Send" stage.

## 168.4 Boundary Scan features

The table below shows the IEEE1149.1-2001 instructions and instruction codes for GEN4.

**Table 168.2 Boundary scan instruction codes**

| | Instruction code | |
|---|---|---|
| **Instruction** | **Bit [7:0]** | **Notes** |
| BYPASS | B' 1111 1111 | |
| EXTEST | B' 0000 0000 | |
| SAMPLE/PRELOAD | B' 0100 0000 | |
| IDCODE | B' 0101 0101 | |

## 168.5 Boundary Scan applicable pins

Note: The ports of the Boundary Scan applicable pins don't need to be configured by use of any port configuration register during Boundary Scan.

Boundary Scan (EXTEST) is applicable to all pins except pins indicated in the following table.

**Table 168.3 Boundary Scan inapplicable pins**

| Type | Pin Name |
|---|---|
| Reset | PRESET# |
| Test pins | TEST1, TEST2, TEST3 |
| Mode control | BSMODE |
| JTAG interface | DCUTCK_LPDCLK, DCUTDO_LPDO, DCURDY#_LPDCLKOUT, DCUTDI_LPDI, DCUTMS, DCUTRST#_LPDRST# |
| Clock inputs/outputs | XTAL, EXTAL, EXTALR |
| LPDDR4X/LPDDR5 interface | All LPDDR4X/LPDDR5 interface pins |
| PCIeGen4 interface | PCIeGen4 interface pins except for PCIE0_CLKREQ# |
| CSI2 interface | All CSI2 interface pins |
| DSI interface | All DSI interface pins |
| SYSTEM interface | SYSTEM interface pins except for PRESETOUT#, AVS1, AVS0, ERROROUT#, ERROROUTC |

## 168.6 Device ID Register (DID)

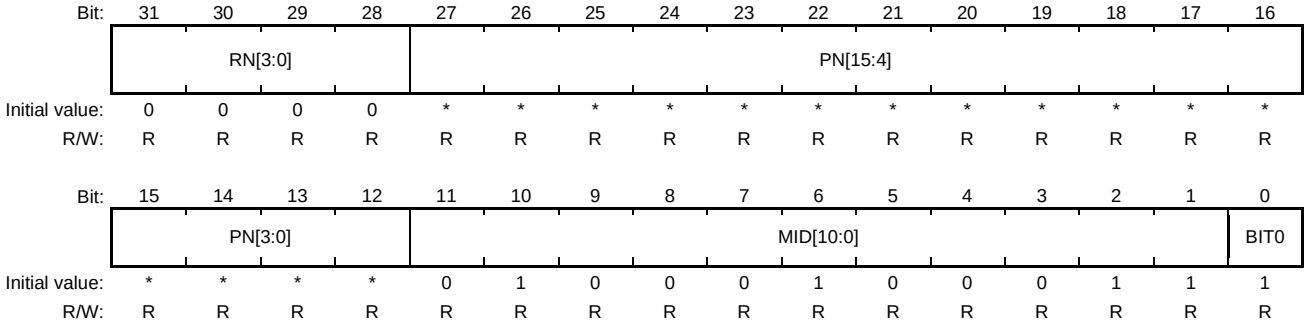
The GEN3 specific ID code is provide by the device identification register DID. The value of DID is read with IDCODE instruction and cannot be accessed by CPU.

### 168.6.1 DID – Boundary Scan ID Register

This register stores the boundary scan ID.

Access: This register cannot be read by the CPU.

Address: -




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | RN[3:0] | H'0 | R | Revision number |
| 27 to 12 | PN[15:0] | H'85D9 | R | Product number |
| 11 to 1 | MID[10:0] | H'447 | R | Manufacturer ID: H'447 = {MID[10:0] + 1} |
| 0 | BIT0 | | R | |

## 168.7 Usage Notes

1. Once a instruction is set, it is not modified until another instruction is issued. To issue the same instruction twice in a row, insert a instruction that has no effect on chip operation (such asBYPASS) between the instructions.

2. To start the system in boundary scan mode, de-assert DCUTRST#_LPDRST# while PRESET# is high.
Also be sure toset DCUTMS to high before de-asserting DCUTRST#_LPDRST# and ensure that DCUTMS remains high for 600 ns + five DCUTCK_LPDCLK clock cycles after de-asserting DCUTRST#_LPDRST#.

3. For details of boundary scan timing constraint, see Section 200, Electrical Characteristics.

4. If serial transfer is performed exceeding the number of bits of the register connected between TDI and DCUTDO_LPDO, the data that is input from TDI is output from DCUTDO_LPDO as is.

5. If the serial transfer sequence is corrupted, be sure to assert DCUTRST#_LPDRST#. In this case, transfer starts again from the beginning regardless of the point of transfer corruption.

6. Data is output via DCUTDO_LPDO at the falling edge of DCUTCK_LPDCLK.

7. To facilitate debugging, route DCUTRST#_LPDRST# on the board in such a way that patterns can be easily cut.

8. For IO pins that have POCn(POWER Condition Control Register) specifications in PFC section,
Boundary Scan only supports the supply voltage setting on the higher voltage side as indicated by POCn register initial value.

# 169. HSSTP

## 169.1 Overview

HSSTP (High Speed Serial Trace Port) is a replacement for existing parallel data output port suitable for transmitting high bandwidth data off-chip such as from Arm's Embedded Trace Macrocells. This will lower ASIC pin count.

### 169.1.1 Features

- Serial interface
  - 10Gbps*6 or 5Gbps output bandwidth through PCIE ch0

- The trace data output from following IPs:
  - Cortex-A76, Cortex-R52, System Trace Macrocell
- Supported trace format mode
  - Continuous mode*3
- Supported data width setting on TPIU*4
  - 32bit

Notes:

*3: Trace output from TPIU
When using HSSTP, please do not use either Bypass mode or Normal mode in the trace formatter of TPIU.
For more details, please refer to chapter 9 on Arm CoreSight SoC-400 r3p2 Technical Reference Manual.

*4: Please set Current Port Size Register in each TPIU.

*6: The effective throughput is 10Gbps.

### 169.1.2 Block Diagram

Figure 169.1 shows the block diagram of HSSTP for 2-lane.
The dotted line means the boundary of the clock domain, ZTRφ, ZSφ and PCI PIPEφ are the names of clock domains.

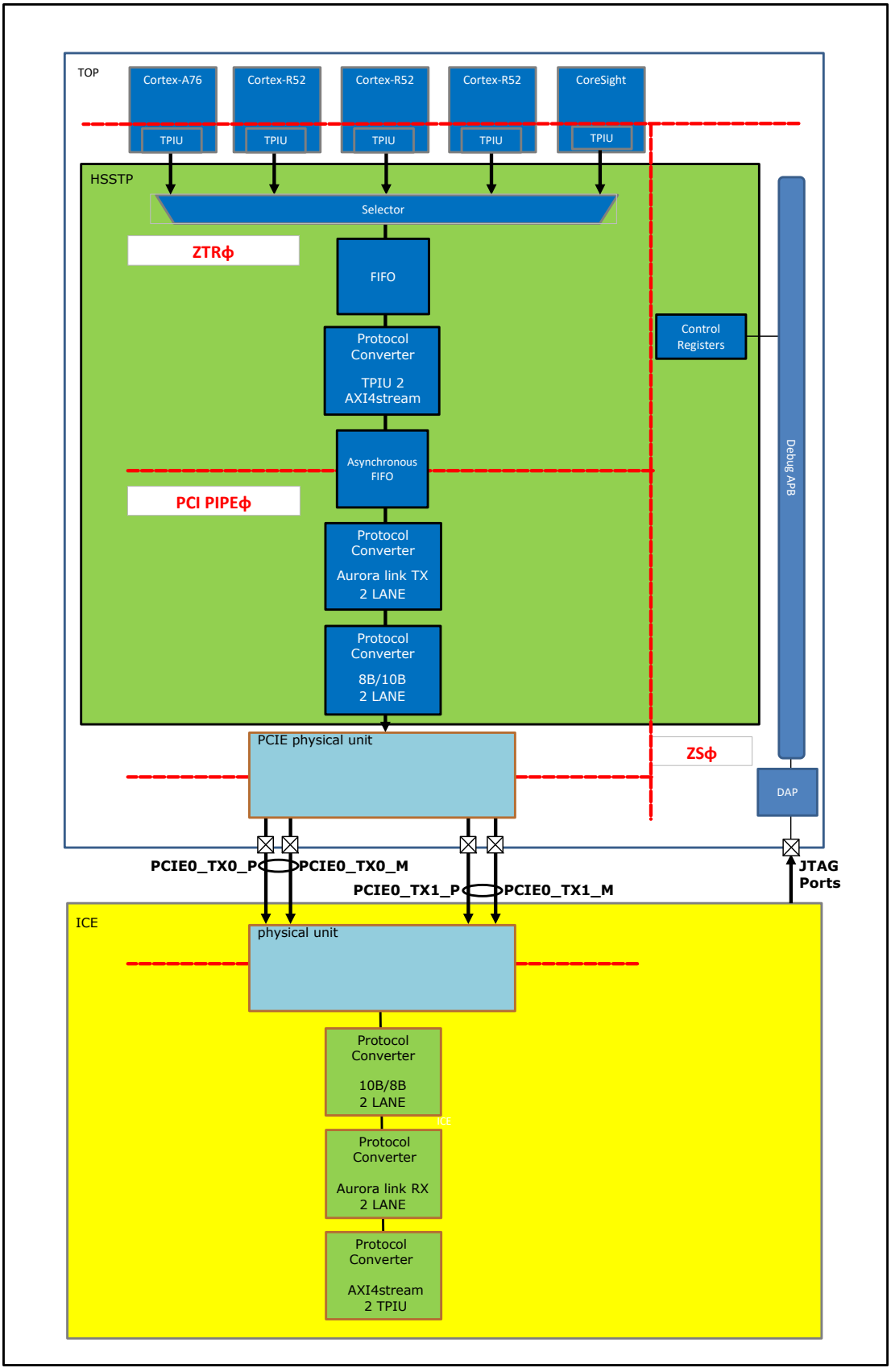



**Figure 169.1 HSSTP Block Diagram when using PCIE physical unit for 2-lane**

**[Abbreviation in block diagram]**

| | |
|---|---|
| APB: | Advanced Peripheral Bus |
| DAP: | Debug Access Port |
| FIFO: | First-In First-Out |
| HSSTP: | High Speed Serial Trace Port |
| ICE: | In-Circuit Emulator |
| JTAG: | Joint Test Action Group |
| PCIE: | Peripheral Component Interconnect Express |
| RX: | Receiver |
| TPIU: | Trace Port Interface Unit |
| TX: | Transmitter |

### 169.1.3 External Pins

#### (1) Serial interface Pins

HSSTP is implemented with PCIE physical unit. Since this product has 2 sets of 2-lane PCIE interface (PCIE0 as ch0, PCIE1 as ch1). When HSSTP uses a PCIE output, the PCIE function cannot be used for ch0. When using 2-lane PCIE interface, please use PCIE0. Please refer to chapter 104. PCIe Controller for more details.

### 169.1.4 Register Configuration

Please do not access to registers not listed below. Operations to undefined addresses are not guaranteed.

[Legend]

Initial value: Register value after a reset

-: Undefined value

R/W: Readable/writable. The written value can be read.

R: Read-only. The write value should always be the initial value.

W: Write-only. The read value is undefined.

#### (1) HSSTP Register Configuration

One Aurora links is installed for using 2-lane PCIE physical unit.

Register Base Address when using 2-lane PCIE physical unit:

| | |
|---|---|
| CPU View | H’F822 0000 |
| Debugger View | H’8022 0000 |

**Table 169.1 Register summary**

(1) Register Summary

| Register Name | Abbreviation | R/W | Offset Address | Initial Value | Size | R-Car V4M with PCIE 2-lane |
|---|---|---|---|---|---|---|
| PHY Clock Synchronization Register 2 | PCSR2 | R/W | H'0540 | H'1A06 0000 | 32 | √ |
| TX FRAME LENGTH Register | TXFLR | R/W | H'0544 | H'0000 FFFF | 32 | √ |
| Mode Control Register | MCR | R/W | H'0550 | H'0000 0004 | 32 | √ |
| PLL Lock Management Register | PLMR | R/W | H'0558 | H'0000 0000 | 32 | √ |
| Trace Source Select Register | TSSR | R/W | H'05C0 | H'0000 0001 | 32 | √ |
| Aurora PHY Control Register | APCR | R/W | H'0630 | H'0000 0000 | 32 | √ |
| Aurora PHY Status Register | APSR | R | H'0638 | H'0000 0000 | 32 | √ |

(2) Register Summary without Description

| Register Name | Abbreviation | R/W | Offset Address | Initial Value | Size | R-Car V4M with PCIE 2-lane |
|---|---|---|---|---|---|---|
| STP Status Register | STPSR | R | H'0500 | H'0000 0000 | 32 | √ |
| STP Control Register | STPCR | R/W | H'0504 | H'0000 0000 | 32 | √ |
| STP Lanes Select Register | STPLSR | R/W | H'0508 | H'0000 0001 | 32 | √ |
| PHY Clock Synchronization Register | PCSR | R/W | H'050C | H'0000 2710 | 32 | √ |
| Link Initialization Register | LLIR | R/W | H'0510 | H'2020 1370 | 32 | √ |
| LINK and PHY Capability Register | LPCAPR | R | H'0514 | H'0000 2802 | 32 | √ |
| Link and PHY Identification Register | LPIDR | R | H'0518 | H'0021 4231 | 32 | √ |
| Link Initialization Register 2 | LLIR2 | R/W | H'051C | H'0000 0401 | 32 | √ |
| Integration Mode Ctrl Register | ITCTRL | R/W | H'0F00 | H'0000 0000 | 32 | √ |
| Claim Tag Bit Set Register | CLAIMSET | R/W | H'0FA0 | H'0000 000F | 32 | √ |
| Claim Tag Bit Clear Register | CLAIMCLR | R/W | H'0FA4 | H'0000 0000 | 32 | √ |
| Device Affinity 0 Register | DEVAFF0 | R | H'0FA8 | H'0000 0000 | 32 | √ |
| Device Affinity 1 Register | DEVAFF1 | R | H'0FAC | H'0000 0000 | 32 | √ |
| Lock Access Register | LAR | W | H'0FB0 | H'0000 0000 | 32 | √ |
| Lock Status Register | LSR | R | H'0FB4 | H'0000 0003 | 32 | √ |
| Authentication Status Register | AUTHSTATUS | R | H'0FB8 | H'0000 0000 | 32 | √ |
| Device Architecture Register | DEVARCH | R | H'0FBC | H'4771 0A50 | 32 | √ |
| Device ID Register 2 | DEVID2 | R | H'0FC0 | H'0000 0000 | 32 | √ |
| Device ID Register 1 | DEVID1 | R | H'0FC4 | H'0000 0000 | 32 | √ |
| Device ID Register | DEVID | R | H'0FC8 | H'0000 0000 | 32 | √ |
| Device Type Register | DEVTYPE | R | H'0FCC | H'0000 0050 | 32 | √ |
| Peripheral ID4 Register | PIDR4 | R | H'0FD0 | H'0000 0004 | 32 | √ |
| Peripheral ID5 Register | PIDR5 | R | H'0FD4 | H'0000 0000 | 32 | √ |
| Peripheral ID6 Register | PIDR6 | R | H'0FD8 | H'0000 0000 | 32 | √ |
| Peripheral ID7 Register | PIDR7 | R | H'0FDC | H'0000 0000 | 32 | √ |
| Peripheral ID0 Register | PIDR0 | R | H'0FE0 | H'0000 000A | 32 | √ |
| Peripheral ID1 Register | PIDR1 | R | H'0FE4 | H'0000 0030 | 32 | √ |
| Peripheral ID2 Register | PIDR2 | R | H'0FE8 | H'0000 00EA | 32 | √ |
| Peripheral ID3 Register | PIDR3 | R | H'0FEC | H'0000 0000 | 32 | √ |
| Component ID0 Register | CIDR0 | R | H'0FF0 | H'0000 000D | 32 | √ |
| Component ID1 Register | CIDR1 | R | H'0FF4 | H'0000 0090 | 32 | √ |
| Component ID2 Register | CIDR2 | R | H'0FF8 | H'0000 0005 | 32 | √ |
| Component ID3 Register | CIDR3 | R | H'0FFC | H'0000 00B1 | 32 | √ |

### (2) DBE registers configuration related to HSSTP

DBE has trace pin control and debugging status registers for chip-specific debug features. DBGREG2 and DBGREG9 in DBE are the HSSTP related registers. It is possible to write to DBGREG2 (Debug Function Selection Register), only after writing a key code to the DBGREG9 (KEY register). Please refer to the section 169.3.1 HSSTP start-up sequence for the detailed access sequence for more information.

**Table 169.2 Register summary**

| Name of Register | Abbreviation | R/W | Address (CPU/ Debugger View) | Initial Value | Size |
|---|---|---|---|---|---|
| Debug Function Selection Register | DBGREG2 | R/W | H'E610 0024 | H'0000 0000 | 32 |
| KEY register | DBGREG9 | R/W | H'E610 0040 | H'0000 0000 | 32 |

### (3) Physical unit registers configuration related to HSSTP

HSSTP shares PCIE physical unit with PCIE LINK. The setting to PCIE physical unit is required before using the HSSTP function. For details of PCIE register, please refer to the section 169.3.1 (1) HSSTP with PCIE0 physical unit for 2-lane.

## 169.2 Register Description

This section describes the register specification not described in the Arm HSSTP Specification.

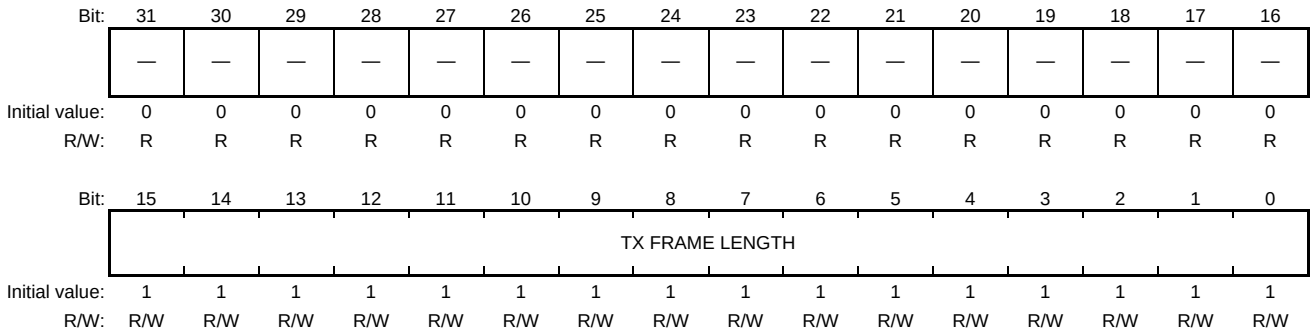
### 169.2.1 PHY Clock Synchronization Register 2 (PCSR2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | CC_DURATION | | | | | |
| Initial value: | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | Reserved | B’000 | R/W | Reserved<br>These bits are always read as initial value.<br>The write value should always be initial value. |
| 28 | Reserved | B’1 | R/W | Reserved<br>This bit is always read as initial value.<br>The write value should always be initial value. |
| 27 to 24 | Reserved | H’A | R/W | Reserved<br>These bits are always read as initial value.<br>The write value should always be initial value. |
| 23 to 22 | Reserved | B’00 | R/W | Reserved<br>These bits are always read as initial value.<br>The write value should always be initial value. |
| 21 to 16 | CC_DURATION | H'06 | R/W | The output period of Clock Compensation<br>(CC_DURATION cycles in PCI PIPEφ) |
| 15 to 0 | Reserved | H'0000 | R/W | Reserved<br>These bits are always read as initial value.<br>The write value should always be initial value. |

### 169.2.2 TX FRAME LENGTH Register (TXFLR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | Reserved | H'0000 | R | Reserved<br>These bits are always read as initial value.<br>The write value should always be initial value. |
| 15 to 0 | TX_FRAME_LENGTH | H'FFFF | R/W | The frame length of Aurora frame.<br>This bit field represents number of cycles per Aurora frame.<br>Number of cycles per Aurora frame is<br>INT(TX_FRAME_LENGTH/2+1).<br>In 2-lane case, size of Aurora frame is<br>INT(TX_FRAME_LENGTH/2+1)*4 [byte].<br>In 4-lane case, size of Aurora frame is<br>INT(TX_FRAME_LENGTH/2+1)*8 [byte]. |

### 169.2.3 Trace Source Select Register (TSSR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | Trace Source Select | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | Reserved | All 0 | R/W | Reserved<br>These bits are always read as initial value.<br>The write value should always be initial value. |
| 5 to 0 | Trace_Source_Select | H'1 | R/W | Trace source selection<br>When using PCIE physical unit for 2-lane<br>H'0: CoreSight is selected.<br>H'1: Cortex-A76 is selected.<br>H'2: Reserved.<br>H'3: Cortex-R52 #0 is selected.<br>H'4: Cortex-R52 #1 is selected.<br>H'5: Cortex-R52 #2 is selected.<br>Others: Reserved for future use. |

### 169.2.4 Aurora PHY Control Register (APCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | TISD REN | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R | R | R/W | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | Reserved | All 0 | R | Reserved for future use. |
| 28 to 24 | Reserved | All 0 | R/W | Reserved for future use. |
| 23 to 17 | Reserved | All 0 | R | Reserved for future use. |
| 16 | Reserved | 0 | R/W | Reserved for future use. |
| 15 to 9 | Reserved | All 0 | R | Reserved for future use. |
| 8 | Reserved | 0 | R/W | Reserved for future use. |
| 7 to 4 | Reserved | All 0 | R | Reserved for future use. |
| 3 | TISDREN | 0 | R/W | Tx Driver Enable<br>0: Disabled (default)<br>1: Enabled |
| 2 to 0 | Reserved | All 0 | R | Reserved for future use. |

### 169.2.5 Aurora PHY Status Register (APSR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | COSPLOCK |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | Reserved | All 0 | R | Reserved for future use. |
| 0 | COSPLOCK | 0 | R | PLL Lock Detect<br>0: Unlocked<br>1: Locked |

### 169.2.6 PLL Lock Management Register (PLMR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | LTS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | Reserved | All 0 | R | Reserved for future use. |
| 0 | LTS | 0 | R/W | Reserved for future use. |

### 169.2.7 Mode Control Register (MCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | ECP lane select | | CRC octets | CRC EN | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | * | 0 | 1 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | Reserved | All 0 | R/W | Reserved for future use. |
| 5 to 4 | ECP_lane_select | *[2-lane]<br>B’00 | R/W | [2-lane]<br>B’00: ECP is output from Lane0. (Set this value when CRC is embedded.)<br>B’01: ECP is output from Lane1. (Set this value When CRC is not embedded.)<br>B’10: Reserved.<br>B’11: Reserved. |
| 3 | CRC_octets | B’0 | R/W | B’0: (Set this value when CRC is embedded.)<br>B’1: (Set this value when CRC is not embedded.) |
| 2 | CRC_EN | B’1 | R/W | B’0: CRC is not embedded in trace data frame.<br>B’1: CRC is embedded in trace data frame. |
| 1 to 0 | Reserved | All 0 | R/W | Reserved.<br>These bits are always read as initial value.<br>The write value should always be initial value. |

## 169.3 Operation

The HSSTP start-up sequence is described in this section.

### 169.3.1 HSSTP start-up sequence

#### (1) HSSTP with PCIE0 physical unit for 2-lane

**Table 169.3a Start-up sequence for HSSTP with PCIE0 physical unit**

| Procedure | | Software Setting | HSSTP Status | PCIE physical unit Status | Note |
|---|---|---|---|---|---|
| Category | No | | | | |
| PCIE physical unit Initialization. | 0 | PCIe Reference Clock Gen setting (depends on board) | Stopped | Power Down | - |
| | 1 | DBE setting for PCIE physical unit selection.<br>DBGREG9 (KEY Register)<br>Address: 0xE6100040<br>Write 0x0000A500 to DBGREG9<br>(1st step to allow the access to DBGREG2)<br>Write 0x0000A501 to DBGREG9<br>(2nd step to allow the access to DBGREG2)<br>DBGREG2 (Debug Function Selection Register)<br>Address: 0xE6100024<br>Write 0b01 to Bit [1:0]<br>(for using PCIE0 physical unit) | Stopped | Power Down | DAP accesses via AXI-AP. |
| | 2 | Release MSTP on PCIE0 physical unit<br>MSTPCR6 Address: 0xE6152D18<br>Prepare the write value for the following steps:<br>1. Read MSTPCR6. We call the value (A).<br>2. Clear Bit [24] of (A) to 0. We call the value (B).<br>3. Inverse all bits of (B). We call the value (C).<br>CPGWPR Address: 0xE6150000<br>Write (C) to CPGWPR<br>MSTPCR6 Address: 0xE6152D18<br>Write (B) to MSTPCR6<br>MSTPSR6 Address: 0xE6152E18<br>Poll Bit [24] until 0 (for confirming MSTP release) | Stopped | Power Down | DAP accesses via AXI-AP. |
| | 3 | PCIE physical unit mode setting<br>Step1. PCIE physical unit 2-lane mode<br>PCIEMSR0 Address: 0xE65D6200<br>Write 0x00000011<br>Step2. PCIE0 physical unit reset negate<br>SRSTCLR6 Address: 0xE6152C98<br>Prepare the write value for the following steps:<br>1. Prepare 0x00000000 as (A)<br>2. Set Bit [24] of (A) to 1. We call the value (B).<br>3. Inverse all bits of (B). We call the value (C).<br>CPGWPR Address: 0xE6150000<br>Write (C) to CPGWPR<br>SRSTCLR6 Address: 0xE6152C98<br>Write (B) to SRSTCLR6<br>Step3. Start initialization sequence on PCIE0 physical unit<br>PCIE0 PHY RSTCTRL1 Address: 0xE65D6214<br>Write 0x00000000 | Stopped | Mode setting and reset clear | DAP accesses via AXI-AP. |
| | 4 | PCIE0 LINK setting<br>Step1. PLL lock<br>PCIE0 MPLLABCTRL2P0 Address: 0xE65D70C4<br>Poll Bit [5] until 1<br>Bit [4] indicates that the PLL for PCIE physical unit is locked.<br>Step2. Physical unit ready<br>PCIE0 TRNSREQP0 Address: 0xE65D7020<br>Poll Bit [3:2] until 0b00<br>Bit [3:2] indicates that PCIE physical unit is ready. | Stopped | PLL on PCIE physical unit locked. | DAP accesses via AXI-AP. |

| Procedure Category | No | Software Setting | HSSTP Status | PCIE physical unit Status | Note |
|---|---|---|---|---|---|
| | 5 | PCIE0 physical unit setting<br>* If the clock frequency of Aurora link does not have to be changed from 125MHz, this item is not necessary.<br>PCIE0 PHY CNFGP0 Address: 0xE65D62C4<br>Write 0b0101 to Bit [11:8]<br>Write 0b1010 to Bit [3:0]<br>Bit [11:10] is clock frequency of lane-1<br>Bit [9:8] is clock frequency of lane-0<br>0b01: 250MHz, 0b00: 125MHz<br>(The clock frequency is changed from 125MHz (5Gbps) to 250MHz (10Gbps).)<br>Bit [3:2] is width of lane-1<br>Bit [1:0] is width of lane-0<br>0b10: 32bit<br>PCIE0 PHY TXACKSTSCLR Address: 0xE65D7104<br>Write 0x00010001<br>PCIE0 PHY OVRDEN2 Address: 0xE65D71BC<br>Write 12'hFFF to Bit [29:18] | Stopped | PLL on PCIE physical unit locked. | DAP accesses via AXI-AP. |
| | 7 | PHYENABLE assert (software setting)<br>APCR offset: 0x630<br>Write 1 to Bit [3] | Stopped | PLL on PCIE physical unit locked. | - |
| | 8 | Poll COSPLLLOCK<br>APSR offset: 0x638<br>Poll Bit [0] until 1 (PLL Lock Detection) | Stopped | PLL on PCIE physical unit locked. | |
| HSSTP Initialization | 9 | STP_PWRUP assert<br>STPCR offset: 0x504<br>Write 1 to Bit [0] | /SP/ is sent to PCIE physical unit. (/SP/=/K28.5/D10.2/D10.2/D10.2) | /SP/ is sent to the outside of the LSI. | - |
| | 10 | Register initial setting<br>Update the settings upon necessity.<br>PCSR offset: 0x50C<br>Bit [15:0] is clock compensation (CC) generation interval.<br>Initial value: 0x2710<br>CC is inserted every 0x2710*2 Byte data out.<br>LLIR offset: 0x510<br>Bit [30:22] is VERIFY_LEN.<br>Bit [21:13] is BOND_LEN.<br>Bit [12:0] is ALIGN_LEN.<br>Initial value: 0x20201370<br>VERIFY_LEN: 0x080<br>BOND_LEN: 0x100<br>ALIGN_LEN: 0x1370<br>LLIR2 *4 offset: 0x51C<br>Bit [15:12] is VERIFY_MUL.<br>Bit [11:8] is BOND_MUL.<br>Bit [7:4] is ALIGN_MUL.<br>Initial value: 0x00000401<br>VERIFY_MUL: 0x0<br>BOND_MUL: 0x4<br>ALIGN_MUL: 0x0<br>TSSR offset: 0x5C0<br>Bit [3:0] is Trace Source Select<br>0x0: System Trace Macrocell is selected.<br>0x1: Cortex-A76 (Initial value)<br>0x2: Reserved.<br>0x3: Cortex-R52 #0 is selected.<br>0x4: Cortex-R52 #1 is selected.<br>0x5: Cortex-R52 #2 is selected.<br>PCSR2 offset: 0x540<br>Bit [21:16] is the output period of the CC.<br>Initial value: 0x06<br>TXFLR offset: 0x544<br>Bit [15:0] is frame size<br>Initial value: 0xFFFF<br>MCR offset: 0x550<br>With CRC<br>Bit [7:0] 0x04<br>Without CRC<br>Bit [7:0] 0x18 | /SP/ is sent to PCIE physical unit. (/SP/=/K28.5/D10.2/D10.2/D10.2) | /SP/ is sent to the outside of the LSI. | - |

| Procedure Category | Procedure No | Software Setting | HSSTP Status | PCIE physical unit Status | Note |
|---|---|---|---|---|---|
| | 11 | STP_nRST negate<br>STPCR offset: 0x504<br>Write 1 to Bit [2]<br>* Please don't control both STP_PWRUP and STP_nRST at the same time. | Align + Verify started | /SP/ is sent to the outside of the LSI. | - |
| | 12 | Poll LANE_UP<br>STPSR offset: 0x500<br>Poll Bit [0] until 1 (LANE_UP [0]) | Align completed (Verification pattern = /V/I/I/I/) | Verification pattern is sent to the outside of the LSI. | - |
| | 13 | Poll CHANNEL_UP<br>STPSR offset: 0x500<br>Poll Bit [6] until 1 (CHANNEL_UP [0]) | Verify completed<br>/I/ pattern is sent to PCIE physical unit.<br>A clock compensation (CC) pattern is inserted in transmission data. | /I/ pattern is sent to the outside of the LSI. | If VERIFICATION PATTERN is not recognized by RX (ICE) side and CHANNEL_UP is not asserted, the following procedure should be executed.<br>(1) Assert STP_nRST<br>(2) ALIGN_LEN and VERIFY_LEN<br>(3) Go back to procedure No.11 “STP_nRST negate” |
| Trace Transmission Setting | 14 | STP_nSWRST negate<br>STPCR offset: 0x504<br>Write 1 to Bit [3] | Releasing reset to TPIU interface | /I/ pattern is sent to the outside of the LSI. | - |
| | 15 | STP_EN assert<br>STPCR offset: 0x504<br>Write 1 to Bit [1] | Ready to receive the trace information from TPIU | /I/ pattern is sent to the outside of the LSI. | - |
| Coresight Trace Setting | - | Transmission setting of the TPIU<br>It is possible to set the procedure either before No.9 “STP_PWRUP assert” or after No.15 “STP_EN assert” because clock is supplied to all TPIU.<br>If it is set before No.9, TPIU output will start after assertion of STP_EN. If it is set after No.15, TPIU output will start immediately. | The trace data received from selected TPIU, and sent to PCIE physical unit. | The trace data received from HSSTP begins to send to the outside of the LSI. | - |

## 169.4 Usage Notes

It is prohibited to change DBGREG2 to change the PCIE physical unit selection while STPCR Bit [2] (nRST) is NOT zero.

## 169.5 Safety Mechanisms

No description on this section.

# 170. Debug and Trace

## 170.1 Overview

This module provides the debug and trace function for the processors including Cortex CPUs.

### 170.1.1 Features

- JTAG interface
  - Supports dedicated 5-pin JTAG and LPD[*1] (1.8V interface)
  - Additional pin-multiplexed 1-JTAG port (3.3V interface)
    - JTAG I/F for PAP
    - The JTAG I/F for GPU is not supported in R-Car V4H and V4M.
- The cross trigger function between following IPs:
  - Cortex-A76
  - Cortex-R52
  - VDSP 0/1
  - IMP DTA
  - CoreSight
  - ICUMX
- Tracing function
  - 10Gbps or 5Gbps HSSTP output bandwidth through PCIe 2-lane[*2] (full run mode only)
  - 32Gbps or 16Gbps ETR output bandwidth through PCIe 2-lane[*4] (full run mode only)
    - Trace capture using external DRAM via ETR (low power mode only)
  - 32-Kbyte embedded trace FIFO for Cortex-A76 CPU cluster
  - 32-Kbyte embedded trace FIFO for Cortex-R52
  - 32-Kbyte embedded trace FIFO for CoreSight
  - 16-Kbyte embedded trace FIFO for VDSP
  - System Trace Macrocell (ISP/MTI, AXI Masters, IMP DTA)
    - An additional STM for IMP DTA
    - STM3 for MTI isn't used in R-Car V4H and V4M.
  - CRC calculation for trace data
- Secure debug
  - Debugging is disabled if ICUMX and Life Cycle are used in Secure LCS[*6] and it can be enabled with secure debugging authentication.
  - Secure debug authentication is performed with ICUMX via Main-JTAG. Sub-JTAG is disconnected until the authentication completes.
  - On Chip Debug ID (OCDID) authentication. Debugger can't be connected unless this ID authentication passes.
  - JTAG debugger connection isn't allowed when the FUSE bit named 'fuse_jtag_debug_disable' is set to 1.

Note:
*1: LPD stands for Low Pin Debug, which is used for ICUMX
*2: Available bandwidth selectable from either 5Gbps per lane or 2.5Gbps per lane

*4: Available bandwidth selectable from either 16Gbps per lane or 8Gbps per lane
*6: ICUMX and Life Cycle define Life Cycle State (LCS). Secure (SE) LCS is one of Life Cycle states.

### 170.1.2 Block Diagram

## R-Car V4M CoreSight Sub System

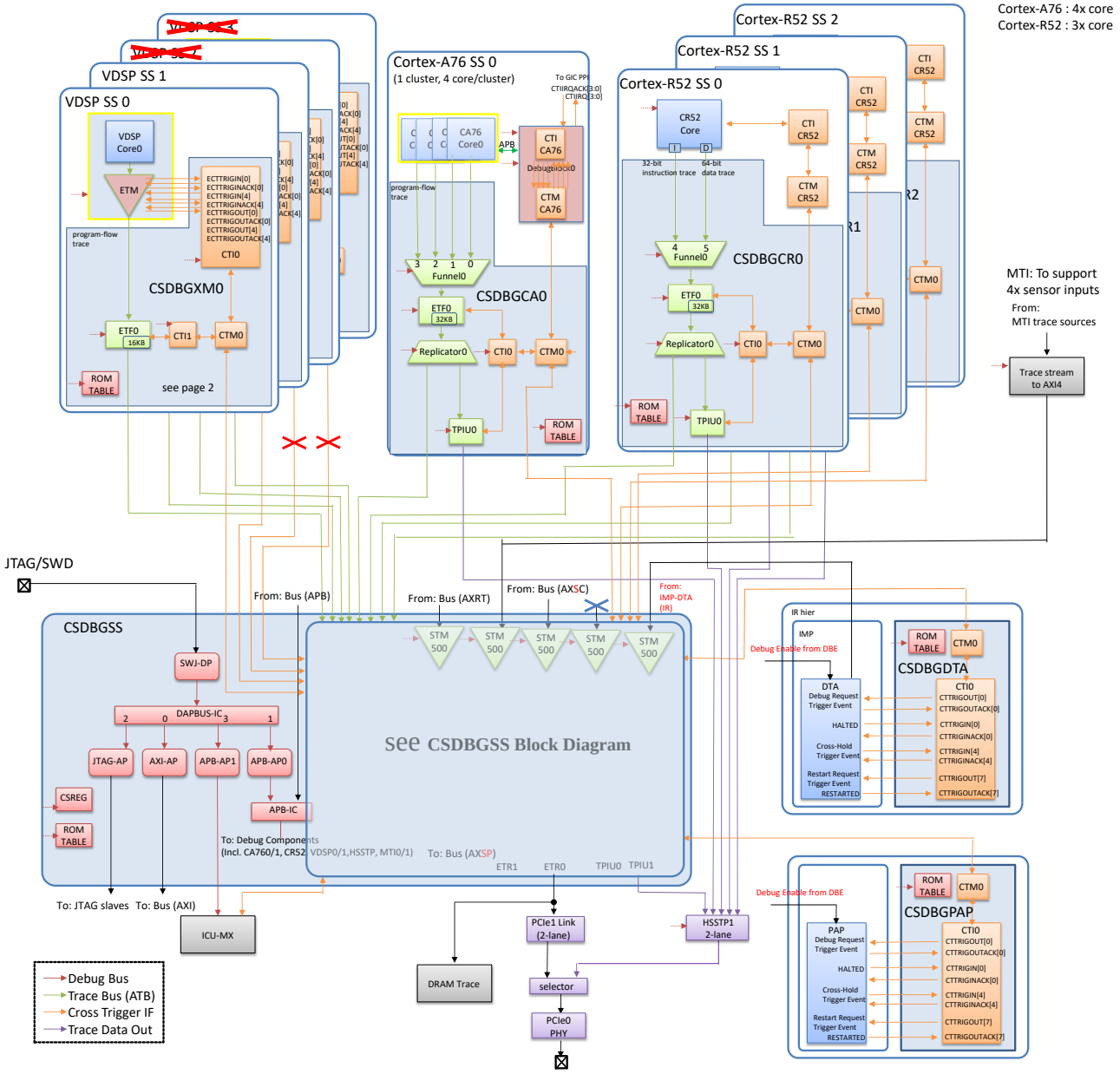



**Figure 170.1 CoreSight Sub-System Block Diagram**

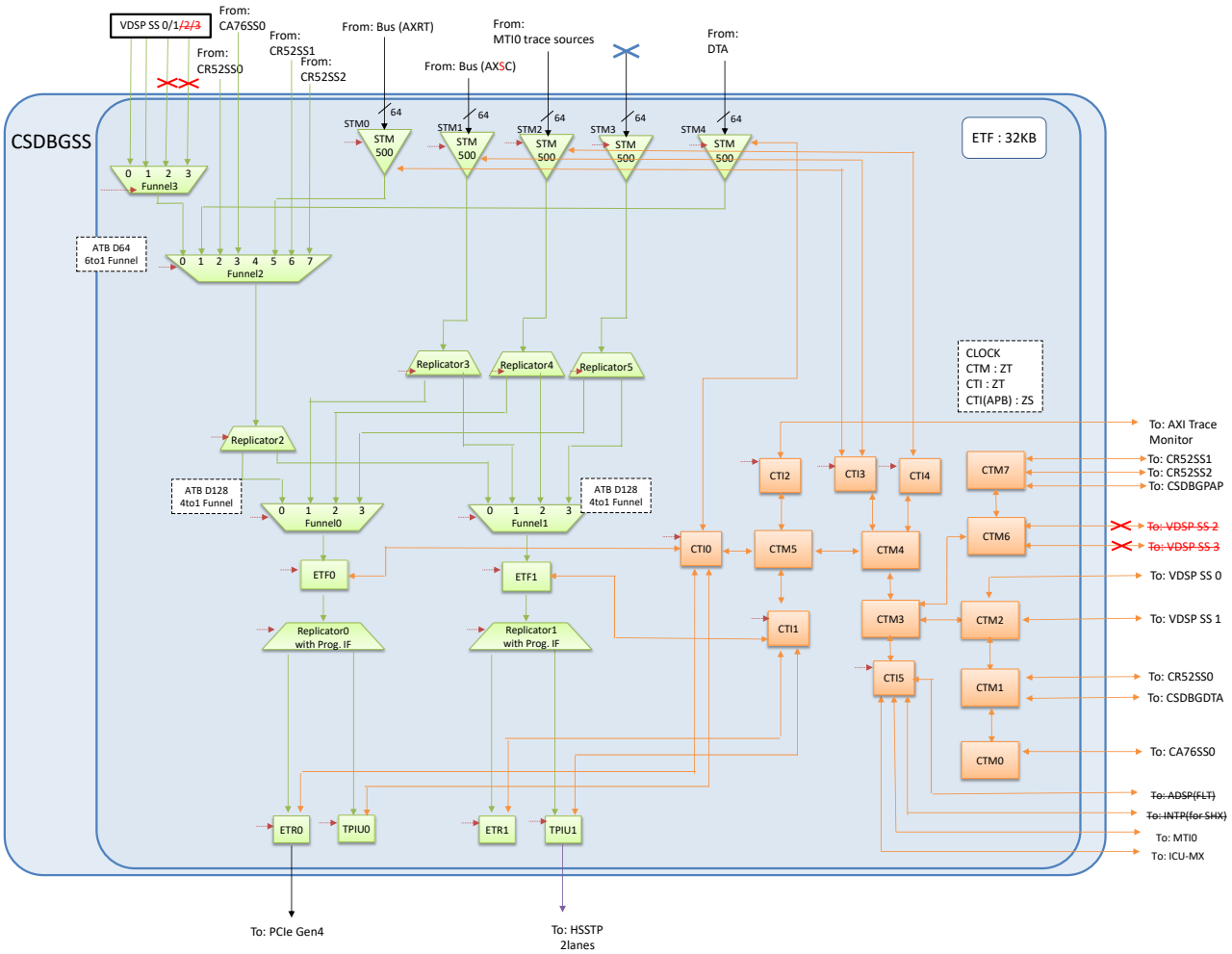



**Figure 170.2 CSDBGSS Block Diagram**

### 170.1.3 JTAG Pins

5-pin JTAG and SWD are supported as dedicated JTAG port. Sub-JTAG is additional JTAG ports as pin-multiplexed pins. Main-JTAG ports support 1.8V only. Sub-JTAG ports support 3.3V only. Each JTAG port can be connected to different TAP controller through MD pins. Refer to JTAG selection tables in Table 170.1.

**Table 170.1 JTAG pin and LPD pin for R-Car V4H and V4M**

| IF | Pin name | I/O | Function | | | |
|---|---|---|---|---|---|---|
| | | | JTAG IF | JTAG IF for ICUMX | LPD IF for ICUMX | SWD |
| Main-JTAG | DCUTRST#_LPDRST# | Input | TRST# | DCUTRST# | LPDRST# | - |
| | DCUTCK_LPDCLK | Input | TCK | DCUTCK | LPDCLK | SWCLK |
| | DCUTMS | I/O | TMS | DCUTMS | - | SWDIO |
| | DCUTDI_LPDI | Input | TDI | DCUTDI | LPDI | - |
| | DCUTDO_LPDO | Output | TDO | DCUTDO | LPDO | - |
| | DCURDY_N_LPDCLKOU | Output | - | DCURDY# | LPDCLKOUT | - |
| Sub-JTAG | RXDB_EXTFXR | Input | TRST# | - | - | - |
| | TPU0TO1 | Input | TCK | - | - | SWCLK |
| | FXR_TXDB | I/O | TMS | - | - | SWDIO |
| | CANFD2_TX | Input | TDI | - | - | - |
| | TPU0TO0 | Output | TDO | - | - | - |
| | FXR_TXENB_N | Output | - | - | - | - |

### 170.1.4 Register Configuration

Do not write to addresses other than those listed below, otherwise normal operation cannot be guaranteed. Values read from other addresses are undefined.

[Legend]
Initial value: Register value after a reset
-: Undefined value
R/W: Readable/writable. The written value can be read.
R: Read-only. The write value should always be the initial value.
-/W: Write-only. The read value is undefined.

#### (1) CoreSight & CPU Debug Resource Register Configurations

**Table 170.2 CoreSight Address Map**

| Module | Component | | Base Address | | Offset from Base Address |
|---|---|---|---|---|---|
| | Name | Abbreviation | CPU view Address | Debugger view Address | |
| CSDBGSS | ROM Table | CSD-ROM | H'00 F800 0000 | H'8000 0000 | H'00 0000 |
| | STM0 | CSD-STM0 | | | H'01 0000 |
| | STM1 | CSD-STM1 | | | H'02 0000 |
| | STM2 | CSD-STM2 | | | H'03 0000 |
| | STM3 | CSD-STM3 | | | H'04 0000 |
| | ETF0 | CSD-ETF0 | | | H'05 0000 |
| | ETF1 | CSD-ETF1 | | | H'06 0000 |
| | STM4 | CSD-STM4 | | | H'07 0000 |
| | ATB Funnel3 | CSD-FUN3 | | | H'08 0000 |
| | RESERVED | | | | H'09 0000 |
| | RESERVED | | | | H'0A 0000 |
| | TPIU0 | CSD-TPI0 | | | H' 0B0000 |
| | TPIU1 | CSD-TPI1 | | | H' 0C0000 |
| | ETR0 | CSD-ETR0 | | | H' 0D0000 |
| | ETR1 | CSD-ETR1 | | | H' 0E0000 |
| | ATB Funnel0 | CSD-FUN0 | | | H' 0F0000 |
| | ATB Funnel1 | CSD-FUN1 | | | H'10 0000 |
| | ATB Funnel2 | CSD-FUN2 | | | H'11 0000 |
| | CTI0 | CSD-CTI0 | | | H'12 0000 |
| | CTI1 | CSD-CTI1 | | | H'13 0000 |
| | CTI2 | CSD-CTI2 | | | H'14 0000 |
| | CTI3 | CSD-CTI3 | | | H'15 0000 |
| | CTI4 | CSD-CTI4 | | | H'16 0000 |

| Module | | Component | | Base Address | | Offset from Base Address |
|---|---|---|---|---|---|---|
| | | Name | Abbreviation | CPU view Address | Debugger view Address | |
| | | CTI5 | CSD-CTI5 | | | H'17 0000 |
| | | Replicator0 | CSD-REP0 | | | H'18 0000 |
| | | Replicator1 | CSD-REP1 | | | H'19 0000 |
| | | CSREG | CSD-REG | | | H'1F 0000 |
| CSDBG Peripheral | | MTICTRL0 | | H'00 F820 0000 | H'8020 0000 | H'00 0000 |
| | | MTICTRL1 | | | | H'01 0000 |
| | | HSSTP2P0 | | | | H'02 0000 |
| | | XMON_STM0 | | | | H'08 0000 |
| | | XMON_CTRL0 | | | | H'09 0000 |
| CSDBGDTA | | ROM Table | DTA-ROM | H'00 F830 0000 | H'8030 0000 | H'00 0000 |
| | | CTI0 | DTA-CTI0 | | | H'01 0000 |
| CSDBGPAP | | ROM Table | PAP-ROM | 40'h00_F838_0000 | 32'h8038_0000 | 20'h0_0000 |
| | | CTI0 | PAP-CTI0 | | | 20'h1_0000 |
| VDSP SS 0 | CSDBG XM0 | ROM Table | D0XM-ROM | H'00 F840 0000 | H'8040 0000 | H'00 0000 |
| | | ETF0 | D0XM-ETF0 | | | H'01 0000 |
| | | CTI0 | D0XM-CTI0 | | | H'02 0000 |
| | | CTI1 | D0XM-CTI1 | | | H'03 0000 |
| | dsp top 0 | ETM-R4 | D0XM-ETMR4 | 40'h00_F848_0000 | 32'h8048_0000 | H'00 0000 |
| | | CEVA XM8 | D0XM-XM8 | 40'h00_F849_0000 | 32'h8049_0000 | H'00 0000 |
| VDSP SS 1 | CSDBG XM1 | ROM Table | D1XM-ROM | H'00 F850 0000 | H'8050 0000 | H'00 0000 |
| | | ETF0 | D1XM-ETF0 | | | H'01 0000 |
| | | CTI0 | D1XM-CTI0 | | | H'02 0000 |
| | | CTI1 | D1XM-CTI1 | | | H'03 0000 |
| | dsp top 1 | ETM-R4 | D1XM-ETMR4 | 40'h00_F858_0000 | 32'h8058_0000 | 20'h0_0000 |
| | | CEVA XM8 | D1XM-XM8 | 40'h00_F859_0000 | 32'h8059_0000 | 20'h0_0000 |
| CR52SS 0 | CSDBG CR0 | ROM Table | R0CR-ROM | H'00 F880 0000 | H'8080 0000 | H'00 0000 |
| | | ETF0 | R0CR-ETF0 | | | H'01 0000 |
| | | CTI0 | R0CR-CTI0 | | | H'02 0000 |
| | | TPIU0 | R0CR-TPI0 | | | H'03 0000 |
| | | ATB Funnel0 | R0CR-FUN0 | | | H'04 0000 |
| | | Replicator0 | R0CR-REP0 | | | H'05 0000 |
| | MP Core 0 | ROM Table | R0MP-ROM | 40'h00_F8C0_0000 | 32'h80C0_0000 | 20'h0_0000 |
| | | Processor 0 Debug | R0MP-DBG0 | | | 20'h1_0000 |
| | | Processor 0 CTI | R0MP-CTI0 | | | 20'h2_0000 |
| | | Processor 0 PMU | R0MP-PMU0 | | | 20'h3_0000 |
| | | Processor 0 ETM | R0MP-ETM0 | | | 20'h4_0000 |
| CR52SS 1 | CSDBG CR1 | ROM Table | R1CR-ROM | 40'h00_F890_0000 | 32'h8090_0000 | 20'h0_0000 |
| | | ETF0 | R1CR-ETF0 | | | 20'h1_0000 |

| Module | | Component | | Base Address | | Offset from Base Address |
|---|---|---|---|---|---|---|
| | | Name | Abbreviation | CPU view Address | Debugger view Address | |
| | | CTI0 | R1CR-CTI0 | | | 20'h2_0000 |
| | | TPIU0 | R1CR-TPI0 | | | 20'h3_0000 |
| | | ATB Funnel0 | R1CR-FUN0 | | | 20'h4_0000 |
| | | Replicator0 | R1CR-REP0 | | | 20'h5_0000 |
| | MP Core 0 | ROM Table | R1MP-ROM | 40'h00_F8D0_0000 | 32'h80D0_0000 | 20'h0_0000 |
| | | Processor 0 Debug | R1MP-DBG0 | | | 20'h1_0000 |
| | | Processor 0 CTI | R1MP-CTI0 | | | 20'h2_0000 |
| | | Processor 0 PMU | R1MP-PMU0 | | | 20'h3_0000 |
| | | Processor 0 ETM | R1MP-ETM0 | | | 20'h4_0000 |
| CR52SS2 | CSDBG CR2 | ROM Table | R2CR-ROM | 40'h00_F8A0_0000 | 32'h80A0_0000 | 20'h0_0000 |
| | | ETF0 | R2CR-ETF0 | | | 20'h1_0000 |
| | | CTI0 | R2CR-CTI0 | | | 20'h2_0000 |
| | | TPIU0 | R2CR-TPI0 | | | 20'h3_0000 |
| | | ATB Funnel0 | R2CR-FUN0 | | | 20'h4_0000 |
| | | Replicator0 | R2CR-REP0 | | | 20'h5_0000 |
| | MP Core 0 | ROM Table | R2MP-ROM | 40'h00_F8E0_0000 | 32'h80E0_0000 | 20'h0_0000 |
| | | Processor 0 Debug | R2MP-DBG0 | | | 20'h1_0000 |
| | | Processor 0 CTI | R2MP-CTI0 | | | 20'h2_0000 |
| | | Processor 0 PMU | R2MP-PMU0 | | | 20'h3_0000 |
| | | Processor 0 ETM | R2MP-ETM0 | | | 20'h4_0000 |
| CA76SS0 | CSDBG CA0 | ROM Table | E0CA-ROM | H'00 F900 0000 | H'8100 0000 | H'00 0000 |
| | | ETF0 | E0CA -ETF0 | | | H'01 0000 |
| | | CTI0 | E0CA -CTI0 | | | H'02 0000 |
| | | TPIU0 | E0CA -TPIU0 | | | H'03 0000 |
| | | ATB Funnel0 | E0CA -FUN0 | | | H'04 0000 |
| | | Replicator0 | E0CA-REP0 | | | H'05 0000 |
| | Cluster 0 (MP Core 0, 1, 2, 3) | ROM Table | E0C0MP-ROM | H'00 F940 0000 | H'8140 0000 | H'00 0000 |
| | | CPU 0 Debug | E0C0MP-DBG0 | | | H'01 0000 |
| | | CPU 0 CTI | E0C0MP-CTI0 | | | H'02 0000 |
| | | CPU 0 PMU | E0C0MP-PMU0 | | | H'03 0000 |
| | | CPU 0 ETM | E0C0MP-ETM0 | | | H'04 0000 |
| | | CPU 1 Debug | E0C0MP-DBG1 | | | H'11 0000 |
| | | CPU 1 CTI | E0C0MP-CTI1 | | | H'12 0000 |
| | | CPU 1 PMU | E0C0MP-PMU1 | | | H'13 0000 |
| | | CPU 1 ETM | E0C0MP-ETM1 | | | H'14 0000 |
| | | CPU 2 Debug | E0C0MP-DBG2 | | | H'21 0000 |
| | | CPU 2 CTI | E0C0MP-CTI2 | | | H'22 0000 |
| | | CPU 2 PMU | E0C0MP-PMU2 | | | H'23 0000 |
| | | CPU 2 ETM | E0C0MP-ETM2 | | | H'24 0000 |

| Module | | Component | | Base Address | | Offset from Base Address |
|---|---|---|---|---|---|---|
| | | Name | Abbreviation | CPU view Address | Debugger view Address | |
| | | CPU 3 Debug | E0C0MP-DBG3 | | | H'31 0000 |
| | | CPU 3 CTI | E0C0MP-CTI3 | | | H'32 0000 |
| | | CPU 3 PMU | E0C0MP-PMU3 | | | H'33 0000 |
| | | CPU 3 ETM | E0C0MP-ETM3 | | | H'34 0000 |

### (2) CSreg Register Configurations

CSreg is the collection of control registers for chip-specific debug features. These registers can be accessed from both Debugger and CPU via Debug-APB.

**Table 170.3 CSreg register configurations**

| Name of Register | R/W | CPU view Address | Debugger view Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| PMUSNAPSHOTREQ Register | R/W | H' F81F 0800 | H' 801F 0800 | H'0000 0000 | 32 |
| PMUSNAPSHOTACK Register | R | H' F81F 0804 | H' 801F 0804 | H'0000 000X | 32 |
| CSREG_E00 Register | R/W | H' F81F 0E00 | H' 801F 0E00 | H'0000 001F | 32 |
| CSREG_E01 Register | R/W | H' F81F 0E04 | H' 801F 0E04 | H'0000 0000 | 32 |
| LOCKACCESS Register | W | H' F81F 0FB0 | H' 801F 0FB0 | - | 32 |
| LOCKSTATUS Register | R | H' F81F 0FB4 | H' 801F 0FB4 | H'0000 0003 | 32 |
| Peripheral ID4 Register | R | H' F81F 0FD0 | H' 801F 0FD0 | H'0000 0004 | 32 |
| Peripheral ID0 Register | R | H' F81F 0FE0 | H' 801F 0FE0 | H'0000 00AA | 32 |
| Peripheral ID1 Register | R | H' F81F 0FE4 | H' 801F 0FE4 | H'0000 003F | 32 |
| Peripheral ID2 Register | R | H' F81F 0FE8 | H' 801F 0FE8 | H'0000 001A | 32 |
| Peripheral ID3 Register | R | H' F81F 0FEC | H' 801F 0FEC | H'0000 0000 | 32 |
| Component ID0 Register | R | H' F81F 0FF0 | H' 801F 0FF0 | H'0000 000D | 32 |
| Component ID1 Register | R | H' F81F 0FF4 | H' 801F 0FF4 | H'0000 00F0 | 32 |
| Component ID2 Register | R | H' F81F 0FF8 | H' 801F 0FF8 | H'0000 0005 | 32 |
| Component ID3 Register | R | H' F81F 0FFC | H' 801F 0FFC | H'0000 00B1 | 32 |

### (3) TDBG Register Configurations

TDBG is the collection of trace pin control registers and debugging status registers for chip-specific debug features. Only after write operation to the KEY bit in DBGREG9 is done, it is possible to write to DBGREG2. Note that asserting KEY bit in DBGREG9 register requires two phases of writing as describing in section 170.2.20

**Table 170.4 TDBG register configurations**

| Name of Register | Abbreviation | R/W | Address (CPU/ Debugger View) | Initial Value | Access Size |
|---|---|---|---|---|---|
| Debug Register 1 | DBGREG1 | R | H'E610 0020 | H'X00X 0000* | 32 |
| Debug Register 2 | DBGREG2 | R/W | H'E610 0024 | H'0000 0000 | 32 |
| Debug Register 4 | DBGREG4 | R | H'E610 002C | H'00XX XXXX* | 32 |
| Debug Register 9 | DBGREG9 | R/W | H'E610 0040 | H'0000 0000 | 32 |

* The initial value is determined by input of LSI pin.

### 170.1.5 Connected Module

**Table 170.5 Connected module**

| Module name | Connected module name | Function of connected module |
|---|---|---|
| CoreSight | AP-System Core | Processor |
| | Arm Realtime Core | Processor |
| | IMP | Image recognition engine |
| | HSSTP | High Speed Serial Trace Port |
| | CPG | Clock Pulse Generator |
| | PFC | Pin function |
| | PAP | PAP-System |
| | MTI | Measurement Transmission Interface |
| | ICUMX | Processor |
| | VDSP | Processor |

### 170.1.6 Part Number and TARGETID

Part number is a product identification. This number is stored in Peripheral Identification Registers of ROM Table and TARGETID Register. Debugger reads those registers to identify the product. Each product has unique value of part number as shown in Table 170.6:

Note: Refer to Arm CoreSight Architecture Specification v2.0 for more detail information.

**Table 170.6 Part Number and TARGETID**

| Forth Generation of R-Car Series Prosucts | Part Number | TARGETID |
|---|---|---|
| R-Car V4M | H’051 | H’0051 0447 |

## 170.2 Register Description

### 170.2.1 CoreSight & CPU Debug Resource Registers

Refer to following manuals for generic CoreSight components.

- Arm Architecture Reference Manual, Armv8 for ARMv8-A architecture profile (ARM DDI 0487B.b)
- Arm DynamIQ Shared Unit Technical Reference Manual (ARM 100453_0401_03_en, Revision: r4p1)
- Arm Cortex-R52 Processor Technical Reference Manual (ARM 100026_0103_00_en, Revision: r1p3)
- Arm CoreSight SoC-400 Technical Reference Manual (ARM 100536_0302_01_en, Revision: r3p2)
- Arm CoreSight Trace Memory Controller Technical Reference Manual (ARM DDI 0451, Revision: r0p1)
- Arm CoreSight STM-500 System Trace Macrocell Technical Reference Manual (ARM DDI 0528B, Revision: r0p1)

### 170.2.2 PMUSNAPSHOTREQ Register

Address CPU view: H' F81F 0800 Debugger view: H' 801F 0800

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | - | - | - | - | - | - | - | CA760_ DSU0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | Reserved | H'0000_000 | R/W | Reserved |
| 0 | CA760_DSU0 | 0 | R/W | Write signal value of PMUSNAPSHOTREQ to CA760 DSU0 |

### 170.2.3 PMUSNAPSHOTACK Register

Address CPU view: H' F81F 0804 Debugger view: H' 801F 0804

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | - | - | - | - | - | - | - | CA760_ DSU0 |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | - | - | - |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | Reserved | H’0000_000 | R | Reserved |
| 0 | CA760_DSU0 | - | R | Read signal value of PMUSNAPSHOTACK from CA760 DSU0 |

### 170.2.4 CSREG_E00 Register

Address CPU view: H' F81F 0E00 Debugger view: H' 801F 0E00

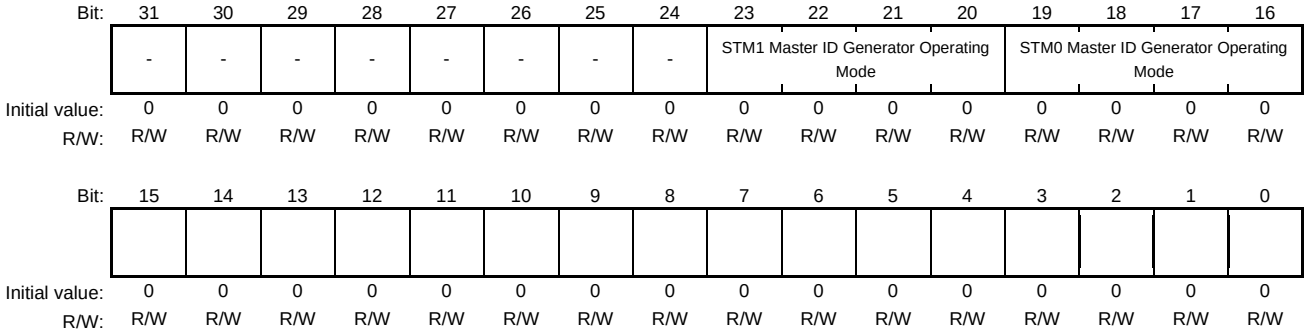
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | STM4 FIXED AWBURST enable | STM3 FIXED AWBURST enable | STM2 FIXED AWBURST enable | STM1 FIXED AWBURST enable | STM0 FIXED AWBURST enable |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | - | H'00000000 | R/W | Reserved |
| 4 | STM4_FIXED_AWBURST_enable | B'1 | R/W | Override AWBURST at input of STM4 to FIXED type.<br>0: disable<br>1: enable |
| 3 | STM3_FIXED_AWBURST_enable | B'1 | R/W | Override AWBURST at input of STM3 to FIXED type.<br>0: disable<br>1: enable |
| 2 | STM2_FIXED_AWBURST_enable | B'1 | R/W | Override AWBURST at input of STM2 to FIXED type.<br>0: disable<br>1: enable |
| 1 | STM1_FIXED_AWBURST_enable | B'1 | R/W | Override AWBURST at input of STM1 to FIXED type.<br>0: disable<br>1: enable |
| 0 | STM0_FIXED_AWBURST_enable | B'1 | R/W | Override AWBURST at input of STM0 to FIXED type.<br>0: disable<br>1: enable |

### 170.2.5 CSREG_E01 Register

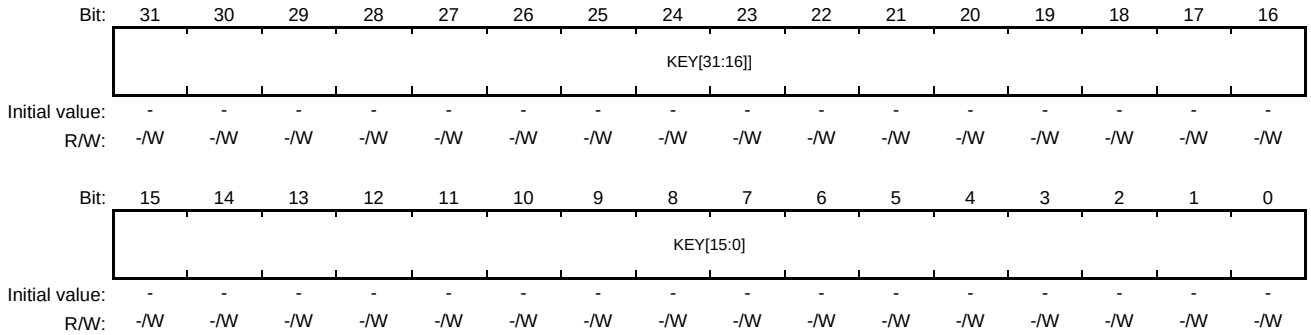
Address CPU view: H' F81F 0E04 Debugger view: H' 801F 0E04




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | Reserved | H'00 | R/W | Reserved |
| 23 to 20 | STM1_Master_ID_Generator_Operating_Mode | H'0 | R/W | 4'b0000: Normal Mode<br>4'b0001: Burst Mode |
| 19 to 16 | STM0_Master_ID_Generator_Operating_Mode | H'0 | R/W | See above |
| 15 to 0 | Reserved | H'00 | R/W | Reserved |

### 170.2.6 LOCKACCESS Register

Address CPU view: H' F81F 0FB0 Debugger view: H' 801F 0FB0




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | KEY[31:0] | - | W | Lock Access Register. See CoreSight Architecture Specification |

### 170.2.7 LOCKSTATUS Register

Address CPU view: H' F81F 0FB4 Debugger view: H' 801F 0FB4

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | nTT | SLK | SLI |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | H’0000 0000 | R | Lock Status Register. See CoreSight Architecture Specification |
| 2 | nTT | 0 | R | |
| 1 | SLK | 1 | R | |
| 0 | SLI | 1 | R | |

### 170.2.8 Peripheral ID0 Register

Address CPU view: H' F81F 0FE0 Debugger view: H' 801F 0FE0

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | Part No.0[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | - | H'000000 | R | Reserved |
| 7 to 0 | Part_No.0[7:0] | H'AA | R | part number of the component |

### 170.2.9 Peripheral ID1 Register

Address CPU view: H' F81F 0FE4 Debugger view: H' 801F 0FE4

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | JEP106 ID code[3:0] | | | | Part No.1[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | - | H'000000 | R | Reserved |
| 7 to 4 | JEP106_ID_code[3:0] | H'3 | R | JEDEC code |
| 3 to 0 | Part_No.1[3:0]] | H'F | R | part number of the component |

### 170.2.10 Peripheral ID2 Register

Address CPU view: H' F81F 0FE8 Debugger view: H' 801F 0FE8

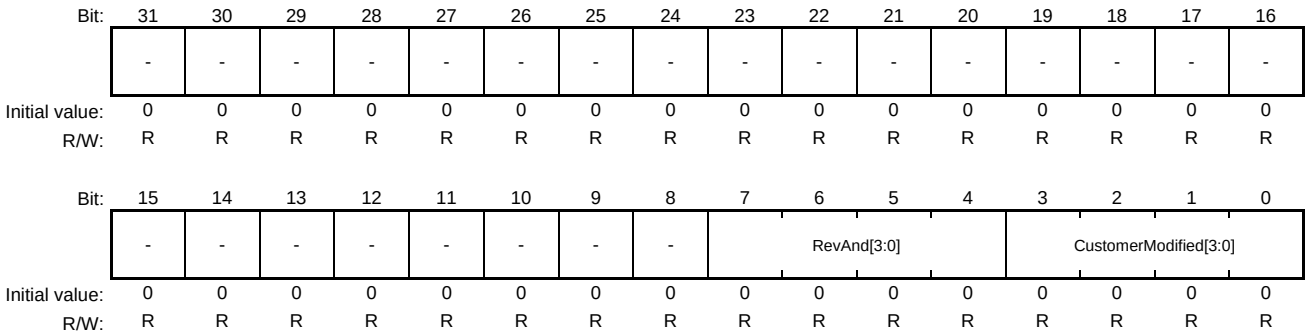
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | Rev[3:0] | | | | - | JEP106 ID code[6:4] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | H'000000 | R | Reserved |
| 7 to 4 | Rev[3:0] | H’1 | R | Revision |
| 3 | — | B'1 | R | |
| 2 to 0 | JEP106_ID_code[6:4] | H'2 | R | JEDEC code |

### 170.2.11 Peripheral ID3 Register

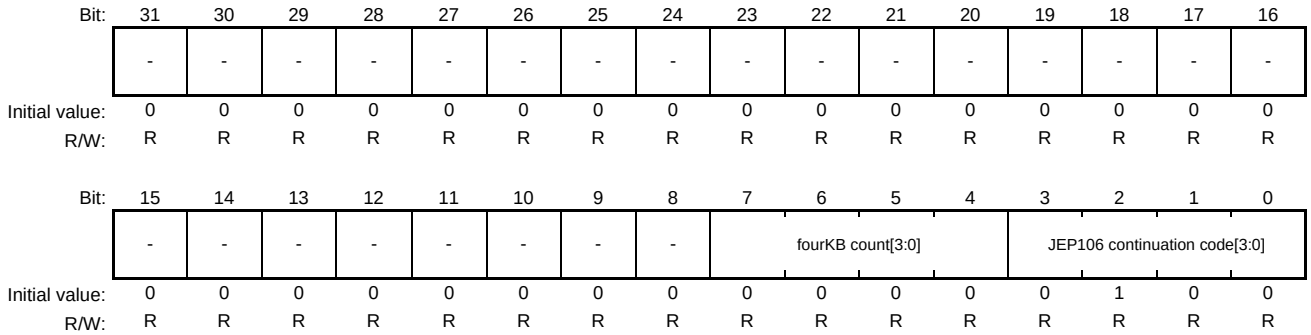
Address CPU view: H' F81F 0FEC Debugger view: H' 801F 0FEC




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | H'000000 | R | Reserved |
| 7 to 4 | RevAnd[3:0] | H'0 | R | Indicates that there are no errata fixes to this component |
| 3 to 0 | CustomerModified[3:0] | H'0 | R | Indicates that the customer has not modified this component |

### 170.2.12 Peripheral ID4 Register

Address CPU view: H' F81F 0FD0 Debugger view: H' 801F 0FD0




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | H'000000 | R | Reserved |
| 7 to 4 | fourKB_count[3:0] | H'0 | R | Indicates that the device only occupies 4KB of memory |
| 3 to 0 | JEP106_continuation_code[3:0] | H'4 | R | JEDEC code |

### 170.2.13 Component ID0 Register

Address CPU view: H' F81F 0FF0 Debugger view: H' 801F 0FF0

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | Preamble[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | H'000000 | R | Reserved |
| 7 to 0 | Preamble[7:0] | H'0D | R | component identification code |

### 170.2.14 Component ID1 Register

Address CPU view: H' F81F 0FF4 Debugger view: H' 801F 0FF4

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | Component class[3:0] | | | | Preamble[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | H'000000 | R | Reserved |
| 7 to 4 | Component_class[3:0] | H'F | R | Class of the component |
| 3 to 0 | Preamble[3:0] | H'0 | R | component identification code |

### 170.2.15 Component ID2 Register

Address CPU view: H' F81F 0FF8 Debugger view: H' 801F 0FF8

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | Preamble[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | H'000000 | R | Reserved |
| 7 to 0 | Preamble[7:0] | H'05 | R | component identification code |

### 170.2.16 Component ID3 Register

Address CPU view: H' F81F 0FFC Debugger view: H' 801F 0FFC

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | Preamble[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | H'000000 | R | Reserved |
| 7 to 0 | Preamble[7:0] | H'B1 | R | component identification code |

### 170.2.17 Debug Register 1 (DBGREG1)

This register is intended to reflect the values entered into external pins. The values for external pins are set when initialization is performed.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MD11 | MD10 | MD21 | MD20 | - | - | - | - | - | - | - | - | - | - | - | MDT[0] |
| Initial value: | Undefined | Undefined | Undefined | Undefined | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Undefined |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | MD11 | Undefined | R | The value of the MD11 pin input is reflected in the initial value. |
| 30 | MD10 | Undefined | R | The value of the MD10 pin input is reflected in the initial value. |
| 29, 28 | MD21_MD20 | Undefined | R | The value of MD21 and MD20 pin input is reflected in the initial value. |
| 27 to 18 | - | All 0 | R | Reserved<br>These bits are always read as 0. |
| 17 | - | 0 | R | Reserved<br>This bit is always read as 0. |
| 16 | MDT[0] | Undefined | R | The value of the MDT0 pin input is reflected in the initial value. |
| 15 to 0 | - | All 0 | R | Reserved<br>These bits are always read as 0. |

### 170.2.18 Debug Register 2 (DBGREG2)

This register is for setting the PCIe Trace port to enable or disable.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | DMON2EN | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 2 | - | All 0 | R | Reserved<br>Initial value = all 0. |
| 1 to 0 | DMON2EN | All 0 | R/W | Selects the 2-lane HSSTP or not used<br>11: Reserved<br>10: Reserved<br>01: 2-lane HSSTP uses PCIE port0 trace output<br>00: PCIE trace output isn't used by HSSTP |

### 170.2.19 Debug Register 4 (DBGREG4)

This is a status register showing debug features are enable or disable. It is possible to perform read operations only from the CPU.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | CRIMP_HNIDEN | CRIMP_HIDEN | CRIMP_NIDEN | CRIMP_DBGEN | PAP_NIDEN | PAP_DBGEN | DTA_NIDEN | DTA_DBGEN | VDSP_SPNIDEN | VDSP_SPIDEN | VDSP_NIDEN | VDSP_DBGEN |
| Initial value: | 0 | 0 | 0 | 0 | * | * | * | * | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | CS_DEVEN | CS_SPNIDEN | CS_SPIDEN | CS_NIDEN | CS_DBGEN | CR_HNIDEN | CR_HIDEN | CR_NIDEN | CR_DBGEN | CA_SPNIDEN | CA_SPIDEN | CA_NIDEN | CA_DBGEN |
| Initial value: | 0 | 0 | 0 | * | * | * | * | * | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | - | All 0 | R | Reserved<br>These bits are always read as 0. |
| 27 | CRIMP_HNIDEN | *1 | R | Indicates disable or enable of the HNIDEN of Cortex-R52 2(for IMP).<br>0: Disable<br>1: Enable |
| 26 | CRIMP_HIDEN | *1 | R | Indicates disable or enable of the HIDEN of Cortex-R52 2(for IMP).<br>0: Disable<br>1: Enable |
| 25 | CRIMP_NIDEN | *1 | R | Indicates disable or enable of the NIDEN of Cortex-R52 2(for IMP).<br>0: Disable<br>1: Enable |
| 24 | CRIMP_DBGEN | *1 | R | Indicates disable or enable of the DBGEN of Cortex-R52 2(for IMP).<br>0: Disable<br>1: Enable |
| 23 | PAP_NIDEN | *1 | R | Indicates disable or enable of the NIDEN of PAP.<br>0: Disable<br>1: Enable |
| 22 | PAP_DBGEN | *1 | R | Indicates disable or enable of the DBGEN of PAP.<br>0: Disable<br>1: Enable |
| 21 | DTA_NIDEN | *1 | R | Indicates disable or enable of the NIDEN of DTA.<br>0: Disable<br>1: Enable |
| 20 | DTA_DBGEN | *1 | R | Indicates disable or enable of the DBGEN of DTA.<br>0: Disable |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| | | | | 1: Enable |
| 19 | VDSP_SPNIDEN | *1 | R | Indicates disable or enable of the SPNIDEN of VDSP.<br>0: Disable<br>1: Enable |
| 18 | VDSP_SPIDEN | *1 | R | Indicates disable or enable of the SPIDEN of VDSP.<br>0: Disable<br>1: Enable |
| 17 | VDSP_NIDEN | *1 | R | Indicates disable or enable of the NIDEN of VDSP.<br>0: Disable<br>1: Enable |
| 16 | VDSP_DBGEN | *1 | R | Indicates disable or enable of the DBGEN of VDSP.<br>0: Disable<br>1: Enable |
| 15 to 13 | - | All 0 | R | Reserved<br>These bits are always read as 0. |
| 12 | CS_DEVEN | *1 | R | Indicates disable or enable of the DAP_DEVICEEN of CoreSight.<br>0: Disable<br>1: Enable |
| 11 | CS_SPNIDEN | *1 | R | Indicates disable or enable of the SPNIDEN of CoreSight.<br>0: Disable<br>1: Enable |
| 10 | CS_SPIDEN | *1 | R | Indicates disable or enable of the SPIDEN of CoreSight.<br>0: Disable<br>1: Enable |
| 9 | CS_NIDEN | *1 | R | Indicates disable or enable of the NIDEN of CoreSight.<br>0: Disable<br>1: Enable |
| 8 | CS_DBGEN | *1 | R | Indicates disable or enable of the DBGEN of CoreSight.<br>0: Disable<br>1: Enable |
| 7 | CR_HNIDEN | *1 | R | Indicates disable or enable of the HNIDEN of Cortex-R52 0, Cortex-R52 1.<br>0: Disable<br>1: Enable |
| 6 | CR_HIDEN | *1 | R | Indicates disable or enable of the HIDEN of Cortex-R52 0, Cortex-R52 1.<br>0: Disable<br>1: Enable |
| 5 | CR_NIDEN | *1 | R | Indicates disable or enable of the NIDEN of Cortex-R52 0, Cortex-R52 1.<br>0: Disable<br>1: Enable |
| 4 | CR_DBGEN | *1 | R | Indicates disable or enable of the DBGEN of Cortex-R52 0, Cortex-R52 1.<br>0: Disable<br>1: Enable |
| 3 | CA_SPNIDEN | *1 | R | Indicates disable or enable of the SPNIDEN of Cortex-A76. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| | | | | 0: Disable<br>1: Enable |
| 2 | CA_SPIDEN | *1 | R | Indicates disable or enable of the SPIDEN of Cortex-A76.<br>0: Disable<br>1: Enable |
| 1 | CA_NIDEN | *1 | R | Indicates disable or enable of the NIDEN of Cortex-A76.<br>0: Disable<br>1: Enable |
| 0 | CA_DBGEN | *1 | R | Indicates disable or enable of the DBGEN of Cortex-A76.<br>0: Disable<br>1: Enable |

Note: *1 Depends on the setting of the MD21, MD20, MD11,MD10,MDT[1:0] pins and Security state.

### 170.2.20 Debug Register 9 (DBGREG9)

This register has a key role in preventing invalid access. It is possible to perform read/write operations only from the CPU. DBGREG2 can only be written when the KEY bit in DBGREG9 is updated as below procedure.

First, write H'A500 to DBGREG9[15:0]. This enables to write to DBGREG9[0]. Then, write H'A501 to DBGREG9[15:0]. After this writing is done, DBGREG2 can be written.

It is recommended to clear the KEY bit in DBGREG9 after finishing writing to DBGREG2. To clear KEY bit in DBGREG9, write H'0000_0000 to DBGREG9.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | AID[7:0] | | | | | | | | - | - | - | - | - | - | - | KEY |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 8 | AID[7:0] | H'00 | R/W | KEY Bit Write Aid<br>Use this bit to write a bit pattern (H'A5) to be written to the KEY bit. |
| 7 to 1 | - | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | KEY | 0 | R/W | DBGREG2 access control<br>Provides access control to the DBGREG2.<br>0: It is not possible to perform write operations to the DBGREG2.<br>1: It is possible to perform write operations to the DBGREG2.<br>When writing 1 to this bit, be sure to write H'A5 to the AID bits at the same time. |

## 170.3 Operation

### 170.3.1 JTAG Selection

When LCS is not Secure LCS, both Main-JTAG and Sub-JTAG can be selected through MD pins.

-: No connection with TAP controller

*1: Don't care

**Table 170.7 JTAG selection table (LCS is NOT equal to Secure LCS)**

| MDT0 | MD21 | MD20 | MD11 | MD10 | Main-JTAG | Sub-JTAG |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | - | Normal Function |
| | | | | 1 | Reserved | Reserved |
| | | | 1 | 0 | ICUMX JTAG | Normal Function |
| | | | | 1 | ICUMX LPD | Normal Function |
| | | 1 | 0 | 0 | ICUMX JTAG | CoreSight |
| | | | | 1 | ICUMX LPD | CoreSight |
| | | | 1 | *1 | Reserved | Reserved |
| | 1 | 0 | 0 | 0 | CoreSight | Normal Function |
| | | | | 1 | Reserved | Reserved |
| | | | 1 | *1 | Reserved | Reserved |
| | | 1 | *1 | *1 | Reserved | Reserved |
| 1 | 0 | 0 | 0 | 0 | VDSP0 | Normal Function |
| | | | | 1 | VDSP1 | Normal Function |
| | | | 1 | 0 | Reserved | Reserved |
| | | | | 1 | Reserved | Reserved |
| | | 1 | 0 | 0 | CoreSight | VDSP0 |
| | | | | 1 | CoreSight | VDSP1 |
| | | | 1 | 0 | Reserved | Reserved |
| | | | | 1 | Reserved | Reserved |
| | 1 | 0 | 0 | 0 | PAP | Normal Function |
| | | | | 1 | Reserved | Reserved |
| | | | 1 | *1 | Reserved | Reserved |
| | | 1 | 0 | 0 | CoreSight | PAP |
| | | | | 1 | Reserved | Reserved |
| | | | 1 | *1 | Reserved | Reserved |

When LCS is Secure LCS, Main-JTAG can be selected through MD pins and debug authentication is performed via Main-JTAG.

-: No connection with TAP controller

*1: Don't care

*2: Sub-JTAG is disconnected during the authentication. After the authentication, Sub-JTAG is connected and internal debug authentication signals are asserted.

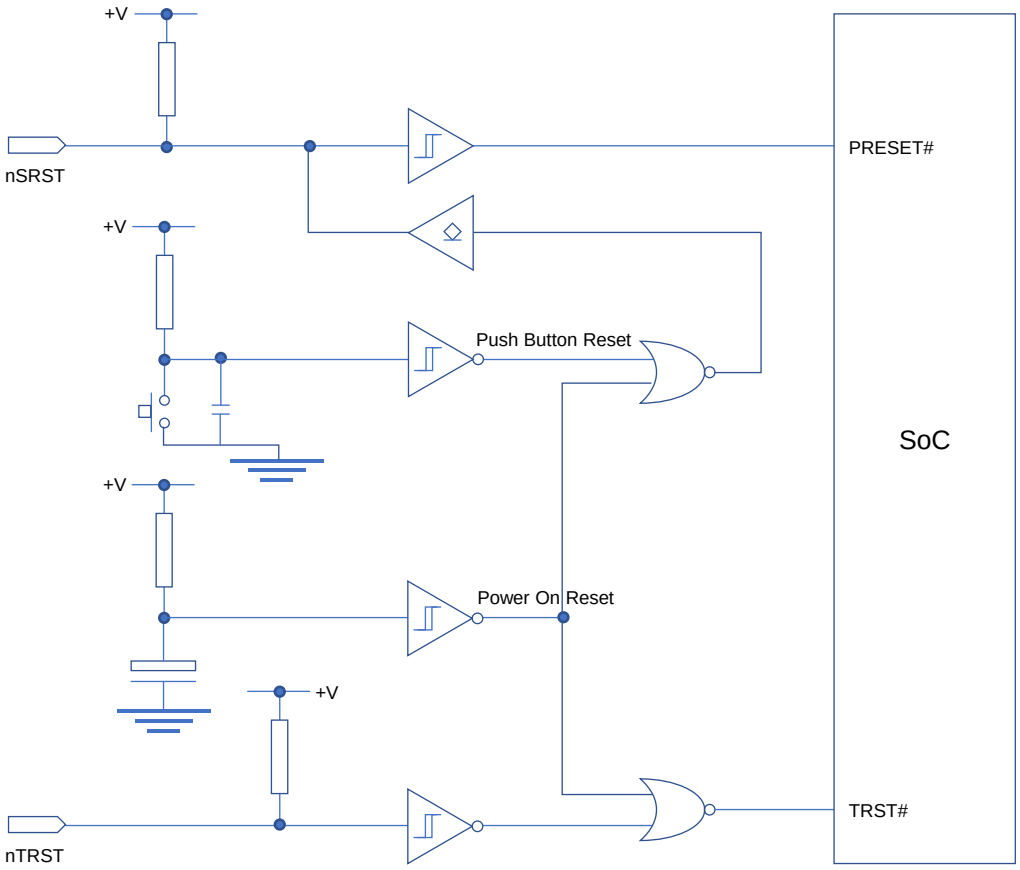
*3: VDSPs and PAP don't have authentication register. They can't be connected to main-JTAG.

**Table 170.8 JTAG selection table (LCS is equal to Secure LCS)**

| MDT0 | MD21 | MD20 | MD11 | MD10 | Main-JTAG | Sub-JTAG |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | - | Normal Function |
| | | | | 1 | Reserved | Reserved |
| | | | 1 | 0 | ICUMX JTAG | Normal Function |
| | | | | 1 | ICUMX LPD | Normal Function |
| | | 1 | 0 | 0 | ICUMX JTAG | *2 |
| | | | | 1 | ICUMX LPD | *2 |
| | | | 1 | *1 | Reserved | Reserved |
| | 1 | 0 | 0 | 0 | CoreSight | Normal Function |
| | | | | 1 | Reserved | Reserved |
| | | | 1 | *1 | Reserved | Reserved |
| | | 1 | *1 | *1 | Reserved | Reserved |
| 1 | 0 | 0 | 0 | *1 | *3 | Reserved |
| | | | 1 | *1 | Reserved | Reserved |
| | | 1 | 0 | 0 | CoreSight | *2 |
| | | | | 1 | CoreSight | *2 |
| | | | 1 | *1 | Reserved | Reserved |
| | 1 | 0 | 0 | 0 | *3 | Reserved |
| | | | | 1 | Reserved | Reserved |
| | | | 1 | *1 | Reserved | Reserved |
| | | 1 | 0 | 0 | CoreSight | *2 |
| | | | | 1 | Reserved | Reserved |
| | | | 1 | *1 | Reserved | Reserved |

### 170.3.2 JTAG Reset Circuit Logic

When designing a target hardware on which the debugger can be used, a pull-up resistor must be included on both PRESET# and TRST# signals. Apply a low level to TRST# at power-on.




Example Reset Circuit Logic

The table shows the pin processing for TRST# pin on both Main-JTAG and Sub-JTAG.

| MDT0 | MD21 | MD20 | MD11 | MD10 | Main-JTAG | Sub-JTAG | Debugger connection required or not | TRST# Pin Processing | Debugger connection required or not | TRST# Pin Processing |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Main-JTAG | Main-JTAG | Sub-JTAG | Sub-JTAG |
| 0 | 0 | 0 | 0 | 0 | - | Normal Function | Do not connect | PU, PD or OPEN *3 | Do not connect | *4 |
| | | | | 1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | | | 1 | 0 | ICUMX JTAG | Normal Function | Connect | *2 | Do not connect | *4 |
| | | | | 1 | ICUMX LPD | Normal Function | Connect | *2 | Do not connect | *4 |
| | | 1 | 0 | 0 | ICUMX JTAG | CoreSight | Connect | *2 | Connect | *2 |
| | | | | 1 | ICUMX LPD | CoreSight | Connect | *2 | Connect | *2 |
| | | | 1 | *1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | 1 | 0 | 0 | 0 | CoreSight | Normal Function | Connect | *2 | Do not connect | *4 |
| | | | | 1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | | | 1 | *1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | | 1 | *1 | *1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| 1 | 0 | 0 | 0 | 0 | VDSP0 | Normal Function | Connect | *2 | Do not connect | *4 |
| | | | | 1 | VDSP1 | Normal Function | Connect | *2 | Do not connect | *4 |
| | | | 1 | *1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | | 1 | 0 | 0 | CoreSight | VDSP0 | Connect | *2 | Connect | *2 |
| | | | | 1 | CoreSight | VDSP1 | Connect | *2 | Connect | *2 |
| | | | 1 | *1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | 1 | 0 | 0 | 0 | PAP | Normal Function | Connect | *2 | Do not connect | *4 |
| | | | | 1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | | | 1 | *1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | | 1 | 0 | 0 | CoreSight | PAP | Connect | *2 | Connect | *2 |
| | | | | 1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |
| | | | 1 | *1 | Reserved | Reserved | prohibited MD setting | prohibited MD setting | prohibited MD setting | prohibited MD setting |

*1: Don't care

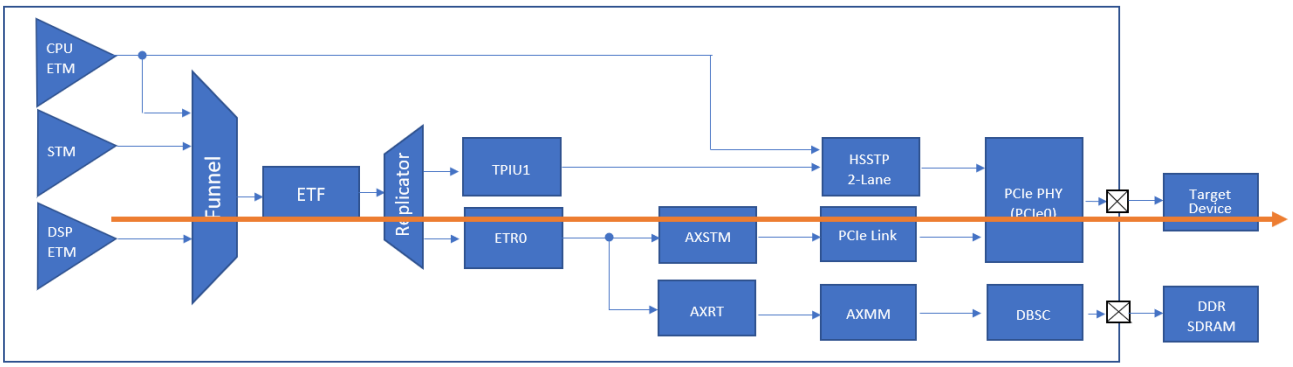
*2: Follow the example of reset circuit logic
*3: When OPEN, enable the built-in Pull Up.
*4: Follow processing as a functional pin

### 170.3.3 Trace Output Selection

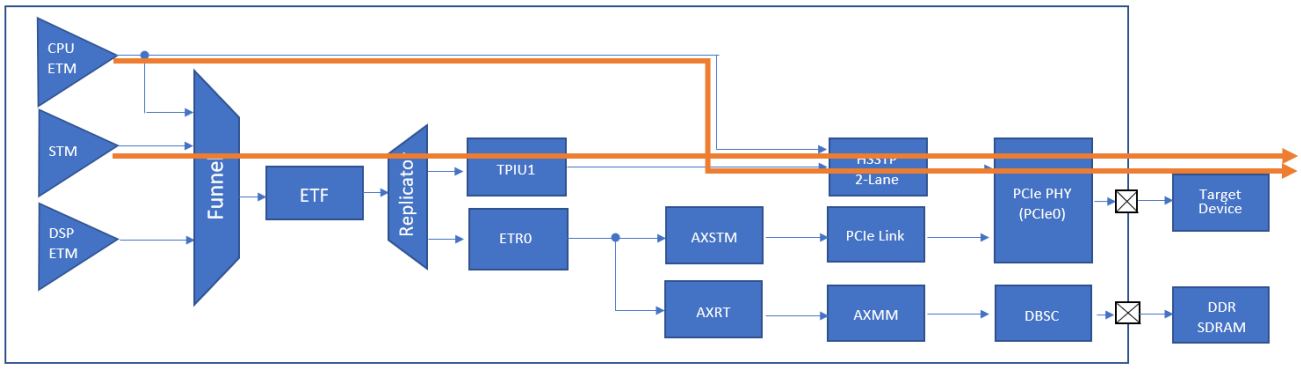
The supported 3 trace output port possibilities are summarized. The 1x trace can be selected from PCIe trace or HSSTP trace. See also Section 170.3.4 HSSTP PCIE channel selection for HSSTP trace.

[1x Trace]

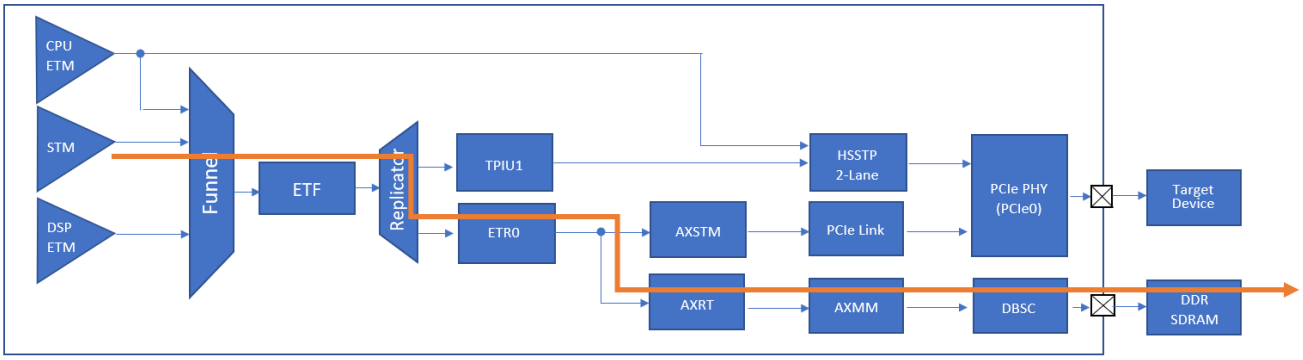
- 32Gbps or 16Gbps PCIe 2-lane Trace
  (1) CSDBGSS/ETR0 -> PCIe Link/PHY -> PCIe0
- 10Gbps or 5Gbps HSSTP 2-lane Trace
  (2) CSDBGSS/TPIU1 -> HSSTP Link/PCIe PHY -> PCIe0
  CSDBGCA0/TPIU0 -> HSSTP Link/PCIe PHY -> PCIe0
  CSDBGCR0/TPIU0 -> HSSTP Link/PCIe PHY -> PCIe0
  CSDBGCR1/TPIU0 -> HSSTP Link/PCIe PHY -> PCIe0
  CSDBGCR2/TPIU0 -> HSSTP Link/PCIe PHY -> PCIe0
- Trace capture using external DRAM via ETR
  (3) CSDBGSS/ETR0 -> DBSC -> DDR SDRAM




(1) PCIe 2-lane Trace (PCIe0)




(2) HSSTP 2-lane Trace (PCIe0)




(3) Trace capture using external DRAM via ETR

**Figure 170.3 Trace Path**

### 170.3.4 STM Master ID Mapping

STM-500 is used for system trace to output some information from CPU and monitor the contents stored in memories such as DDR memories and internal SRAMs by utilizing DMA transfer. Following table lists of Master ID value (output of STM) to identify the master module sending the transaction to STM-500 in each product.

STM0 is used by the masters which can access to STM0 via AXRT (AXI bus of RT hierarchy). STM1 is dedicated for Vision IP.

**Table 170.9 STM Master ID Mapping**

| Master Name | STM Number | Secure Access | Master_ID |
|---|---|---|---|
| Cortex-R52 SS0 | STM0 | Secure | H’03 |
| Cortex-R52 SS1 | STM0 | Secure | H’04 |
| Cortex-R52 SS2 | STM0 | Secure | H’05 |
| Cortex-A76 Core#0_0 | STM0 | Secure | H’08 |
| Cortex-A76 Core#0_1 | STM0 | Secure | H’09 |
| Cortex-A76 Core#0_2 | STM0 | Secure | H’0A |
| Cortex-A76 Core#0_3 | STM0 | Secure | H’0B |
| SYS-DMAC1 PER/RT-DMAC0 PER | STM0 | Secure | H’31 |
| SYS-DMAC2 PER/RT-DMAC1 PER | STM0 | Secure | H’34 |
| RT-DMAC2 PER | STM0 | Secure | H’37 |
| RT-DMAC3 PER | STM0 | Secure | H’3A |
| Cortex-R52 SS0 | STM0 | Non-secure | H’03 |
| Cortex-R52 SS1 | STM0 | Non-secure | H’04 |
| Cortex-R52 SS2 | STM0 | Non-secure | H’05 |
| Cortex-A76 Core#0_0 | STM0 | Non-secure | H’08 |
| Cortex-A76 Core#0_1 | STM0 | Non-secure | H’09 |
| Cortex-A76 Core#0_2 | STM0 | Non-secure | H’0A |
| Cortex-A76 Core#0_3 | STM0 | Non-secure | H’0B |
| IMP0/IMP1 #0 | STM0 | Non-secure | H’10 |
| IMP0/IMP1 #1 | STM0 | Non-secure | H’11 |
| IMP0/IMP1 #2 | STM0 | Non-secure | H’12 |
| IMP0/IMP1 #3 | STM0 | Non-secure | H’13 |
| IMP0/IMP1 #4 | STM0 | Non-secure | H’14 |

| | | | |
|---|---|---|---|
| IMP0/IMP1 #5 | STM0 | Non-secure | H’15 |
| IMP0/IMP1 #6 | STM0 | Non-secure | H’16 |
| IMP0/IMP1 #7 | STM0 | Non-secure | H’17 |
| IMP0/IMP1 #8 | STM0 | Non-secure | H’18 |
| IMP0/IMP1 #9 | STM0 | Non-secure | H’19 |
| UMFL0 | STM1 | Non-secure | H’20 |
| SMPS0 | STM1 | Non-secure | H’22 |
| SMPO0 | STM1 | Non-secure | H’24 |
| SYS-DMAC1 PER/RT-DMAC0 PER | STM0 | Non-secure | H’31 |
| SYS-DMAC2 PER/RT-DMAC1 PER | STM0 | Non-secure | H’34 |
| RT-DMAC2 PER | STM0 | Non-secure | H’37 |
| RT-DMAC3 PER | STM0 | Non-secure | H’3A |

Note : *1: SYS-DMAC and RT-DMAC are not distinguishable. Either one DMA Controller is expected to use at the same time.

### 170.3.5 HSSTP PCIE Channel Selection

DBGREG2.DMON2EN can select which PCI-express channel via 2-lane HSSTP is used.

### 170.3.6 Cross Trigger Connection

This section shows the cross trigger connections.

**Table 170.10 CSD-CTI0 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | ETF0 FULL | - |
| | | 1 | ETF0 ACQCOMP | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | STM4 TRIGOUTSPTE | - |
| | | 5 | STM4 TRIGOUTSW | - |
| | | 6 | STM4 TRIGOUTHETE | - |
| | | 7 | STM4 ASYNCOUT | - |
| | | | Out | outack |
| | out | 0 | ETF0 FLUSHIN | - |
| | | 1 | ETF0 TRIGIN | - |
| | | 2 | ETR0 FLUSHIN | - |
| | | 3 | ETR0 TRIGIN | - |
| | | 4 | TPIU0 FLUSHIN | TPIU0 FLUSHINACK |
| | | 5 | TPIU0 TRIGIN | TPIU0 TRIGINACK |
| | | 6 | STM4 HWEVENT[0], ~STM4 HWEVENT[1] | - |
| | | 7 | STM4 HWEVENT[2], ~STM4 HWEVENT[3] | - |

-: no use
~: inverted

**Table 170.11 CSD-CTI1 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | ETF1 FULL | - |
| | | 1 | ETF1 ACQCOMP | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |
| | | | Out | outack |
| | out | 0 | ETF1 FLUSHIN | - |
| | | 1 | ETF1 TRIGIN | - |
| | | 2 | ETR1 FLUSHIN | - |
| | | 3 | ETR1 TRIGIN | - |
| | | 4 | TPIU1 FLUSHIN | TPIU1 FLUSHINACK |
| | | 5 | TPIU1 TRIGIN | TPIU1 TRIGINACK |
| | | 6 | - | - |
| | | 7 | - | - |

-: no use

**Table 170.12 CSD-CTI2 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | TRIGOUTSPTE | - |
| | | 1 | TRIGOUTSW | - |
| | | 2 | TRIGOUTHETE | - |
| | | 3 | ASYNCOUT | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |
| | | | Out | outack |
| | out | 0 | - | - |
| | | 1 | - | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |

-: no use

**Table 170.13 CSD-CTI3 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | STM0 TRIGOUTSPTE | - |
| | | 1 | STM0 TRIGOUTSW | - |
| | | 2 | STM0 TRIGOUTHETE | - |
| | | 3 | STM0 ASYNCOUT | - |
| | | 4 | STM1 TRIGOUTSPTE | - |
| | | 5 | STM1 TRIGOUTSW | - |
| | | 6 | STM1 TRIGOUTHETE | - |
| | | 7 | STM1 ASYNCOUT | - |
| | | | Out | outack |
| | out | 0 | STM0 HWEVENT[0], ~STM0 HWEVENT[1] | - |
| | | 1 | STM0 HWEVENT[2], ~STM0 HWEVENT[3] | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | STM1 HWEVENT[0], ~STM1 HWEVENT[1] | - |
| | | 5 | STM1 HWEVENT[2], ~STM1 HWEVENT[3] | - |
| | | 6 | - | - |
| | | 7 | - | - |

-: no use

**Table 170.14 CSD-CTI4 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | STM2 TRIGOUTSPTE | - |
| | | 1 | STM2 TRIGOUTSW | - |
| | | 2 | STM2 TRIGOUTHETE | - |
| | | 3 | STM2 ASYNCOUT | - |
| | | 4 | STM3 TRIGOUTSPTE | - |
| | | 5 | STM3 TRIGOUTSW | - |
| | | 6 | STM3 TRIGOUTHETE | - |
| | | 7 | STM3 ASYNCOUT | - |
| | | | Out | outack |
| | out | 0 | STM2 HWEVENT[0], ~STM2 HWEVENT[1] | - |
| | | 1 | STM2 HWEVENT[2], ~STM2 HWEVENT[3] | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | STM3 HWEVENT[0], ~STM3 HWEVENT[1] | - |
| | | 5 | STM3 HWEVENT[2], ~STM3 HWEVENT[3] | - |
| | | 6 | - | - |
| | | 7 | - | - |

-: no use

**Table 170.15 CSD-CTI5 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | - | - |
| | | 1 | mtirt0_common_ovf_err | - |
| | | 2 | mtirt1_common_ovf_err | - |
| | | 3 | - | - |
| | | 4 | - | - |
| | | 5 | icumx_haltreq_csd | csd_haltack_icumx |
| | | 6 | icumx_restartreq_csd | csd_restartack_icumx |
| | | 7 | TDBG tg_ctiack_cs | - |
| | | | Out | outack |
| | out | 0 | - | - |
| | | 1 | - | - |
| | | 2 | cs_rtirq_zt | - |
| | | 3 | - | - |
| | | 4 | - | - |
| | | 5 | csd_stopreq_icumx | icumx_stopack_csd |
| | | 6 | csd_restartreq_icumx | icumx_restartack_csd |
| | | 7 | TDBG cs_ctireq_tg | - |

-: no use

**Table 170.16 E0CA-CTI0 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | - | - |
| | | 1 | - | - |
| | | 2 | ETF0 FULL | - |
| | | 3 | ETF0 ACQCOMP | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |
| | | | Out | outack |
| | out | 0 | ETF0 FLUSHIN | - |
| | | 1 | ETF0 TRIGIN | - |
| | | 2 | TPIU0 FLUSHIN | TPIU0 FLUSHINACK |
| | | 3 | TPIU0 TRIGIN | TPIU0 TRIGINACK |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |

-: no use

**Table 170.17 R0/1/2CR-CTI0 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | - | - |
| | | 1 | - | - |
| | | 2 | ETF0 FULL | - |
| | | 3 | ETF0 ACQCOMP | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |
| | | | Out | outack |
| | out | 0 | ETF0 FLUSHIN | - |
| | | 1 | ETF0 TRIGIN | - |
| | | 2 | TPIU0 FLUSHIN | TPIU0 FLUSHINACK |
| | | 3 | TPIU0 TRIGIN | TPIU0 TRIGINACK |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |

-: no use

**Table 170.18 DTA-CTI0 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | HALTED | csd_halted_dta |
| | | 1 | - | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | Cross-Hold Trigger Event | csd_cross_hold_dta |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |
| | | | Out | outack |
| | out | 0 | Debug Request Trigger Event | dta_debug_request_csd |
| | | 1 | - | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | Restart Request Trigger Event | dta_restarted_csd |

-: no use

**Table 170.19 VDSP XM-CTI0 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | In | inack |
| | in | 0 | cevaxm8_ocm_debug_noss_r | - |
| | | 1 | cevaxm8_opa1_mtch_level_r | cti_pa_mtch_ack |
| | | 2 | cevaxm8_obpint_trig_level_r[4] | cti_trig_ack[4] |
| | | 3 | cevaxm8_obpint_trig_level_r[3] | cti_trig_ack[3] |
| | | 4 | cevaxm8_obpint_trig_level_r[2] | cti_trig_ack[2] |
| | | 5 | cevaxm8_obpint_trig_level_r[1] | cti_trig_ack[1] |
| | | 6 | cevaxm8_obpint_trig_level_r[0] | cti_trig_ack[0] |
| | | 7 | cevaxm8_etm_trigger | etm_trigger_ack |
| | | | Out | outack |
| | out | 0 | ext_bp1_req | cevaxm8_ocm_ext1_ack_r |
| | | 1 | ext_bp2_req | cevaxm8_ocm_ext2_ack_r |
| | | 2 | ext_exit_debug | ~cevaxm8_ocm_debug_noss_r |
| | | 3 | - | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |

-: no use
~: inverted

**Table 170.20 VDSP XM-CTI1 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | - | - |
| | | 1 | - | - |
| | | 2 | ETF FULL | - |
| | | 3 | ETF ACQCOMP | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |
| | | | Out | outack |
| | out | 0 | ETF FLUSHIN | - |
| | | 1 | ETF TRIGIN | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |

-: no use

**Table 170.21 PAP-CTI0 Connections**

| IF | Port | | Wire | |
|---|---|---|---|---|
| Trigger IF | | | in | inack |
| | in | 0 | dbg_halt_outreq | dbg_halt_outreqack |
| | | 1 | - | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | dbg_resume_outreqCross-Hold Trigger Event | dbg_resume_outreqack |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |
| | | | Out | outack |
| | out | 0 | dbg_halt_outreq | dbg_halt_outreqack |
| | | 1 | - | - |
| | | 2 | - | - |
| | | 3 | - | - |
| | | 4 | - | - |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | dbg_resume_outreqRestart Request Trigger Event | dbg_resume_outreqack |

-: no use

### 170.3.7 ATB Funnel Information

The CoreSight infrastructure's ATB funnel ports are connected as in the following table;

**Table 170.22 ATB Funnel Port connection**

| Module | | Connection | |
|---|---|---|---|
| | | ATB Funnel Port | Abbreviation |
| CSDBGSS | ATB Funnel0 | Port0 | Funnel2 |
| | | Port1 | CSD-STM1 |
| | | Port2 | CSD-STM2 |
| | | Port3 | CSD-STM3 |
| | ATB Funnel1 | Port0 | Funnel2 |
| | | Port1 | CSD-STM1 |
| | | Port2 | CSD-STM2 |
| | | Port3 | CSD-STM3 |
| | ATB Funnel2 | Port0 | Funnel3 |
| | | Port1 | CSD-STM4 |
| | | Port2 | CR52SS0 |
| | | Port3 | CA76SS0 |
| | | Port4 | - |
| | | Port5 | CSD-STM0 |
| | | Port6 | CR52SS1 |
| | | Port7 | CR52SS2 |
| | ATB Funnel3 | Port0 | VDSP0 |
| | | Port1 | VDSP1 |
| CA76SS CSDBGCA | | Port0 | CPU0 |
| | | Port1 | CPU1 |
| | | Port2 | CPU2 |
| | | Port3 | CPU3 |
| CR52SS CSDBGCR | | Port0 | - |
| | | Port1 | - |
| | | Port2 | - |
| | | Port3 | - |
| | | Port4 | Instruction trace stream |
| | | Port5 | Data trace stream |

-: no use

### 170.3.8 AXI Trace Monitor

The AXI transactions issued by the master IPs that access the DDR main memory always pass through AXMM, which is one of AXI bus components in R-Car V4H. The AXMM incorporates a monitor function which can trace AXI write transactions. Address and data can be traced w/wo timestamp information. Enabling/disabling timestamp can be selected by register setting. The AXMM duplicates the AXI write transactions and diverts them to the PCIe trace output for debugging purpose.

The AXI trace monitor consists of AXI_Trace_Guard, AXI_Trace_W, RAM and AXI_Trace_R. The AXI_Trace_Guard controls acknowledge signal not to stall the accesses for DDR main memory excessively. The following two different modes are provided and either of them can be selected by register setting;

1. The AXI trace is suspended not to affect the accesses for DDR main memory when trace path is stalled (ex. RAM/BUF full due to PCIe not responding).
2. The AXI trace isn't suspended even if trace path is stalled. This mode does affect the accesses for DDR main memory (i.e. invasive to application behavior).

The AXI_Trace_W writes the trace data into RAM. The AXI_Trace_R reads the trace data from RAM, packetizes it and transmits it out towards the PCIe trace output. The AXI_Trace_W and the AXI_Trace_R provide the same function as the Arm's CoreSight components: STM and TMC (ETR). The AXI trace monitor generates trace packets and formats them. The trace packets which conform to MIPI System Trace Protocol version 2 (STPv2) are formatted into 16-byte frames (CoreSight Trace Formatter).

- RAM size (intermediate trace memory): 512-bit * 128-word
- Buffer size (buffer for AXI transaction output stage): 256-byte

The AXMM can filter the AXI write transactions by trace source and write address. The filtering function is divided into the two parts. One (pre-filter) is implemented in the access protection block and the other (post_filiter) in AXI_Trace_W.

The registers STMPFLTM0-7 having 256-bits in total are used to enable/disable this trace monitor for each AXI master. 256 different SRC_IDs are assigned to all AXI masters in R-Car V4H. The pre-filtering functions based on Region ID/Sec ID (STMPFLTRS) and DRAM/SYSRAM area (STMPFLTDA/STMPFLTSA) are provided as well. The registers STMFLT0-15 are used to set the 16 different post filtering conditions.

**Types of Pre-filters**

**1. SRC_ID Filter**

The registers STMPFLTM0-7 having 256-bits in total are used to enable/disable this trace monitor for each AXI master. 256 different SRC_IDs are assigned to all AXI masters in R-Car V4H. The assignment of SRC_IDs is described in the SoC bus section of this TS document.

As an example, here is the STMPFLTM0-7 register setting to enable this trace monitor for the AXI masters having SRC_ID = 1 and 32:
STMPFLTM0 = 0x0000_0002
STMPFLTM1 = 0x0000_0001
STMPFLTM2-7 = 0x0000_0000

**2. Access Authentication Filter**

The pre-filtering function based on Region ID/Sec ID is provided as well. The STMPFLTRS[15:0] bit field corresponds to 16 Region IDs and STMPFLTRS[19:16] to 4 Sec IDs. For each Region ID/Sec ID, this trace monitor can be enabled/disabled.

**3. System RAM Filter**

The pre-filtering function based on SYSRAM area is also provided. The STMPFLTSA[15:0] bit field corresponds to 16 access protect areas of System RAM. For each protect area of System RAM, this trace monitor can be enabled/disabled.

**4. DRAM Filter**

The pre-filtering function based on DRAM area is provided as well. The number of the protect areas for DRAM is

64. STMPFLTDAH is used for the protected areas 63-32 and STMPFLTDAL for the protected areas 31-0. For each protect area of DRAM, this trace monitor can be enabled/disabled.

The AXI_Trace_Guard has the control registers which can be accessed from CPU. Those registers can be accessed from DEBUG_APB ('XMON_STM0' in CoreSight address map) only when a control bit is set (STMFLTIDR/DBGALLOW = 1). The AXI_Trace_W has the control registers (REG_STM) which can be accessed from DEBUG_APB ('XMON_STM0' in CoreSight address map). The AXI_Trace_R also has the control register (REG_TMC) which can be accessed from DEBUG_APB ('XMON_CTRL0' in CoreSight address map).

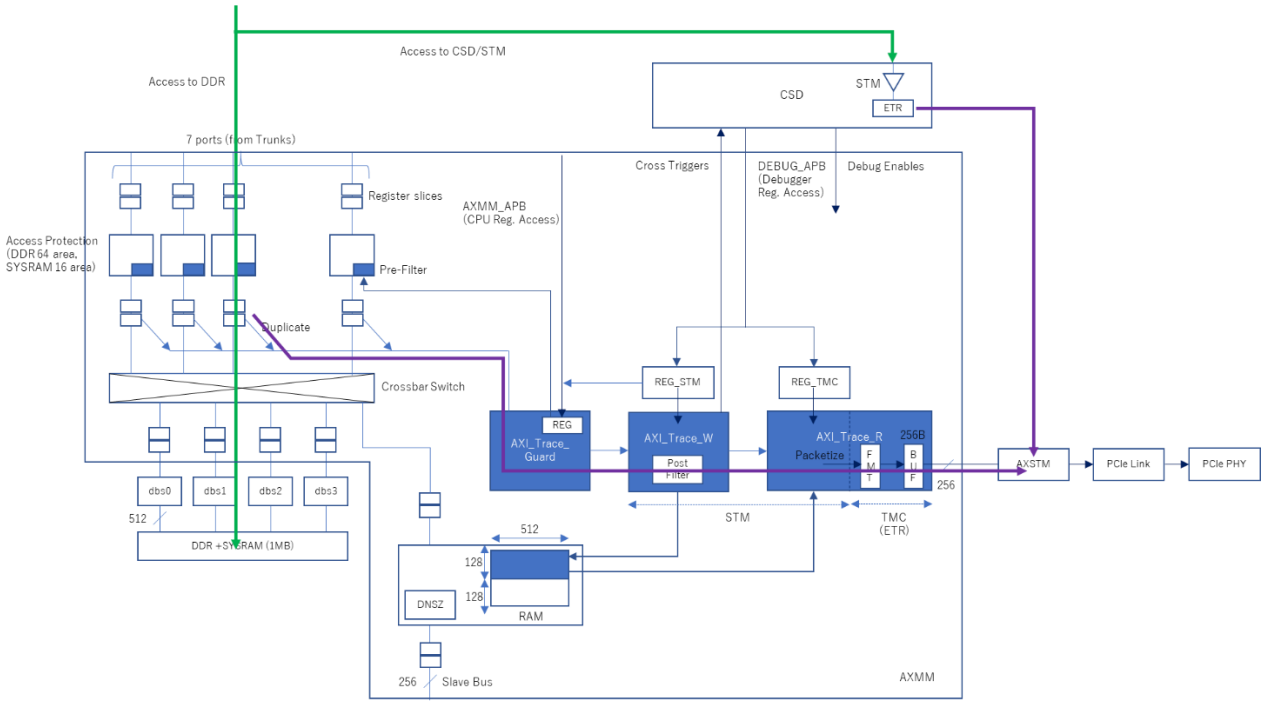



**Figure 170.4 AXI Trace Monitor in AXMM**

**[Example of Initial Register Setting]**

The guard control, STM/TMC control and filtering control registers need to be set before enabling STM and TMC. The unlocking is required in prior to register setting.

1. Unlock Control Register Access
   DBGALLOW bit in STMFLTIDR
   LAR for STM and TMC

2. Set Guard Control Registers
   STMGCTR0 to control the invasion for AXI bus
   STMGCTR1 for AWREADY negate control

3. Set STM/TMC Control Registers
   STMAUXCR for STM FIFO auto-flush control
   STMTCSR to specify TRACE ID
   RSZ (RAM Size register)
   DBA (Data Buffer Address register)
   RWP (RAM Write Pointer register)
   AXICTL for AWCACHE/AWPROT control bits
   FFCR to enable formatting

4. Set Filtering Control Registers
   Prefiltering
      Master ID prefiltering (STMPFLTM)
      Region ID and Sec ID prefiltering (STMPFLTRS)
      DRAM Area prefiltering (STMPFLTDA)
      SYSRAM Area prefiltering (STMPFLTSA)
   Post filtering (STMFLT)

5. Enable STM/TMC
   STMTCSR to enable STM
   CTL to enable TMC trace capture

## 170.4 Usage Notes

### 170.4.1 TRST control in no JTAG selection

Refer to 170.3.2 JTAG Reset Circuit Logic.

### 170.4.2 Pseudo Power Down

Pseudo Power down is not supported at debugging.

### 170.4.3 Warm Reset Request from Cortex-A76

There is a register bit to request a warm reset against Cortex-A76 (CWRR, bit[1] in EDPRCR). But a bit setting by external debugger doesn't have any effect in R-Car V4H and Cortex-A76 warm reset doesn't take place.

### 170.4.4 Debug & Trace on VDSP Power Shut Off

Debugger isn't allowed to access VDSP when VDSP is powered off or under reset state. Power-shut-off isn't allowed when VDSP debug and trace function is in use.

### 170.4.5 Software Reset

The software reset for CoreSight Debug, by setting bit 30 in SFTRST register No. 3, can't be used in user and in debug modes.

### 170.4.6 Debug & Trace Continuity after NSI Reset

The debug and trace function can't be continued after NSI reset except for Cortex-A76/Cortex-R52.

### 170.4.7 Debug & Trace Continuity after Power-on Reset

When a power-on reset is asserted during Cortex-R52's debug and trace, the Cortex- R52's debug and trace can't be continued after power-on reset. This is because many of Cortex- R52's debug resources are powered off during power-on reset. But setting of Reset Vector Break can be preserved. So, re-setting of previous debugger setting is possible after Reset Vector Break.

When a power-on reset is asserted during Cortex-A76's debug and trace, the Cortex-A76's debug and trace can't be continued after power-on reset. This is because all of Cortex-A76's debug resources are powered off during power-on reset. Reset Vector Break is not supported for Cortex-A76 because the core can't be selected as a master boot processor in R-Car V4M.

### 170.4.8 Debug Resource Access Restriction

Any operation to debug resource of Cortex-R52, except for setting of Reset Vector Break, is prohibited while the core is powered off.

Any operation to debug resource of Cortex-A76 is prohibited while the core is powered off or in Runtime Test.

### 170.4.9 Access to debug resource of Cortex-R52 in core standby mode

When Cortex-R52 processor is in ‘core standby mode’, clock supply to the debug resource of Cortex-R52 is stopped. Any access to the debug resource of Cortex-R52 from external debugger or any other processor is prohibited. For example, when ICUMX processor is selected as a master boot processor, Cortex-R52 processors (core 1, core 2) are in ‘core standby mode’ unless booted by another processor. The reset vector catch setting for Cortex-R52 processor (core 0) is possible when Cortex-R52 processor is selected as a master boot processor.

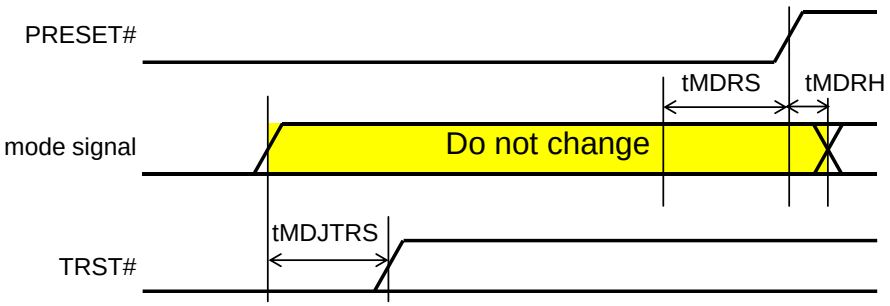
### 170.4.10 Applying a low level to TRST# at power-on

Apply a low level to TRST# at power-on. However, it is not necessary to apply a low level in the following cases.

- For Main-JTAG TRST# when VDSP0 or VDSP1 is connected to Main-JTAG (when (MDT0, MD21, MD20, MD11, MD10) = (10000), (10001))
- For Sub-JTAG TRST# when VDSP0 or VDSP1 is connected to Sub-JTAG (when (MDT0, MD21, MD20, MD11, MD10) = (10100), (10101))

### 170.4.11 Mode Signal Timing Requirements for TRST# and PRESET#

When TRST# is deasserted before PRESET# is deasserted, make sure that the mode signal value does not change during the period from when TRST# is deasserted to when PRESET# is deasserted.
The setup time of mode signal for TRST# deassert (tMDJTRS) is the same as tMDRS. For tMDRS, refer to chapter 200. Electrical Characteristics.



## 170.5 Safety Mechanisms

Nothing special to describe for Debug and Trace as it is a non-safety related module.

# 171. Power On Self Test (POST) and Runtime Test

## 171.1 Power On Self Test (POST)

### 171.1.1 Overview

The POST is a safety mechanism to detect faults before the SoC is used. This safety mechanism is executed only once when power is supplied to the SoC. The POST is performed using the field BIST[*1] modules.
The logic and memory built-in self tests (BISTs) are executed consecutively on the specified hierarchies within the POST operation. The memory BIST is applied to all RAMs. The order of tests and setting values for tests are written on the ROM, and the tests are executed according to the contents of the ROM.

Note *1: The field BIST is a collective term for the POST and Runtime Test.

### 171.1.2 Hardware Description

**Glossary**

| Term | Contents |
|---|---|
| SCAN TEST | It will test the logic section.<br>This test can achieve high fault coverage (90% or more), but need the test pattern of each test target.<br>It uses in the Runtime Test. |
| Logic BIST | It will test the logic section.<br>This test uses random pattern generator. Therefore, there is no need to have a test pattern of each test target.<br>It uses in the POST. |
| Memory BIST | It will test the memory (RAM) section.<br>This test uses an algorithm test pattern generator to achieve full fault coverage.<br>It uses in both POST and Runtime Test. |

#### 171.1.2.1 Features

POST performs test by dividing the CHIP in multiple hierarchies.

There are 2 types of hierarchy ("Stop" and "Boot start").
When POST detects fault in "Stop" hierarchy, SoC boot is stopped and error is notified to external device.
On the other hand, SoC boot is not stopped regardless of POST results of "Boot start" hierarchy.
In this case, the Fault occurrence shall be confirmed by using software. For details, see the section 171.1.2.4 Register Configuration.

POST is splitted into 46 hierarchies.
POST runs the Logic BIST and Memory BIST.

**Table 171.1 Hierarchy of POST**

| No. | Hierarchy Name | Description | Power Domain | Hierarchy Type | Test Coverage(*) |
|---|---|---|---|---|---|
| 1 | TOP | TOP Hierarchy | C5 | Stop | (90%) |
| 2 | VIO | Video IO Hierarchy | C5, C4 | Boot start | (90%) |
| 3 | ISP0 | Video IO Hierarchy (ISP0) | A3ISP0 | Boot start | (60%) |
| 4 | VC | Video Codec Hierarchy | C5, C4 | Boot start | (90%) |
| 5 | IMR0(IMR0) | Video Codec Hierarchy (IMR0(IMR0)) | A3IMR0 | Boot start | (60%) |
| 6 | IMR1(IMR1) | Video Codec Hierarchy (IMR1(IMR1)) | A3IMR1 | Boot start | (60%) |
| 7 | IMR2(IMS0) | Video Codec Hierarchy (IMR2(IMS0)) | A3IMR2 | Boot start | (60%) |
| 8 | IMR3(IMS1) | Video Codec Hierarchy (IMR3(IMS1)) | A3IMR3 | Boot start | (60%) |
| 9 | MM | Main Memory Hierarchy | C5 | Stop | (90%) |
| 10 | DBSC0 | Main Memory Hierarchy (DBSC0) | C5 | Stop | (90%) |
| 11 | DDR | Main Memory Hierarchy (DDR) | C5 | Stop | (90%) |
| 12 | IR | Image Recognition Hierarchy | C5 | Boot start | (90%) |
| 13 | A3IR Bus | Image Recognition Hierarchy (A3IR Bus) | C4 | Boot start | (90%) |
| 14 | IMP0 | Image Recognition Hierarchy (IMP core0) | A2IMP01 | Boot start | (60%) |
| 15 | IMP1 | Image Recognition Hierarchy (IMP core1) | A2IMP01 | Boot start | (60%) |
| 16 | OCV0 | Image Recognition Hierarchy (OCV core0) | A2CV0 | Boot start | (60%) |
| 17 | OCV1 | Image Recognition Hierarchy (OCV core1) | A2CV1 | Boot start | (60%) |
| 18 | OCV2 | Image Recognition Hierarchy (OCV core2) | A2CV2 | Boot start | (60%) |
| 19 | OCV3 | Image Recognition Hierarchy (OCV core3) | A2CV3 | Boot start | (60%) |
| 20 | DP0 | Image Recognition Hierarchy (IMP DMAC0, IMP PSC0) | A2PSC | Boot start | (60%) |
| 21 | DP1 | Image Recognition Hierarchy (IMP DMAC1) | A2DMA | Boot start | (60%) |
| 22 | CNN | Image Recognition Hierarchy (IMP CNN) | A1CN0 | Boot start | (60%) |
| 23 | DSP0 | Image Recognition Hierarchy (DSP0) | A1DSP0 | Boot start | (60%) |
| 24 | DSP1 | Image Recognition Hierarchy (DSP1) | A1DSP1 | Boot start | (60%) |
| 25 | CNRAM | Image Recognition Hierarchy (CNRAM) | A2CN0 | Boot start | (60%) |
| 26 | IMPRAM | Image Recognition Hierarchy (IMPRAM) | C4 | Boot start | (60%) |
| 27 | RT | RealTime Hierarchy | C5 | Stop | (90%) |
| 28 | Cortex-R52 | RealTime Hierarchy (Cortex-R52) | C5, A3CR0, A3CR1, A3CR2 | Stop | (90%) |
| 29 | CA76SS0 C4 | Cortex-A76 CPU Sub-system Hierarchy (C4 domain) | C4 | Boot start | (90%) |
| 30 | CA76SS0 Cluster0 | Cortex-A76 CPU Cluster 0 Hierarchy (A2 domain) | A2E0D0 | Boot start | (60%) |
| 31 | CA76SS0 Core0 | Cortex-A76 CPU Core 0 Hierarchy (A1 domain) | A1E0D0C0 | Boot start | (60%) |
| 32 | CA76SS0 Core1 | Cortex-A76 CPU Core 1 Hierarchy (A1 domain) | A1E0D0C1 | Boot start | (60%) |
| 33 | CA76SS0 Core2 | Cortex-A76 CPU Core 2 Hierarchy (A1 domain) | A1E0D0C2 | Boot start | (60%) |

| No. | Hierarchy Name | Description | Power Domain | Hierarchy Type | Test Coverage(*) |
|---|---|---|---|---|---|
| 34 | CA76SS0 Core3 | Cortex-A76 CPU Core 3 Hierarchy (A1 domain) | A1E0D0C3 | Boot start | (60%) |
| 35 | PER | Peripheral-DMAC Hierarchy | C5 | Boot start | (90%) |
| 36 | HSC | High Speed Communication Hierarchy | C5, C4 | Boot start | (90%) |
| 37 | PVD0 | 3D Graphics (RGX PVD0) Hierarchy | A23DGB | Boot start | (60%) |
| 38 | PVD1 | 3D Graphics (RGX PVD1) Hierarchy | A23DGB | Boot start | (60%) |
| 39 | PVD2 | 3D Graphics (RGX PVD2) Hierarchy | A33DGA | Boot start | (60%) |
| 40 | PVD | 3D Graphics (CONST RGX Bridge) Hierarchy | C4 | Boot start | (60%) |
| 41 | PVC | 3D Graphics (CONST) Hierarchy | C5, C4 | Boot start | (90%) |
| 42 | VIP | Vision IP Hierarchy | C5, C4 | Boot start | (90%) |
| 43 | UMFL0 | Vision IP Hierarchy (Optical Flow 0) | A3VIP0 | Boot start | (60%) |
| 44 | SMPS0 | Vision IP Hierarchy (SMPS0) | A3VIP2 | Boot start | (60%) |
| 45 | SMPO0 | Vision IP Hierarchy (SMPO0) | A3VIP2 | Boot start | (60%) |
| 46 | PAP | Vision IP Hierarchy (PAP) | A3DUL | Boot start | (60%) |

(*) SM coverage is important, not hierarchical coverage.

The POST is automatically applied on all hierarchies by setting the levels on the mode pins MD25 to high (1) and the levels on the mode pins MD23 to low (0).
The POST is automatically applied on partial hierarchies by setting the levels on the mode pins MD25 to high (1) and the levels on the mode pins MD23 to high (1).

POST is started at different time for different domains to reduce the rush current during power up of the SoC.

**Table 171.2 Pin function of POST**

| Pins | | POST |
|---|---|---|
| MD25 | MD23 | Description |
| 0 | - | No operation |
| 1 | 0 | The test is executed on all hierarchies. |
| 1 | 1 | The test is executed on partial hierarchies. |

### 171.1.2.2 Block Diagram

Figure 171.1 shows the block diagram of field BIST system.

The field BIST modules is constituted by one of field BIST controller (FBC) and each hierarchy of the field BIST activators (FBAs), JTAG-sequencer (FBJs).

**(1) Field BIST controller (FBC)**

- Create a POST start signal.
  Notifies the finish of POST to the clock pulse generator (CPG)
- Interrupt Controller (INTC) interface
  "Field BIST pre" signal is Test ready.
  "Field BIST finish" signal is Test finish.
  These signals asserted at the time of Runtime Test.
- Error Control Module (ECM) interface
  "fbc_rt_fail" is Test Fail for Runtime Test.
  "fbc_rt_overerr" is Time out Error for Runtime Test.
  "fbc_rt_saftyerr" is Bus Error for Runtime Test.

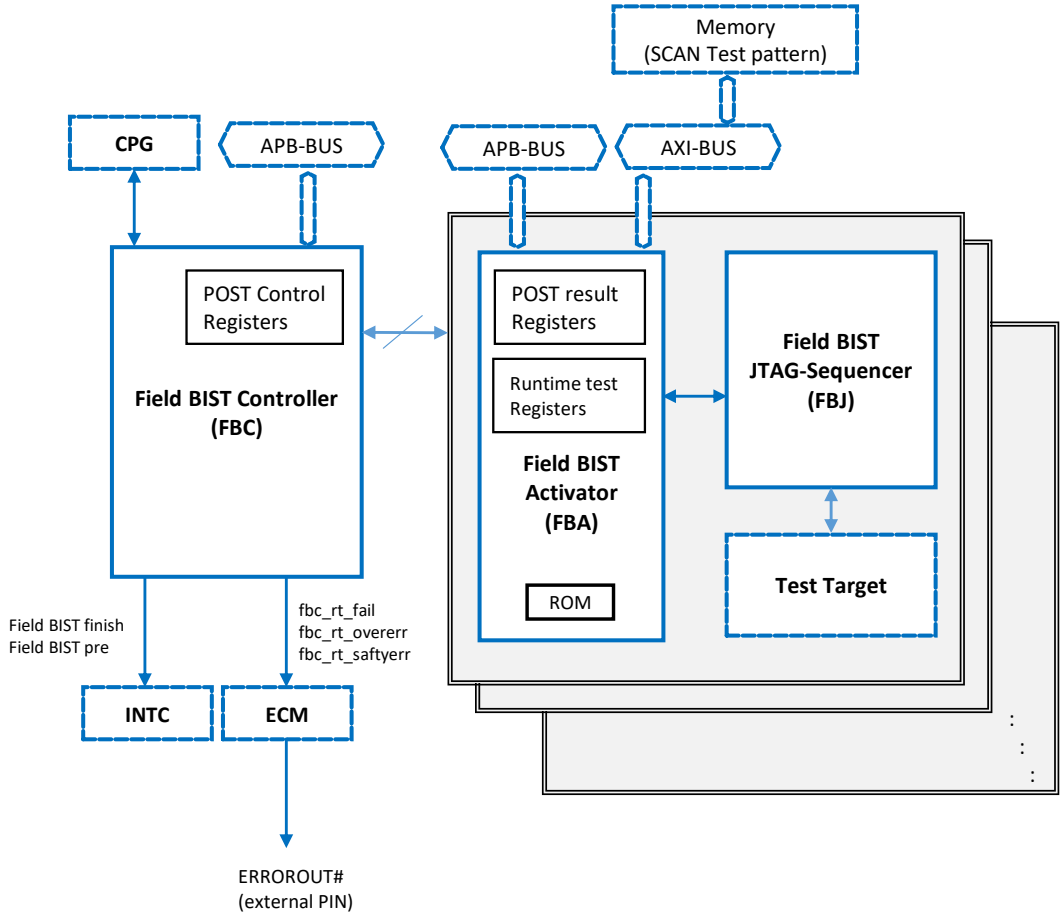



**Figure 171.1 Overview of Field-BIST system**

**(2) Field BIST Activator (FBA)**
Test procedures and test conditions are written in the ROM.
Reads the external memory SCAN pattern at AXI bus, transfer to FBJ.
Stored in the register to the test results SISR (Single Input Signature Register).

**(3) JTAG Sequencer (FBJ)**
Run the test of the test target by issuing the test code.

#### 171.1.2.3 External Pins

When POST detect fault in "STOP" hierarchy, the PRESETOUT# signal is not de-asserted.
On the other hand, when POST detects no fault or detects fault only in "Boot start" hierarchy, the PRESETOUT# signal is de-asserted.

#### 171.1.2.4 Register Configuration

**(1) FBA registers**
Most of the registers works at the initial value.
The FBA registers are used for the scan test in the runtime test and for the memory built-in self test (MBIST).
Renesas Electronics provides the setting values of the registers.

(a) **Hierarchy registers**

**Table 171.3 Register Configuration (Hierarchy)**

| No. | Address [31:12] | Hierarchy Name | Description | POST MD23=0 (POST0) | POST MD23=1 (POST1) | Runtime Test |
|---|---|---|---|---|---|---|
| 1 | H'FF820 | TOP | TOP Hierarchy | √ | √ | - |
| 2 | H'FEAC0 | VIO | Video IO Hierarchy | √ | √ | - |
| 3 | H'FEAC1 | ISP0 | Video IO Hierarchy (ISP0) | √ | - | √ |
| 4 | H'FE600 | VC | Video Codec Hierarchy | √ | √ | - |
| 5 | H'FE601 | IMR0(IMR0) | Video Codec Hierarchy (IMR0(IMR0)) | √ | - | √ |
| 6 | H'FE602 | IMR1(IMR1) | Video Codec Hierarchy (IMR1(IMR1)) | √ | - | √ |
| 7 | H'FE605 | IMR2(IMS0) | Video Codec Hierarchy (IMR2(IMS0)) | √ | - | √ |
| 8 | H'FE606 | IMR3(IMS1) | Video Codec Hierarchy (IMR3(IMS1)) | √ | - | √ |
| 9 | H'E67A0 | MM | Main Memory Hierarchy | √ | √ | - |
| 10 | H'E67A1 | DBSC0 | Main Memory Hierarchy (DBSC0) | √ | √ | - |
| 11 | H'E67FF | DDR | Main Memory Hierarchy (DDR) | √ | √ | - |
| 12 | H'FF8C8 | IR | Image Recognition Hierarchy | √ | √ | - |
| 13 | H'FF8C9 | A3IR Bus | Image Recognition Hierarchy (A3IR Bus) | √ | √ | - |
| 14 | H'FF8C0 | IMP0 | Image Recognition Hierarchy (IMP core0) | √ | - | √ |
| 15 | H'FF8C1 | IMP1 | Image Recognition Hierarchy (IMP core1) | √ | - | √ |
| 16 | H'FF8C4 | OCV0 | Image Recognition Hierarchy (OCV core0) | √ | - | √ |
| 17 | H'FF8C5 | OCV1 | Image Recognition Hierarchy (OCV core1) | √ | - | √ |
| 18 | H'FF8C6 | OCV2 | Image Recognition Hierarchy (OCV core2) | √ | - | √ |
| 19 | H'FF8C7 | OCV3 | Image Recognition Hierarchy (OCV core3) | √ | - | √ |
| 20 | H'FFF56 | DP0 | Image Recognition Hierarchy (IMP DMAC0, IMP PSC0) | √ | - | √ |
| 21 | H'FFF57 | DP1 | Image Recognition Hierarchy (IMP DMAC1) | √ | - | √ |
| 22 | H'FFF50 | CNN | Image Recognition Hierarchy (IMP CNN) | √ | - | √ |
| 23 | H'FFF64 | DSP0 | Image Recognition Hierarchy (DSP0) | √ | - | √ |
| 24 | H'FFF65 | DSP1 | Image Recognition Hierarchy (DSP1) | √ | - | √ |
| 25 | H'FFF55 | CNRAM | Image Recognition Hierarchy (CNRAM) | √ | - | √ |
| 26 | H'FFF76 | IMPRAM | Image Recognition Hierarchy (IMPRAM) | √ | - | - |
| 27 | H'FFFE0 | RT | RealTime Hierarchy | √ | √ | - |
| 28 | H'FFFC0 | Cortex-R52 | RealTime Hierarchy (Cortex-R52) | √ | √ | - |
| 29 | H'FF878 | CA76SS0 C4 | Cortex-A76 CPU Sub-system Hierarchy (C4 domain) | √ | √ | - |
| 30 | H'FF870 | CA76SS0 Cluster0 | Cortex-A76 CPU Cluster 0 Hierarchy (A2 domain) | √ | - | √ |
| 31 | H'FF871 | CA76SS0 Core0 | Cortex-A76 CPU Core 0 Hierarchy (A1 domain) | √ | - | √ |
| 32 | H'FF872 | CA76SS0 Core1 | Cortex-A76 CPU Core 1 Hierarchy (A1 domain) | √ | - | √ |
| 33 | H'FF874 | CA76SS0 Core2 | Cortex-A76 CPU Core 2 Hierarchy (A1 domain) | √ | - | √ |
| 34 | H'FF875 | CA76SS0 Core3 | Cortex-A76 CPU Core 3 Hierarchy (A1 domain) | √ | - | √ |
| 35 | H'E6680 | PER | Peripheral-DMAC Hierarchy | √ | √ | - |
| 36 | H'E6450 | HSC | High Speed Communication Hierarchy | √ | √ | - |
| 37 | H'FD900 | PVD0 | 3D Graphics (RGX PVD0) Hierarchy | √ | - | - |
| 38 | H'FD910 | PVD1 | 3D Graphics (RGX PVD1) Hierarchy | √ | - | - |
| 39 | H'FD920 | PVD2 | 3D Graphics (RGX PVD2) Hierarchy | √ | - | - |
| 40 | H'FD930 | PVD | 3D Graphics (CONST RGX Bridge) Hierarchy | √ | - | - |

| No. | Address [31:12] | Hierarchy Name | Description | POST MD23=0 (POST0) | POST MD23=1 (POST1) | Runtime Test |
|---|---|---|---|---|---|---|
| 41 | H'FD940 | PVC | 3D Graphics (CONST) Hierarchy | √ | √ | - |
| 42 | H'E7B60 | VIP | Vision IP Hierarchy | √ | √ | - |
| 43 | H'E7B61 | UMFL0 | Vision IP Hierarchy (Optical Flow 0) | √ | - | √ |
| 44 | H'E7B64 | SMPS0 | Vision IP Hierarchy (SMPS0) | √ | - | √ |
| 45 | H'E7B63 | SMPO0 | Vision IP Hierarchy (SMPO0) | √ | - | √ |
| 46 | H'E7B62 | PAP | Vision IP Hierarchy (PAP) | √ | - | √ |

√: Support for the product
-: Not support for the product

(b) **Common Address**

**Table 171.4 Register Configuration**

| Register Name | Abbreviation | R/W | Address[11:0] | Initial Value | Access Size |
|---|---|---|---|---|---|
| **POST Result** | | | | | |
| Total Test result for Power On Self Test | PSTRSTT | R | H'400 | H'0000 1300 | 32 |
| Counter Value for Power On Self Test | PSTCNT | R | H'440 | H'0000 0000 | 32 |

### 171.1.2.5 Connected Module

**Table 171.5 Connected Module**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| FBC | CPG | Assert start of POST.<br>Provide finish of POST. |
| | INTC | Provide Field BIST end of preparation interrupt to INTC.<br>Provide Field BIST finish to INTC. |
| | ECM | Provide Field BIST test fail to ECM.<br>Provide Time out error to ECM.<br>Provide BUS error to ECM. |

### 171.1.2.6 Register Description

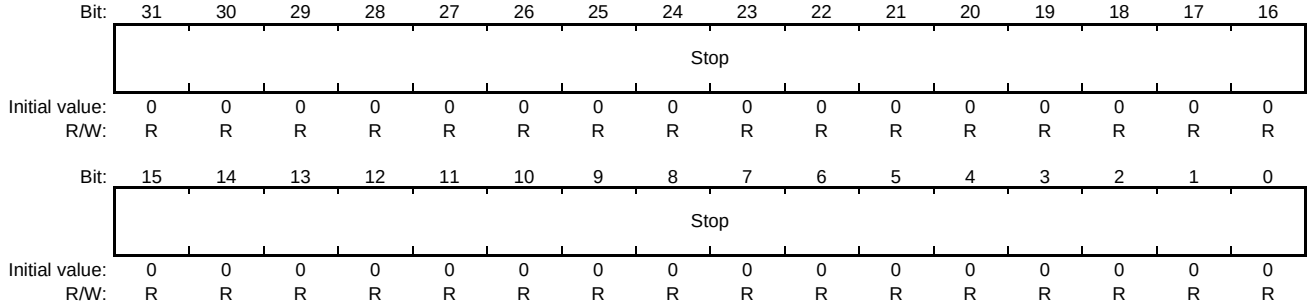
#### (1) Total Test result for Power On Self Test (PSTRSTT)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | R-rst_T | - | error_T | | | testnum_T | | | | status_R | | status_T | |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 13 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 12 | R-rst_T | 1 | R | BIST Redundant result. (Safety function)<br>0: pass<br>1: fail |
| 11 | - | 0 | R | Reserved.<br>This bit is always read as 0. The write value should always be 0. |
| 10 to 8 | error_T | 011 | R | BIST error.<br>000: Pass.<br>001: Fail.<br>010: TimeOut. (FieldBIST internal WDT)<br>100: BUS error.<br>This error occurs when a slave error or decoder error is detected in the read response RRESP on the AXI bus. |
| 7 to 4 | testnum_T | 0000 | R | Number of tests. |
| 3 to 2 | status_R | 00 | R | BIST Redundant status (Safety function)<br>00: unexec.<br>01: wait. (preparing for execution of a test)<br>10: exec.<br>11: finish. |
| 1 to 0 | status_T | 00 | R | BIST status.<br>00: unexec.<br>01: wait. (preparing for execution of a test)<br>10: exec.<br>11: finish. |

**(2) Counter Value for Power On Self Test (PSTCNT)**




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Stop | All 0 | R | FieldBIST internal WDT (high counter) stop value. |

FieldBIST internal WDT is made of a down counter.
Time-out error occurs when the WDT is zero.

### 171.1.2.7 Debugger Usage Note

Reset vector catch works when POST is enabled. For the reset release order, release TRST# first, then PRESET#. Set the reset vector catch between these two resets. POST after power-on reset in debug mode is emulation (dummy). The reset vector catch setting is not deleted, and the reset vector catch is possible.

[POST execution/skip conditions]
When MD25 = 0, POST is not executed.
When MD25 = 1,
  When not in debug mode (mode pin setting {MDT0, MD21, MD20, MD11, MD10} = 5'b00000), POST is executed.
  When in debug mode
    When the mode pin setting is that CoreSight is connected to Main-JTAG,
      POST is skipped when TRST# is released before PRESET#.
      POST is executed when PRESET# is released before TRST#.
    When the mode pin setting is that CoreSight is connected to Sub-JTAG,
      POST is skipped when TRST# is released before PRESET#.
      POST is executed when PRESET# is released before TRST#.
    When the mode pin setting is that CoreSight is connected neither to Main-JTAG nor Sub-JTAG, POST is executed.

Note: "POST is skipped" differs from "POST is not executed" in that the Self Test is not performed, but the POST execution period is kept.

## 171.2 Runtime Test

### 171.2.1 Overview

The Runtime Test is a safety mechanism to detect faults during the SoC is used. This safety mechanism needs to be periodically executed by software. The Runtime Test is performed using the field BIST[*1] modules.

Note *1: The field BIST is a collective term for the POST and Runtime Test.

### 171.2.2 Hardware Description

#### 171.2.2.1 Runtime Test

This module is a safety mechanism in case of functional safety supported product. The function of Runtime Test shares the hardware configuration with POST. See the section on POST for hardware configuration.

Runtime Test is only run for a specified hierarchy when power is supplied to a specified hierarchy.

Scan tests are applied to logic blocks to accomplish a high failure coverage. The memory built-in self test (MBIST) is applied as shown in Table 171.6.

Accomplishing failure coverage of at least 90% by the scan tests requires many test patterns in almost every hierarchy, leading to the tests taking a longer time (Hardware processing time is within 10ms. In addition, software processing time is required.).

Tests require that the normal operation of the hierarchies that are their targets be stopped, and having them out of operation for long periods affects system performance. The time-sliced method can be used to reduce this effect where this is not allowable. (For example, the time-sliced method is to divide the test in about 2ms.)

Before executing Runtime Test, set module stop control register "for target module of Runtime Test" to stop, except Cortex-A76. For details, refer to section 9 Module Standby, Software Reset.

Before executing Runtime Test, set software reset register "for target module of Runtime Test" to reset, except Cortex-A76 and Vision-IP (except PAP). For details, refer to section 9 Module Standby, Software Reset.

Do not access modules in Runtime Test

**(A) Features**
Runtime Test performs test by dividing the CHIP in multiple hierarchies.

Runtime Test is applied into 26 hierarchies.
Table 171.6 is a list of the Runtime test methods for the respective hierarchies.

The field BIST (*1) circuit has two modes of operation for Runtime Test: ROM batch mode and single-step mode.

In ROM batch mode, some tests execute in sequence according to information in the ROM.
Control of the Software is only once, but test time is longer.
This mode is recommended to use for MBIST and couldn't be used for scan test because there are no ROM for scan test.

In Single-step mode, each tests executes independently by using register data and scan pattern in RAM.
Each test could take short time but control of software is required multiple times.
This mode should be used for scan test and is not recommended to use for MBIST.

Note *1: The field BIST is a collective term for POST and Runtime Test.

R-Car V4M all of the hierarchy to adopt the Time-sliced test.

Running scan tests requires the writing of test patterns provided by Renesas Electronics to the memory in advance.
Align the test patterns with a 128-byte boundary in 32-bit address range.

**Table 171.6 Hierarchy of Runtime Test**

| No. | Hierarchy Name | Description | Runtime Test | Runtime Test MBIST |
|---|---|---|---|---|
| 1 | TOP | TOP Hierarchy | - | - |
| 2 | VIO | Video IO Hierarchy | - | - |
| 3 | ISP0 | Video IO Hierarchy (ISP0) | √ | - |
| 4 | VC | Video Codec Hierarchy | - | - |
| 5 | IMR0(IMR0) | Video Codec Hierarchy (IMR0(IMR0)) | √ | √ |
| 6 | IMR1(IMR1) | Video Codec Hierarchy (IMR1(IMR1)) | √ | √ |
| 7 | IMR2(IMS0) | Video Codec Hierarchy (IMR2(IMS0)) | √ | √ |
| 8 | IMR3(IMS1) | Video Codec Hierarchy (IMR3(IMS1)) | √ | √ |
| 9 | MM | Main Memory Hierarchy | - | - |
| 10 | DBSC0 | Main Memory Hierarchy (DBSC0) | - | - |
| 11 | DDR | Main Memory Hierarchy (DDR) | - | - |
| 12 | IR | Image Recognition Hierarchy | - | - |
| 13 | A3IR Bus | Image Recognition Hierarchy (A3IR Bus) | - | - |
| 14 | IMP0 | Image Recognition Hierarchy (IMP core0) | √ | - |
| 15 | IMP1 | Image Recognition Hierarchy (IMP core1) | √ | - |
| 16 | OCV0 | Image Recognition Hierarchy (OCV core0) | √ | - |
| 17 | OCV1 | Image Recognition Hierarchy (OCV core1) | √ | - |
| 18 | OCV2 | Image Recognition Hierarchy (OCV core2) | √ | - |
| 19 | OCV3 | Image Recognition Hierarchy (OCV core3) | √ | - |
| 20 | DP0 | Image Recognition Hierarchy (IMP DMAC0, IMP PSC0) | √ | - |
| 21 | DP1 | Image Recognition Hierarchy (IMP DMAC1) | √ | - |
| 22 | CNN | Image Recognition Hierarchy (IMP CNN) | √ | - |
| 23 | DSP0 | Image Recognition Hierarchy (DSP0) | √ | - |
| 24 | DSP1 | Image Recognition Hierarchy (DSP1) | √ | - |
| 25 | CNRAM | Image Recognition Hierarchy (CNRAM) | √ | - |
| 26 | IMPRAM | Image Recognition Hierarchy (IMPRAM) | - | - |
| 27 | RT | RealTime Hierarchy | - | - |
| 28 | Cortex-R52 | RealTime Hierarchy (Cortex-R52) | - | - |
| 29 | CA76SS0 C4 | Cortex-A76 CPU Sub-system Hierarchy (C4 domain) | - | - |
| 30 | CA76SS0 Cluster0 | Cortex-A76 CPU Cluster 0 Hierarchy (A2 domain) | √ | - |
| 31 | CA76SS0 Core0 | Cortex-A76 CPU Core 0 Hierarchy (A1 domain) | √ | √ |
| 32 | CA76SS0 Core1 | Cortex-A76 CPU Core 1 Hierarchy (A1 domain) | √ | √ |
| 33 | CA76SS0 Core2 | Cortex-A76 CPU Core 2 Hierarchy (A1 domain) | √ | √ |
| 34 | CA76SS0 Core3 | Cortex-A76 CPU Core 3 Hierarchy (A1 domain) | √ | √ |
| 35 | PER | Peripheral-DMAC Hierarchy | - | - |
| 36 | HSC | High Speed Communication Hierarchy | - | - |
| 37 | PVD0 | 3D Graphics (RGX PVD0) Hierarchy | - | - |
| 38 | PVD1 | 3D Graphics (RGX PVD1) Hierarchy | - | - |
| 39 | PVD2 | 3D Graphics (RGX PVD2) Hierarchy | - | - |
| 40 | PVD | 3D Graphics (CONST RGX Bridge) Hierarchy | - | - |
| 41 | PVC | 3D Graphics (CONST) Hierarchy | - | - |
| 42 | VIP | Vision IP Hierarchy | - | - |
| 43 | UMFL0 | Vision IP Hierarchy (Optical Flow 0) | √ | - |
| 44 | SMPS0 | Vision IP Hierarchy (SMPS0) | √ | - |

| No. | Hierarchy Name | Description | Runtime Test | Runtime Test MBIST |
|---|---|---|---|---|
| 45 | SMPO0 | Vision IP Hierarchy (SMPO0) | √ | - |
| 46 | PAP | Vision IP Hierarchy (PAP) | √ | - |

√: Support for the product
-: Not support for the product

**(B) Interrupt Signals**

The following interrupt signals and error interrupt signals are provided.

**(1) Interrupt Signals (the logical OR of interrupt signals of each hierarchy)**

"Field BIST finish"
Test completed.
"Field BIST pre"
Test preparation completed.

**(2) Interrupt Signals (the independent interrupt signal for each hierarchy)**

"Field BIST finish"
Test completed.
"Field BIST pre"
Test preparation completed.

**(3) Error interrupt signals (the logical OR of interrupt signals of each hierarchy)**

"Field-BIST controller.rt_fail"
The result of the test was failure.
"Field-BIST controller.rt_overerr"
Timeout. (There is no response from the Test Target)
"Field-BIST controller.rt_saftyerr"
Bus error.

**(4) Error interrupt signals (the independent interrupt signal for each hierarchy)**

"Field-BIST controller.rt_fail"
The result of the test was failure.
"Field-BIST controller.rt_overerr"
Timeout. (There is no response from the Test Target)
"Field-BIST controller.rt_saftyerr"
Bus error.

When use the interrupt signals of (1) or (3), if multiple hierarchies issue interrupts simultaneously, the first to be issued will be in effect because the logical OR of the interrupt signals is taken.
To check the detailed Field BIST finish interrupt, read the RTTFINISH1, RTTFINISH2 and RTTFINISH3 registers.

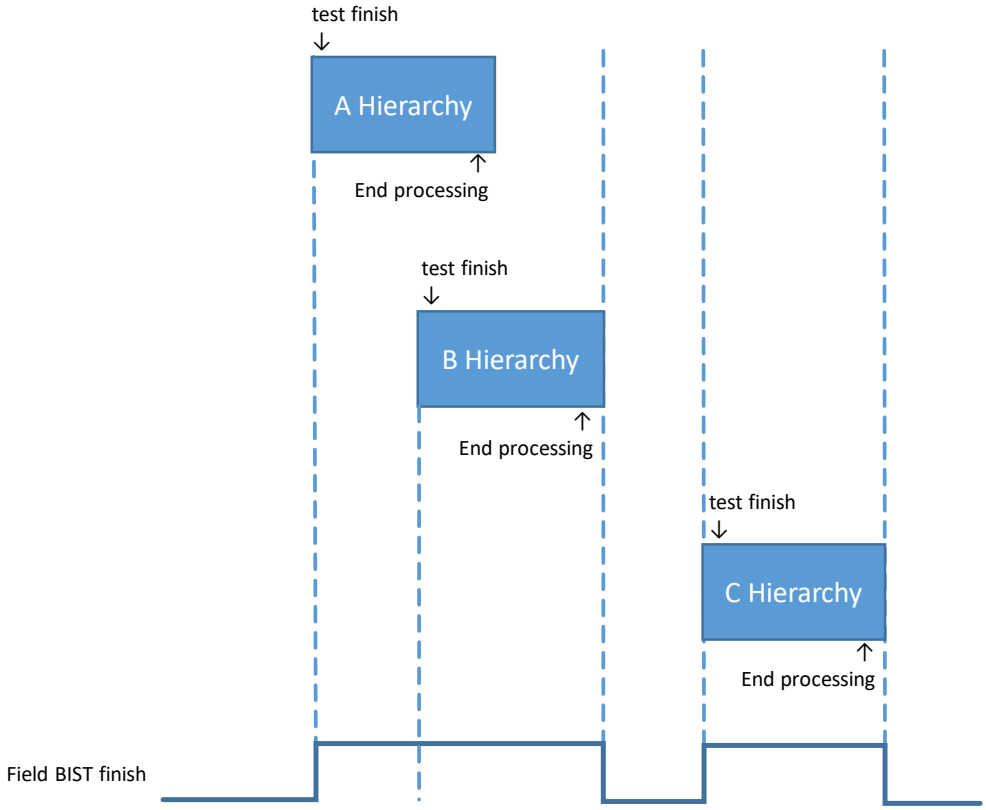



**Figure 171.2 Interrupt Signal for example**

**(C) Register Configuration**

**Table 171.7 Register Configuration**

| Register Name | Abbreviation | R/W | Address[11:0] | Initial Value | Access Size |
|---|---|---|---|---|---|
| Runtime Test parameter | | | | | |
| Execution option | RTTEX | R/W | H'000 | H'0000 0000 | 32 |
| Memory information (1) | RTTMI1 | R/W | H'004 | H'0000 0000 | 32 |
| Memory information (2) | RTTMI2 | R/W | H'008 | H'0000 0000 | 32 |
| Expected value of SISR (1) | RTTSISR1 | R/W | H'00C | H'0000 0000 | 32 |
| Upper counter of WDT | RTTUC | R/W | H'014 | H'0000 0000 | 32 |
| Lower counter of WDT | RTTDC | R/W | H'018 | H'0000 0000 | 32 |
| Single Step Mode setting register0 (Test clock condition 1) | RTTSET0 | R/W | H'080 | H'0000 0000 | 32 |
| Single Step Mode setting register1 (Test clock condition 2) | RTTSET1 | R/W | H'084 | H'0000 0000 | 32 |
| Single Step Mode setting register32 (Memory BIST condition) | RTTSET32 | R/W | H'100 | H'0000 0000 | 32 |
| Single Step Mode setting register33 (Memory BIST condition) | RTTSET33 | R/W | H'104 | H'0000 0000 | 32 |
| Single Step Mode setting register34 (Memory BIST condition) | RTTSET34 | R/W | H'108 | H'0000 0000 | 32 |
| Single Step Mode setting register35 (Memory BIST condition) | RTTSET35 | R/W | H'10C | H'0000 0000 | 32 |
| Single Step Mode setting register36 (Memory BIST condition) | RTTSET36 | R/W | H'110 | H'0000 0000 | 32 |
| Single Step Mode setting register37 (Memory BIST condition) | RTTSET37 | R/W | H'114 | H'0000 0000 | 32 |
| Single Step Mode setting register38 (Memory BIST condition) | RTTSET38 | R/W | H'118 | H'0000 0000 | 32 |
| Single Step Mode setting register39 (Memory BIST condition) | RTTSET39 | R/W | H'11C | H'0000 0000 | 32 |
| Single Step Mode setting register40 (Memory BIST condition) | RTTSET40 | R/W | H'120 | H'0000 0000 | 32 |
| Single Step Mode setting register41 (Memory BIST condition) | RTTSET41 | R/W | H'124 | H'0000 0000 | 32 |
| Single Step Mode setting register42 (Memory BIST condition) | RTTSET42 | R/W | H'128 | H'0000 0000 | 32 |
| Single Step Mode setting register43 (Memory BIST condition) | RTTSET43 | R/W | H'12C | H'0000 0000 | 32 |
| Single Step Mode setting register44 (SCAN test condition) | RTTSET44 | R/W | H'130 | H'0000 0000 | 32 |
| | | | | | |
| Total Test result for Runtime Test | RTTRSTT | R | H'444 | H'0200 1700 | 32 |
| Counter Value for Runtime Test | RTTCNT | R | H'458 | H'0000 0000 | 32 |
| | | | | | |
| Expected value reverse for Self-Check | SLFRVS | R/W | H'4A0 | H'0000 0000 | 32 |
| Self-check result | SLFRST | R | H'4A4 | H'0000 0000 | 32 |
| Dummy Error set | ERRSET | R/W | H'4C0 | H'0000 0000 | 32 |

For Address [31:12] assignment of the registers, see tables (Table 171.3 Register Configuration (Hierarchy)).

**Table 171.8 Register Configuration (Field BIST Controller)**

| Register Name | Abbreviation | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| Runtime Test Finish Interrupt check1 | RTTFINISH1 | R | H’FF83 0100 | H'0000 0000 | 32 |
| Runtime Test Finish Interrupt check2 | RTTFINISH2 | R | H’FF83 0104 | H'0000 0000 | 32 |
| Runtime Test Finish Interrupt check3 | RTTFINISH3 | R | H’FF83 0108 | H'0000 0000 | 32 |

**Table 171.9 Register Configuration**

| Register Name | Abbreviation | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|---|
| Power-on self test clock frequency control register | POSTCKCR | R/W | H’E615 0890 | H’0000 *-0B | 32 |
| Power-on self test 2 clock frequency control register | POST2CKCR | R/W | H’E615 0894 | H’0000 *-05 | 32 |
| Power-on self test 4 clock frequency control register | POST4CKCR | R/W | H’E615 089C | H’0000 *-0B | 32 |

Note: * See section 171.2.2.1 Runtime Test (D) (30) to (32).

### (D) Register Description

#### (1) Execution option (RTTEX)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | SLFR | SLFE | - | - | - | - | KCD | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

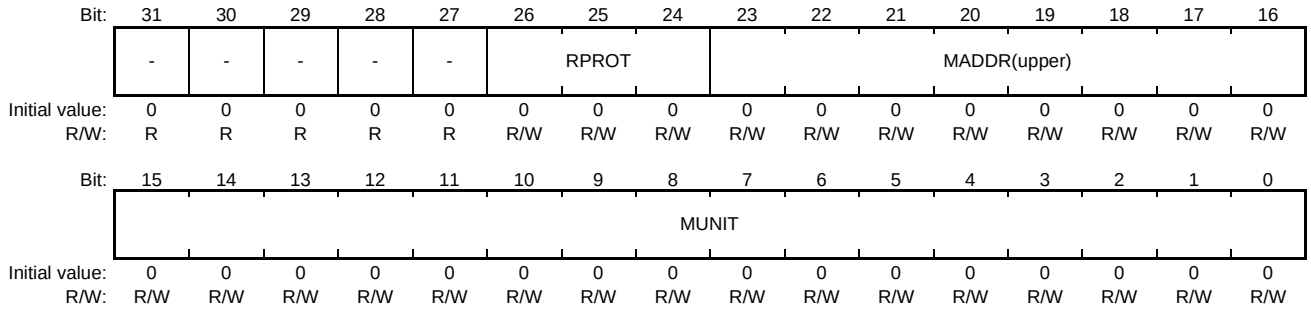
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | STM | | | INTM | | SSZ | TR | WM | TM | STP | EX |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 29 | SLFR | 0 | R/W | Self-Check Counter Reset for field BIST activator.<br>0: Counter Reset<br>1: Counter not Reset |
| 28 | SLFE | 0 | R/W | Execution of Self-Check for Field BIST activator.<br>0:Not Execution of Self-Check for Field BIST activator.<br>1:Execution of Self-Check for Field BIST activator. |
| 27 to 24 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 23 to 16 | KCD | All 0 | R/W | Key Code<br>H'5A: Effective execution of Run Time Test<br>H'F1: Effective execution of Self-Check for field BIST activator.<br>Other: No reaction. |
| 15 to 11 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 10 to 8 | STM | 000 | R/W | Single Step Test Mode.<br>Valid only when STP=1. Other state can be anything<br>001: LBIST.<br>010: MBIST.<br>011: SCAN.<br>Other: Do not use. |
| 7 to 6 | INTM | All 0 | R/W | Interrupt mask<br>00: non mask<br>*1: Always mask(fba_rt_pre and fba_rt_finish are not assert)<br>10: Only at the time of the fail(fba_rt_finish are not assert at fail) |
| 5 | SSZ | 0 | R/W | SCAN Pattern size<br>Valid only when STM=011. Other state can be anything.<br>0: 32bit mode<br>1: 16bit mode |
| 4 | TR | 0 | R/W | Select the value of the timer<br>0: ROM<br>1: register |
| 3 | WM | 0 | R/W | Wfi mask<br>0: not mask<br>1: mask |
| 2 | TM | 0 | R/W | Mask the judgment of timer<br>0: not mask<br>1: mask |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 1 | STP | 0 | R/W | Single Step Mode.<br>0: ROM batch mode.<br>1: Single Step Mode. |
| 0 | EX | 0 | R/W | Execution of Runtime Test.<br>0:Not Execution of Runtime Test.<br>1:Execution of Runtime Test. |

Note *1: SLFR and SLFE bit of the RTTEX are effective only when KCD bit is H'F1.

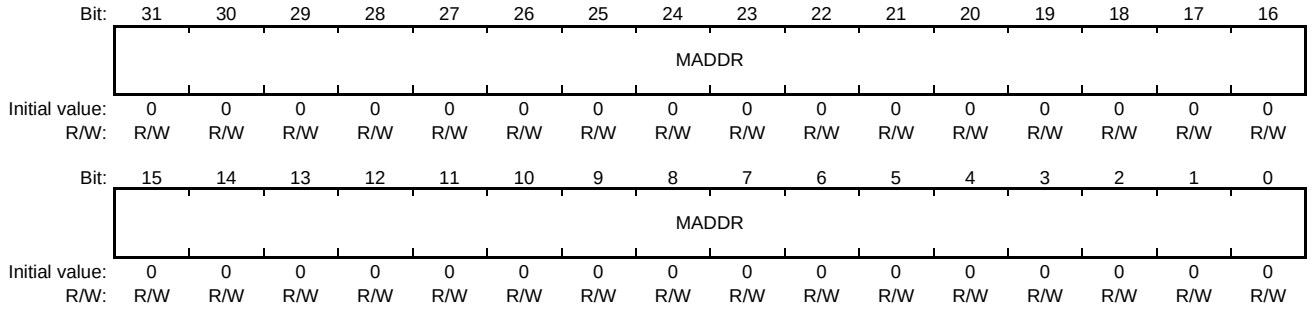
### (2) Memory information (1) (RTTMI1)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 26 to 24 | RPROT | 000 | R/W | Protection type of MEMORY.<br><br>[26]RPROT[2] 0: Data access, 1: Instruction access.<br>[25]RPROT[1] 0: Secure access, 1: Non-secure access.<br>[24]RPROT[0] 0: Unprivileged access, 1: Privileged access. |
| 23 to 16 | MADDR(upper) | 00000000 | R/W | Memory upper address[39:32] that stores the SCAN pattern. |
| 15 to 0 | MUNIT | H'0000 | R/W | Number of units (1 unit = 128 Byte). |

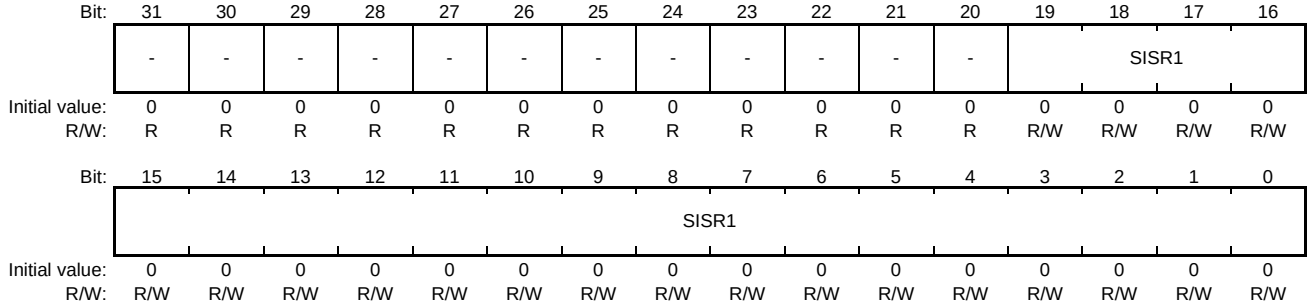
Write the values separately provided by Renesas Electronics.

### (3) Memory information (2) (RTTMI2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | MADDR | All 0 | R/W | Memory address that stores the SCAN pattern. |

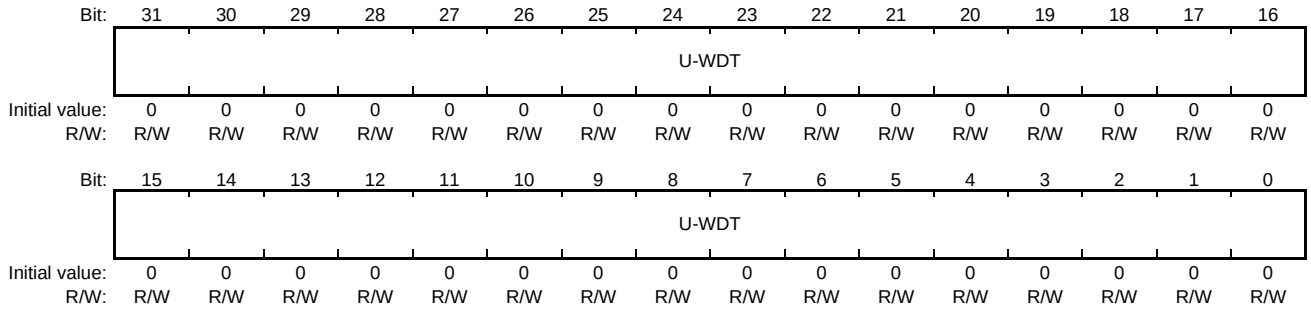
### (4) Expected value of SISR (1) (RTTSISR1)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 19 to 0 | SISR1 | All 0 | R/W | SISR-TEST#1 Expected value. |

The packed result (SISR)
Write the values separately provided by Renesas Electronics.

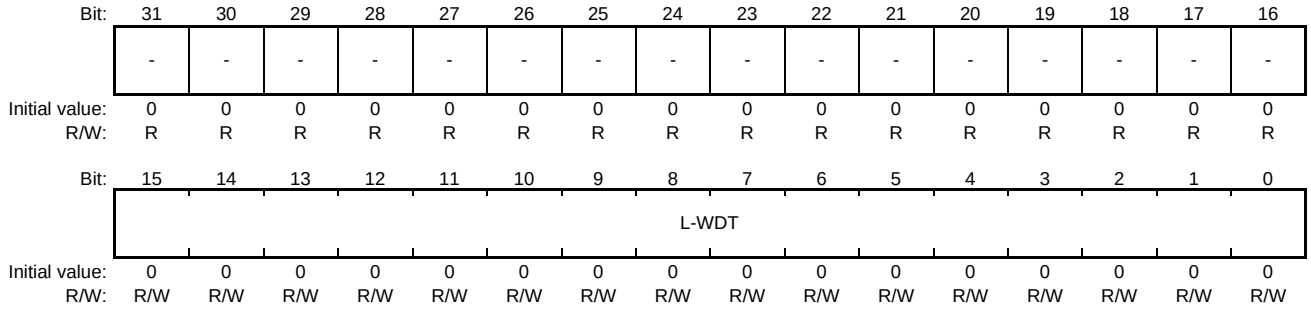
### (5) Upper counter of WDT (RTTUC)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | U-WDT | All 0 | R/W | Upper counter of Watchdog Timer. |

This register is used to detect timeout in the runtime test.
The timeout occurs when the runtime test is not completed within the number of counts specified in RTTUC x RTTDC cycles of the clock signal (66 MHz) dedicated to the POST.
Write the values separately provided by Renesas Electronics.

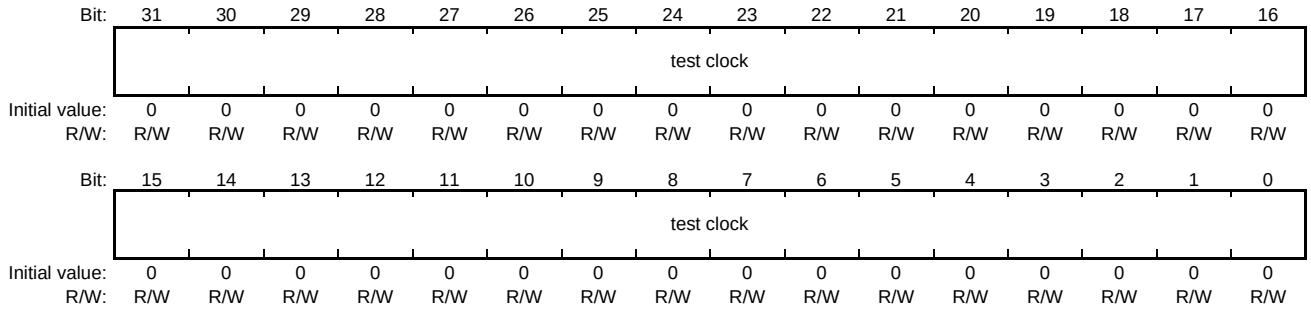
### (6) Lower counter of WDT (RTTDC)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 15 to 0 | L-WDT | All 0 | R/W | Lower counter of Watchdog Timer. |

Write the values separately provided by Renesas Electronics.

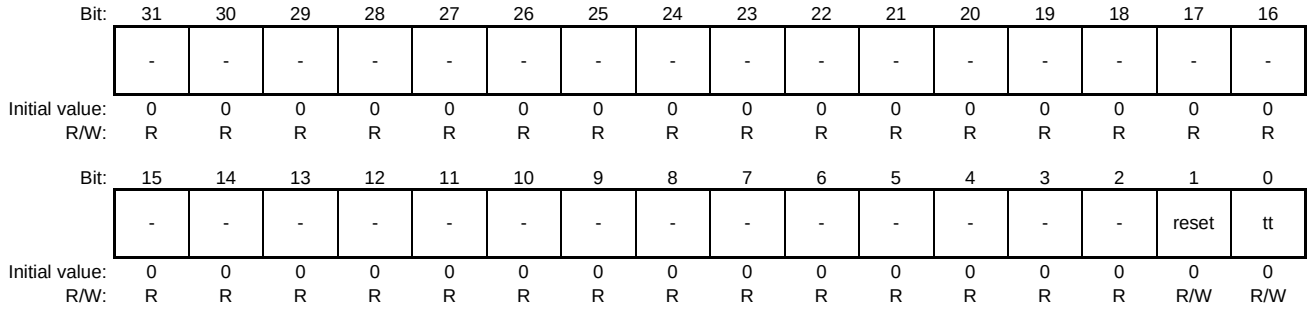
### (7) Single Step Mode setting register0 (Test clock condition 1) (RTTSET0)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | test clock | All 0 | R/W | Test setting input. (Test clock selection for user domain) |

Write the values separately provided by Renesas Electronics.

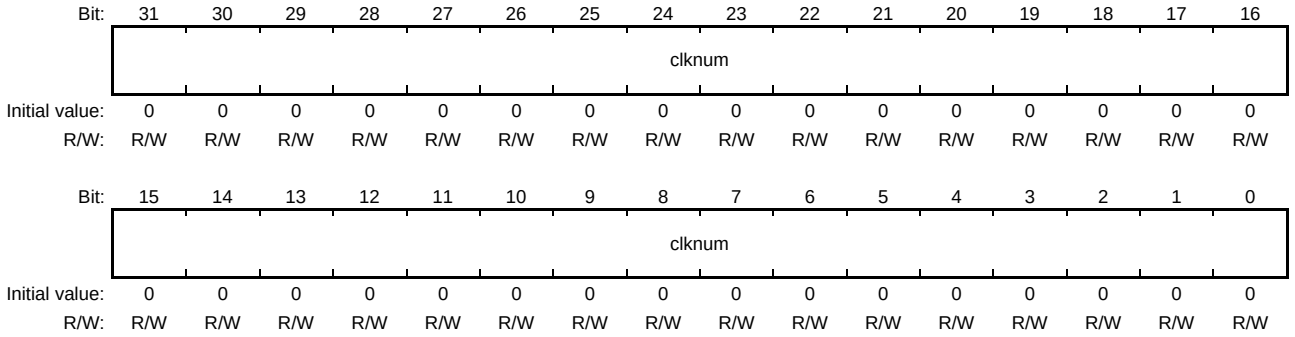
### (8) Single Step Mode setting register1 (Test clock condition 2) (RTTSET1)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 1 | reset | 0 | R/W | Test setting input. (Reset test enable)<br>0: Reset test disabled<br>1: Reset test enabled |
| 0 | tt | 0 | R/W | Test setting input. (Test clock selection for test domain)<br>0: Selected other clock<br>1: Selected test clock for test Domain |

Write the values separately provided by Renesas Electronics.

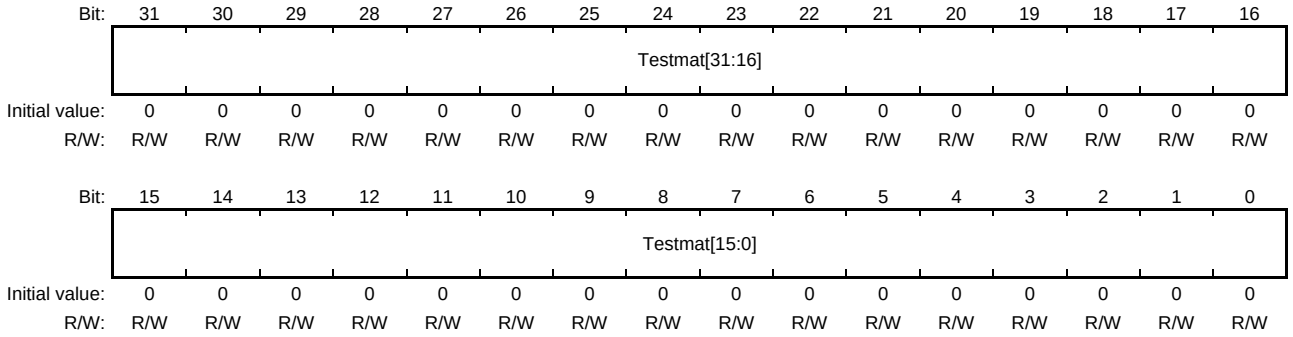
### (9) Single Step Mode setting register32 (Memory BIST condition) (RTTSET32)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | clknum[31:0] | All 0 | R/W | Test setting input. (Test cycle of MBIST) |

Write the values separately provided by Renesas Electronics.

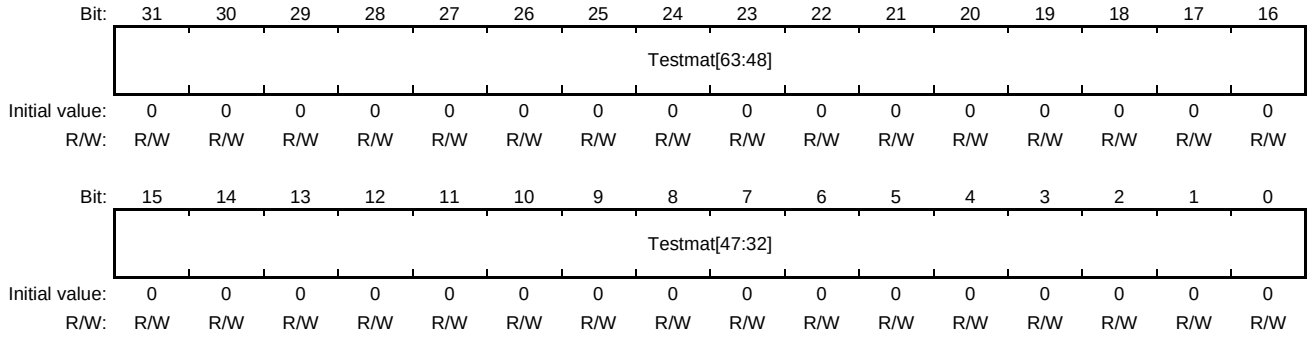
### (10) Single Step Mode setting register33 (Memory BIST condition) (RTTSET33)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Testmat[31:0] | All 0 | R/W | Test setting input. (Test target bridge selection of MBIST) |

Write the values separately provided by Renesas Electronics.

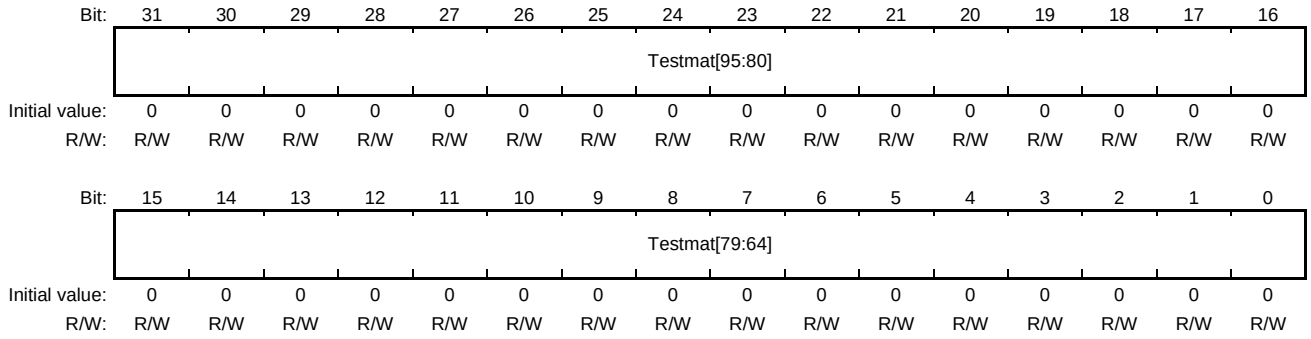
### (11) Single Step Mode setting register34 (Memory BIST condition) (RTTSET34)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Testmat[63:32] | All 0 | R/W | Test setting input. (Test target bridge selection of MBIST) |

Write the values separately provided by Renesas Electronics.

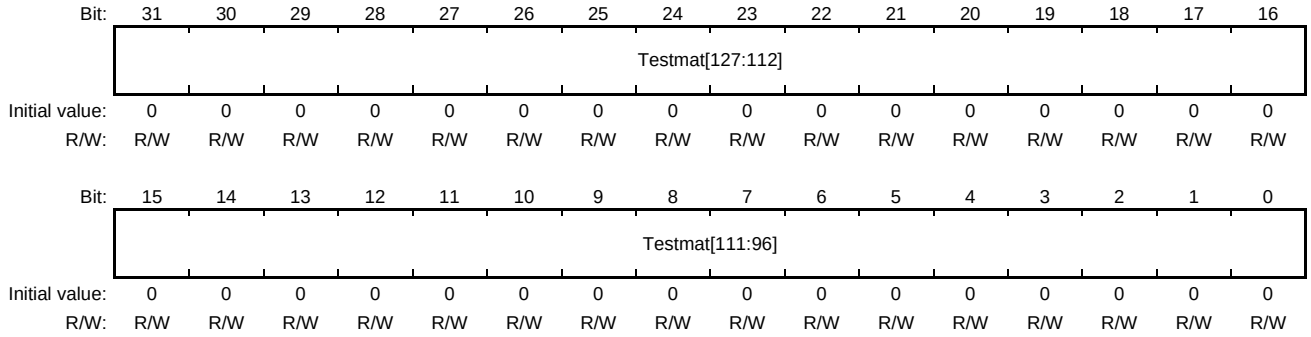
### (12) Single Step Mode setting register35 (Memory BIST condition) (RTTSET35)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Testmat[95:64] | All 0 | R/W | Test setting input. (Test target bridge selection of MBIST) |

Write the values separately provided by Renesas Electronics.

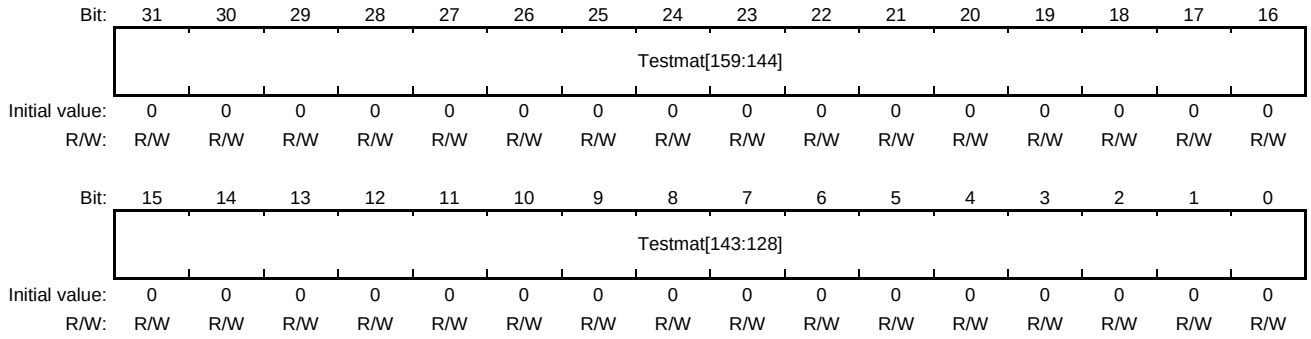
**(13) Single Step Mode setting register36 (Memory BIST condition) (RTTSET36)**




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Testmat[127:96] | All 0 | R/W | Test setting input. (Test target bridge selection of MBIST) |

Write the values separately provided by Renesas Electronics.

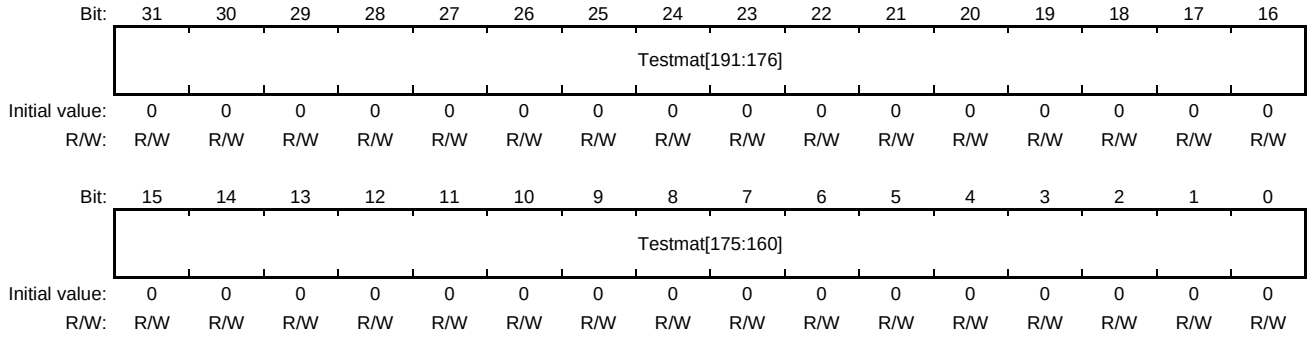
**(14) Single Step Mode setting register37 (Memory BIST condition) (RTTSET37)**




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Testmat[159:128] | All 0 | R/W | Test setting input. (Test target bridge selection of MBIST) |

Write the values separately provided by Renesas Electronics.

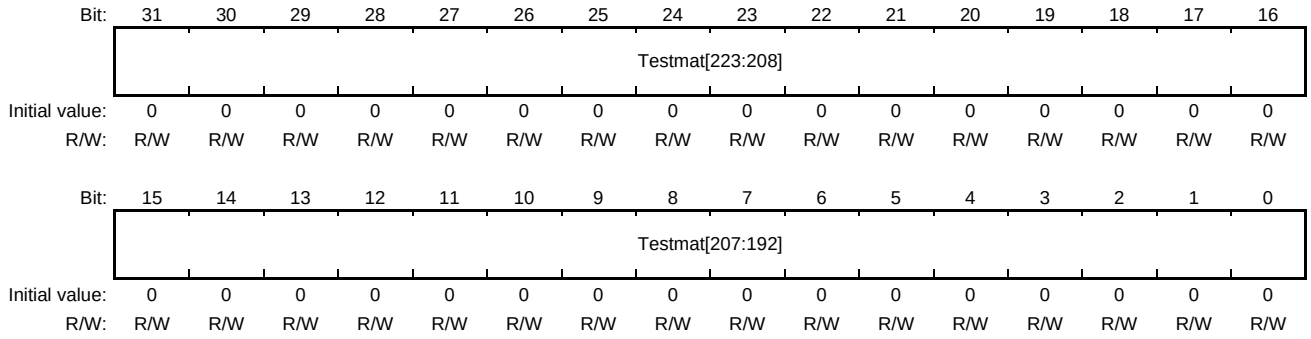
### (15) Single Step Mode setting register38 (Memory BIST condition) (RTTSET38)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Testmat[191:160] | All 0 | R/W | Test setting input. (Test target bridge selection of MBIST) |

Write the values separately provided by Renesas Electronics.

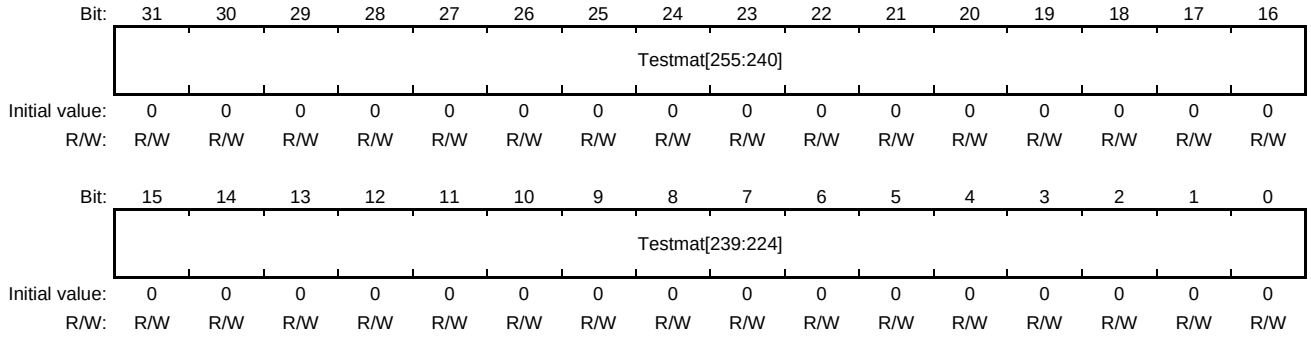
### (16) Single Step Mode setting register39 (Memory BIST condition) (RTTSET39)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Testmat[223:192] | All 0 | R/W | Test setting input. (Test target bridge selection of MBIST) |

Write the values separately provided by Renesas Electronics.

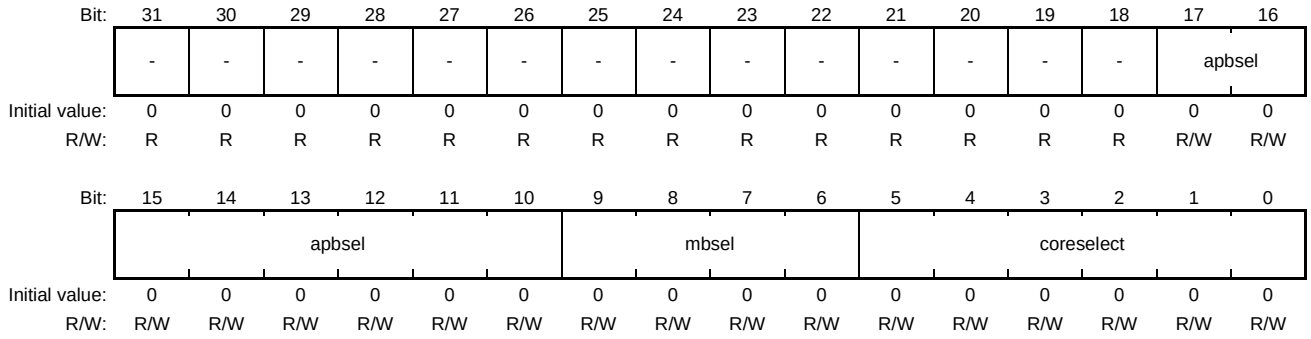
### (17) Single Step Mode setting register40 (Memory BIST condition) (RTTSET40)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Testmat[255:224] | All 0 | R/W | Test setting input. (Test target bridge selection of MBIST) |

Write the values separately provided by Renesas Electronics.

### (18) Single Step Mode setting register41 (Memory BIST condition) (RTTSET41)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | - | All 0 | R | Reserved<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 17 to 10 | apbsel | All 0 | R/W | Test setting input. (Test target APG selection of MBIST) |
| 9 to 6 | mbsel | All 0 | R/W | Test setting input. (Test domain selection of MBIST) |
| 5 to 0 | coreselect | All 0 | R/W | Test setting input. (MBIST target CORE selection) |

Write the values separately provided by Renesas Electronics.

**(19) Single Step Mode setting register42 (Memory BIST condition) (RTTSET42)**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | Y address | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | X address | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | - | All 0 | R | Reserved<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 29 to 16 | Y address | All 0 | R/W | Test setting input. (Y address of MBIST) |
| 15 to 14 | - | All 0 | R | Reserved<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 13 to 0 | X address | All 0 | R/W | Test setting input. (X address of MBIST) |

Write the values separately provided by Renesas Electronics.

**(20) Single Step Mode setting register43 (Memory BIST condition) (RTTSET43)**

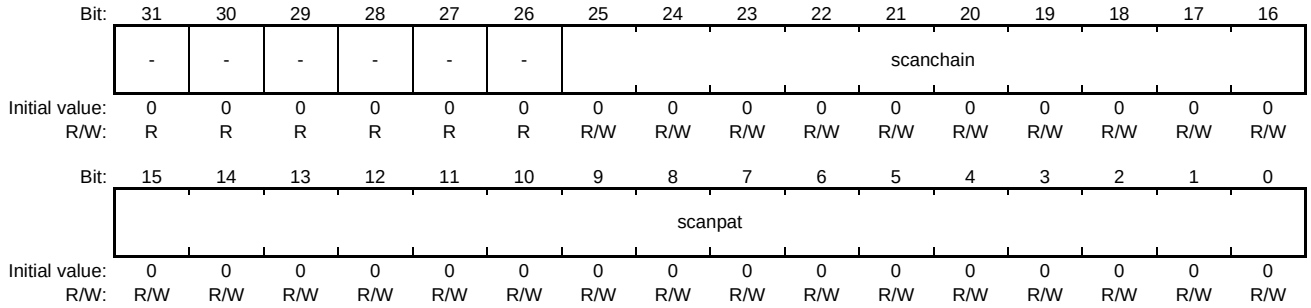
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | XY | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | XY | | Z address | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | - | 0 | R | Reserved<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 30 to 14 | XY | All 0 | R/W | Test setting input. (XY of multiplication of MBIST) |
| 13 to 0 | Z address | All 0 | R/W | Test setting input. (Z address of MBIST) |

Write the values separately provided by Renesas Electronics.

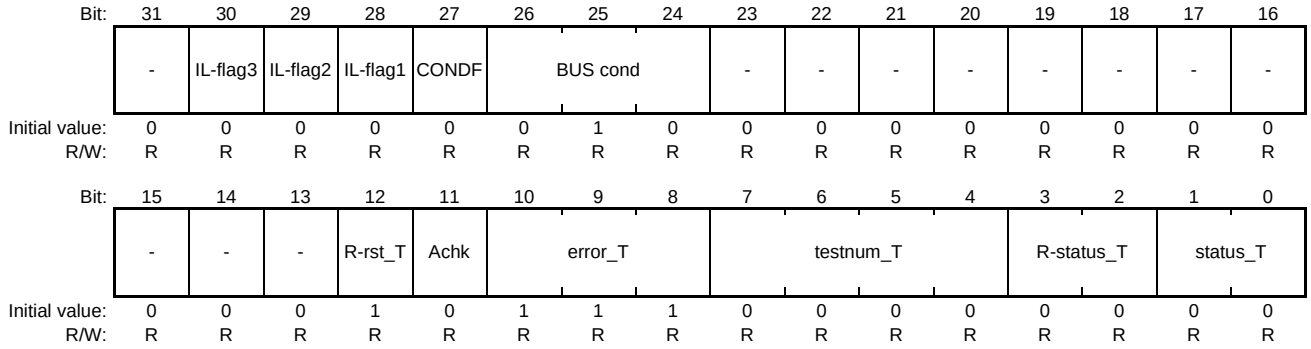
### (21)Single Step Mode setting register44 (SCAN test condition) (RTTSET44)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 25 to 16 | scanchain | All 0 | R/W | Test setting input for hardware-SCAN. (Chain length) |
| 15 to 0 | scanpat | All 0 | R/W | Test setting input for hardware-SCAN. (Pattern count) |

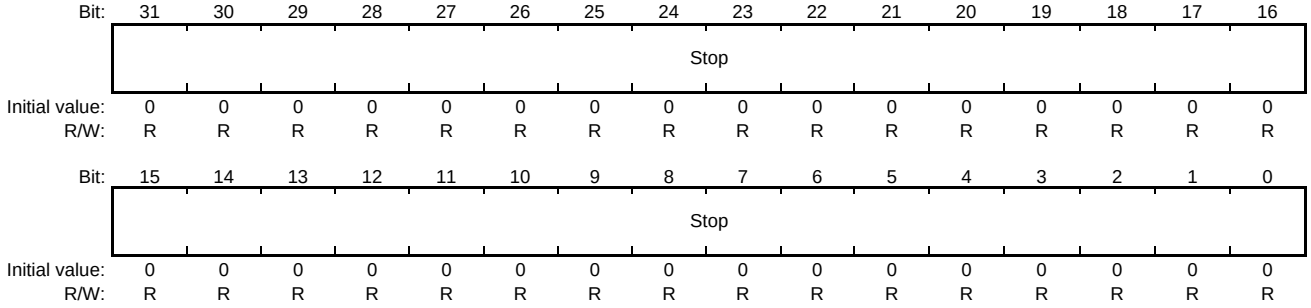
Write the values separately provided by Renesas Electronics.

### (22)Total Test result for Runtime Test (RTTRSTT)



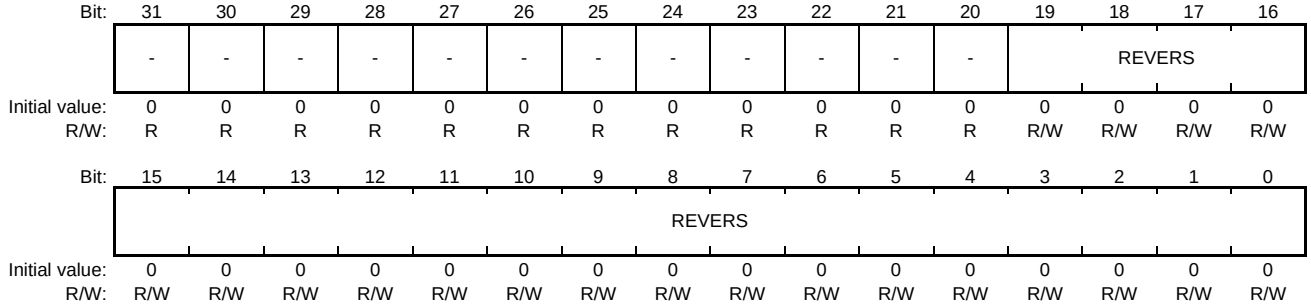

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | - | 0 | R | Reserved.<br>The bit is always read as 0. The write value should always be 0. |
| 30 | IL-flag3 | 0 | R | Runtime Test Illegal suspension flag-3 |
| 29 | IL-flag2 | 0 | R | Runtime Test Illegal suspension flag-2 |
| 28 | IL-flag1 | 0 | R | Runtime Test Illegal suspension flag-1 |
| 27 | CONDF | 0 | R | Debug monitor of FieldBIST condition.<br>0: not response.<br>1: responded |
| 26 to 24 | BUS cond | 010 | R | Debug code of BUS condition.<br>[26] AXI BUS busy.<br>[25] Waiting for response BUS.<br>[24] SCAN pattern request. |
| 23 to 13 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 12 | R-rst_T | 1 | R | BIST Redundant result (Safety function)<br>0: Pass<br>1: Fail |
| 11 | Achk | 0 | R | Debug code of address check.<br>0: normal<br>1: warning |
| 10 to 8 | error_T | 111 | R | BIST error.<br>000: Pass.<br>001: Fail.<br>010: TimeOut.<br>100: BUS error. |
| 7 to 4 | testnum_T | 0000 | R | Number of tests. |
| 3 to 2 | R-status_T | All 0 | R | BIST redundant status. (Safety function)<br>00: unexec.<br>01: wait.<br>10: exec.<br>11: finish. |
| 1 to 0 | status_T | 00 | R | BIST status.<br>00: unexec.<br>01: wait.<br>10: exec.<br>11: finish.<br>Note:* When status_T in the RTTRSTT register is 01:wait or 10:exec, do not clear EX bit of the RTTEX register.<br>The status is in the finish state until the next RTT execute. |

### (23) Counter Value for Runtime Test (RTTCNT)



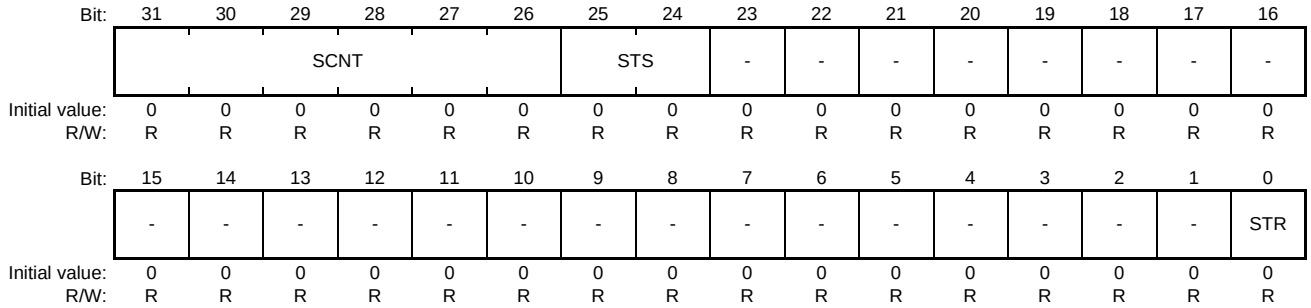

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Stop | All 0 | R | WDT (high counter) stop value. |

### (24) Expected value reverse for Self-Check (SLFRVS)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 19 to 0 | REVERS | All 0 | R/W | Expected value reverse for Self-Check SISR |

### (25) Self-check result (SLFRST)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | SCNT | All 0 | R | Self-Check counter |
| 25 to 24 | STS | 00 | R | Self-Check status<br>00: unexec<br>01: exec<br>11: finish |
| 23 to 1 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | STR | 0 | R | Self -Check Result<br>0: Pass<br>1: Fail |

Note *1: In case of the SLFR bits of the RTTEX register is 1B'1, SCNT bit returns to 1.
In case of the SLFR bits of the RTTEX register is 1B'0, SCNT bit is count up.

### (26) Dummy Error set (ERRSET)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | DSB | DST | DSF |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | - | All 0 | R | Reserved.<br>The bit or field is readable only. When writing to the register, write 0 to it. |
| 2 | DSB | 0 | R/W | Dummy set BUS-Error (set of fbc_rt_safetyerr) |
| 1 | DST | 0 | R/W | Dummy set Time out (set of fbc_rt_overerr) |
| 0 | DSF | 0 | R/W | Dummy set Fail (set of fbc_rt_fail) |

### (27) Runtime Test Finish Interrupt check1 register (RTTFINISH1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IMR0 (IMR0) | - | - | - | - | ISP0 | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | CA76SS0 Core3 | CA76SS0 Core2 | CA76SS0 Core1 | CA76SS0 Core0 | - | CA76SS0 Cluster0 | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | IMR0(IMR0) | 0 | R | finish interrupt observation Video Codec Hierarchy (IMR0(IMR0)) |
| 30 to 27 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 26 | ISP0 | 0 | R | finish interrupt observation Video IO Hierarchy (ISP0) |
| 25 to 15 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 14 | CA76SS0 Core3 | 0 | R | finish interrupt observation Cortex-A76 CPU Core 3 Hierarchy (A1 domain) |
| 13 | CA76SS0 Core2 | 0 | R | finish interrupt observation Cortex-A76 CPU Core 2 Hierarchy (A1 domain) |
| 12 | CA76SS0 Core1 | 0 | R | finish interrupt observation Cortex-A76 CPU Core 1 Hierarchy (A1 domain) |
| 11 | CA76SS0 Core0 | 0 | R | finish interrupt observation Cortex-A76 CPU Core 0 Hierarchy (A1 domain) |
| 10 | - | 0 | R | Reserved.<br>The bit is always read as 0. The write value should always be 0. |
| 9 | CA76SS0 Cluster0 | 0 | R | finish interrupt observation Cortex-A76 CPU Cluster 0 Hierarchy (A2 domain) |
| 8 to 0 | - | 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (28) Runtime Test Finish Interrupt check2 register (RTTFINISH2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | OCV3 | OCV2 | OCV1 | OCV0 | DP1 | DP0 | - | - | IMP1 | IMP0 | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SMPO | - | - | - | SMPS | - | UMFL | - | PAP | - | - | IMR3 (IMS1) | IMR2 (IMS0) | - | - | IMR1 (IMR1) |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | - | 0 | R | Reserved.<br>The bit is always read as 0. The write value should always be 0. |
| 30 | OCV3 | 0 | R | finish interrupt observation Image Recognition Hierarchy (OCV core3) |
| 29 | OCV2 | 0 | R | finish interrupt observation Image Recognition Hierarchy (OCV core2) |
| 28 | OCV1 | 0 | R | finish interrupt observation Image Recognition Hierarchy (OCV core1) |
| 27 | OCV0 | 0 | R | finish interrupt observation Image Recognition Hierarchy (OCV core0) |
| 26 | DP1 | 0 | R | finish interrupt observation Image Recognition Hierarchy (IMP DMAC1) |
| 25 | DP0 | 0 | R | finish interrupt observation Image Recognition Hierarchy (IMP DMAC0, IMP PSC0) |
| 24 to 23 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 22 | IMP1 | 0 | R | finish interrupt observation Image Recognition Hierarchy (IMP core1) |
| 21 | IMP0 | 0 | R | finish interrupt observation Image Recognition Hierarchy (IMP core0) |
| 20 to 16 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 15 | SMPO | 0 | R | finish interrupt observation Vision IP Hierarchy (SMPO) |
| 14 to 12 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 11 | SMPS | 0 | R | finish interrupt observation Vision IP Hierarchy (SMPS) |
| 10 | - | 0 | R | Reserved.<br>The bit is always read as 0. The write value should always be 0. |
| 9 | UMFL | 0 | R | finish interrupt observation Vision IP Hierarchy (Optical Flow) |
| 8 | - | 0 | R | Reserved.<br>The bit is always read as 0. The write value should always be 0. |
| 7 | PAP | 0 | R | finish interrupt observation Vision IP Hierarchy (PAP) |
| 6 to 5 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 4 | IMR3(IMS1) | 0 | R | finish interrupt observation Video Codec Hierarchy (IMR3(IMS1)) |
| 3 | IMR2(IMS0) | 0 | R | finish interrupt observation Video Codec Hierarchy (IMR2(IMS0)) |
| 2 to 1 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 0 | IMR1(IMR1) | 0 | R | finish interrupt observation Video Codec Hierarchy (IMR1(IMR1)) |

### (29) Runtime Test Finish Interrupt check3 register (RTTFINISH3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | CNRAM | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSP1 | DSP0 | - | - | - | - | - | - | - | - | - | - | CNN | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 21 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 20 | CNRAM | 0 | R | finish interrupt observation Image Recognition Hierarchy (CNRAM) |
| 19 to 16 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 15 | DSP1 | 0 | R | finish interrupt observation Image Recognition Hierarchy (DSP1) |
| 14 | DSP0 | 0 | R | finish interrupt observation Image Recognition Hierarchy (DSP0) |
| 13 to 4 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 3 | CNN | 0 | R | finish interrupt observation Image Recognition Hierarchy (IMP CNN) |
| 2 to 0 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |

### (30) Power-On Self Test Clock Frequency Control Register (POSTCKCR)

POSTCKCR is a 32-bit readable/writable register. This register controls the Power-On Self Test clock (POSTCK).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ~MD25 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | ~MD25 | R/W | Clock Stop<br>0: Supplies POSTCK clock<br>1: Stops POSTCK clock |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV[5:0] | 00 1011 | R/W | Division Ratio<br>These bits set the frequency division ratio of POSTCK clock.<br>00 1011 : 66.66MHz<br>Others : Setting prohibited |

This register is not used for Runtime Test.

### (31) Power-On Self Test 2 Clock Frequency Control Register (POST2CKCR)

POST2CKCR is a 32-bit readable/writable register. This register controls the Power-On Self Test clock (POST2CK).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ~MD25 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | ~MD25 | R/W | Clock Stop<br>0: Supplies POST2CK clock<br>1: Stops POST2CK clock |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV[5:0] | 00 0101 | R/W | Division Ratio<br>These bits set the frequency division ratio of POST2CK clock.<br>00 0101 : 133.33MHz<br>Others : Setting prohibited |

This register is for the Cortex-A76 Hierarchy.
This register is also used for Runtime Test.

### (32) Power-On Self Test 4 Clock Frequency Control Register (POST4CKCR)

POST4CKCR is a 32-bit readable/writable register. This register controls the Power-On Self Test clock (POST4CK).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | CKSTP | - | - | DIV[5:0] | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ~MD25 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 9 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 8 | CKSTP | ~MD25 | R/W | Clock Stop<br>0: Supplies POST4CK clock<br>1: Stops POST4CK clock |
| 7 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. |
| 5 to 0 | DIV[5:0] | 00 1011 | R/W | Division Ratio<br>These bits set the frequency division ratio of POST4CK clock.<br>00 0101 : 133.33MHz<br>00 1011 : 66.66MHz<br>Others : Setting prohibited |

This register is for the Vision IP hierarchy.
This register is also used for Runtime Test.

#### 171.2.2.2 Runtime Test for AP-System Core

**(A) Overview**

To do Runtime Test for AP-System Core, using APMU and AP-System Core is necessary.

**(B) Related Registers**

To do Runtime Test for AP-System Core, the following registers of APMU have to be used. For more details of each register, refer to Section 10.

**Table 171.10 Register Configuration of Runtime Test for AP-System Core**

| Register Name | Abbreviation |
|---|---|
| Power Control Register for Cluster 0 | PWRCTRLCL0 |
| Safety Control Register for Cluster 0 | SAFECTRLCL0 |
| Power Control Register for Core z | PWRCTRLCz |
| Safety Control Register for Core z | SAFECTRLCz |

z=[0-3]

**(C) Hardware Description**

When Cortex-A76 CPU core's Runtime Test is done, a power mode of Cortex-A76 CPU core goes to Pseudo OFF mode from ON mode. At that time, L1/L2 cache in Cortex-A76 CPU core is flushed automatically. So, program code for cache flush operation is not necessary. In other words, cache flush operation cannot be skipped.

When Cortex-A76 CPU cluster's Runtime Test is done, a power mode of Cortex-A76 CPU cluster goes to Pseudo OFF mode or MEM_RET mode. These two power modes can be selected by user before Runtime Test is started. If a power mode of Cortex-A76 CPU cluster goes to Pseudo OFF mode, L3 cache in Cortex-A76 CPU cluster is flushed automatically. If a power mode of Cortex-A76 CPU cluster goes to MEM_RET mode, L3 cache in Cortex-A76 CPU cluster is preserved.

### 171.2.2.3 Self-Check for Field-BIST

Figure 171.3 shows the Self-Check section of Field-BIST judgement circuit.
The judgement circuit compares SISR and expected SISR value and notifies error.

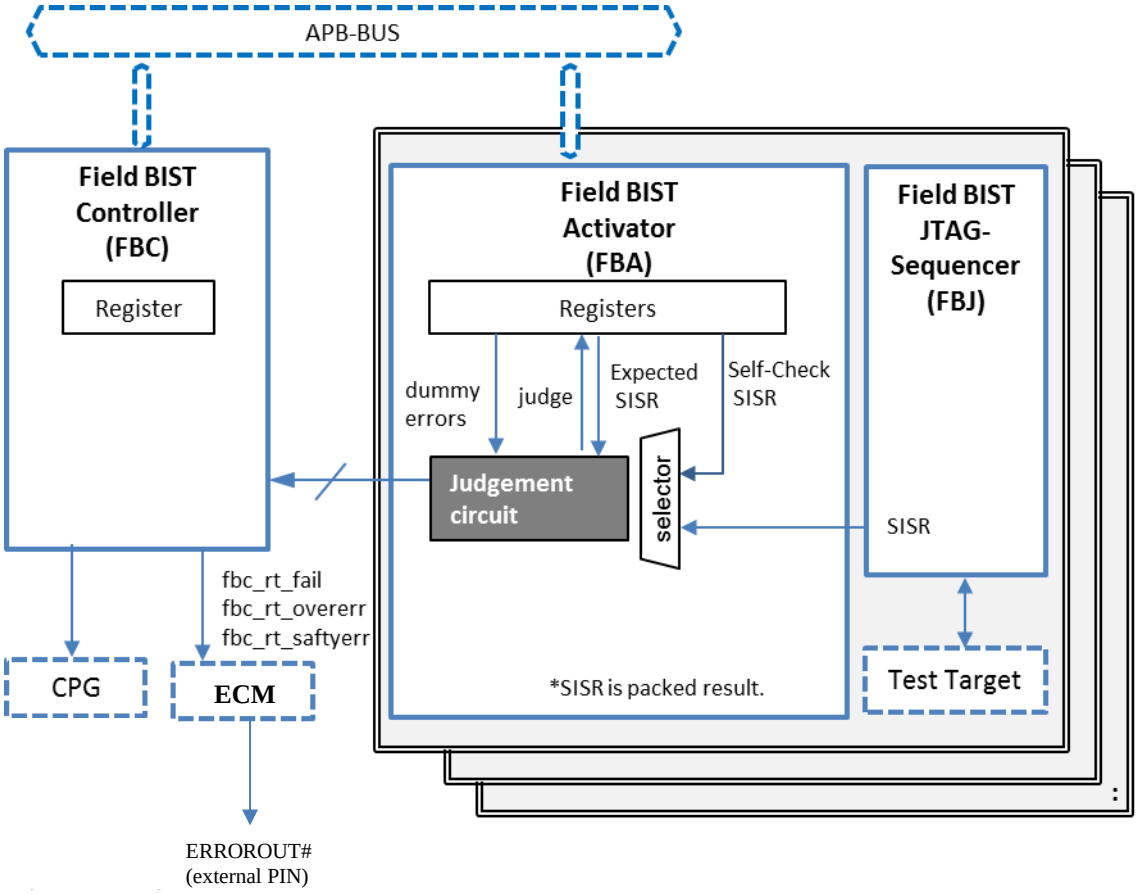



**Figure 171.3 Overview of Field-BIST judgement circuit**

Field BIST Self-Check Handling Process.

(1) Write an arbitrary value to SISR1 bit of RTTSISR1 register and decide Expected SISR.
(2) Write the inverted bit in REVERS bits of SLFRVS register to determine the Self-Check SISR value.
Self-Check SISR is an exclusive OR of SISR1 bits and REVERS bits.
In case REVERS bits is ALL 0, it is Pass expectation.
In cases where REVERS bits is other than ALL 0, it is Fail expectation.
(3) Self-check is executed when SLFE bit of RTTEX register writes B'1 and KCD bits writes H'F1.
(4) When STS bits of SLFRST register is B'11, the value of STR bit is valid for Self-Check result.
In the case where the SLFR bit of RTTEX register is B'1, SCNT bits returns to H'1.
In the case where the SLFR bit of RTTEX register is B'0, SCNT bits is counted up.
In the case where Self-Check is Fail, an error interrupt of fbc_rt_fail is asserted.
(5) Write ALL 0 to RTTEX register and end Self-Check.
De-assert error interrupt of fbc_rt_fail.

#### 171.2.2.4 Error injection

Confirm the fault of the error interrupt for FieldBIST.

Field BIST Error injection Handling Process.

(1) Error injection is executed by writing 1B'1 to each bit of ERRSET register.
In the case where DSF bit is B'1, fbc_rt_fail of error interrupt is asserted.
In the case where DST is B'1, fbc_rt_overerr of error interrupt is asserted.
In the case where DSB is B'1, fbc_rt_safetyerr of error interrupt is asserted.

(2) Write ALL 0 to ERRSET and finish Error injection.
De-assert each error interrupt.

# 200. Electrical Characteristics

## 200.1 Absolute Maximum Ratings

**Table 200.1 Absolute Maximum Ratings**

| Item | Value | Unit | Remarks |
|---|---|---|---|
| Power supply voltage (3.3 V)<br>(VDDQ33,<br>VDDQ18_33_SDHI,<br>VDDQ18_33_I2C,<br>VDDQ18_33_SPI) | -0.3 to +3.84 | V | |
| Power supply voltage (2.5 V)<br>(VDDQ18_25_AVB <for AVB pins>) | -0.3 to +2.9 | V | |
| Power supply voltage (1.8 V)<br>(VDDQ18,<br>VDDQ18_PCIEn (n = 0),<br>VDDQ18_CSIn (n=0,1),<br>VDDQ18_DSIn (n=0,1)<br>VDDQ18_25_AVB <for AVB pins>,<br>VDDQ18_33_SDHI,<br>VDDQ18_33_I2C,<br>VDDQ18_33_SPI,<br>VDD18_OCO,<br>VDD18_OTP,<br>VDD18_TRNG,<br>VDD18_CPGPLLn (n=1 to 7)) | -0.3 to +2.34 | V | |
| Power supply voltage<br>(0.6V) (VDDQVA_DDRn (n=0,1)) | -0.3 to +0.96 | V | LPDDR4X |
| Power supply voltage<br>(1.1 V) (VDDQX_DDRn (n=0,1)) | -0.3 to +1.32 | V | LPDDR4X |
| Power supply voltage<br>(0.5V) (VDDQVA_DDRn (n=0,1)) | -0.3 to +0.96 | V | LPDDR5 |
| Power supply voltage<br>(1.05 V) (VDDQX_DDRn (n=0,1)) | -0.3 to +1.32 | V | LPDDR5 |
| Power supply voltage (0.75 V)<br>(VDD<br>VDD_PCIEn (n=0),<br>VDD_CSIn (n=0,1),<br>VDD_DSIn (n=0,1),<br>VDD_DSIPLLn (n=0,1),<br>VDD_DDRPLLn (n=01)) | -0.3 to +1.05 | V | |
| Input voltage (3.3-V I/O) | -0.3 to VDDQ33 + 0.3 | V | |
| Input voltage (3.3-V I/O [SDHI]) | -0.3 to VDDQ18_33_SDHI + 0.3 | V | |
| Input voltage (3.3-V I/O [I2C]) | -0.3 to VDDQ18_33_I2C + 0.3 | V | |
| Input voltage (3.3-V I/O [SPI]) | -0.3 to VDDQ18_33_SPI + 0.3 | V | |
| Input voltage (2.5-V I/O [ETH]<AVB>) | -0.3 to VDDQ18_25_AVB + 0.3 | V | |
| Input voltage (1.8-V I/O) [SDHI]) | -0.3 to VDDQ18_33_SDHI + 0.3 | V | |
| Input voltage (1.8-V I/O) [I2C]) | -0.3 to VDDQ18_33_I2C + 0.3 | V | |
| Input voltage (1.8-V I/O) [SPI]) | -0.3 to VDDQ18_33_SPI + 0.3 | V | |
| Input voltage (1.8-V I/O) | -0.3 to VDDQ18 + 0.3 | V | |

| Item | Value | Unit | Remarks |
|---|---|---|---|
| Input voltage (1.8-V I/O [PCIEn (n = 0)]) | -0.4 to VDDQ18_PCIEn + 0.3 (n = 0) | V | |
| Input voltage (1.2-V I/O [CSIn (n=0,1)]) | -0.15 to 1.45 (n = 0,1) | V | |
| Input voltage (1.2-V I/O [DSI n (n=0,1)]) | -0.15 to 1.45 (n = 0,1) | V | |
| Input voltage (0.6-V I/O [VDDQVA_DDRn (n=0,1)]) | -0.3 to VDDQVA_DDRn + 0.3 (n=0,1) | V | LPDDR4X |
| Input voltage (1.1-V I/O [VDDQX_DDRn (n=0,1)]) | -0.3 to 1.32 | V | LPDDR4X |
| Input voltage (0.5-V I/O [VDDQVA_DDRn (n=0,1)]) | -0.3 to VDDQVA_DDRn + 0.3 (n=0,1) | V | LPDDR5 |
| Input voltage (1.05-V I/O [VDDQX_DDRn (n=0,1)]) | -0.3 to 1.32 | V | LPDDR5 |
| Output voltage (3.3-V I/O) | -0.3 to VDDQ33 + 0.3 | V | |
| Output voltage (3.3-V I/O [SDHI]) | -0.3 to VDDQ18_33_SDHI + 0.3 | V | |
| Output voltage (3.3-V I/O [I2C]) | -0.3 to VDDQ18_33_I2C + 0.3 | V | |
| Output voltage (3.3-V I/O [SPI]) | -0.3 to VDDQ18_33_SPI + 0.3 | V | |
| Output voltage (2.5-V I/O [ETH]<AVB>) | -0.3 to VDDQ18_25_AVB + 0.3 | V | |
| Output voltage (1.8-V I/O) [SDHI]) | -0.3 to VDDQ18_33_SDHI + 0.3 | V | |
| Output voltage (1.8-V I/O) [I2C]) | -0.3 to VDDQ18_33_I2C + 0.3 | V | |
| Output voltage (1.8-V I/O) [SPI]) | -0.3 to VDDQ18_33_SPI + 0.3 | V | |
| Output voltage (1.8-V I/O) | -0.3 to VDDQ18 + 0.3 | V | |
| Output voltage (1.8-V I/O [PCIEn (n = 0)]) | -0.3 to VDDQ18_PCIEn + 0.3 (n = 0) | V | |
| Output voltage (1.2-V I/O [DSIn (n = 0 to 1)]) | -0.15 to 1.45 (n = 0 to 1) | V | |
| Output voltage (0.6-V I/O [VDDQVA_DDRn (n=0,1)]) | -0.3 to VDDQVA_DDRn + 0.3 (n=0,1) | V | LPDDR4X |
| Output voltage (1.1-V I/O [VDDQX_DDRn (n=0,1)]) | -0.3 to 1.32 | V | LPDDR4X |
| Output voltage (0.5-V I/O [VDDQVA_DDRn (n=0,1)]) | -0.3 to VDDQVA_DDRn + 0.3 (n=0,1) | V | LPDDR5 |
| Output voltage (1.05-V I/O [VDDQX_DDRn (n=0,1)]) | -0.3 to 1.32 | V | LPDDR5 |
| Operating temperature | -40 to +125 | °C | Tj (junction) |
| Storage temperature | -55 to +125 | °C | Ta (ambient) |

Notes: Permanent damage to the LSI may result if absolute maximum ratings are exceeded. In normal operation, this LSI shall be used within the specifications described in the following descriptions. If this LSI is not used within the specifications, the reliability of this LSI may lower.

Voltages are referenced at GND = VSS = AVSS = 0 V.

## 200.2 Power Supply

**Table 200.2 Power Supply**

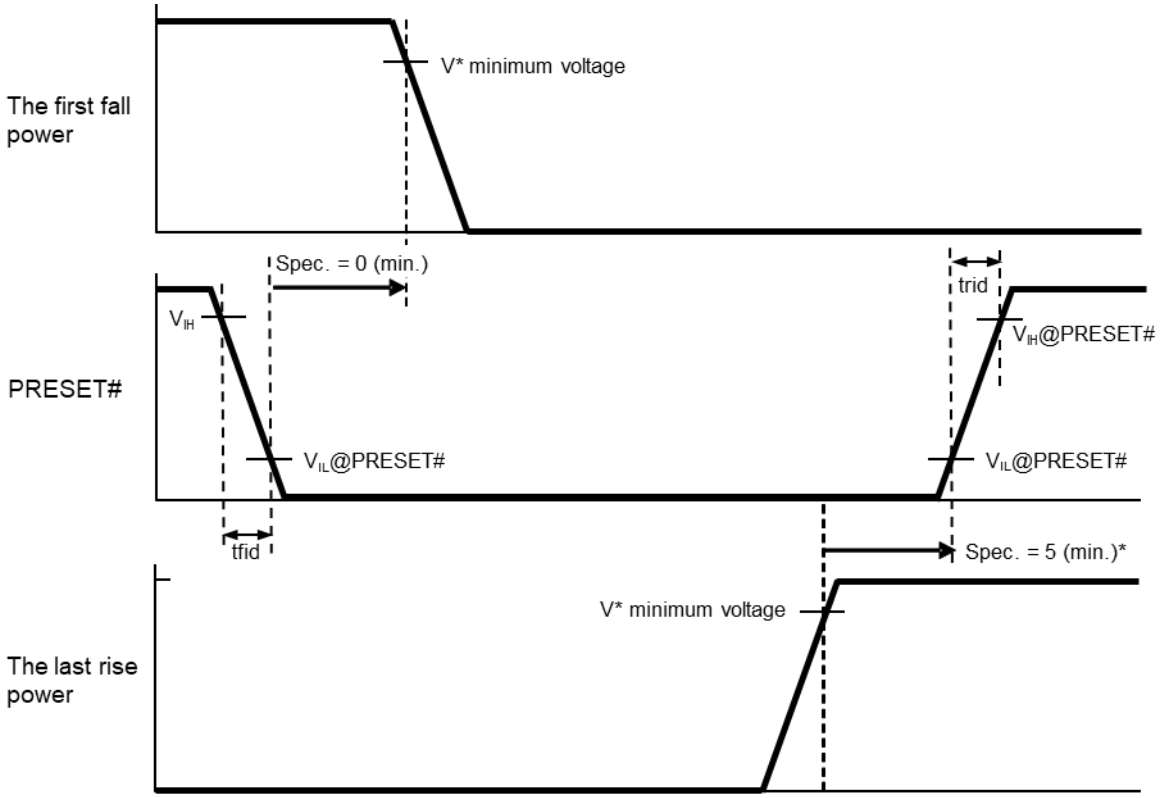
| Item | Symbol | Voltage | | | Unit | Remarks |
|---|---|---|---|---|---|---|
| | | Min. | Typ. | Max. | | |
| Power supply (DC+AC) | VDD_AVS0 | 0.710 | 0.765 | 0.825 | V | From the power estimation, the lowest limit of the supply voltage at the SOC ball terminal is narrowed regarding the IR drop. |
| | VDD_AVS1 | 0.700 | 0.755 | 0.815 | | |
| | VDD_AVS2 | 0.675 | 0.730 | 0.790 | | |
| Power supply (DC) | VDD_dc_AVS0 | 0.740 | 0.765 | 0.795 | V | Based on the power estimation, this product requires Feedback control using the PMIC remote sense signal.<br>This is the recommended DC setting value for PMIC.<br>Long-term reliability is guaranteed at the maximum voltage of VDD_dc_AVS0 (0.795V). |
| | VDD_dc_AVS1 | 0.730 | 0.755 | 0.785 | | |
| | VDD_dc_AVS2 | 0.705 | 0.730 | 0.760 | | |
| Power supply (AC) | VDD_ac | -0.03 | — | 0.03 | V | The AC fluctuation due to the load fluctuation. |
| Power supply (3.3-V I/O) | VDDQ33 | 3.1 | 3.3 | 3.5 | V | VDDQ33 for power supply and VSS for ground |
| Power supply (2.5-V I/O [ETH]<AVB>) | VDDQ18_25_AVB | 2.4 | 2.5 | 2.6 | V | VDDQ18_25_AVB for power supply and VSS for ground |
| Power supply (1.8-V I/O [ETH]<AVB>) | VDDQ18_25_AVB | 1.7 | 1.8 | 1.9 | V | VDDQ18_25_AVB for power supply and VSS for ground |
| Power supply (1.8-V I/O [SDHI]) | VDDQ18_33_SDHI | 1.7 | 1.8 | 1.9 | V | VDDQ18_33_SDHI for power supply and VSS for ground |
| Power supply (3.3-V I/O [SDHI]) | VDDQ18_33_SDHI | 3.1 | 3.3 | 3.5 | V | VDDQ18_33_SDHI for power supply and VSS for ground |
| Power supply (1.8-V I/O [I2C]) | VDDQ18_33_I2C | 1.7 | 1.8 | 1.9 | V | VDDQ18_33_I2C for power supply and VSS for ground |
| Power supply (3.3-V I/O [I2C]) | VDDQ18_33_I2C | 3.1 | 3.3 | 3.5 | V | VDDQ18_33_I2C for power supply and VSS for ground |
| Power supply (1.8-V I/O [SPI]) | VDDQ18_33_SPI | 1.7 | 1.8 | 1.9 | V | VDDQ18_33_SPI for power supply and VSS for ground |
| Power supply (3.3-V I/O [SPI]) | VDDQ18_33_SPI | 3.1 | 3.3 | 3.5 | V | VDDQ18_33_SPI for power supply and VSS for ground |
| Power supply (1.8-V I/O [OCO]) | VDD18_OCO | 1.7 | 1.8 | 1.9 | V | VDD18_OCO for power supply and VSS for ground |
| Power supply (1.8-V I/O [OTP]) | VDD18_OTP | 1.7 | 1.8 | 1.9 | V | VDD18_OTP for power supply and VSS for ground |
| Power supply (1.8-V I/O [TRNG]) | VDD18_TRNG | 1.7 | 1.8 | 1.9 | V | VDD18_TRNG for power supply and VSS for ground |
| Power supply (1.8-V I/O [CPGPLL]) | VDD18_CPGPLLn (n=1 to 7) | 1.7 | 1.8 | 1.9 | V | VDD18_CPGPLLn (n=1 to 7) for power supply and VSS for ground |
| Power supply (1.8-V I/O [VDDQ18]) | VDDQ18 | 1.7 | 1.8 | 1.9 | V | VDDQ18 for power supply and VSS for ground |
| Power supply (1.8-V I/O [PCIE]) | VDDQ18_PCIEn (n = 0) | 1.7 | 1.8 | 1.9 | V | VDDQ18_PCIEn (n = 0) for power supply and VSS for ground |

| Item | Symbol | Voltage | | | Unit | Remarks |
|---|---|---|---|---|---|---|
| | | Min. | Typ. | Max. | | |
| Power supply (1.2-V I/O [CSI]) | VDDQ18_CSIn (n = 0 to 1) | 1.7 | 1.8 | 1.9 | V | VDDQ18_CSI n (n = 0 to 1) for power supply and VSS for ground |
| Power supply (1.2-V I/O [DSI]) | VDDQ18_DSIn (n = 0 to 1) | 1.7 | 1.8 | 1.9 | V | VDDQ18_DSI n (n = 0 to 1) for power supply and VSS for ground |
| Power supply (0.6-V I/O [DDR (LPDDR4X)]) | VDDQVA_DDRn (n=0,1) | 0.57 | 0.6 | 0.65 | V | VDDQVA_DDRn (n=0,1) for power supply and VSS for ground |
| Power supply (1.1-V I/O [DDR (LPDDR4X)]) | VDDQX_DDRn (n=0,1) | 1.06 | 1.10 | 1.17 | V | VDDQX_DDRn (n=0,1) for power supply and VSS for ground |
| Power supply (0.5-V I/O [DDR (LPDDR5)]) | VDDQVA_DDRn (n=0,1) | 0.47 | 0.5 | 0.57 | V | VDDQVA_DDRn (n=0,1) for power supply and VSS for ground |
| Power supply (1.05-V I/O [DDR (LPDDR5)]) | VDDQX_DDRn (n=0,1) | 1.01 | 1.05 | 1.12 | V | VDDQX_DDRn (n=0,1) for power supply and VSS for ground |
| Power supply (0.75-V I/O [PCI]) | VDD_PCI0 | *1 | *1 | *1 | V | VDD_PCIE0 for power supply and VSS for ground |
| Power supply (0.75-V I/O [CSI]) | VDD_CSIn (n=0,1) | *1 | *1 | *1 | V | VDD_CSIn (n=0,1) for power supply and VSS for ground |
| Power supply (0.75-V I/O [DSI]) | VDD_DSIn (n=0,1) | *1 | *1 | *1 | V | VDD_DSIn (n=0,1) for power supply and VSS for ground |
| Power supply (0.75-V I/O [DSIPLL]) | VDD_DSIPLLn (n=0,1) | *1 | *1 | *1 | V | VDD_DSIPLLn (n=0,1) for power supply and VSS for ground |
| Power supply (0.75-V I/O [DDRPLL]) | VDD_DDRPLL01 | *1 | *1 | *1 | V | VDD_DDRPLL01 for power supply and VSS for ground |

Notes:*1...Refer to VDD_dc_AVSn(n=0 to 2)

## 200.3 Sequence of Turning On/Off Power Supplies

### (1) PRESET# VS. Power

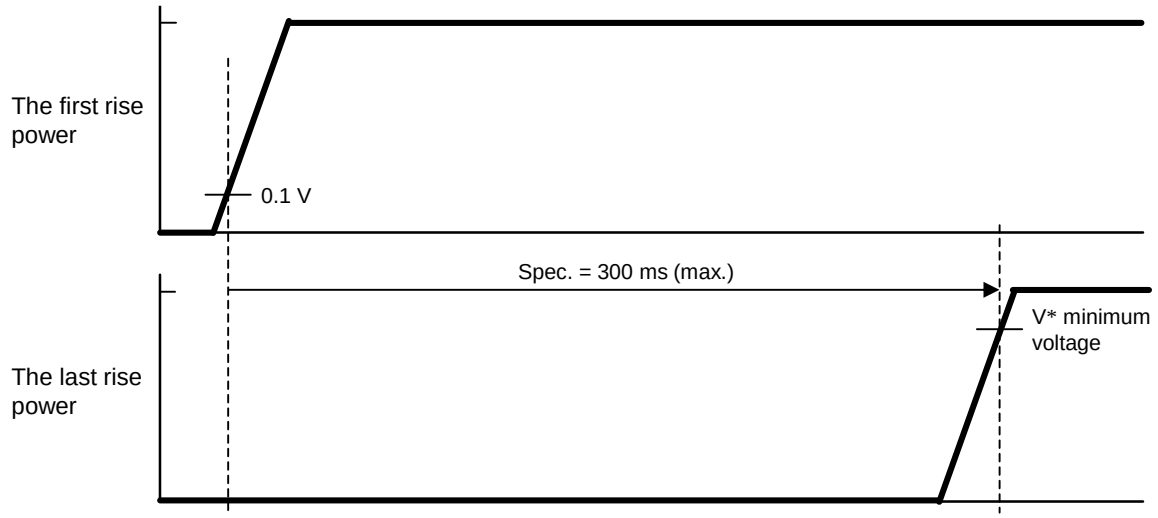



*PRESET# negate timing is defined by "tOSC + T". For details, see chapter 200.6.1 Clock and Reset Timings

**Table 200.3 PRESET# Input Timing**

| Item | Symbol | Min. | Typ. | Max. | Unit | Remarks |
|---|---|---|---|---|---|---|
| PRESET# Input fall time | tfid | — | — | 4 | us | — |
| PRESET# Input rise time | trid | — | — | 4 | us | — |

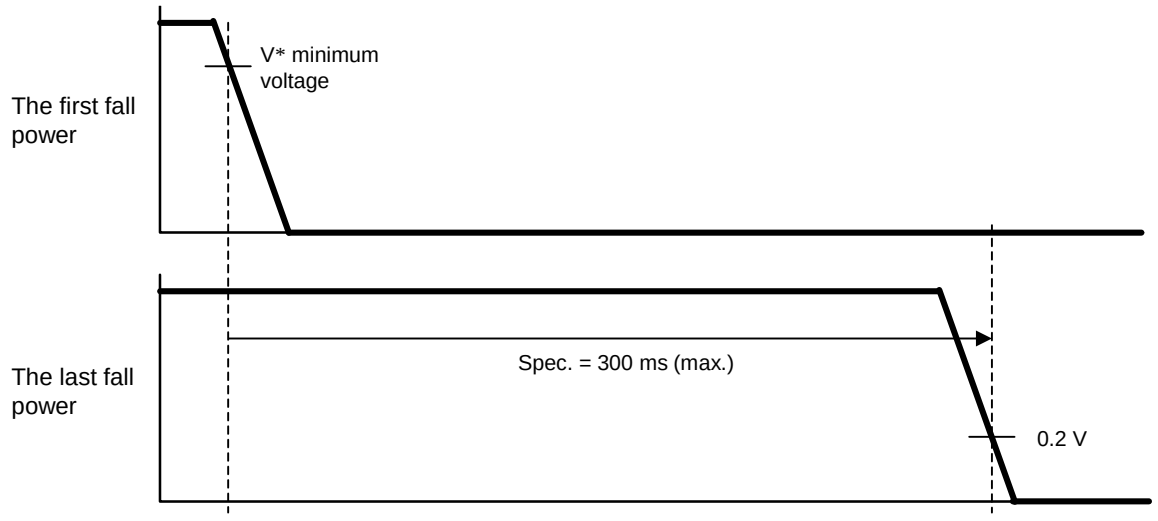
**(2) Period for Power Rise**




For all V*, power off state must be 0 V (GND), power rising must be started from 0 V, and end to 0 V respectively. (excluding DDR-SDRAM power-supply backup state)
Period from 0 V to 0.1V at power-on and from 0.2 V to 0 V at power-off should be shorten as much as possible.

*) Please also refer to the Power-on/off sequence of DDR-SDRAM datasheet.

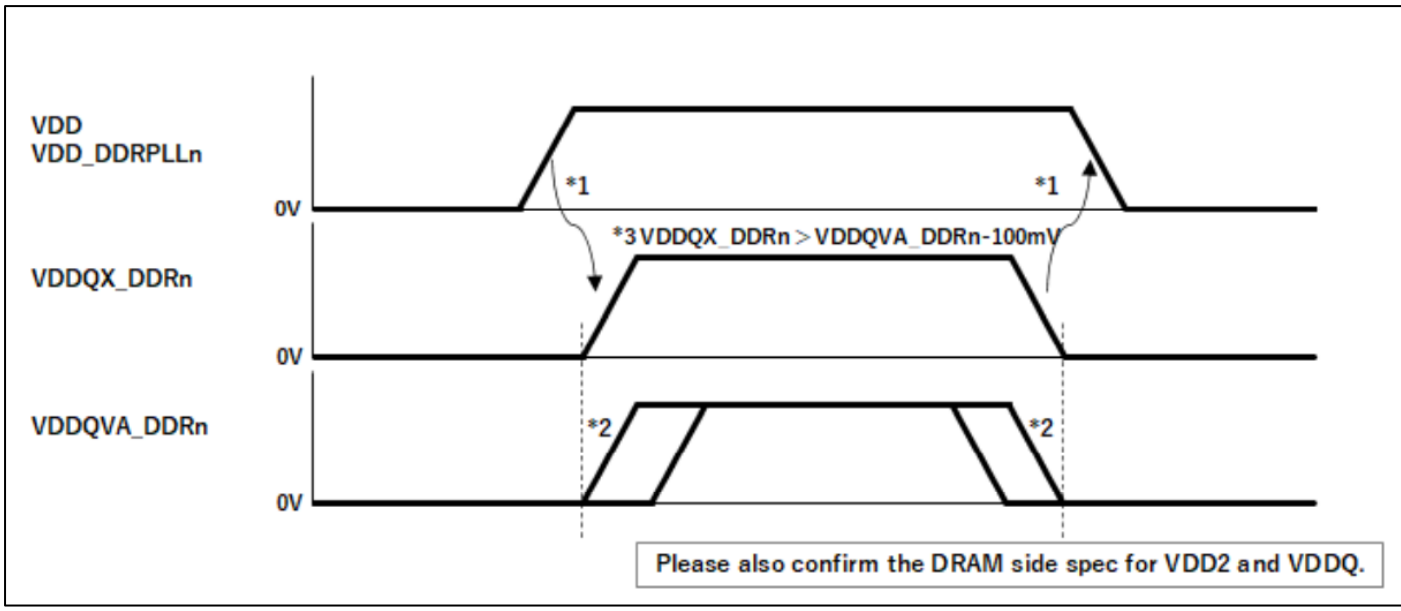
**(3) Period for Power Fall**




For all V*, power off state must be 0 V (GND), power rising must be started from 0 V, and end to 0 V respectively. (excluding DDR-SDRAM power-supply backup state)
Period from 0 V to 0.1V at power-on and from 0.2 V to 0 V at power-off should be shorten as much as possible.

*) Please also refer to the Power-on/off sequence of DDR-SDRAM datasheet.

**(4) VDD and VDD_DDRPLLn vs VDDQX_DDRn vs VDDQVA_DDRn (n=0,1)**




*1:VDD power-on must be earlier than VDDQX_DDRn, and VDD power-off must be later than VDDQX_DDRn. (n=0,1)
*2:VDDQX_DDRn power-on must be earlier than VDDQVA_DDRn, and VDDQX_DDRn power-off must be later than VDDQVA_DDRn. (n=0,1)
*3:VDDQX_DDRn must be greater than VDDQVA_DDRn-100mV. (n=0,1)

## 200.4 Output Voltage Characteristics

**(1) V/xDrive strength Table**

**Table 200.4 Drive strength and setting register(3.3V IO, 2.5/1.8V IO, 3.3/1.8V IO)**

| Drive Strength | Drive Capacitance | Resister bit | | |
|---|---|---|---|---|
| | | DRV3 | DRV2 | DRV1 |
| 2 | 2/8 | 0 | 0 | 1 |
| 3 | 3/8 | 0 | 1 | 0 |
| 4 | 4/8 | 0 | 1 | 1 |
| 5 | 5/8 | 1 | 0 | 0 |
| 6 | 6/8 | 1 | 0 | 1 |
| 7 | 7/8 | 1 | 1 | 0 |
| 8 | 8/8 | 1 | 1 | 1 |

**(2) V/xDrive strength Table**

**Table 200.5 Drive strength and setting register(1.8V IO)**

| Drive Strength | Drive Capacitance | Resister bit | |
|---|---|---|---|
| | | DRV2 | DRV1 |
| 1 | 1/4 | 0 | 0 |
| 2 | 2/4 | 0 | 1 |
| 3 | 3/4 | 1 | 0 |
| 4 | 4/4 | 1 | 1 |

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

## 200.5 DC Characteristics

### 200.5.1 Supply Current

**Table 200.6 Supply Current**

| Item | | Symbol | Power Min. | Typ. | Max. | Unit | Remarks |
|---|---|---|---|---|---|---|---|
| Supply current (Internal) | | IDD | — | — | 23520 | mA | VDD = VDD_dc(Max) |
| Supply current (3.3-V I/O) | | IDDQ33 | — | — | 35 | mA | VDDQ33 (Max) |
| Supply current (3.3-V I/O [SDHI]) | | IDDQ18_33_SDHI | — | — | 25 | mA | VDDQ18_33_SDHI=3.5V |
| Supply current (3.3-V I/O [SPI]) | | IDDQ18_33_SPI | — | — | 3 | mA | VDDQ18_33_SPI=3.5V |
| Supply current (3.3-V I/O [I3C]) | | IDDQ18_33_I2C | — | — | 1 | mA | VDDQ18_33_I2C=3.5V |
| Supply current (2.5-V I/O [ETH]<AVB>) | | IDDQ18_25_ETH | — | — | 300 | mA | VDDQ18_25_AVB=2.6V |
| Supply current (1.8-V I/O [ETH]<AVB>) | | IDDQ18_25_ETH | — | — | 250 | mA | VDDQ18_25_AVB=1.9V |
| Supply current (1.8-V I/O [VDDQ18]) | | IDDQ18 | — | — | 35 | mA | VDDQ18(Max) |
| Supply current (1.8-V I/O [PCIE]) | | IDDQ18_PCIEn (n = 0) | — | — | 150 | mA | VDDQ18_PCIEn(Max) (n = 0) |
| Supply current (1.2-V I/O [CSI]) | | IDDQ18_CSIn (n = 0,1) | — | — | 18 | mA | VDDQ18_CSIn(Max) (n = 0,1) |
| Supply current (1.2-V I/O [DSI]) | | IDDQ18_DSIn (n = 0,1) | — | — | 25 | mA | VDDQ18_DSIn(Max) (n = 0,1) |
| Supply current (1.8-V I/O [SDHI]) | | IDDQ18_33_SDHI | — | — | 14 | mA | VDDQ18_33_SDHI=1.9V |
| Supply current (1.8-V I/O [SPI]) | | IDDQ18_33_SPI | — | — | 2 | mA | VDDQ18_33_SPI=1.9V |
| Supply current (1.8-V I/O [I2C]) | | IDDQ18_33_I2C | — | — | 1 | mA | VDDQ18_33_I2C=1.9V |
| Supply current (0.6-V I/O [DDR(LPDDR4X)]) | Normal | IDDQVA_DDRn (n=0,1) | — | — | 169 | mA | VDDQVA_DDRn = 0.65 V (n=0,1) Condition: Driver impedance: 40Ω No termination Load capacitance: 2.5pF Write Read rate: 50%, 50% I/O Toggle rate: 100% |
| | DDR power supply backup | | — | — | 1.0 | mA | VDDQVA_DDRn = 0.65 V (n=0,1) |
| Supply current (1.1-V I/O [DDR(LPDDR4X)]) | Normal | IDDQX_DDRn (n=0,1) | — | — | 4.2 | mA | VDDQX_DDRn = 1.17 V (n=0,1) Condition: Driver impedance: 40Ω No termination Load capacitance: 2.5pF Write Read rate: 50%, 50% I/O Toggle rate: 100% |
| | DDR power supply backup | | — | — | 0.5 | mA | VDDQX_DDRn = 1.17 V (n=0,1) |
| Supply current (0.5-V I/O [DDR(LPDDR5)]) | Normal | IDDQVA_DDRn (n=0,1) | — | — | 156 | mA | VDDQVA_DDRn = 0.57 V (n=0,1) Condition: Driver impedance: 40Ω No termination Load capacitance: 2.5pF Write Read rate: 50%, 50% I/O Toggle rate: 100% |
| | DDR power supply backup | | — | — | 0.83 | mA | VDDQVA_DDRn = 0.57 V (n=0,1) |

| Item | | Symbol | Power | | | Unit | Remarks |
|---|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | | |
| Supply current (1.05-V I/O [DDR(LPDDR5)]) | Normal | IDDQX_DDRn (n=0,1) | — | — | 8.4 | mA | VDDQX_DDRn = 1.12 V (n=0,1) Condition: Driver impedance: 40Ω No termination Load capacitance: 2.5pF Write Read rate: 50%, 50% I/O Toggle rate: 100% |
| | DDR power supply backup | | — | — | 0.48 | mA | VDDQX_DDRn = 1.12 V (n=0,1) |
| Supply current (PLL [CPG]) | | IDD18_CPGPLLn (n = 1 to 7) | — | — | 3.3 | mA | VDDQ18_CPGPLLn = 1.9 V (n = 1 to 7) |
| Supply current (PLL [DDR(LPDDR4X)]) | | VDD_DDRPLLn (n=01) | — | — | 46 | mA | VDD_DDRPLLn = 0.825 V (n=01) |
| Supply current (PLL [DDR(LPDDR5)]) | | VDD_DDRPLLn (n=01) | | | 43 | mA | VDD_DDRPLLn = 0.825 V (n=01) |
| Supply current (1.8-V I/O [OCO]) | | IDD18_OCO | — | — | 0.06 | mA | VDD18_OCO = 1.9V |
| Supply current (1.8-V I/O [OTP]) | | IDD18_OTP | — | — | 12.6 | mA | VDD18_OTP = 1.9V |
| Supply current (1.8-V I/O [TRNG]) | | IDD18_TRNG | — | — | 1.2 | mA | VDD18_TRNG = 1.9V |
| Supply current (0.75-V I/O [PCIE]) | | IDD_PCIEn (n=0) | — | — | 241 | mA | VDD_PCIEn = 0.825V (n=0) |
| Supply current (CSI) | | IDD_CSIn (n=0,1) | — | — | 49 | mA | VDD_CSIn = 0.825V (n=0,1) |
| Supply current (DSI) | | IDD_DSIn (n=0,1) | — | — | 57 | mA | VDD_DSIn = 0.825V (n=0,1) |
| Supply current (PLL [DSI]) | | IDD_DSInPLL (n=0,1) | — | — | 15 | mA | VDD_DSInPLL = 0.825V (n=0,1) |

### 200.5.2 DC Characteristics (3.3-V I/O)

**Table 200.7 DC Characteristics (3.3-V I/O)**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | 2.0 | — | VDDQ33 + 0.3 | V | VDDQ33 = 3.1 V to 3.5 V, VSS = 0 V, VDDQ18 = 1.7 V to 1.9 V, VSS = 0 V*2 | |
| Input low voltage | VIL18 | -0.3 | — | 0.3*VDDQ18*2 | V | | PRESET# =Low*1 |
| Input low voltage | VIL33 | -0.3 | — | 0.8 | V | | PRESET# =High*1 |
| Output high voltage | VOH | 2.4 | | VDDQ33 + 0.3 | V | | IOH = - 4 mA |
| Output low voltage | VOL | -0.3 | — | 0.4 | V | | IOL = 4 mA |
| Pin capacitance | CL | — | — | 10 | pF | | |
| Input leakage current | ILI | — | — | 10 | µA | VDDQ33 = 3.1 V to 3.5 V, VSS = 0 V Without pull-up or pull-down resistor | All input pins |
| Output leakage current | ILO | __ | — | 10 | µA | | Hi-Z output*1 |
| Pull-up current | IPU | -35 | — | -115 | µA | VI = 0V With pull-up resistor (50 kohm(Typ.)) | |
| Pull-down current | IPD | 35 | — | 115 | µA | VI = VDDQ33 (3.1 to 3.5 V) With pull-down resistor (50 kohm(Typ.)) | |

Note *1 : About timing of decoding MD pin's value triggered by rising edge of PRESET#, refer to "200.6.1 Clock and Reset Timings"

*2 : Because VIL affects VDDQ18 instead of VDDQ33 only during PRESET#=Low period due to the circuit structure of the pin.

**Table 200.8 Output leakage current by V/xDrive strength (3.3-V I/O)**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 6.5 | — | — | mA | VDDQ33 = 3.1V to 3.5V, VSS = 0V without pull-up or pull-down, VOL = 0.4V, VOH = VDDQ33 – 0.4V | V/xDrive = 2/8 |
| | 10.2 | — | — | mA | | V/xDrive = 3/8 |
| | 11.1 | — | — | mA | | V/xDrive = 4/8 |
| | 12.7 | — | — | mA | | V/xDrive = 5/8 |
| | 13.3 | — | — | mA | | V/xDrive = 6/8 |
| | 14.7 | — | — | mA | | V/xDrive = 7/8 |
| | 15.6 | — | — | mA | | V/xDrive = 8/8 |

### 200.5.3 DC Characteristics (3.3-V I/O [SPI])

**Table 200.9 DC Characteristics (3.3-V I/O [SPI])**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | 2.0 | — | VDDQ18_33_SPI + 0.3 | V | VDDQ18_33_SPI = 3.1 V to 3.5 V, VSS = 0 V, VDDQ18 = 1.7 V to 1.9 V, VSS = 0 V*2 | |
| Input low voltage | VIL18 | -0.3 | — | 0.3*VDDQ18*2 | V | | PRESET# =Low*1 |
| Input low voltage | VIL33 | -0.3 | — | 0.8 | V | | PRESET# =High*1 |
| Output high voltage | VOH | 2.4 | | VDDQ18_33_SPI + 0.3 | V | | IOH = - 4 mA |
| Output low voltage | VOL | -0.3 | — | 0.4 | V | | IOL = 4 mA |
| Pin capacitance | CL | — | — | 10 | pF | | |
| Input leakage current | ILI | — | — | 10 | µA | VDDQ18_33_SPI = 3.1 V to 3.5 V, VSS = 0 V Without pull-up or pull-down resistor | All input pins |
| Output leakage current | ILO | __ | — | 10 | µA | | Hi-Z output*1 |
| Pull-up current | IPU | -35 | — | -115 | µA | VI = 0V With pull-up resistor (50 kohm(Typ.)) | |
| Pull-down current | IPD | 35 | — | 115 | µA | VI = VDDQ18_33_SPI (3.1 to 3.5 V) With pull-down resistor (50 kohm(Typ.)) | |

Note *1 : About timing of decoding MD pin's value triggered by rising edge of PRESET#, refer to "200.6.1 Clock and Reset Timings"

*2 : Because VIL affects VDDQ18 instead of VDDQ18_33_SPI only during PRESET#=Low period due to the circuit structure of the pin.

**Table 200.10 Output leakage current by V/xDrive strength (3.3-V I/O)[SPI]**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 6.5 | — | — | mA | VDDQ18_33_SPI = 3.1V to 3.5V, VSS = 0V without pull-up or pull-down, VOL = 0.4V, VOH = VDDQ18_33_SPI – 0.4V | V/xDrive = 2/8 |
| | 10.2 | — | — | mA | | V/xDrive = 3/8 |
| | 11.1 | — | — | mA | | V/xDrive = 4/8 |
| | 12.7 | — | — | mA | | V/xDrive = 5/8 |
| | 13.3 | — | — | mA | | V/xDrive = 6/8 |
| | 14.7 | — | — | mA | | V/xDrive = 7/8 |
| | 15.6 | — | — | mA | | V/xDrive = 8/8 |

### 200.5.4 DC Characteristics (3.3-V I/O [SDHI])

**Table 200.11 DC Characteristics (3.3-V I/O [SDHI])**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | 0.625*VDDQ18_33_SDHI | — | VDDQ18_33_SDHI + 0.3 | V | VDDQ18_33_SDHI = 3.1 to 3.5 V, VSS = 0 V | — |
| Input low voltage | VIL | VSS - 0.3 | — | 0.25*VDDQ18_33_SDHI | V | VDDQ18_33_SDHI = 3.1 to 3.5 V, VSS = 0 V | — |
| Output high voltage | VOH | 0.75*VDDQ18_33_SDHI | — | — | V | VDDQ18_33_SDHI = 3.1 to 3.5 V, VSS = 0 V | IOH = -2 mA VDDQ18_33_SDHI min |
| Output low voltage | VOL | — | — | 0.125*VDDQ18_33_SDHI | V | VDDQ18_33_SDHI = 3.1 to 3.5 V, VSS = 0 V | IOL = 2 mA VDDQ18_33_SDHI min |
| Pin capacitance | CL | — | — | 10 | pF | VDDQ18_33_SDHI = 3.1 to 3.5 V, VSS = 0 V | |
| Input leakage current | ILI | — | — | 10 | µA | VDDQ18_33_SDHI = 3.1 to 3.5 V, VSS = 0 V Without pull-up or pull-down resistor | — |
| Output leakage current | ILO | — | — | 10 | µA | VDDQ18_33_SDHI = 3.1 to 3.5 V, VSS = 0 V Without pull-up or pull-down resistor | Hi-Z output |

**Table 200.12 Output leakage current by V/xDrive strength (3.3-V I/O [SDHI])**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 4.7 | — | — | mA | VDDQ18_33_SDHI = 3.1V to 3.5V, VSS = 0V without pull-up or pull-down, VOL = 0.4V VOH = VDDQ18_33_SDHI – 0.4V | V/xDrive = 2/8 |
| | 5.9 | — | — | mA | | V/xDrive = 3/8 |
| | 6.3 | — | — | mA | | V/xDrive = 4/8 |
| | 6.8 | — | — | mA | | V/xDrive = 5/8 |
| | 7.0 | — | — | mA | | V/xDrive = 6/8 |
| | 7.3 | — | — | mA | | V/xDrive = 7/8 |
| | 7.6 | — | — | mA | | V/xDrive = 8/8 |

### 200.5.5 DC Characteristics (1.8-V I/O [SDHI (SDR50, SDR104)])

**Table 200.13 DC Characteristics (1.8-V I/O [SDHI (SDR50, SDR104)])**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | 1.27 | — | VDDQ18_33_SDHI + 0.3 | V | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | — |
| Input low voltage | VIL | VSS - 0.3 | — | 0.58 | V | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | — |
| Output high voltage | VOH | 1.4 | — | — | V | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | IOH = -2 mA |
| Output low voltage | VOL | — | — | 0.45 | V | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | IOL = 2 mA |
| Pin capacitance | CL | — | — | 10 | pF | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | |
| Input leakage current | ILI | — | — | 10 | µA | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V Without pull-up or pull-down resistor | — |
| Output leakage current | ILO | — | — | 10 | µA | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V Without pull-up or pull-down resistor | Hi-Z output |

**Table 200.14 Output leakage current by V/xDrive strength (1.8-V I/O [SDHI (SDR50, SDR104)])**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 5.1 | — | — | mA | VDDQ18_33_SDHI = 1.7V to 1.9V, VSS = 0V without pull-up or pull-down, VOL = 0.45V VOH = VDDQ18_33_SDHI – 0.5V | V/xDrive = 2/8 |
| | 6.5 | — | — | mA | | V/xDrive = 3/8 |
| | 7.0 | — | — | mA | | V/xDrive = 4/8 |
| | 7.6 | — | — | mA | | V/xDrive = 5/8 |
| | 7.8 | — | — | mA | | V/xDrive = 6/8 |
| | 8.2 | — | — | mA | | V/xDrive = 7/8 |
| | 8.6 | — | — | mA | | V/xDrive = 8/8 |

### 200.5.6 DC Characteristics (1.8-V I/O [MMC])

**Table 200.15 DC Characteristics (1.8-V I/O [MMC])**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | 0.65*VDDQ18_33_SDHI | — | VDDQ18_33_SDHI + 0.3 | V | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | — |
| Input low voltage | VIL | VSS - 0.3 | — | 0.35*VDDQ18_33_SDHI | V | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | — |
| Output high voltage | VOH | VDDQ18_33_SDHI - 0.45 | — | — | V | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | IOH = -2 mA |
| Output low voltage | VOL | — | — | 0.45 | V | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | IOL = 2 mA |
| Pin capacitance | CL | — | — | 10 | pF | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V | — |
| Input leakage current | ILI | — | — | 10 | µA | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V Without pull-up or pull-down resistor | — |
| Output leakage current | ILO | — | — | 10 | µA | VDDQ18_33_SDHI = 1.7 V to 1.9 V, VSS = 0 V Without pull-up or pull-down resistor | Hi-Z output |

**Table 200.16 Output leakage current by V/xDrive strength (1.8-V I/O [MMC])**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 5.1 | — | — | mA | VDDQ18_33_SDHI = 1.7V to 1.9V, VSS = 0V without pull-up or pull-down, VOL = 0.45V VOH = VDDQ18_33_SDHI – 0.45V | V/xDrive = 2/8 |
| | 6.5 | — | — | mA | | V/xDrive = 3/8 |
| | 7.0 | — | — | mA | | V/xDrive = 4/8 |
| | 7.6 | — | — | mA | | V/xDrive = 5/8 |
| | 7.8 | — | — | mA | | V/xDrive = 6/8 |
| | 8.2 | — | — | mA | | V/xDrive = 7/8 |
| | 8.6 | — | — | mA | | V/xDrive = 8/8 |

### 200.5.7 DC Characteristics (1.8-V I/O [SPI])

**Table 200.17 DC Characteristics (1.8-V I/O [SPI])**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | 0.65*VDDQ18_33_SPI | — | VDDQ18_33_SPI + 0.3 | V | VDDQ18_33_SPI = 1.7 V to 1.9 V, VSS = 0 V | — |
| Input low voltage | VIL | VSS - 0.3 | — | 0.35*VDDQ18_33_SPI | V | VDDQ18_33_SPI = 1.7 V to 1.9 V, VSS = 0 V | — |
| Output high voltage | VOH | VDDQ18_33_SPI - 0.45 | — | — | V | VDDQ18_33_SPI = 1.7 V to 1.9 V, VSS = 0 V | IOH = -2 mA |
| Output low voltage | VOL | — | — | 0.45 | V | VDDQ18_33_SPI = 1.7 V to 1.9 V, VSS = 0 V | IOL = 2 mA |
| Pin capacitance | CL | — | — | 10 | pF | VDDQ18_33_SPI = 1.7 V to 1.9 V, VSS = 0 V | — |
| Input leakage current | ILI | — | — | 10 | µA | VDDQ18_33_SPI = 1.7 V to 1.9 V, VSS = 0 V Without pull-up or pull-down resistor | — |
| Output leakage current | ILO | — | — | 10 | µA | VDDQ18_33_SPI = 1.7 V to 1.9 V, VSS = 0 V Without pull-up or pull-down resistor | Hi-Z output |

**Table 200.18 Output leakage current by V/xDrive strength (1.8-V I/O [SPI])**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 7.2 | — | — | mA | VDDQ18_33_SPI = 1.7V to 1.9V, VSS = 0V without pull-up or pull-down, VOL = 0.45V VOH = VDDQ18_33_SPI – 0.45V | V/xDrive = 2/8 |
| | 12.2 | — | — | mA | | V/xDrive = 3/8 |
| | 13.1 | — | — | mA | | V/xDrive = 4/8 |
| | 14.7 | — | — | mA | | V/xDrive = 5/8 |
| | 15.3 | — | — | mA | | V/xDrive = 6/8 |
| | 17.0 | — | — | mA | | V/xDrive = 7/8 |
| | 18.1 | — | — | mA | | V/xDrive = 8/8 |

### 200.5.8 DC Characteristics (1.8-V I/O)

**Table 200.19 DC Characteristics (1.8-V I/O)**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Schmitt input high voltage | VT+ | — | — | 0.7*VDDQ18 | V | VDDQ18 = 1.7 V to 1.9 V, VSS = 0 V | PRESET#, BSMODE, NMI pins |
| Schmitt input low voltage | VT- | 0.3*VDDQ18 | — | — | V | | |
| Hysteresis voltage | ΔVT | 0.1*VDDQ18 | — | — | V | | |
| Input high voltage | VIH | 0.8*VDDQ18 | — | VDDQ18 + 0.3 | V | | EXTAL pins<br>Case of MD9=0 (Inputs an external clock to the EXTAL pin) |
| Input low voltage | VIL | -0.3 | — | 0.2*VDDQ18 | V | | |
| Input high voltage | VIH | 0.7*VDDQ18 | — | VDDQ18 + 0.3 | V | | Other pins |
| Input low voltage | VIL | -0.3 | — | 0.3*VDDQ18 | V | | |
| Output high voltage (1.8-V HyperFlash) | VOH | 0.7*VDDQ18 | — | VDDQ18 + 0.3 | V | | IOH = -3.1mA<br>RPC* pins, QSPI* pins |
| Output low voltage (1.8-V HyperFlash) | VOL | -0.3 | — | 0.45 | V | | IOL = 3.1mA<br>RPC* pins, QSPI* pins |
| Output high voltage (Other pins) | VOH | 0.7*VDDQ18 | — | VDDQ18 + 0.3 | V | | IOH = -4mA |
| Output low voltage (Other pins) | VOL | -0.3 | — | 0.45 | V | | IOL = 4mA |
| Pin capacitance | CL | — | — | 10 | pF | | |
| Input leakage current | ILI | — | — | 10 | µA | VDDQ18 = 1.7 V to 1.9 V, VSS = 0 V Without pull-up or pull-down resistor | All input pins |
| Output leakage current | ILO | — | — | 10 | µA | VDDQ18 = 1.7 V to 1.9 V, VSS = 0 V Without pull-up or pull-down resistor | Hi-Z output |
| Pull-up current | IPU | -20 | — | -65 | µA | VI = 0V With pull-up resistor (50 kohm(Typ.)) | |
| | | 10 | — | 33 | µA | VI = VDDQ18 (1.7 to 1.9 V) With pull-down resistor (100 kohm(Typ.)) | PRESET# pin |
| Pull-down current | IPD | 20 | — | 65 | µA | VI = VDDQ18 (1.7 to 1.9 V) With pull-down resistor (50 kohm(Typ.)) | Other pins |

**Table 200.20 Output leakage current by V/xDrive strength (1.8-V I/O)**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 4.0 | — | — | mA | VDDQ18 = 1.7V to 1.9V, VSS = 0V without pull-up or pull-down, VOL = 0.45V VOH = VDDQ18 – 0.45V | V/xDrive = 1/4 |
| | 6.6 | — | — | mA | | V/xDrive = 2/4 |
| | 9.1 | — | — | mA | | V/xDrive = 3/4 |
| | 11.5 | — | — | mA | | V/xDrive = 4/4 |

**Table 200.21 Output leakage current by V/xDrive strength (1.8-V I/O [HyperFlash])**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 3.1 | — | — | mA | VDDQ18 = 1.7V to 1.9V, VSS = 0V without pull-up or pull-down, VOL = 0.45V VOH = VDDQ18 – 0.45V | V/xDrive = 1/4 |
| | 4.5 | — | — | mA | | V/xDrive = 2/4 |
| | 5.5 | — | — | mA | | V/xDrive = 3/4 |
| | 6.2 | — | — | mA | | V/xDrive = 4/4 |

### 200.5.9 DC Characteristics (I2C open-drain 1.8-V I/O, 3.3-V I/O)

**Table 200.22 DC Characteristics (I2C open-drain 1.8-V I/O, 3.3-V I/O)**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| External pull-up voltage | VPU18 | 1.7 | 1.8 | 1.9 | V | VDDQ18_33_I2C = 1.7 to 1.9 V | |
| Input high voltage | VIH | VPU18 × 0.7 | — | VPU18 + 0.3 | V | VDDQ18_33_I2C = 1.7 to 1.9 V | |
| Input low voltage | VIL | -0.3 | — | VPU18 × 0.3 | V | VDDQ18_33_I2C = 1.7 to 1.9 V | |
| Output low voltage | VOL | — | — | VPU18 × 0.2 | V | VDDQ18_33_I2C = 1.7 to 1.9 V | IOL= 2mA |
| Output low current | IOL | 3 | — | — | mA | VDDQ18_33_I2C = 1.7 to 1.9 V | VOL= 0.4V |
| External pull-up voltage | VPU33 | 3.1 | 3.3 | 3.5 | V | VDDQ18_33_I2C = 3.1 to 3.5 V | |
| Input high voltage | VIH | VPU33 × 0.625 | — | VPU33 + 0.3 | V | VDDQ18_33_I2C = 3.1 to 3.5 V | |
| Input low voltage | VIL | -0.3 | — | VPU33 × 0.25 | V | VDDQ18_33_I2C = 3.1 to 3.5 V | |
| Output low voltage | VOL | — | — | 0.4 | V | VDDQ18_33_I2C = 3.1 to 3.5 V | IOL= 3mA |
| Output low current | IOL | 3 | — | — | mA | VDDQ18_33_I2C = 3.1 to 3.5 V | VOL= 0.4V |

### 200.5.10 DC Characteristics (PCIe)

**Table 200.23 DC Characteristics (PCIe)**

| Test Item | Symbol | Min. | Typ. | Max. | Unit | Remarks |
|---|---|---|---|---|---|---|
| DC Differential TX Impedance | ZTX-DIFF-DC | — | — | 120 | Ω | — |
| DC Input Impedance (2.5 & 5 GT/s) | ZRX-DC | 40 | — | 60 | Ω | — |
| DC Input Impedance (8 & 16 GT/s) | ZRX-DC | — | — | — | Ω | Not specified |

### 200.5.11 DC Characteristics (1.8-V I/O [ETH]<AVB>)

**Table 200.24 DC Characteristics (1.8-V I/O [ETH]<AVB>)**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | VDDQ18_25_AVB - 0.45 | — | VDDQ18_25_AVB + 0.3 | V | VDDQ18_25_AVB = 1.7 V to 1.9 V, VSS = 0 V | — |
| Input low voltage | VIL | VSS - 0.3 | — | 0.58 | V | VDDQ18_25_AVB = 1.7 V to 1.9 V, VSS = 0 V | — |
| Output high voltage | VOH | 1.4 | — | — | V | VDDQ18_25_AVB = 1.7 V to 1.9 V, VSS = 0 V | IOH = -2 mA |
| Output low voltage | VOL | — | — | 0.45 | V | VDDQ18_25_AVB = 1.7 V to 1.9 V, VSS = 0 V | IOL = 2 mA |
| Pin capacitance | CL | — | — | 10 | pF | VDDQ18_25_AVB = 1.7 V to 1.9 V, VSS = 0 V | — |
| Input leakage current | ILI | — | — | 10 | µA | VDDQ18_25_AVB = 1.7 V to 1.9 V, VSS = 0 V | — |
| Output leakage current | ILO | — | — | 10 | µA | VDDQ18_25_AVB = 1.7 V to 1.9 V, VSS = 0 V | Hi-Z output |
| Pull-up current | IPU | -20 | — | -60 | µA | VDDQ18_25_AVB = 1.7 V to 1.9 V, VSS = 0 V | Vin = VSS |
| Pull-down current | IPD | 20 | — | 60 | µA | — | Vin = VDDQ18_25_AVB - 0.45 to VDDQ18_25_AVB + 0.3 |

**Table 200.25 Output leakage current by V/xDrive strength (1.8-V I/O [ETH]<AVB>)**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 5.4 | — | — | mA | VDDQ18_25_AVB = 1.7V to 1.9V, VSS = 0V without pull-up or pull-down, VOL = 0.45V, VOH = VDDQ18_25_AVB – 0.45V | V/xDrive = 2/8 |
| | 5.8 | — | — | mA | | V/xDrive = 3/8 |
| | 6.7 | — | — | mA | | V/xDrive = 4/8 |
| | 7.0 | — | — | mA | | V/xDrive = 5/8 |
| | 7.4 | — | — | mA | | V/xDrive = 6/8 |
| | 7.6 | — | — | mA | | V/xDrive = 7/8 |
| | 7.9 | — | — | mA | | V/xDrive = 8/8 |

### 200.5.12 DC Characteristics (2.5-V I/O [ETH]<AVB>)

**Table 200.26 DC Characteristics (2.5-V I/O [ETH]<AVB>)**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | 1.7 | — | — | V | VDDQ18_25_AVB = 2.4 V to 2.6V | |
| Input low voltage | VIL | — | — | 0.7 | V | | |
| Output high voltage | VOH | 2.0 | — | VDDQ25 + 0.3 | V | VDDQ18_25_AVB = 2.4 V to 2.6V | IOH= -1 mA |
| Output low voltage | VOL | -0.3 | — | 0.4 | V | | IOL = 1 mA |
| Pin capacitance | CL | — | — | 10 | pF | | |
| Input leakage current high | ILIH | — | — | 15 | µA | VDDQ18_25_AVB = 2.6 V, Vin = 2.5 V | |
| Input leakage current low | ILIL | -15 | — | — | µA | VDDQ18_25_AVB = 2.6 V, Vin = 0.4 V | |
| Output leakage current high | ILOH | — | — | 10 | µA | — | Hi-Z output*1 |
| Output leakage current low | ILOL | -10 | — | — | µA | — | Hi-Z output*1 |
| Pull-up current | IPU | 30 | — | 85 | µA | VDDQ18_25_AVB = 2.4 to 2.6 V | Vin = VSS |
| Pull-down current | IPD | 30 | — | 85 | µA | — | Vin = 2.4 to 2.6 V |

Note: (T.B.C.)

**Table 200.27 Output leakage current by V/xDrive strength (2.5-V I/O [ETH]<AVB>)**

| Symbol | Min | Typ | Max | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|
| IOL/IOH | 4.9 | — | — | mA | VDDQ18_25_AVB = 2.4V to 2.6V, VSS = 0V without pull-up or pull-down, VOL = 0.4V, VOH = VDDQ18_25_AVB – 0.4V | V/xDrive = 2/8 |
| | 5.3 | — | — | mA | | V/xDrive = 3/8 |
| | 6.0 | — | — | mA | | V/xDrive = 4/8 |
| | 6.3 | — | — | mA | | V/xDrive = 5/8 |
| | 6.6 | — | — | mA | | V/xDrive = 6/8 |
| | 6.8 | — | — | mA | | V/xDrive = 7/8 |
| | 7.0 | — | — | mA | | V/xDrive = 8/8 |

### 200.5.13 DC Characteristics (I2C 3.3-V I/O)

**Table 200.28 DC Characteristics (I2C 3.3-V I/O)**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage | VIH | 2.0 | — | VDDQ_VIN01 + 0.3 | V | VDDQ_VIN01, VDDQ = 3.0 to 3.6 V | |
| Input low voltage | VIL | -0.3 | — | 0.8 | V | | |
| Output low voltage | VOL | -0.3 | — | 0.4 | V | | IOL = 4 mA |
| Pin capacitance | CL | — | — | 10 | pF | — | All pins |
| Input leakage current | ILI | — | — | 1 | µA | VDDQ_VIN01, VDDQ = 3.0 to 3.6 V | All input pins |
| Output leakage current | ILO | — | — | 1 | µA | Without pull-up or pull-down resistor | Hi-Z output |

### 200.5.14 DC Characteristics (1.2-V I/O [CSI2])

**Table 200.29 DC Characteristics (1.2-V I/O [CSI2])**

The description in this section is compliant with the following MIPI standard: 'MIPI D-PHY specification Version 1.2'.

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|---|
| Input signal voltage range | $V_{PIN}$ | -50 | — | 1350 | mV | | |
| Input leakage current | $I_{LEAK}$ | -10 | — | 10 | uA | $V_{GNDSH(min)} \leq V_{PIN} \leq V_{GNDSH(max)} + V_{PIN(absmax)}$ Lane module in LP receive mode | |
| Ground shift | $V_{GNDSH}$ | -50 | — | 50 | mV | | |
| Transient pin voltage level | $V_{PIN(absmax)}$ | -0.15 | — | 1.45 | V | | |
| Input common mode voltage | $V_{CMRX(DC)}$ | 70 | — | 330 | mV | | HS Receiver |
| Differential input high threshold | $V_{IDTH}$ | — | — | 40 | mV | | HS Receiver |
| Differential input low threshold | $V_{IDTL}$ | -40 | — | — | mV | | HS Receiver |
| Single-ended input high voltage | $V_{IHHS}$ | — | — | 460 | mV | | HS Receiver |
| Single-ended input low voltage | $V_{ILHS}$ | -40 | — | — | mV | | HS Receiver |
| Differential input impedance | $Z_{ID}$ | 80 | — | 125 | Ω | | HS Receiver |
| Input low voltage | $V_{IL}$ | — | — | 550 | mV | | LP Receiver |
| Input high voltage | $V_{IH}$ | 740 | — | — | mV | | LP Receiver |
| Input hysteresis | $V_{HYST}$ | 25 | — | — | mV | | LP Receiver |

### 200.5.15 DC Characteristics (1.2-V I/O [DSI])

**Table 200.30 DC Characteristics (1.2-V I/O [DSI])**

The description in this section is compliant with the following MIPI standard: 'MIPI D-PHY specification Version 1.2'.

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|---|
| Input signal voltage range | $V_{PIN}$ | -50 | — | 1350 | mV | | |
| Input leakage current | $I_{LEAK}$ | -10 | — | 10 | uA | $V_{GNDSH(min)} \leq V_{PIN} \leq V_{GNDSH(max)} + V_{PIN(absmax)}$ Lane module in LP receive mode | |
| Ground shift | $V_{GNDSH}$ | -50 | — | 50 | mV | | |
| transient pin voltage level | $V_{PIN(absmax)}$ | -0.15 | — | 1.45 | V | | |
| Maximum transient time above $V_{PIN(max)}$ or below $V_{PIN(min)}$ | $T_{VPIN(absmax)}$ | — | — | 20 | ns | | |
| HS transmit common-mode voltage | $V_{CMTX}$ | 150 | | 250 | mV | 80 Ω ≤ RL ≤ 125 Ω | HS Transmitter |
| HS Transmit differential voltage | $\|V_{OD}\|$ | 140 | | 270 | mV | 80 Ω ≤ RL ≤ 125 Ω | HS Transmitter |
| HS output high voltage | $V_{OHHS}$ | | | 360 | mV | 80 Ω ≤ RL ≤ 125 Ω | HS Transmitter |
| Single ended output impedance | $Z_{OS}$ | 40 | | 62.5 | Ω | | HS Transmitter |
| Output high voltage | $V_{OH}$ | 0.95 | — | 1.3 | V | | LP Transmitter |
| Output low voltage | $V_{OL}$ | -50 | — | 50 | mV | | LP Transmitter |
| Output impedance | $Z_{OLP}$ | 110 | — | — | Ω | | LP Transmitter |
| Input low voltage | $V_{IL}$ | — | — | 550 | mV | | LP Receiver |
| Input high voltage | $V_{IH}$ | 740 | — | — | mV | | LP Receiver |
| Input hysteresis | $V_{HYST}$ | 25 | — | — | mV | | LP Receiver |

### 200.5.16 Reserved

### 200.5.17 DC Characteristics (0.6-V I/O [LPDDR4X])

**Table 200.31 DC Characteristics (0.6-V I/O [LPDDR4X])**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage (MDQ pin) | VIH | VREF + 0.06 | — | — | V | VDDQVA_DDRn = 0.57 to 0.65 V (*1)<br>VDDQX_DDRn = 1.06 to 1.17 V (*1) | MDQ, MDMI and MWCK pins |
| Input low voltage (MDQ pin) | VIL | — | — | VREF – 0.06 | V | VREF= VDDQVA_DDRn*(1/4) (*1) | |
| Input high voltage | VIH | 0.8 × VDDQVA_DDRn (*1) | — | — | V | VDDQVA_DDRn = 0.57 to 0.65 V (*1)<br>VDDQX_DDRn = 1.06 to 1.17 V (*1) | MBKUP pin |
| Input low voltage | VIL | — | — | 0.2 × VDDQVA_DDRn (*1) | V | | |
| DC differential input high voltage | VIHD | 0.09 | — | — | V | MWCK = H,<br>VDDQVA_DDRn = 0.57 to 0.65 V (*1)<br>VDDQX_DDRn = 1.06 to 1.17 V (*1) | MWCK pins |
| DC differential input low voltage | VILD | — | — | -0.09 | V | MWCK = L,<br>VDDQVA_DDRn = 0.57 to 0.65 V (*1)<br>VDDQX_DDRn = 1.06 to 1.17 V (*1) | |
| High Hi-Z leak current | IOZH | — | — | 200 | µA | VDDQVA_DDRn = 0.57 to 0.65 V (*1)<br>VDDQX_DDRn = 1.06 to 1.17 V (*1) | All pins(*2) |
| Low Hi-Z leak current | IOZL | -200 | — | — | µA | | |
| Pin capacitance | CL | — | — | 15 | pF | — | All pins(*2) |

Note: *1 : n = 0,1
*2 : Except power supply pins.

### 200.5.18 DC Characteristics (0.5-V I/O [LPDDR5])

**Table 200.32 DC Characteristics (0.5-V I/O [LPDDR5])**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| Input high voltage (MDQ pin) | VIH | VREF + 0.05 | — | — | V | VDDQVA_DDRn = 0.47 to 0.57 V (*1)<br>VDDQX_DDRn = 1.01 to 1.12 V (*1) | MDQ, MDMI and MRDQS pins |
| Input low voltage (MDQ pin) | VIL | — | — | VREF – 0.05 | V | VREF= VDDQVA_DDRn*(1/4) (*1) | |
| Input high voltage | VIH | 0.8 × VDDQVA_DDRn (*1) | — | — | V | VDDQVA_DDRn = 0.47 to 0.57 V (*1)<br>VDDQX_DDRn = 1.01 to 1.12 V (*1) | MBKUP pin |
| Input low voltage | VIL | — | — | 0.2 × VDDQVA_DDRn (*1) | V | | |
| DC differential input high voltage | VIHD | 0.09 | — | — | V | MRDQS = H,<br>VDDQVA_DDRn = 0.47 to 0.57 V (*1)<br>VDDQX_DDRn = 1.01 to 1.12 V (*1) | MRDQS pins |
| DC differential input low voltage | VILD | — | — | -0.09 | V | MRDQS = L,<br>VDDQVA_DDRn = 0.47 to 0.57 V (*1)<br>VDDQX_DDRn = 1.01 to 1.12 V (*1) | |
| High Hi-Z leak current | IOZH | — | — | 200 | µA | VDDQVA_DDRn = 0.47 to 0.57 V (*1)<br>VDDQX_DDRn = 1.01 to 1.12 V (*1) | All pins(*2) |
| Low Hi-Z leak current | IOZL | -200 | — | — | µA | | |
| Pin capacitance | CL | — | — | 15 | pF | — | All pins(*2) |

Note: *1 : n = 0,1
*2 : Except power supply pins.

### 200.5.19 LPDDR4X Interface ODT Characteristics

**Table 200.33 LPDDR4X Interface ODT Characteristics**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| ODT resistance (40 Ω) | RTT40 | 33.6 | — | 47.7 | Ω | VDDQVA_DDRn = 0.57 to 0.65 V (n=0,1) | — |

### 200.5.20 LPDDR5 Interface ODT Characteristics

**Table 200.34 LPDDR5 Interface ODT Characteristics**

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement conditions | Remarks |
|---|---|---|---|---|---|---|---|
| ODT resistance (40 Ω) | RTT40 | 33.6 | — | 47.7 | Ω | VDDQVA_DDRn = 0.47 to 0.57 V (n=0,1) | — |

### 200.5.21 Overload Condition (Injection Current)

The overload condition describes the behavior in case of current injection to the port pins.

**Table 200.35 Overload Current**

**Conditions: See 200.7.1 AC Characteristic Measurement Condition**

| Parameter | Item | Symbol | Condition | Ratings*1 | Unit | Measurement conditions |
|---|---|---|---|---|---|---|
| Overload Current Vin > VDDQ Vin < VSS | Pins supplied by VDDQ33 | IINJPM | 1 pin | ±2 | mA | |
| | | IINJNM | Sum of all absolute IINJPM + IINJNM of pins supplied as group *2 | \|50\| | mA | 24 signals |
| | Pins supplied by VDDQ18 | IINJPM | 1 pin | ±2 | mA | |
| | | IINJNM | Sum of all absolute IINJPM + IINJNM of pins supplied as group *2 | \|50\| | mA | 36 signals |
| | Pins supplied by VDDQ18_CSI0 VDDQ18_CSI1 | IINJPM | 1 pin | ±2 | mA | |
| | | IINJNM | Sum of all absolute IINJPM + IINJNM of pins supplied as group *2 | \|22\| | mA | 11 signals |
| | Pins supplied by VDDQ18_DSI0 | IINJPM | 1 pin | ±2 | mA | |
| | | IINJNM | Sum of all absolute IINJPM + IINJNM of pins supplied as group *2 | \|22\| | mA | 11 signals |
| | Pins supplied by VDDQ18_25_AVB | IINJPM | 1 pin | ±2 | mA | |
| | | IINJNM | Sum of all absolute IINJPM + IINJNM of pins supplied as group *2 | \|50\| | mA | 63 signals |
| | Pins supplied by VDDQ18_33_I2C | IINJPM | 1 pin | ±2 | mA | |
| | | IINJNM | Sum of all absolute IINJPM + IINJNM of pins supplied as group *2 | \|28\| | mA | 14 signals |
| | Pins supplied by VDDQ18_33_SDHI | IINJPM | 1 pin | ±2 | mA | |
| | | IINJNM | Sum of all absolute IINJPM + IINJNM of pins supplied as group *2 | \|26\| | mA | 13 signals |
| | Pins supplied by VDDQ18_33_SPI | IINJPM | 1 pin | ±2 | mA | |
| | | IINJNM | Sum of all absolute IINJPM + IINJNM of pins supplied as group *2 | \|50\| | mA | 48 signals |

Notes: 1. The Total current may be limited further by the total power dissipation.
2. The total overload current must be within the output current.
3. Observe the absolute maximum rated voltage
4. Do not inject during key-off

## 200.6 General Characteristics

### 200.6.1 Clock and Reset Timings

**Table 200.36 Clock and Reset Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Pin | Item | Symbol | Min. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| PRESET#, EXTAL | Power-on oscillation settling time (Period from power and EXTAL stable clock input to PRESET# rise.) | tOSC | 5 | — | ms | |
| | | T | 0*2 | *3 | ms | |
| TRST#, EXTAL | Power-on oscillation settling time (Period from power and EXTAL stable clock input to TRST#rise.) | tJTOSC | 5 | — | ms | |
| PRESET# | PRESET# input low level width | tWRSL | 5 | — | ms | |
| Mode signal*1 | MD reset setup time | tMDRS | 5 | — | ms | |
| Mode signal*1 | MD reset hold time | tMDRH | 3 | — | ns | |
| Mode signal*1 | MD TRST# setup time | tMDJTRS | 5 | — | ms | |

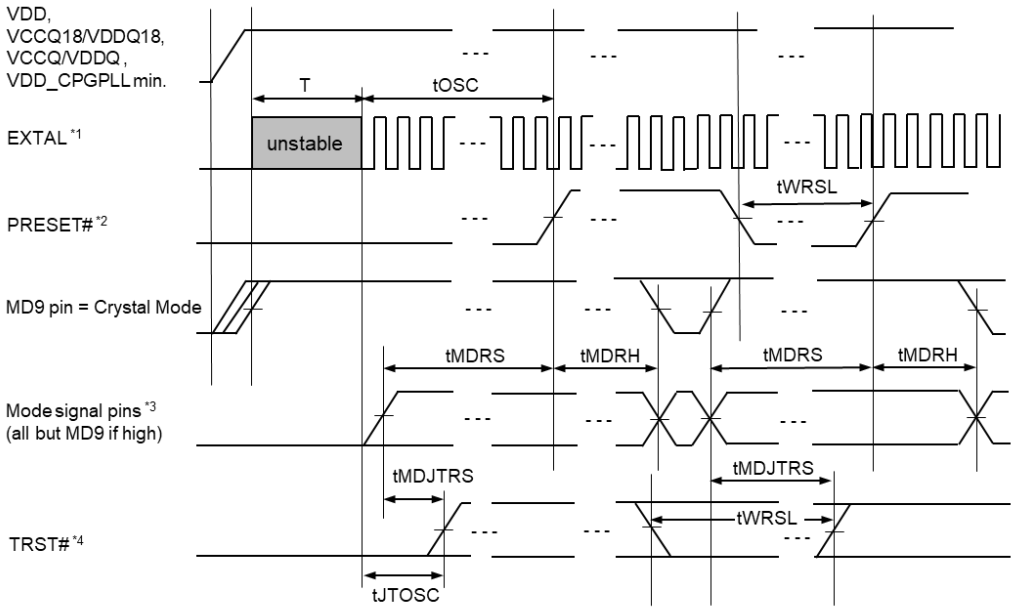
Notes: 1. MDn (n = 0, 1, 2 ...) and MDT [0]. For details of mode signals, refer to section 3.2, Mode Pin Settings.

2: Using 0 stabilization time requires an external clock. For mode setting refer to Table 3.8 MD9 Settings.

3: The oscillation stabilization time differs according to the matching with the external resonator circuit.

It is recommended to determine the oscillation stabilization time by a crystal resonator matching test.

For details of the stabilization time “T”, please contact the crystal manufacture.




Notes 1:
For EXTAL's waveform.
Since the unstable oscillation exists when using a crystal resonator, wait for T until clock is stabilized.
T depends on the circuit on the board.
2: For PRESET# input.
Note electromagnetic noise because that may cause malfunctions of the LSI.
Note an intermediate voltage level input because that may influence the long-term reliability of the LSI
3:
When TRST# is deasserted before PRESET# is deasserted, make sure that the mode signal value does not change during from TRST# deasserting to PRESET# deasserting.
4:
This restriction applies to the "TRST#" signal on both "MAIN JTAG" and "SUB JTAG".

**Figure 200.1 Reset when Turning on Power Supply (PRESET# vs EXTAL and Mode signal)**

### 200.6.2 EXTAL Clock Input Timing

**Table 200.37 EXTAL Clock Input Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|---|
| EXTAL clock input frequency | MD[14:13] = LL | fEX | 15.00 | 16.66 | 16.70 | MHz | — |
| | MD[14:13] = LH | | 18.00 | 20.00 | 20.04 | MHz | |
| | MD[14:13] = HH | | 30.00 | 33.33 | 33.40 | MHz | |
| EXTAL clock input cycle time | MD[14:13] = LL | tEXcyc | 59.88 | 60.00 | 66+2/3 | ns | |
| | MD[14:13] = LH | | 49.90 | 50.00 | 55+5/9 | ns | |
| | MD[14:13] = HH | | 29.94 | 30.00 | 33+1/3 | ns | |
| EXTAL clock input Frequency Tolerance (MD9 = H) | | — | -200 | — | +200 | ppm | |
| EXTAL clock input duty cycle time | | tEXduty | 0.4 | 0.5 | 0.6 | tEXcyc | |
| EXTAL clock input low-level pulse width | | tEXL | 5 | — | — | ns | |
| EXTAL clock input high-level pulse width | | tEXH | 5 | — | — | ns | |
| EXTAL clock input rise time | | tEXr | — | — | 4 | ns | |
| EXTAL clock input fall time | | tEXf | — | — | 4 | ns | |

Notes: 1. Set the MD9 pin to L (low) during power-on reset when using the EXTAL pin as external clock input and the XTAL pin must be open.
2. See Section 8.1.5 Clock Operating Modes to select an oscillator or resonator.

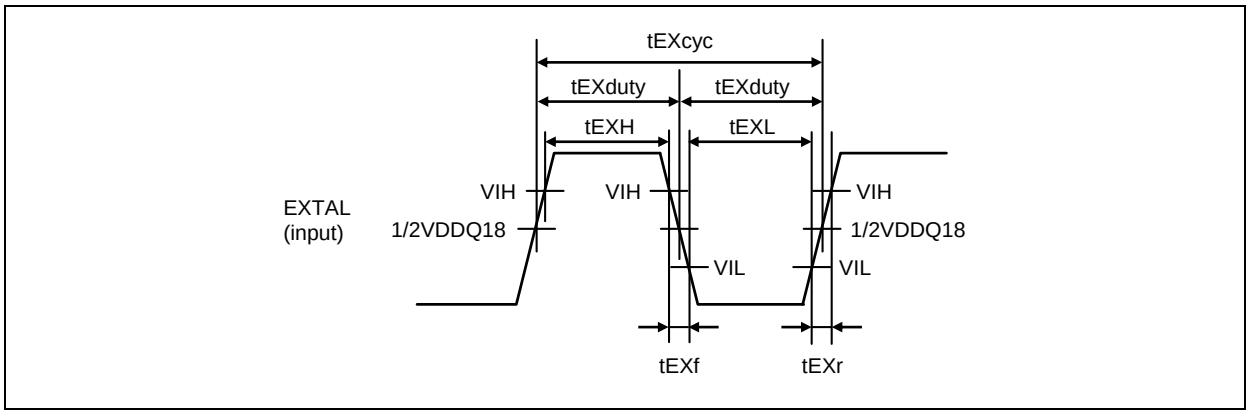



**Figure 200.2 EXTAL Clock Input Timing**

### 200.6.3 CLKOUT Clock Output Timing

**Table 200.38 CLKOUT Clock Output Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| CLKOUT clock output frequency | fCKO | 59.99 | 66.66 | 66.80 | MHz | — |
| CLKOUT clock output cycle time | tCKOcyc | 16.67 | 15.00 | 17.38 | ns | |
| CLKOUT clock output duty cycle time | tCKOduty | 0.4 | 0.5 | 0.6 | tCKOcyc | |
| CLKOUT clock output low pulse width | tCKOL | 4 | — | — | ns | |
| CLKOUT clock output high pulse width | tCKOH | 4 | — | — | ns | |
| CLKOUT clock output rise time | tCKOr | — | — | 4 | ns | |
| CLKOUT clock output fall time | tCKOf | — | — | 4 | ns | |

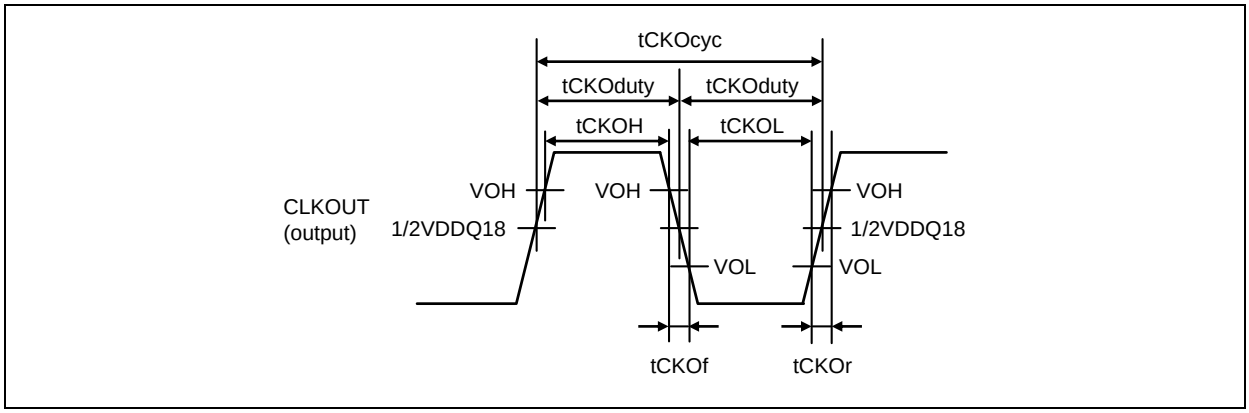



**Figure 200.3 CLKOUT Clock Output Timing**

### 200.6.4 EXTALR Input Timing

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

**Table 200.39 EXTALR Characteristics**

| Item | | Symbol | Min. | Typ. | Max. | Unit | Remarks |
|---|---|---|---|---|---|---|---|
| Input Frequency Range | MD29 = 0 | — | 32.5 | 32.768 | 33 | kHz | |
| Input Frequency Tolerance | | — | -200 | — | +200 | ppm | |
| EXTALR clock input rise time *2 | MD29 = 0 | — | — | — | 1 | µs | |
| EXTALR clock input fall time *2 | MD29 = 0 | — | — | — | 1 | µs | |
| OCO Frequency Deviation | MD29 = 1 | — | 29.5 | 32.786 | 36.06 | kHz | |
| | | | -10 | — | +10 | % | |

Notes: 1. No need to use EXTALR port when MD29 = 1
2. Input rise time and fall time to input clock pulse from crystal oscillator, which is to be applied between VIL and VIH at DC and AC.

### 200.6.5 THS/CIVM Characteristics

**Conditions:**

- **See 200.7.1 AC Characteristic Measurement Condition**.

**Table 200.40 Thermal Sensor Characteristics*1*2*3*4**

| Item | Symbol | Condition | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|---|
| Temperature accuracy | ACCTS1 | – 40 °C ≤ Tj ≤ + 125 °C | -3 | | 3 | °C |
| Temperature update period | tTSUP | | — | — | 1 | ms |
| THS Operation stabilization waiting time | tTSSB | | — | — | 1 | ms |
| Chip internal voltage monitor accuracy | ACCCIV1 | | -60 | | 60 | mV |
| Chip internal voltage update period | tCIVUP | | — | — | 0.5 | ms |
| CIVM Operation stabilization waiting time | tCIVSB | | — | — | 0.5 | ms |

Note 1. The temperature borders need to be set keeping the temperature range (Tj(min) to Tj(max)) including the thermal sensor accuracy.

Note 2. It does not include accuracy of measuring equipment.

Note 3. Thermal sensor cannot detect local heat generation inside the chip.

Note 4. Thermal sensor accuracy is guaranteed within operation temperature.

### 200.6.6 CVM Characteristics

**Table 200.41 CVM Characteristics(1/2)**

| Item | Symbol | Power domain | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Core voltage high detection level | $V_{VDDMDH}$ | VDD | VDD (Max.) – 0.025 | — | VDD (Max.) | V |
| Core voltage low detection level | $V_{VDDMDL}$ | VDD | VDD (Min.) *1 | — | VDD (Min.) *1 + 0.025 | V |

Note 1. Voltage itself depends on supplied one based on AVS configuration.

**Table 200.42 CVM Characteristics**(2/2)

| Item | Symbol | Setting of "CVM digital output filter function control" | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| CVM error detection filtering time | $t_{FCVMOUT}$ | VMFLTFC=B'00 *1 | 5 | — | 15 | µs |
| | | VMFLTFC=B'01 *1 | 2.4 | — | 10 | µs |
| | | VMFLTFC=B'10 *1 | 6.7 | — | 20 | µs |
| CVM_OUT# minimum pulse width | $t_{WCVMOUT}$ | — | 0.5 | — | — | µs |
| VTHSENSE0 output voltage transition time | $t_{SENSE_RISE}$ | Rise: 0V → 0.8V | 90 | — | 195 | µs |
| | $t_{SENSE_FALL}$ | Fall: 0.8V → 0V | 170 | — | 285 | µs |
| VTHREF0 output voltage transition time | $t_{REF_RISE0}$ | Rise: 0V → 0.8V | 90 | — | 195 | µs |
| | $t_{REF_FALL0}$ | Fall: 0.8V → 0V | 170 | — | 285 | µs |
| | $t_{REF_RISE1}$ | Rise: 0.8V → 1.24V | 80 | — | 190 | µs |
| | $t_{REF_FALL1}$ | Fall: 1.24V → 0.8V | 625 | — | 1435 | µs |
| VTHSENSE0 / VTHREF0 output voltage range | $V_{RANGE_L}$ | Output: 0V | — | 0.0 | 0.2 | V |
| | $V_{RANGE_M}$ | Output: 0.8V | 0.7 | 0.8 | 0.9 | V |
| | $V_{RANGE_H}$ | Output: 1.24V | 1.19 | 1.24 | 1.29 | V |
| VTHSENSE0 connect capacitor | C1 | *2 | 0.09 | — | 0.11 | µF |
| VTHREF0 connect capacitor | C2 | *2 | 0.09 | — | 0.11 | µF |

Note 1. See Section 13.2 for register descriptions

Note 2. See Section 13.1.3 for capacitors ("Figure 13.2 Connect capacitors to Analog Pin")

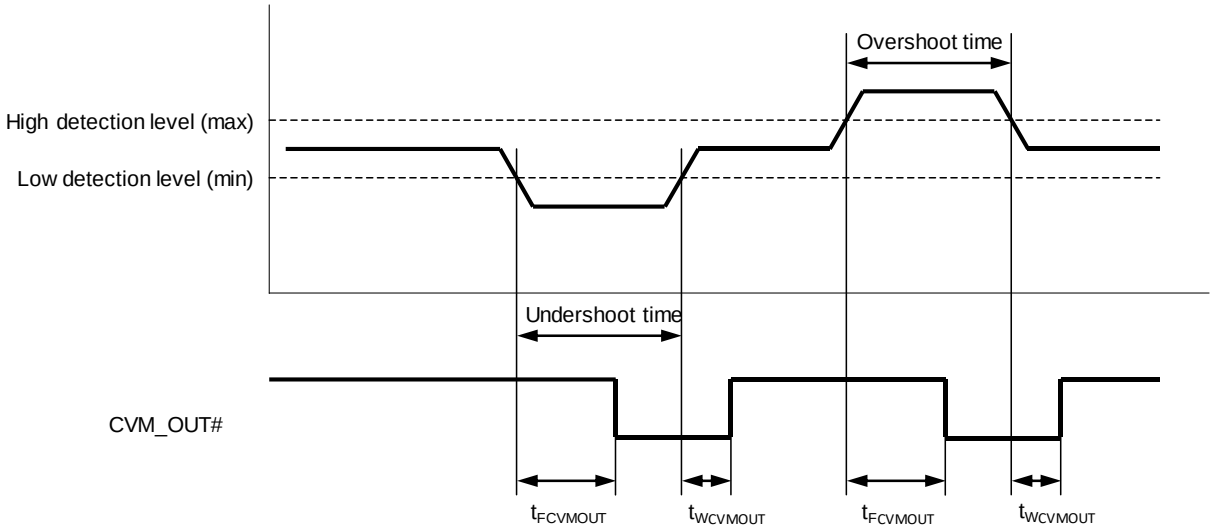



**Figure 200.4 CVM Characteristics**

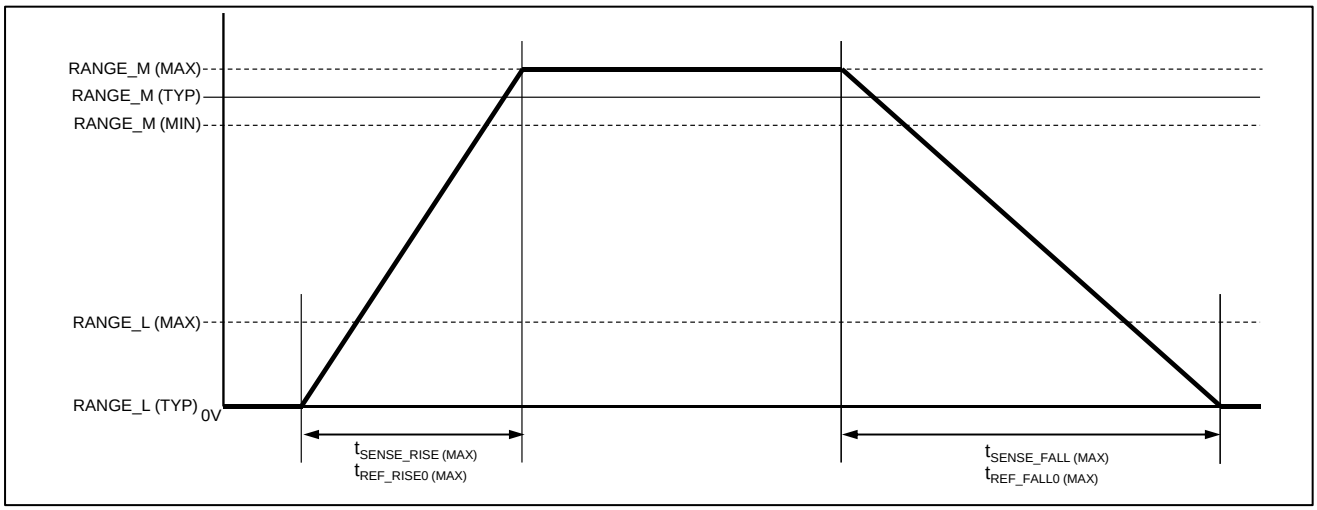



**Figure 200.5 VTHSENSE0/VTHREF0 output voltage transition time (0V → 0.8V, 0.8V → 0V)**

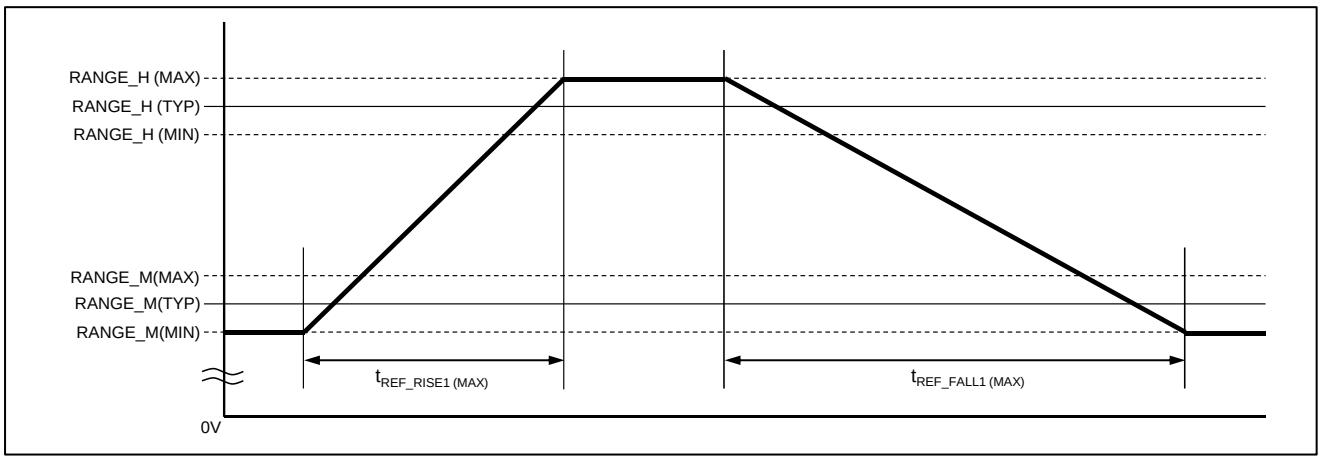



**Figure 200.6 VTHREF0 output voltage transition time (0.8V → 1.24V, 1.24V → 0.8V)**

### 200.6.7 PLL Characteristics

**Conditions:**

- **See 200.7.1 AC Characteristic Measurement Condition**.

**Table 200.43 PLL Characteristics (PLL 1,2,3,4,5,6,7)**

| Item | Symbol | Condition | | MIN | TYP | MAX | Unit |
|---|---|---|---|---|---|---|---|
| Long term jitter | — | SSMODE=000, or 001 *1 | Term=1us | -300 | — | +300 | ps |
| | | | Term=10us | -500 | — | +500 | |
| | | | Term=20us | -500 | — | +500 | |
| | | SSMODE=100 or 101 *1 | Term=1us | -300 | — | +300 | |
| | | | Term=10us | -500 | — | +500 | |
| | | | Term=20us | -500 | — | +500 | |

Note 1. See Section 8.1.5 Clock Operating Modes

## 200.7 AC Characteristics

### 200.7.1 AC Characteristic Measurement Condition

**(1) General Conditions**

Below conditions are valid for all subsequent timing specifications if not noted otherwise:

- See **200.2, Power Supply**.
- Drive capability = **7** (DRV3,DRV2,DRV1=110) for 1.8V/3.3V IO
- Drive capability = **7** (DRV3,DRV2,DRV1=110) for 1.8V/2.5V/ IO
- Drive capability = **7** (DRV3,DRV2,DRV1=110) for 3.3V/ IO
- Drive capability = **3** (DRV2,DRV1=10) for 1.8V IO
- CL = **30** pF
- Buffer type = CMOS

**(2) General Input Measurement Points**

If not stated otherwise, the below given AC timing specification is based on the measurements points as follows:VDD*_ include Refer to Power Supply
the measurement points of the timing chart for AC timing chapter are:
Dot mark indicates 0.5xVDD, bar or dotted line notation indicates 0.7xVDD/0.3xVDD.
As for functions that have individual standards about L/H thresholds,
See "Conditions" described in each function.

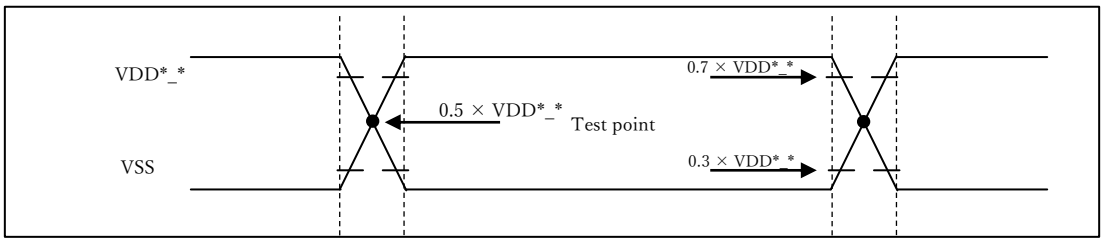



**Figure 200.7 AC Input Measurement Points**

**(3) General Output Measurement Points**

If not stated otherwise, the below given AC timing specification is based on the measurements points as follows: VDD*_ include Refer to Power Supply
the measurement points of the timing chart for AC timing chapter are:
Dot mark indicates 0.5xVDD, bar or dotted line notation indicates 0.7xVDD/0.3xVDD.

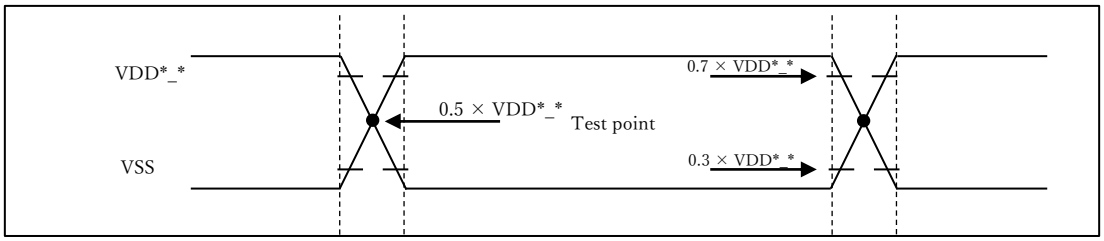



**Figure 200.8 AC Output Measurement Points**

**(4) Load conditions**

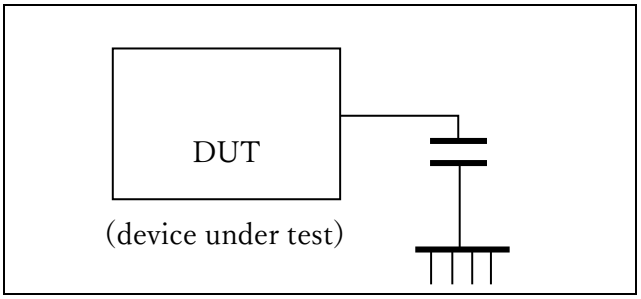



**Figure 200.9 AC Load Conditions**

**(5) General output clock frequency and cycle time (period) minimum specification**

If not stated otherwise, maximum and minimum value described as output clock frequency and cycle time (period) of each module means ideal value derived from setting described in section 8. Clock Pulse Generator(CPG) or module clock division settings.

To obtain detailed information, please refer to section 8. Clock Pulse Generator(CPG) or module clock division settings.

### 200.7.2 AC Characteristics (1.2-V I/O [CSI2])

**Table 200.44 AC Characteristics (1.2-V I/O [CSI2])**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

The description in this section is compliant with the following MIPI standard: 'MIPI D-PHY specification Version 1.2'.

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|---|
| Data to Clock Receiver Setup time | $T_{SETUP(RX)}$ | 0.15 | — | — | UI | >= 0.08Gbps <= 1.0Gbps | HS Receiver |
| | | 0.20 | — | — | UI | > 1.0Gbps <= 1.5Gbps | |
| Data to Clock Receiver Hold time | $T_{HOLD(RX)}$ | 0.15 | — | — | UI | >= 0.08Gbps <= 1.0Gbps | HS Receiver |
| | | 0.20 | — | — | UI | > 1.0Gbps <= 1.5Gbps | |
| Static Data to Clock Skew (RX) | $T_{SKEW[RX]}$ static | -0.20 | | 0.20 | UI | > 1.5Gbps | HS Receiver |
| Dynamic Data to Clock Skew Window RX Tolerance | $T_{SETUP[RX]}$ + $T_{HOLD[RX]}$ dynamic | 0.50 | | | UI | > 1.5Gbps | HS Receiver |
| Minimum pulse width response | $T_{MIN-RX}$ | 20 | — | — | ns | — | LP Receiver |

### 200.7.3 AC Characteristics (1.2-V I/O [DSI])

**Table 200.45 AC Characteristics (1.2-V I/O [DSI])**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

The description in this section is compliant with the following MIPI standard: 'MIPI D-PHY specification Version 1.2'.

| Item | Symbol | Min. | Typ. | Max. | Unit | Measurement condition | Remarks |
|---|---|---|---|---|---|---|---|
| Data to Clock Skew | $T_{SKEW[TX]}$ | -0.15 | — | 0.15 | UI | >= 0.08Gbps<br><= 1.0Gbps | HS Transmitter |
| | | -0.20 | — | 0.20 | UI | > 1.0Gbps<br><= 1.5Gbps | |
| Static Data to Clock Skew (TX) | $T_{SKEW[TX]}$ static | -0.20 | — | 0.20 | UI | > 1.5Gbps | HS Transmitter |
| Dynamic Data to Clock Skew (TX) | $T_{SKEW[TX]}$ dynamic | -0.15 | — | 0.15 | UI | > 1.5Gbps | HS Transmitter |
| Pulse width of the LP exclusive-OR clock | $T_{LP-PULSE-TX}$ | 40 | | | ns | | LP Transmitter |
| Period of the LP exclusive-OR clock | $T_{LP-PER-TX}$ | 90 | | | ns | | LP Transmitter |
| Minimum pulse width response | $T_{MIN-RX}$ | 20 | — | — | ns | — | LP Receiver |

### 200.7.4 SSI Interface

**Table 200.46 SSI Interface Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Note | Figure |
|---|---|---|---|---|---|---|---|
| Output clock cycle | tO | 80 | — | 15625 | ns | — | Figure 200.10 |
| Input clock cycle | tI | 66 | — | 15625 | ns | — | |
| Output clock high-cycle | tHC | 35 | — | — | ns | — | |
| Output clock low-cycle | tLC | 35 | — | — | ns | — | |
| Input clock high-cycle | tHC | 28 | — | — | ns | — | |
| Input clock low-cycle | tLC | 28 | — | — | ns | — | |
| Rise-edge clock timing | tRC | — | — | 20 | ns | Output (100pF) | |
| Output delay | tD | −5 | — | 19 | ns | — | Figure 200.11 to Figure 200.14 |
| | tD | — | — | 19 | ns | — | Figure 200.14 |
| Setup time | tS | 15 | — | — | ns | — | Figure 200.11 to Figure 200.14 |
| Hold time | tH | 5 | — | — | ns | — | |
| Audio clock frequency | fAUDIO | 3.072 | — | 25 | MHz | — | Figure 200.16 |
| Audio clock duty | fAUDIOduty | 0.45 | — | 0.55 | fAUDIO | — | |

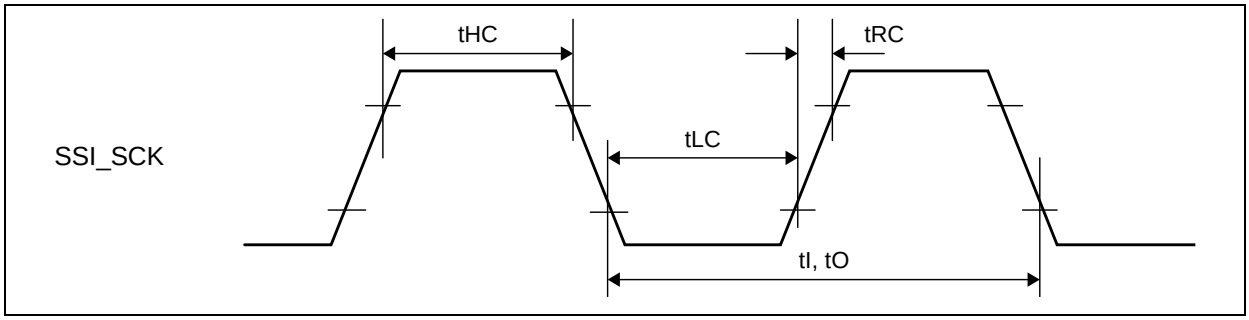



**Figure 200.10 SCK Clock Input/output Timing**

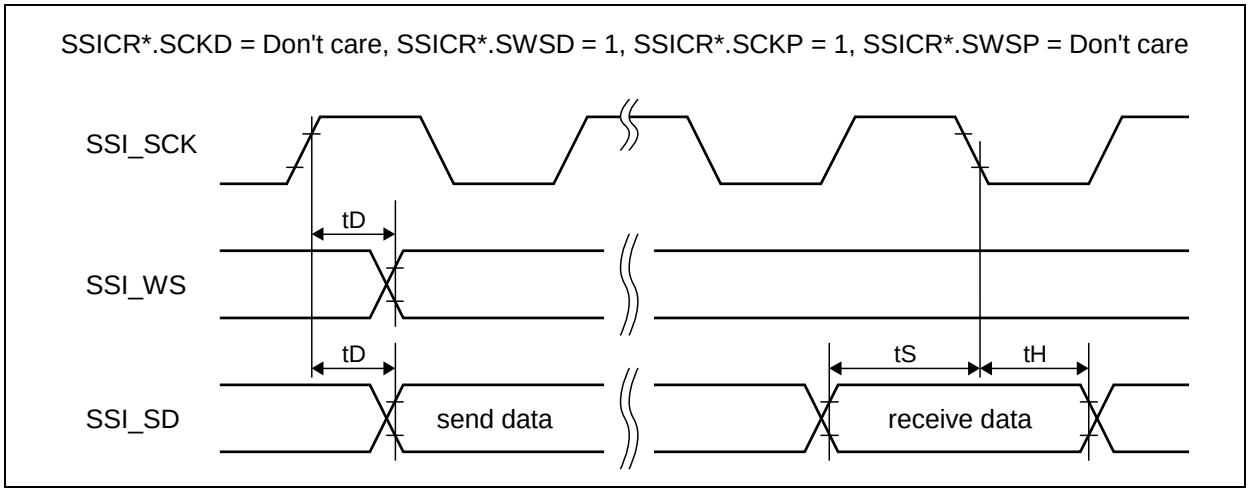



**Figure 200.11 SSI Timing (1)**

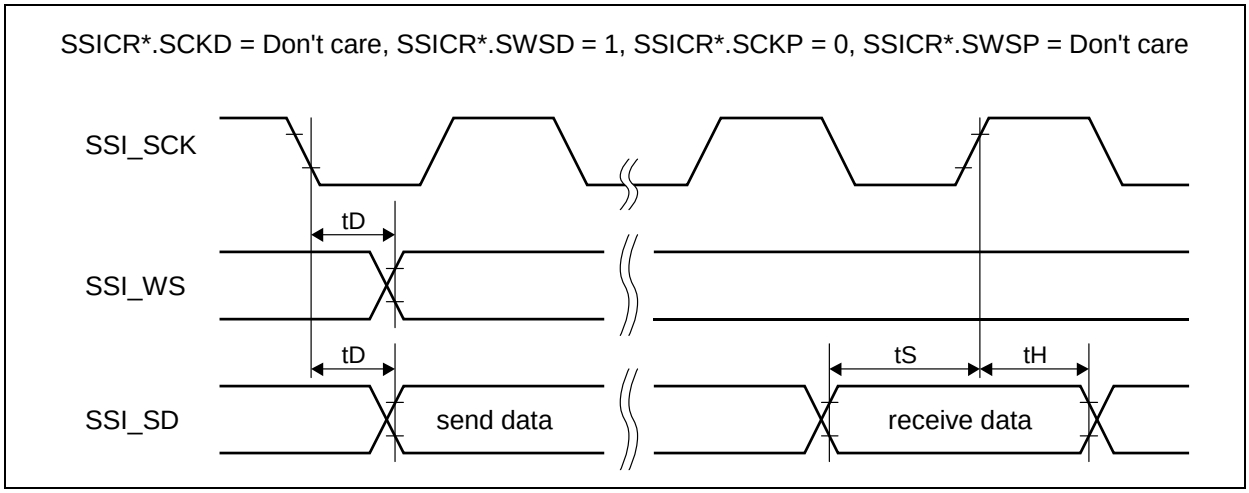



**Figure 200.12 SSI Timing (2)**

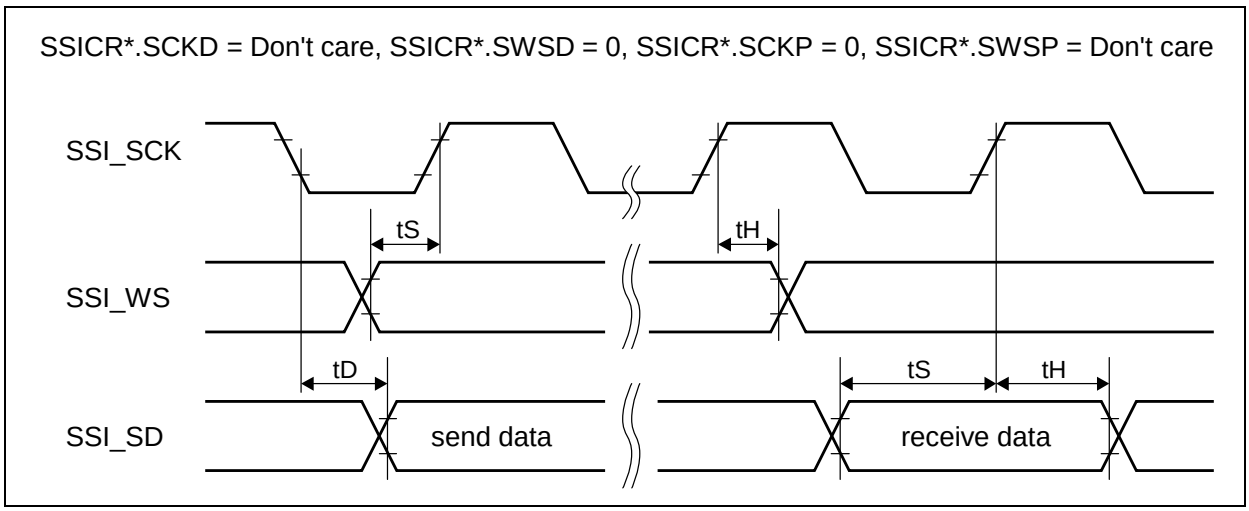



**Figure 200.13 SSI Timing (3)**

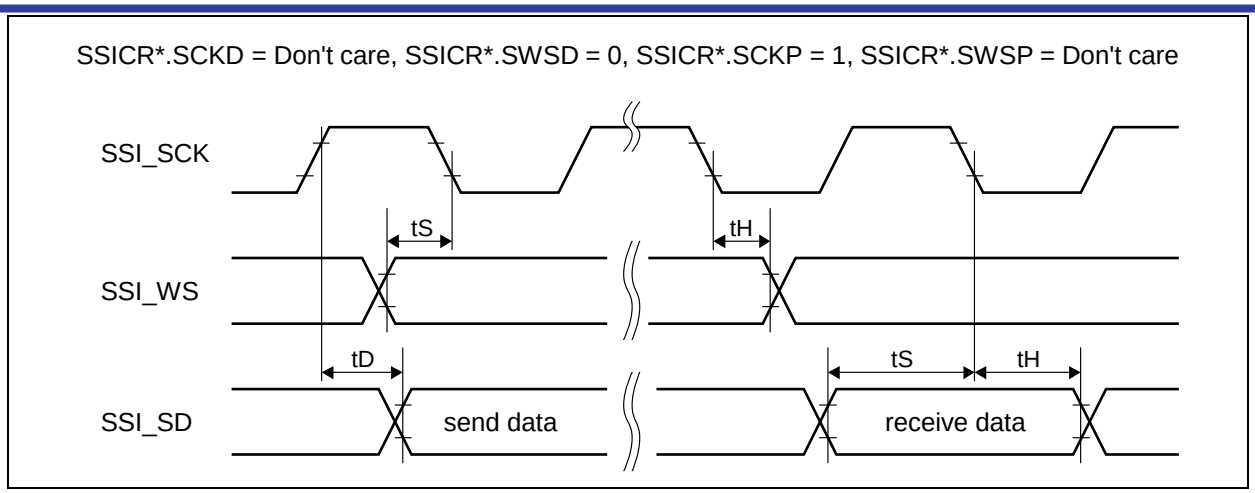



**Figure 200.14 SSI Timing (4)**

The follows are output timings of the MSB bit in the setting of the slave transmission by non-compression.
Send data is no padding bit with a left justify format or a right justify format.
SSICR*.SCKD=0, SSICR*.SWSD=0, SSICR*.DEL=1, SSICR*.TRMD=1
SSICR*.SDTA=0 or SSICR*.SDTA=1 and moreover SSICR*.SWL=SSICR*.DWL

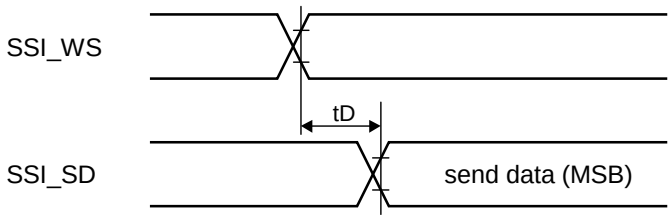



**Figure 200.15 SSI Timing (5)**

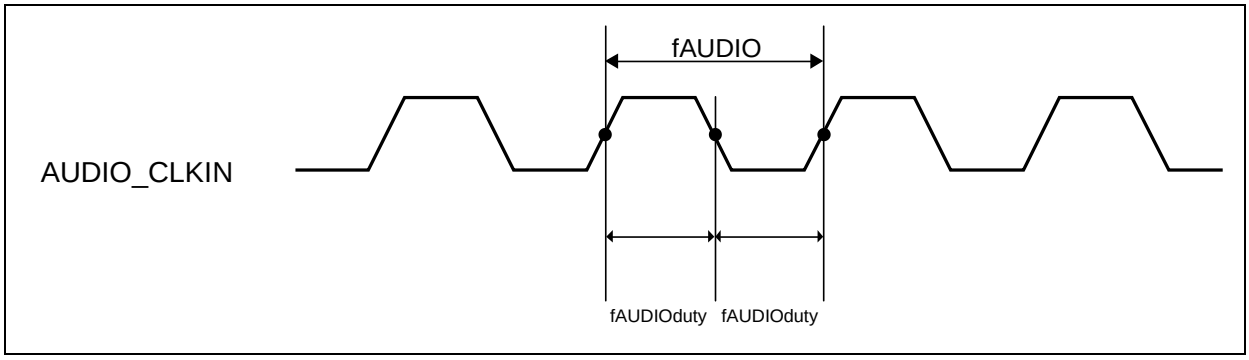



**Figure 200.16 AUDIO_CLK Input Timing**

### 200.7.5 Ethernet Module Signal Timing

#### (1) Ethernet Control Timing (1.8V RGMII 100Mbps Mode)

**Table 200.47 Ethernet Control Timing (1.8V RGMII 100Mbps Mode)**

Conditions: **See (10) RGMII signal AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| ETH_TXCREFCLK cycle time | — | 125 – 50 ppm | — | 125 + 50 ppm | MHz | — |
| ETH_TXC cycle time | Tcyc | 36 | 40 | 44 | ns | |
| ETH_TXC duty cycle | Duty_T | 40 | 50 | 60 | % | |
| ETH_TX_CTL to clock output skew | TskewT | -500 | 0 | 500 | ps | Figure 200.17 |
| ETH_TD[3:0] to clock output skew | TskewT | -500 | 0 | 500 | ps | |
| ETH_RXC cycle time | Tcyc | 36 | 40 | 44 | ns | — |
| ETH_RXC duty cycle | Duty_T | 40 | 50 | 60 | % | |
| ETH_RX_CTL to clock input skew | TskewR | 1 | 1.8 | 17 | ns | Figure 200.18 |
| ETH_RD[3:0] to clock input skew | TskewR | 1 | 1.8 | 17 | ns | |

Notes: 1. ETH is AVB*

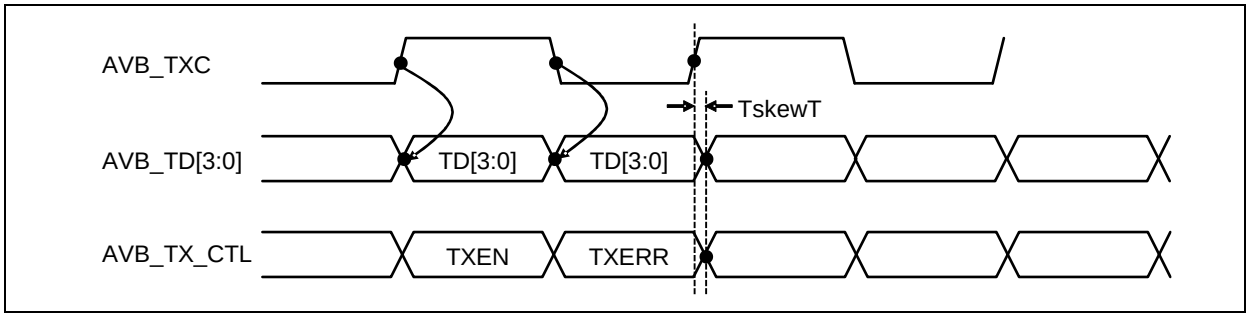



**Figure 200.17 RGMII 100 Mbps Mode Transmission Timing**

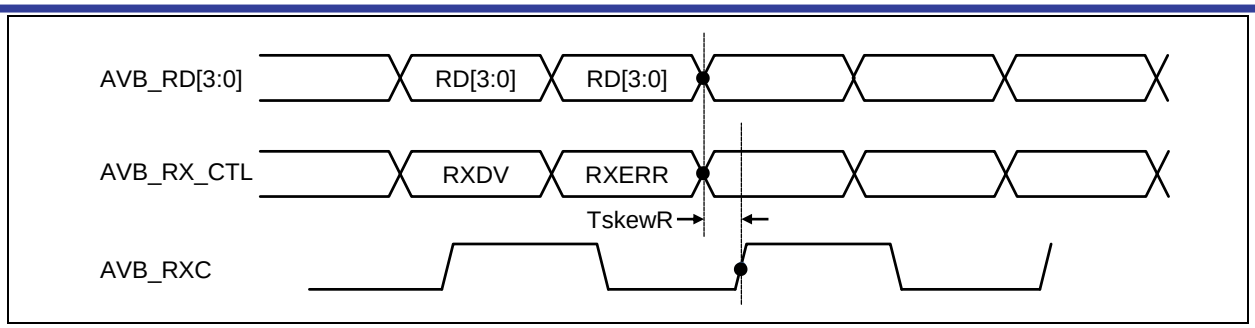



**Figure 200.18 RGMII 100 Mbps Mode Reception Timing**

### (2) Ethernet Control Timing (1.8V RGMII 1Gbps Mode)

**Table 200.48 Ethernet Control Timing (1.8V RGMII 1Gbps Mode)**

Conditions: **See (10) RGMII signal AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| ETH_TXCREFCLK cycle time | — | 125 – 50 ppm | — | 125 + 50 ppm | MHz | — |
| ETH_TXC cycle time | Tcyc | 7.2 | 8 | 8.8 | ns | |
| ETH_TXC duty cycle | Duty_G | 45 | 50 | 55 | % | |
| ETH_TX_CTL to clock output skew | TskewT | -500 | 0 | 500 | ps | Figure 200.19 |
| ETH_TD[3:0] to clock output skew | TskewT | -500 | 0 | 500 | ps | |
| ETH_RXC cycle time | Tcyc | 7.2 | 8 | 8.8 | ns | — |
| ETH_RXC duty cycle | Duty_G | 45 | 50 | 55 | % | |
| ETH_RX_CTL to clock input skew | TskewR | 1 | 1.8 | 2.6 | ns | Figure 200.20 |
| ETH_RD[3:0] to clock input skew | TskewR | 1 | 1.8 | 2.6 | ns | |

Notes: 1. ETH is AVB*

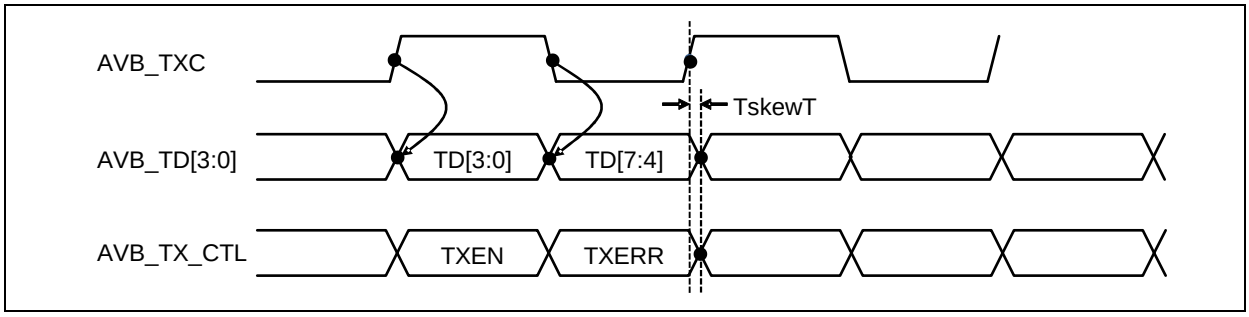



**Figure 200.19 RGMII 1 Gbps Mode Transmission Timing**

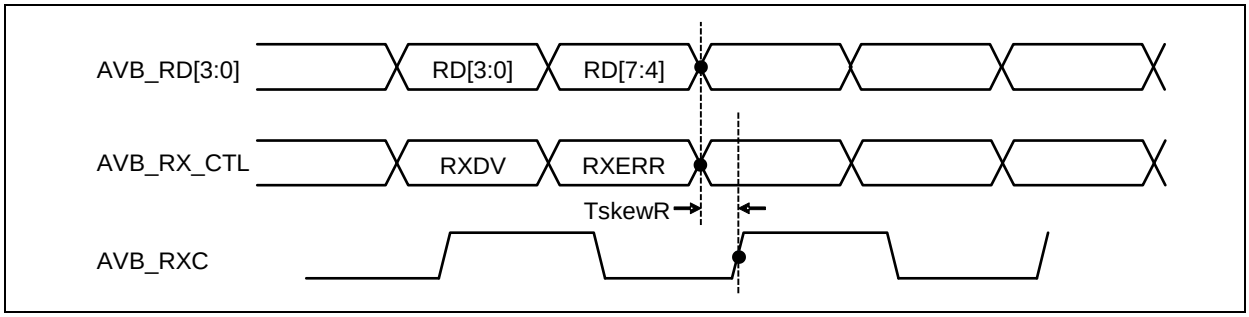



**Figure 200.20 RGMII 1 Gbps Mode Reception Timing**

### (3) Ethernet Control Timing (1.8V RGMII Rx clock internal Delay Mode)

**Table 200.49 Ethernet Control Timing (1.8V RGMII Rx clock internal Delay Mode)**

Conditions: **See (10) RGMII signal AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| ETH_RXC cycle time* | Tcyc | 7.2 | 8 | 8.8 | ns | — |
| ETH_RXC duty cycle (in 1 Gbps) | Duty_G | 45 | 50 | 55 | % | |
| ETH_RXC duty cycle (in 100 Mbps) | Duty_T | 40 | 50 | 60 | % | |
| ETH_RX_CTL to clock input skew | TskewR | -650 | 0 | 650 | ps | Figure 200.21 |
| ETH_RD[3:0] to clock input skew | TskewR | -650 | 0 | 650 | ps | |

Notes: 1. ETH is AVB*

Notes: 2. For 100 Mbps, Tcyc will scale to 40 ns +- 4 ns respectively.

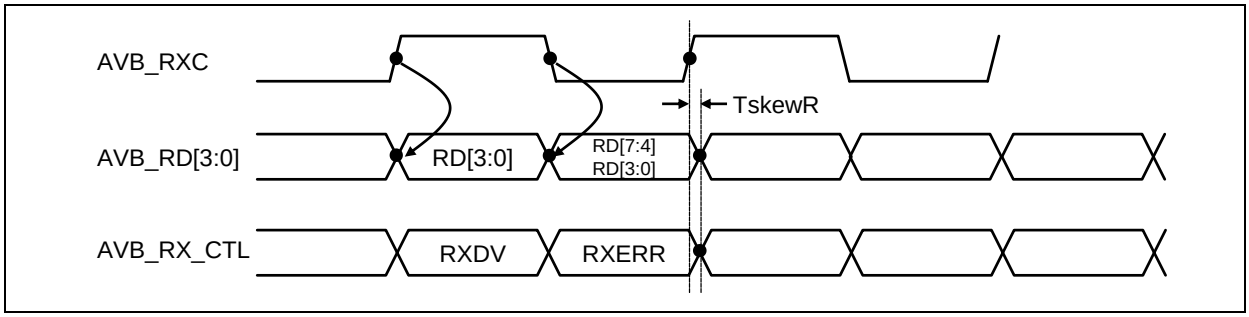



**Figure 200.21 RGMII Delay Mode Reception Timing**

### (4) Ethernet Control Timing (1.8V RGMII Tx clock internal Delay Mode)

**Table 200.50 Ethernet Control Timing (1.8V RGMII Tx clock internal Delay Mode)**

Conditions: **See (10) RGMII signal AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| ETH_TXCREFCLK cycle time | — | 125 - 50 ppm | — | 125 + 50 ppm | MHz | — |
| ETH_TXC cycle time* | Tcyc | 7.2 | 8 | 8.8 | ns | |
| ETH_TXC duty cycle (in 1 Gbps) | Duty_G | 45 | 50 | 55 | % | |
| ETH_TXC duty cycle (in 100 Mbps) | Duty_T | 40 | 50 | 60 | % | |
| ETH_TX_CTL setup time | TsetupT | 1.2 | 2.0 | — | ns | Figure 200.22 |
| ETH_TX_CTL hold time | TholdT | 1.2 | 2.0 | — | ns | |
| ETH_TD[3:0] setup time | TsetupT | 1.2 | 2.0 | — | ns | |
| ETH_TD[3:0] hold time | TholdT | 1.2 | 2.0 | — | ns | |

Notes: 1. ETH is AVB*

Notes: 2. This mode is only valid at 1.8V IO.

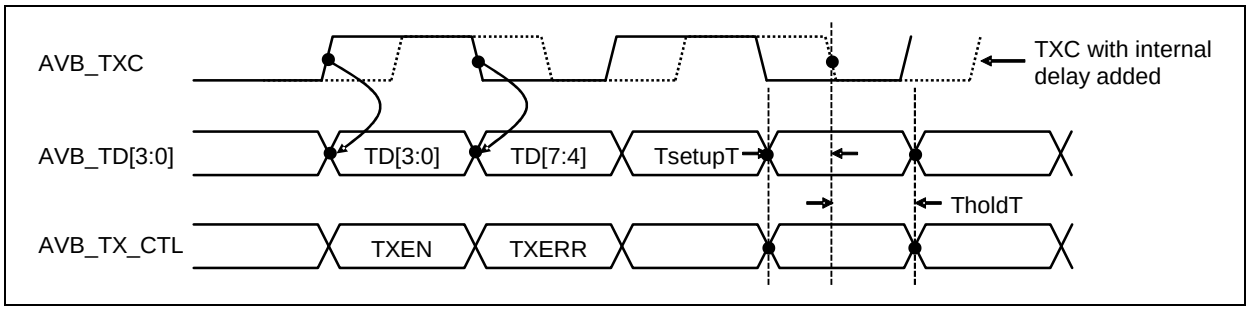



**Figure 200.22 RGMII Delay Mode Reception Timing**

**(5) Ethernet Control Timing (2.5V RGMII 100Mbps Mode)**

**Table 200.51 Ethernet Control Timing (2.5V RGMII 100Mbps Mode)**

Conditions: **See (10) RGMII signal AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| ETH_TXCREFCLK cycle time | — | 125 – 50 ppm | — | 125 + 50 ppm | MHz | — |
| ETH_TXC cycle time | Tcyc | 36 | 40 | 44 | ns | |
| ETH_TXC duty cycle | Duty_T | 40 | 50 | 60 | % | |
| ETH_TX_CTL to clock output skew | TskewT | -510 | 0 | 510 | ps | Figure 200.23 |
| ETH_TD[3:0] to clock output skew | TskewT | -510 | 0 | 510 | ps | |
| ETH_RXC cycle time | Tcyc | 36 | 40 | 44 | ns | — |
| ETH_RXC duty cycle | Duty_T | 40 | 50 | 60 | % | |
| ETH_RX_CTL to clock input skew | TskewR | 1 | 1.8 | 17 | ns | Figure 200.24 |
| ETH_RD[3:0] to clock input skew | TskewR | 1 | 1.8 | 17 | ns | |

Notes: 1. ETH is AVB*

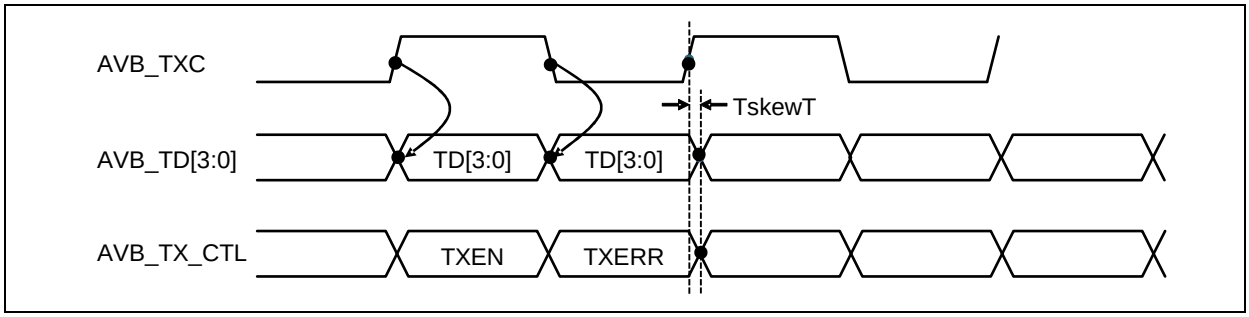



**Figure 200.23 RGMII 100 Mbps Mode Transmission Timing**

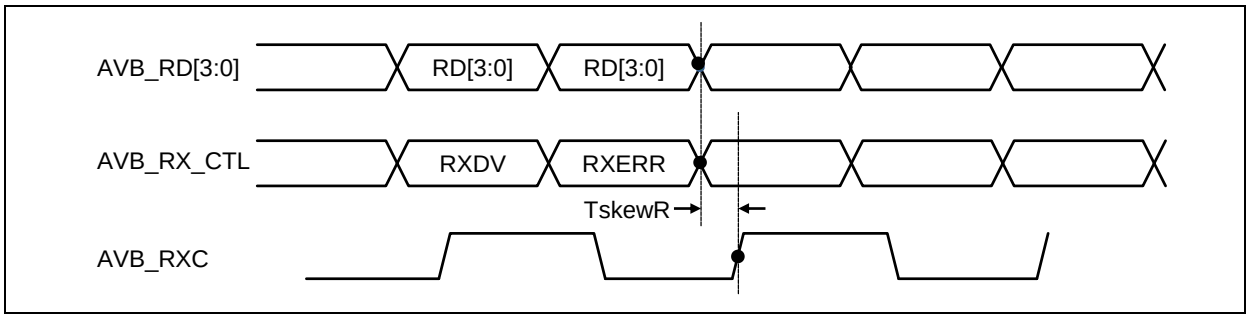



**Figure 200.24 RGMII 100 Mbps Mode Reception Timing**

### (6) Ethernet Control Timing (2.5V RGMII 1Gbps Mode)

**Table 200.52 Ethernet Control Timing (2.5V RGMII 1Gbps Mode)**

Conditions: **See (10) RGMII signal AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| ETH_TXCREFCLK cycle time | — | 125 – 50 ppm | — | 125 + 50 ppm | MHz | — |
| ETH_TXC cycle time | Tcyc | 7.2 | 8 | 8.8 | ns | |
| ETH_TXC duty cycle | Duty_G | 45 | 50 | 55 | % | |
| ETH_TX_CTL to clock output skew | TskewT | -510 | 0 | 510 | ps | Figure 200.25 |
| ETH_TD[3:0] to clock output skew | TskewT | -510 | 0 | 510 | ps | |
| ETH_RXC cycle time | Tcyc | 7.2 | 8 | 8.8 | ns | — |
| ETH_RXC duty cycle | Duty_G | 45 | 50 | 55 | % | |
| ETH_RX_CTL to clock input skew | TskewR | 1 | 1.8 | 2.6 | ns | Figure 200.26 |
| ETH_RD[3:0] to clock input skew | TskewR | 1 | 1.8 | 2.6 | ns | |

Notes: 1. ETH is AVB*

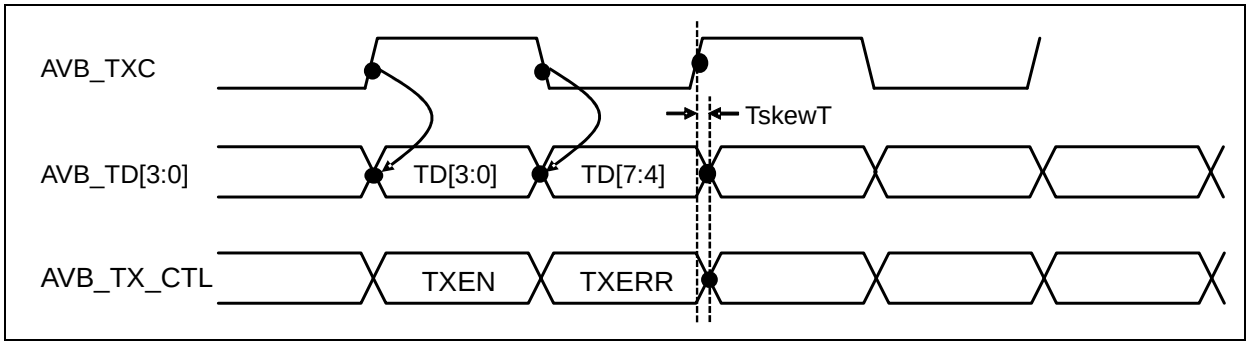



**Figure 200.25 RGMII 1 Gbps Mode Transmission Timing**

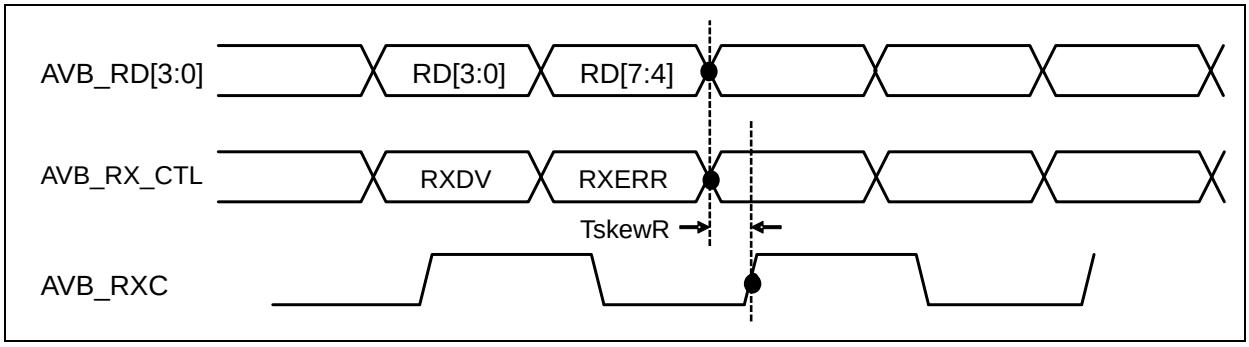



**Figure 200.26 RGMII 1 Gbps Mode Reception Timing**

**(7) Ethernet Control Timing (2.5V RGMII Rx clock internal Delay Mode)**

**Table 200.53 Ethernet Control Timing (2.5V RGMII 1Gbps Mode)**

Conditions: **See (10) RGMII signal AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| ETH_RXC cycle time* | Tcyc | 7.2 | 8 | 8.8 | ns | — |
| ETH_RXC duty cycle (in 1 Gbps) | Duty_G | 45 | 50 | 55 | % | |
| ETH_RXC duty cycle (in 100 Mbps) | Duty_T | 40 | 50 | 60 | % | |
| ETH_RX_CTL to clock input skew | TskewR | -650 | 0 | 650 | ps | Figure 200.27 |
| ETH_RD[3:0] to clock input skew | TskewR | -650 | 0 | 650 | ps | |

Notes: 1. ETH is AVB*

Notes: 2. For 100 Mbps, Tcyc will scale to 40 ns +- 4 ns respectively.

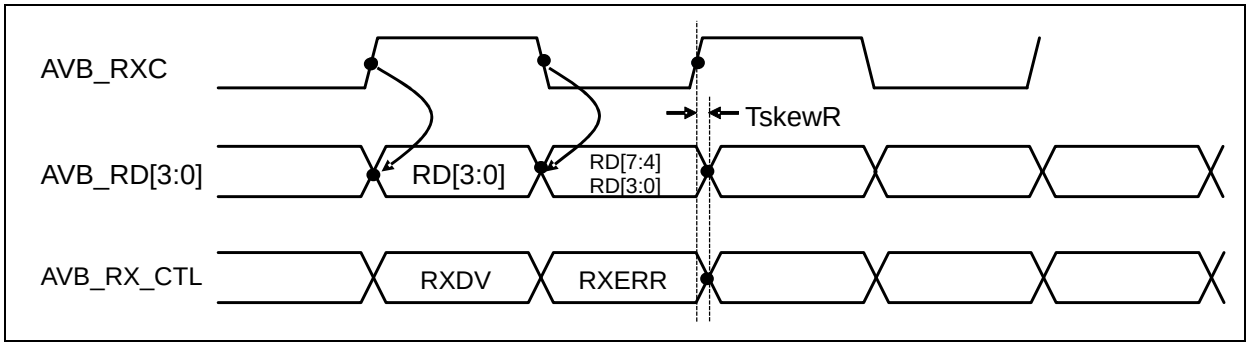



**Figure 200.27 RGMII Delay Mode Reception Timing**

### (8) AVTP pins timing specification

**Table 200.54 AVTP pins timing specification**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| AVB_AVTP_CAPTURE input pulse width | PWidth_capt | 3 | — | — | Cycle* | Figure 200.28 |
| AVB_AVTP_CAPTURE input period | Period_capt | 9 | — | — | Cycle* | |
| AVB_AVTP_MATCH output pulse width | PWidth_match | 1 | — | 4 | Cycle* | Figure 200.29 |
| AVB_AVTP_PPS output pulse width | PWidth_pps | — | 32 | — | Cycle* | Figure 200.30 |

Notes: 1. ETH is AVB*

Notes: 2. 1 cycle = HPϕ (S0D8_HSCϕ) cycle.

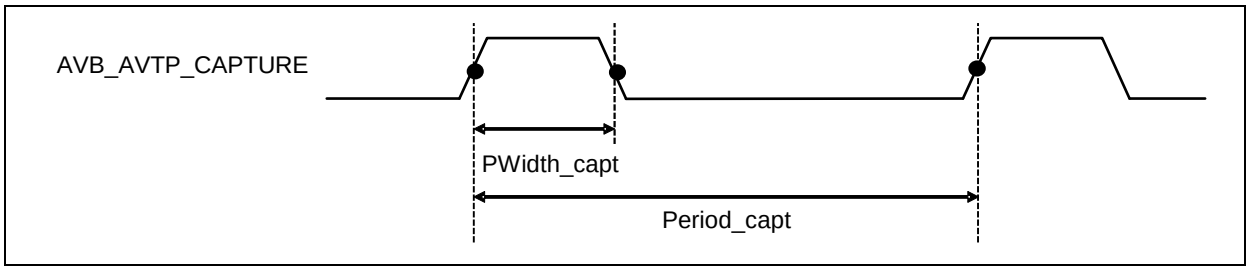



**Figure 200.28 AVB_AVTP_CAPTURE input Timing**

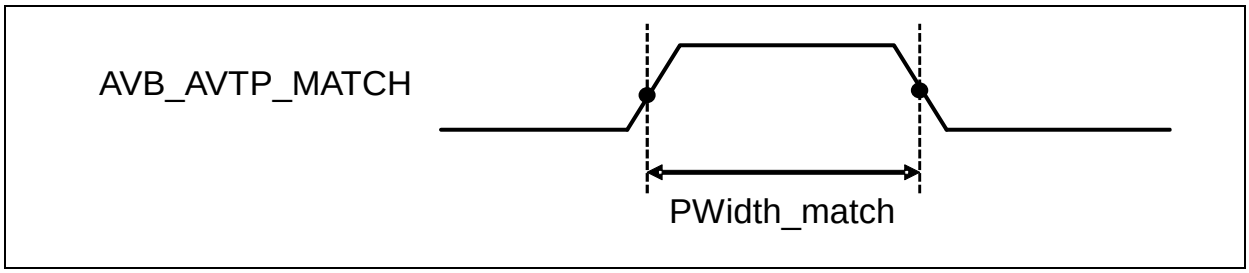



**Figure 200.29 AVB_AVTP_MATCH output Timing**

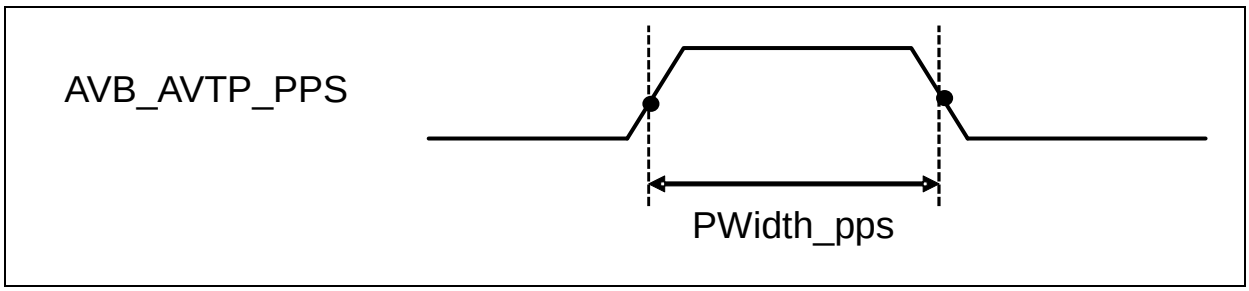



**Figure 200.30 AVB_AVTP_PPS output Timing**

**(9) Sideband pins timing specification**

**Table 200.55 Sideband pins timing specification**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| ETH_MDIO input pulse width | PWidth_side | 2 | — | — | Cycle* | Figure 200.31 |
| ETH_PHY_INT input pulse width | PWidth_side | 2 | — | — | Cycle* | |
| ETH_LINK input pulse width | PWidth_side | 2 | — | — | Cycle* | |

Notes: 1. ETH is AVB*

Notes: 2. 1 cycle = HPϕ (S0D8_HSCϕ) cycle.

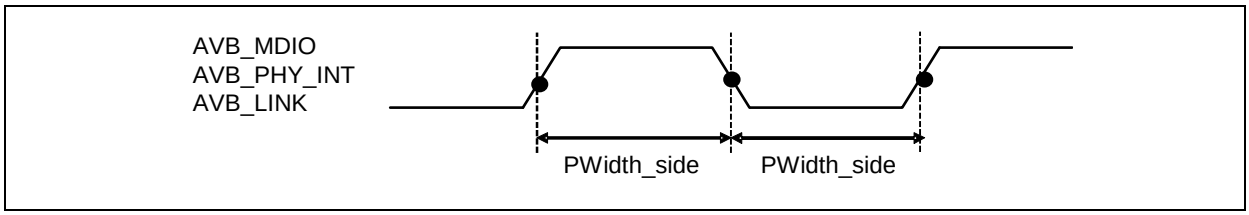



**Figure 200.31 Sideband signals input Timing**

Note: About AVB*_MDC, in case of connecting to a device that does not have a built-in pull resistor, should add pull register at external.

**(10) RGMII signal AC Characteristic Measurement Condition**

Below conditions are valid for RGMII output timing specifications:

- See **200.2, Power Supply**.
- Drive capability = **7** (DRV3,DRV2,DRV1=110) for 1.8V
- Drive capability = **8** (DRV3,DRV2,DRV1=111) for 2.5V

Load condition

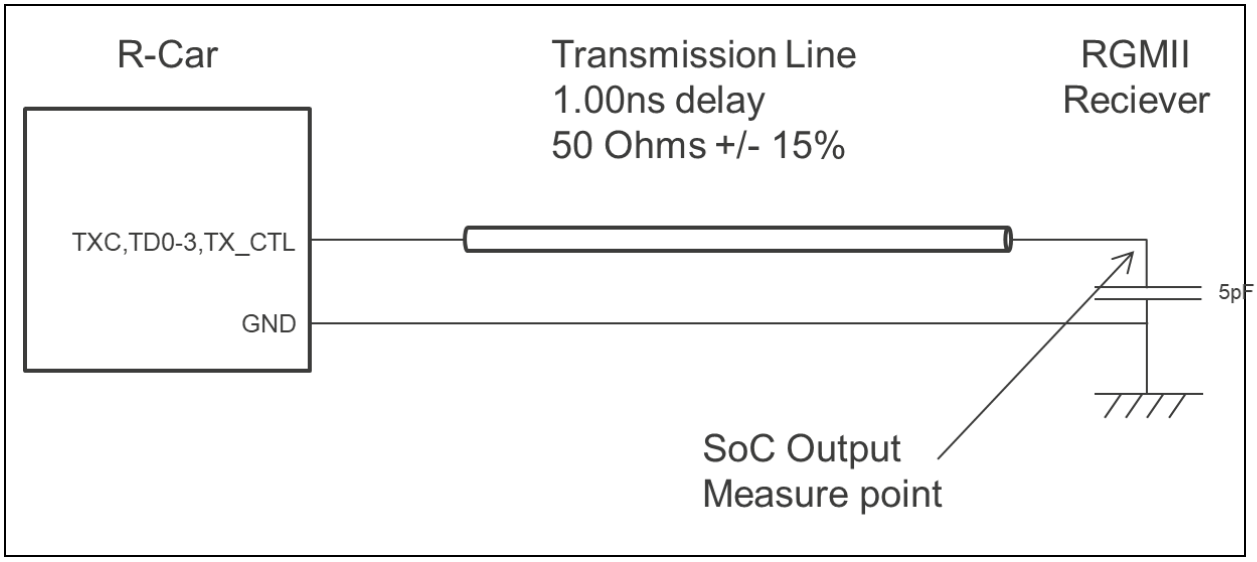



**Figure 200.32 RGMII output signal AC Characteristic Measurement Condition**

The AC timing of the RGMII interface is referenced to the point where the signal passes through half of the power supply voltage.

### 200.7.6 CAN, CAN-FD

**Table 200.56 CAN Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| Input clock cycle (asynchronous) | tSCYC | 20 | — | — | ns | Figure 200.33 |
| Input clock pulse width | tSCKW | 0.4 | — | 0.6 | tSCYC | |
| Input clock rise time | tSCKr | — | — | 0.2 | tCYC | |
| Input clock fall time | tSCKf | — | — | 0.2 | tCYC | |

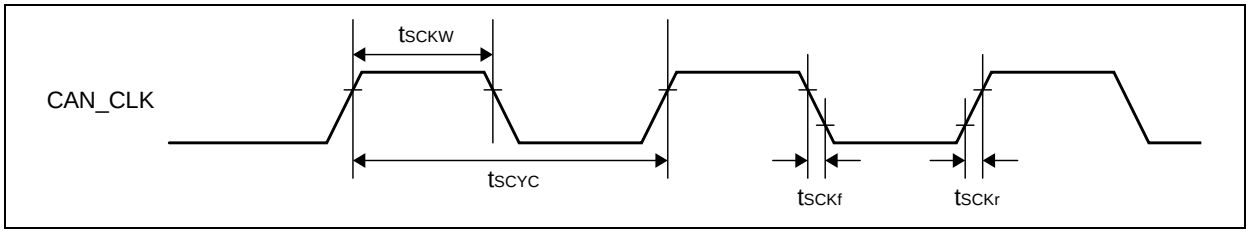



**Figure 200.33 Input Clock Timing**

### 200.7.7 PCIE Controller

The description in this section is compliant with the following PCIe standards:
PCI Express® Base Specification Revision 4.0, Version 1.0, September 27, 2017.
PCI Express® Card Electromechanical Specification, Revision 4.0, Version 1.0, August 7, 2019.

**Table 200.57 PCIE Controller Characteristics**

| Test Item | Symbol | Min. | Typ. | Max. | Unit | Remarks |
|---|---|---|---|---|---|---|
| Differential Peak to Peak Output Voltage (2.5 & 5 GT/s) | VTX-DIFFp-p | 0.800 | — | 1.200 | V | *1 |
| Differential Peak to Peak Output Voltage (8 & 16 GT/s) | VTX-DIFFp-p | 0.800 | — | 1.300 | V | *1 |
| Unit Interval (2.5 GT/s) | UI | 399.88 | — | 400.12 | ps | *2 |
| Unit Interval (5 GT/s) | UI | 199.94 | — | 200.06 | ps | *2 |
| Unit Interval (8 GT/s) | UI | 124.9625 | — | 125.0375 | ps | *2 |
| Unit Interval (16 GT/s) | UI | 62.48125 | — | 62.51875 | ps | *2 |
| External reference clock frequency | — | -100ppm | 100.000 | +100ppm | MHz | — |

Notes: 1. TXP, TXN: DC test
2. Need Reference Clock (Low Voltage Swing, Differential Clocks). The nominal single-ended swing for each clock is 0 to 0.7 V and a nominal frequency of 100 MHz ± 100 ppm.

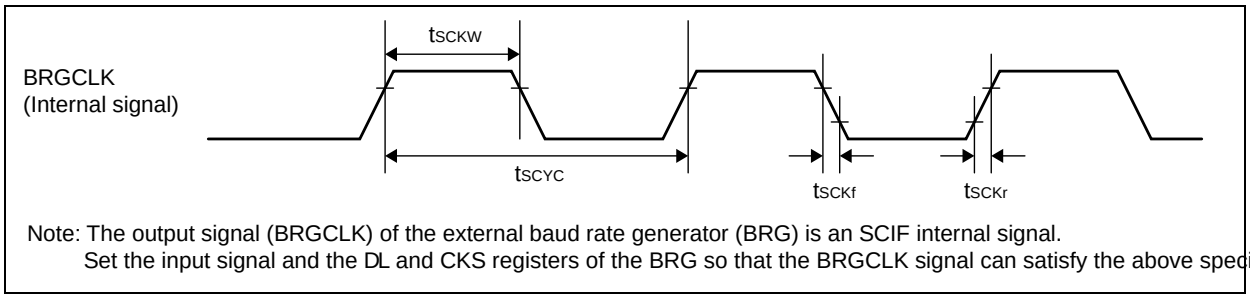
### 200.7.8 SCIF

**Table 200.58 SCIF Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| Input clock cycle (asynchronous) | tSCYC | 4 | — | — | tCYC | Figure 200.34 |
| Input clock cycle (synchronous) | tSCYC | 8 | — | — | tCYC | |
| Input clock pulse width | tSCKW | 0.4 | — | 0.6 | tSCYC | |
| Input clock rise time | tSCKr | — | — | 0.8 | tCYC | |
| Input clock fall time | tSCKf | — | — | 0.8 | tCYC | |
| Transmit data delay time | tTXD | — | — | 4 | tCYC | Figure 200.35 |
| Receive data setup time (synchronous) | tRXS | 5 | — | — | tCYC | |
| Receive data hold time (synchronous) | tRXH | 2 | — | — | tCYC | |

Note: tCYC is for one cycle of the SASYNCPERD4ϕ clock.




Note: The output signal (BRGCLK) of the external baud rate generator (BRG) is an SCIF internal signal.
Set the input signal and the DL and CKS registers of the BRG so that the BRGCLK signal can satisfy the above specification

**Figure 200.34 Input Clock Timing**

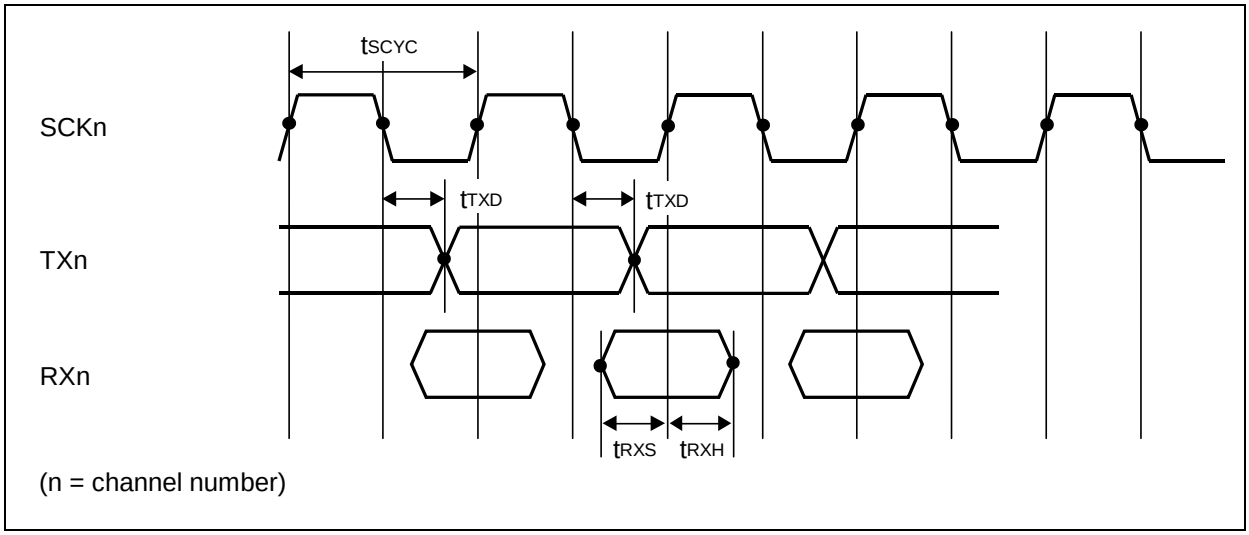



**Figure 200.35 Input/output Timing in Synchronous Mode**

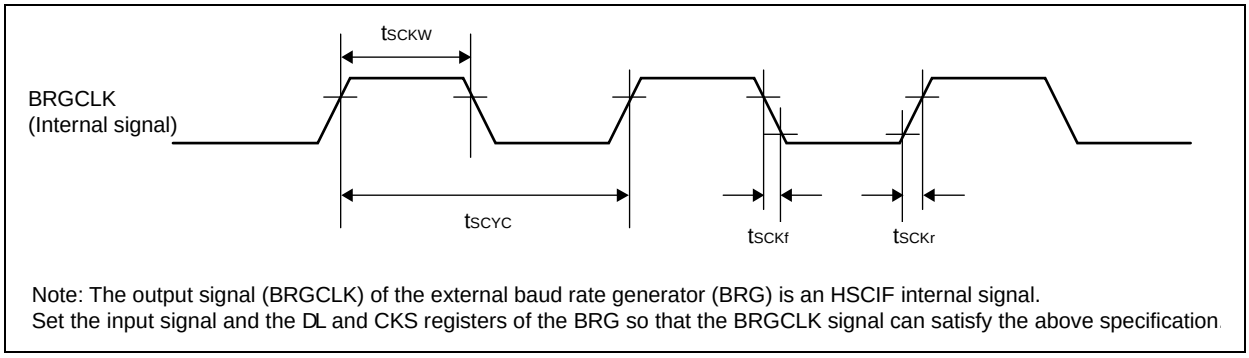
### 200.7.9 HSCIF

**Table 200.59 HSCIF Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| Input clock cycle (asynchronous) | tSCYC | 4 | — | — | tCYC | Figure 200.36 |
| Input clock pulse width | tSCKW | 0.4 | — | 0.6 | tSCYC | |
| Input clock rise time | tSCKr | — | — | 0.8 | tCYC | |
| Input clock fall time | tSCKf | — | — | 0.8 | tCYC | |

Note: tCYC is for one cycle of the SASYNCPERD1ϕ clock.




Note: The output signal (BRGCLK) of the external baud rate generator (BRG) is an HSCIF internal signal.
Set the input signal and the DL and CKS registers of the BRG so that the BRGCLK signal can satisfy the above specification.

**Figure 200.36 Input Clock Timing**

### 200.7.10 I2C

**Table 200.60 I2C Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**
CL =400 pF
Buffer type = Open-Drain type (switching from GPIO mode)

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| I2C_SCL frequency | tICYC | — | — | 400 | kHz | Figure 200.37 |
| I2C_SCL low level time | tLOW | 1/(2 * tICYC) - 100 | — | — | ns | |
| I2C_SCL high level time | tHIGH | 600 | — | — | ns | |
| I2C_SCL/I2C_SDA falling time | tICF | — | — | 250 | ns | |
| I2C_SDA bus free time | tICBF | 1300 | — | — | ns | |
| I2C_SCL start condition hold time | tICH | 600 | — | — | ns | |
| I2C_SCL repeat-start condition setup time | tICS | 600 | — | — | ns | |
| I2C_SDA stop condition setup time | tICST | 600 | — | — | ns | |
| I2C_SDA setup time | tDAS | 100 | — | — | ns | |
| I2C_SDA hold time | tICDH | 0 | — | 900 | ns | |

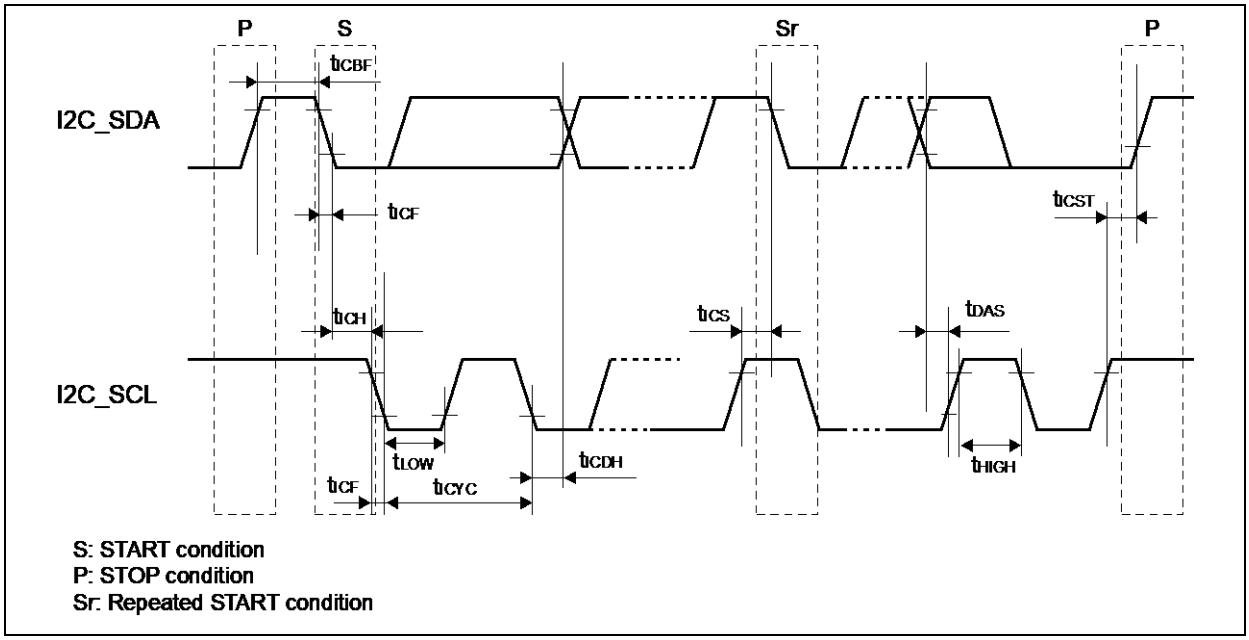



**Figure 200.37 I2C Signal Timing**

**Table 200.61 I2C Fast-Mode Plus Signal Timing**

Conditions: VDDQ33 = 3.3 ± 0.2 V, VDDQ_VIN01 = VDDQ_DU = 3.3 ± 0.2 V / 1.8 V ± 0.1 V, GND = VSS = 0 V,
Ta = -40 to +105 °C, Tj = -40 to +125 °C, CL = 550 pF

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| I2C_SCL frequency | tICYC | — | — | 1000 | kHz | Figure 200.37 |
| I2C_SCL low level time | tLOW | 500 | — | — | ns | |
| I2C_SCL high level time | tHIGH | 260 | — | — | ns | |
| I2C_SCL/I2C_SDA falling time | tICF | — | — | 120 | ns | |
| I2C_SDA bus free time | tICBF | 500 | — | — | ns | |
| I2C_SCL start condition hold time | tICH | 260 | — | — | ns | |
| I2C_SCL repeat-start condition setup time | tICS | 260 | — | — | ns | |
| I2C_SDA stop condition setup time | tICST | 260 | — | — | ns | |
| I2C_SDA setup time | tDAS | 50 | — | — | ns | |
| I2C_SDA hold time | tICDH | 0 | — | — | ns | |

### 200.7.11 MSIOF

**Table 200.62 MSIOF Module Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Max. | Unit | Figures |
|---|---|---|---|---|---|
| MSIOF_SCK clock cycle time (when master TX and master RX) | tMSCYC | 25 | — | ns | Figure 200.38 to Figure 200.41 |
| MSIOF_SCK clock cycle time (when slave RX only) | tMSCYC | 16.666 | — | ns | |
| MSIOF_SCK clock cycle time (when slaver RX and slave TX) | tMSCYC | 40 | — | ns | |
| MSIOF_SCK output clock high-level width | tMSWHO | 0.4 × tMSCYC | — | ns | |
| MSIOF_SCK output clock low-level width | tMSWLO | 0.4 × tMSCYC | — | ns | |
| MSIOF_SCK input high-level width | tMSWHI | 0.4 × tMSCYC | — | ns | |
| MSIOF_SCK input low-level width | tMSWLI | 0.4 × tMSCYC | — | ns | |
| MSIOF_SYNC input setup time in slave mode | tTSFSS | 3 | — | ns | |
| MSIOF_SYNC input hold time in slave mode | tTSFSH | 2 | — | ns | |
| MSIOF_TXD output delay time1 in slave mode | tTSDD1 | 0 | 10 | ns | |
| MSIOF_TXD output delay time2 in slave mode | tTSDD2 | 0 | 12 | ns | |
| MSIOF_RXD input setup time in slave mode | tTSRDS | 3 | — | ns | |
| MSIOF_RXD input hold time in slave mode | tTSRDH | 2 | — | ns | |
| MSIOF_SYNC output delay time in master mode | tTMSFD | -2 | 5 | ns | |
| MSIOF_TXD output delay time in master mode | tTMDD | -2 | 5 | ns | |
| MSIOF_RXD input setup time in master mode | tTMRDS | 3 | — | ns | |
| MSIOF_RXD input hold time in master mode | tTMRDH | 2 | — | ns | |

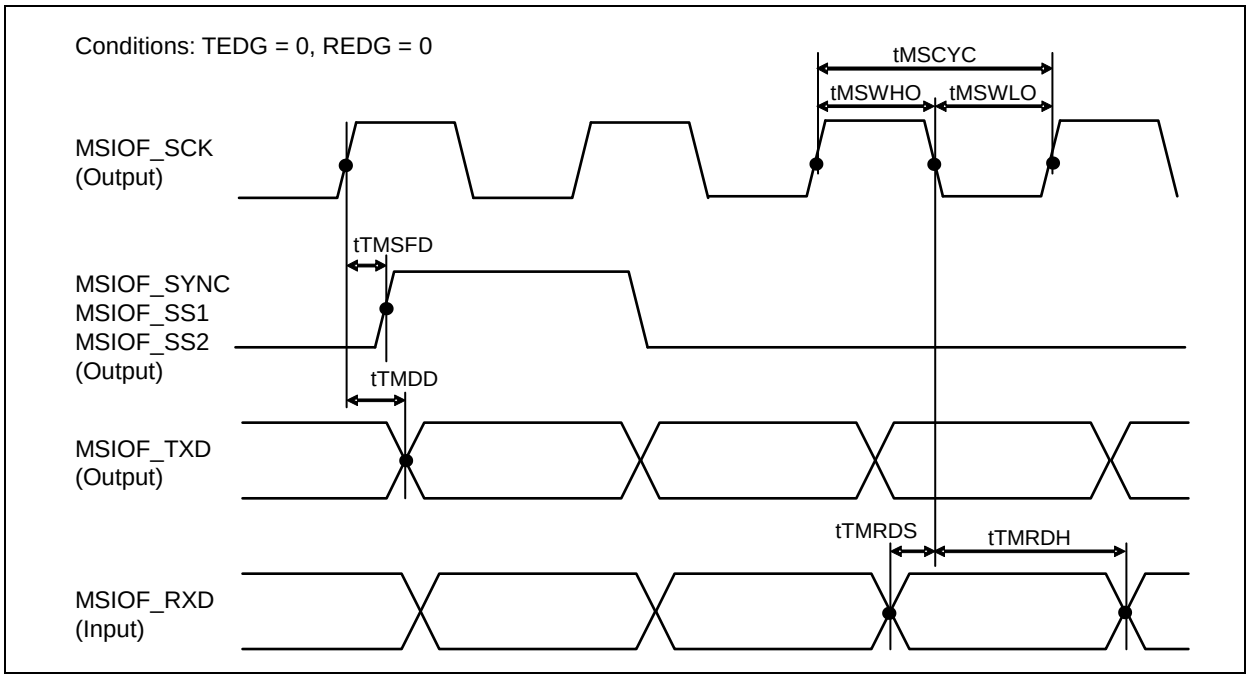



**Figure 200.38 MSIOF Timing (Master Mode)**

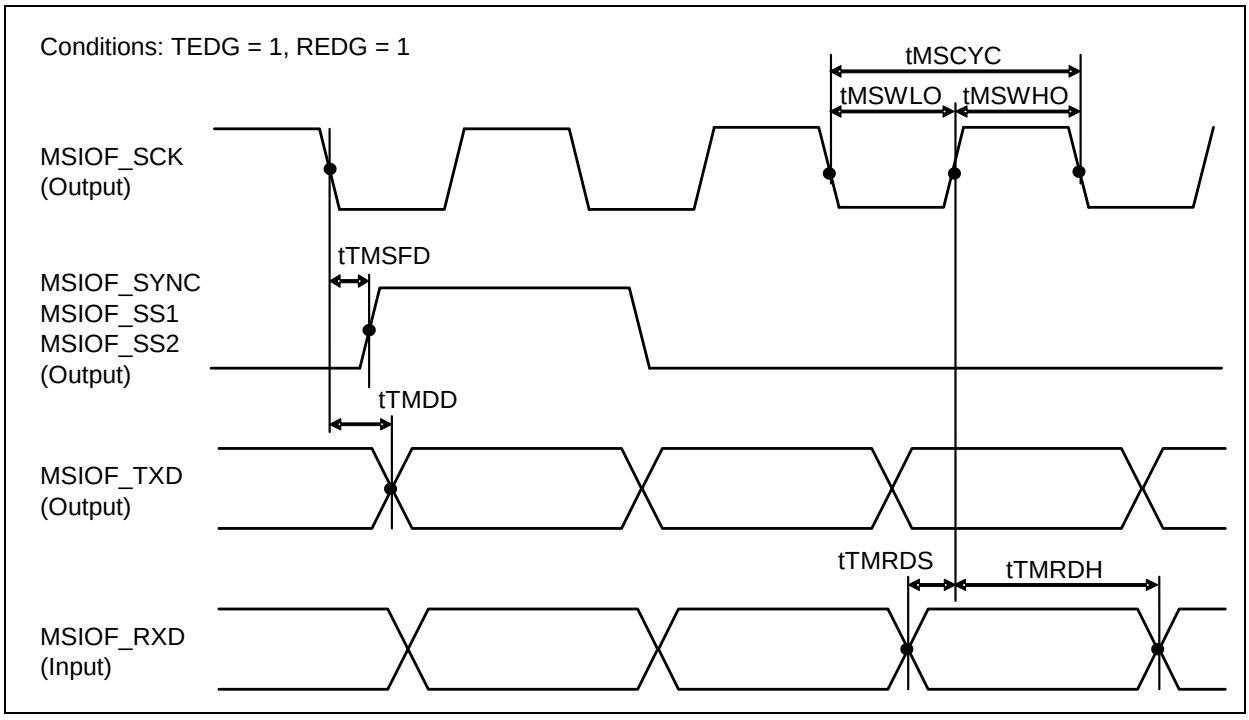



**Figure 200.39 MSIOF Timing (Master Mode)**

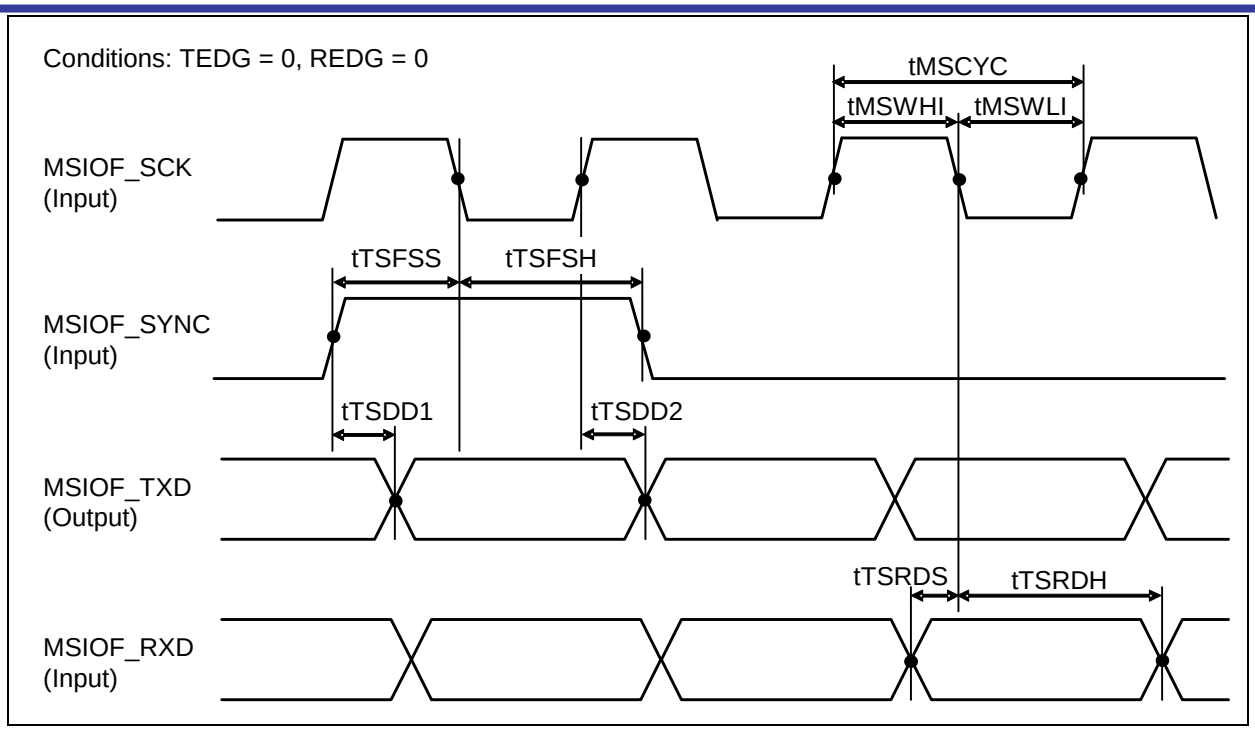



**Figure 200.40 MSIOF Timing (Slave Mode)**

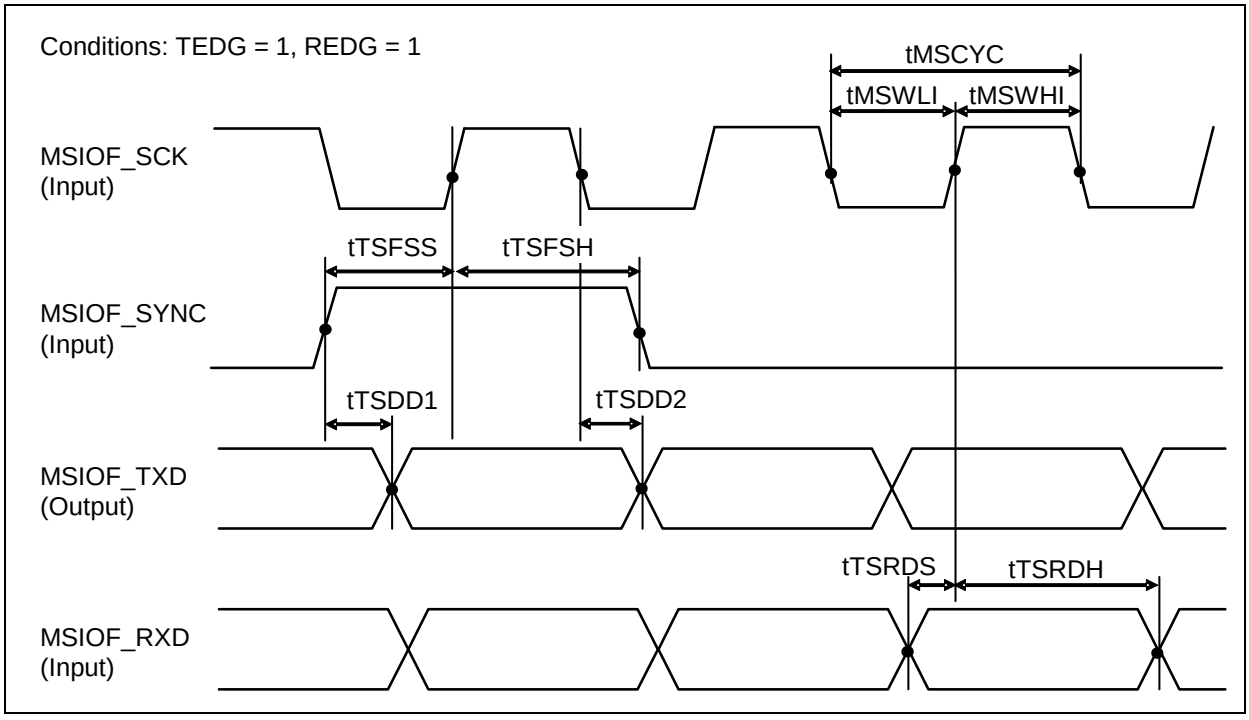



**Figure 200.41 MSIOF Timing (Slave Mode)**

### 200.7.12 RPC Interface

**Table 200.63 RPC Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**
CL = 15 pF

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| SPCLK clock cycle time (Hyper Flash) | tSPCYC | 6.25 | — | — | ns | Figure 200.42 |
| SPCLK clock cycle time (QSPI Flash) | | 12.50 | — | — | ns | |
| SPCLK high pulse width | tSPWH | 0.45 | — | 0.55 | tSPCYC | |
| SPCLK low pulse width | tSPWL | 0.45 | — | 0.55 | tSPCYC | |
| SPCLK slew rate | tSPSR | 0.94 | — | — | V/ns | Figure 200.42 |
| SPCLK differencial crossing voltage | tVOX | 0.40 x VDDQ18 | — | 0.60 x VDDQ18 | V | |
| SSL to setup time | tLEAD | 1 × tSPCYC – 3 | — | 8 × tSPCYC + 3 | ns | Figure 200.43 |
| SSL to hold time | tLAG | 5.5 × tSPCYC – 3 | — | 8.5 × tSPCYC + 3 | ns | Figure 200.44<br>Figure 200.45 |
| Data input valid setup time SDR | tVSS | — | — | 0.5 × tSPCYC + 3 | ns | |
| Data input valid hold time SDR | tVHS | 0 | — | — | ns | |
| Data output setup time | tOS | 0.5 × tSPCYC – 2 | — | — | ns | |
| Data output hold time | tOH | 0.5 × tSPCYC – 2 | — | — | ns | Figure 200.43 |
| Data input valid setup time DDR | tVSD | — | — | 7.0 | ns | |
| Data input valid hold time DDR | tVHD | 0.32 × tSPCYC | — | — | ns | Figure 200.44 |
| Data to Data Strobe Skew | tDDSS | -0.65 | — | 0.65 | ns | |
| SPCLK to Data Strobe Skew | tCDSS | — | — | 7.0 | ns | Figure 200.45 |
| Data Strobe cycle time (HyperFlash) | tDSCYC | 6.25 | — | — | ns | |
| Data Strobe high pulse width | tDSWH | 0.41 | — | 0.59 | tSPCYC | |
| Data Strobe low pulse width | tDSWL | 0.41 | — | 0.59 | tSPCYC | |
| Data Strobe slew rate | tDSSR | 0.94 | — | — | V/ns | Figure 200.47 |
| Output Data Setup time in DDR mode | tSDDR | 0.25 × tSPCYC – 0.6 | — | 0.25 × tSPCYC + 0.6 | ns | |
| Output Data Hold time in DDR mode | tHDDR | 0.25 × tSPCYC – 0.6 | — | 0.25 × tSPCYC + 0.6 | ns | Figure 200.44,<br>Figure 200.45 |
| RPC_RESET assert time | tRPCRA | 0 | — | — | ns | |
| RPC_RESET negate time | tRPCRN | 10 | — | — | ns | Figure 200.46 |

Correspondence of each memory interface is shown.

| Item | Symbol | HyperFlash | Octal-SPI | Serial Flash |
|---|---|---|---|---|
| SPCLK clock cycle time | tSPCYC | ✔ | ✔ | - |
| SPCLK clock cycle time | | - | ✔ *1 | ✔ |
| SPCLK high pulse width | tSPWH | ✔ | ✔ | ✔ |
| SPCLK low pulse width | tSPWL | ✔ | ✔ | ✔ |
| SPCLK slew rate | tSPSR | ✔ | ✔ | - |
| SPCLK differencial crossing voltage | tVOX | ✔ | - | - |
| SSL to setup time | tLEAD | ✔ | ✔ | ✔ |
| SSL to hold time | tLAG | ✔ | ✔ | ✔ |
| Data input valid setup time SDR | tVSS | - | ✔ *1 | ✔ |
| Data input valid hold time SDR | tVHS | - | ✔ *1 | ✔ |
| Data output setup time | tOS | - | ✔ *1 | ✔ |
| Data output hold time | tOH | - | ✔ *1 | ✔ |
| Data input valid setup time DDR | tVSD | - | - | ✔ |
| Data input valid hold time DDR | tVHD | - | - | ✔ |
| Data to Data Strobe Skew | tDDSS | ✔ | ✔ | - |
| SPCLK to Data Strobe Skew | tCDSS | ✔ | ✔ | - |
| Data Strobe cycle time | tDSCYC | ✔ | ✔ | - |
| Data Strobe high pulse width | tDSWH | ✔ | ✔ | - |
| Data Strobe low pulse width | tDSWL | ✔ | ✔ | - |
| Data Strobe slew rate | tDSSR | ✔ | ✔ | - |
| Output Data Setup time in DDR mode | tSDDR | ✔ | ✔ | ✔ |
| Output Data Hold time in DDR mode | tHDDR | ✔ | ✔ | ✔ |
| RPC_RESET assert time | tRPCRA | ✔ | ✔ | ✔ |
| RPC_RESET negate time | tRPCRN | ✔ | ✔ | ✔ |

Notes: 1. only 1bit single data rate (SDR)

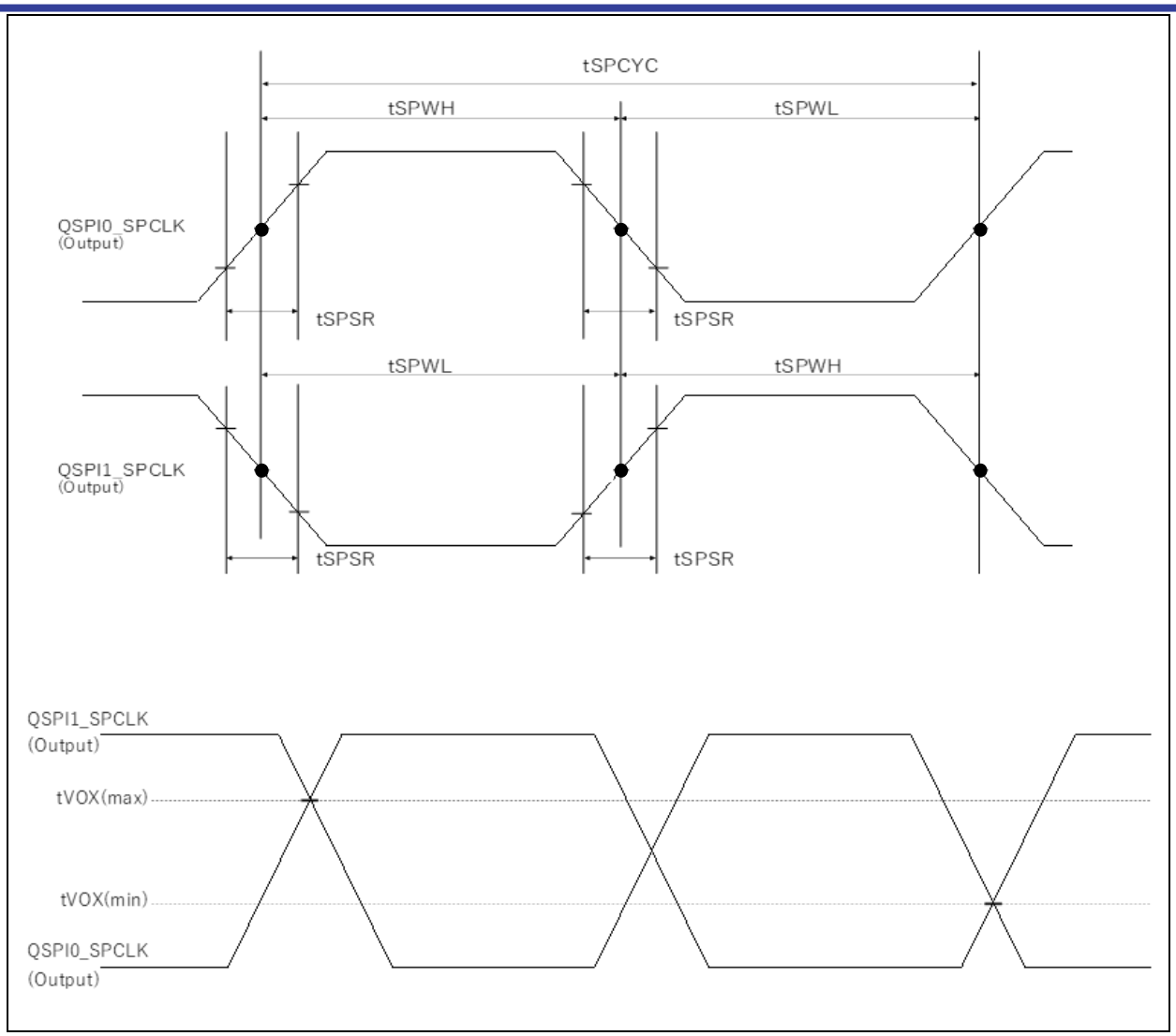
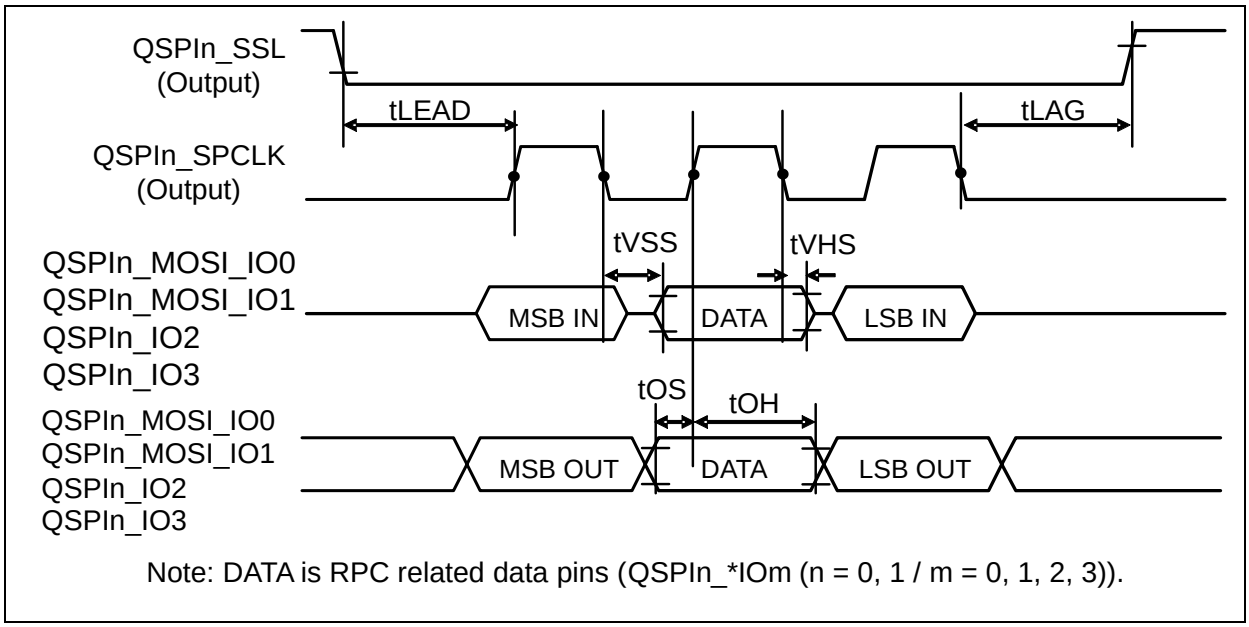



**Figure 200.42 RPC Clock Timing**




Note: DATA is RPC related data pins (QSPIn_*IOm (n = 0, 1 / m = 0, 1, 2, 3)).

**Figure 200.43 RPC SDR Operation Timing (QSPI Flash)**

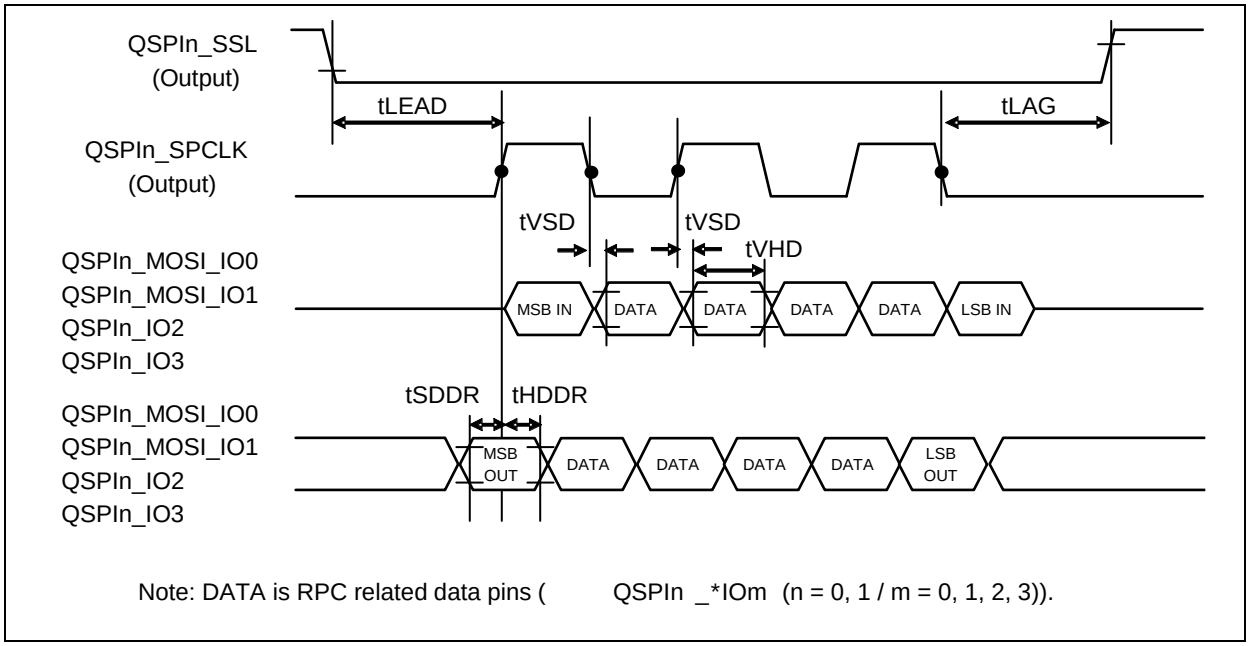



**Figure 200.44 RPC DDR Operation Timing (QSPI Flash)**

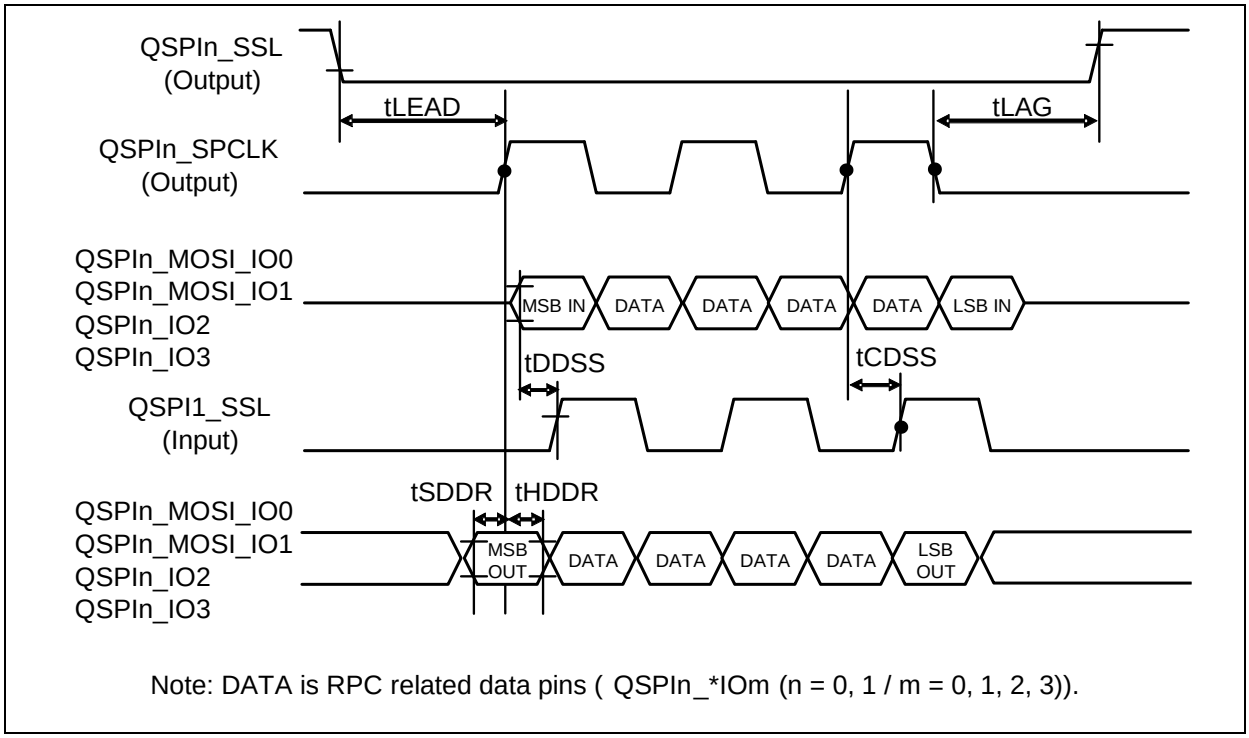



**Figure 200.45 RPC DDR Operation Timing (HyperFlash)**

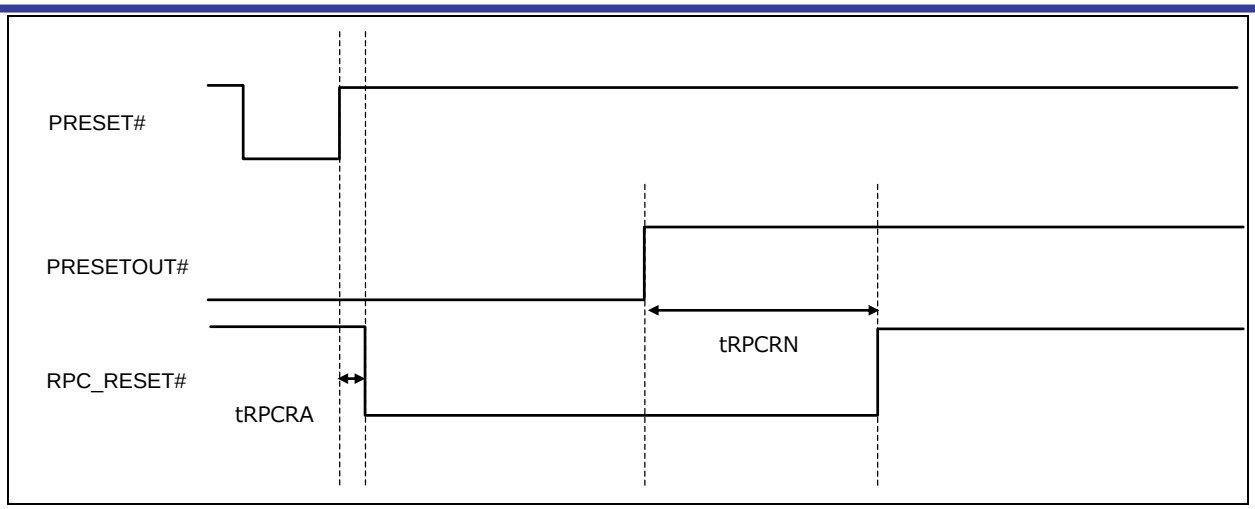



**Figure 200.46 RPC_RESET# Operation Timing**

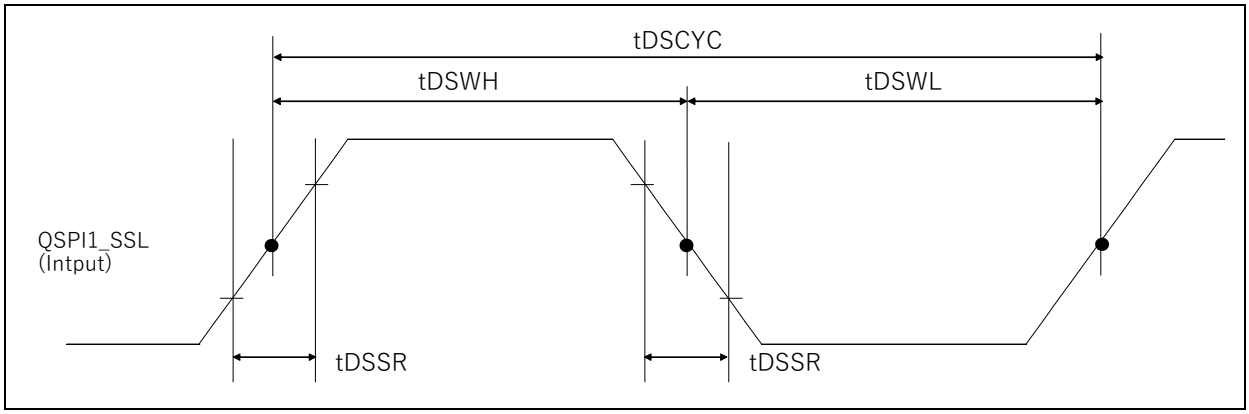



**Figure 200.47 RPC DDR Data Strobe Timing**

### 200.7.13 SD Host Interface (SDHI)

#### (1) SDHI(3.3V) Signal Timing (Default mode)

**Table 200.64 SDHI(3.3V) Signal Timing (Default mode)**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| SDCLK clock cycle time | tSDCYC | 40.0 | — | — | ns | Figure 200.48 |
| SDCMD output data delay time | tSDCMD_S | — | — | 10.0 | ns | |
| | tSDCMD_H | 6.0 | — | — | ns | |
| SDDAT output data delay time | tSDDAD_S | — | — | 10.0 | ns | |
| | tSDDAD_H | 6.0 | — | — | ns | |
| SDCMD input data setup time | tSDCMS | 3.0 | — | — | ns | |
| SDCMD input data hold time | tSDCMH | 2.0 | — | — | ns | |
| SDDAT input data setup time | tSDDAS | 3.0 | — | — | ns | |
| SDDAT input data hold time | tSDDAH | 2.0 | — | — | ns | |

Note: This interface can be used for debug purpose. Electrical characteristics described in this table is a target specification but not guaranteed in V4H and V4M officially.

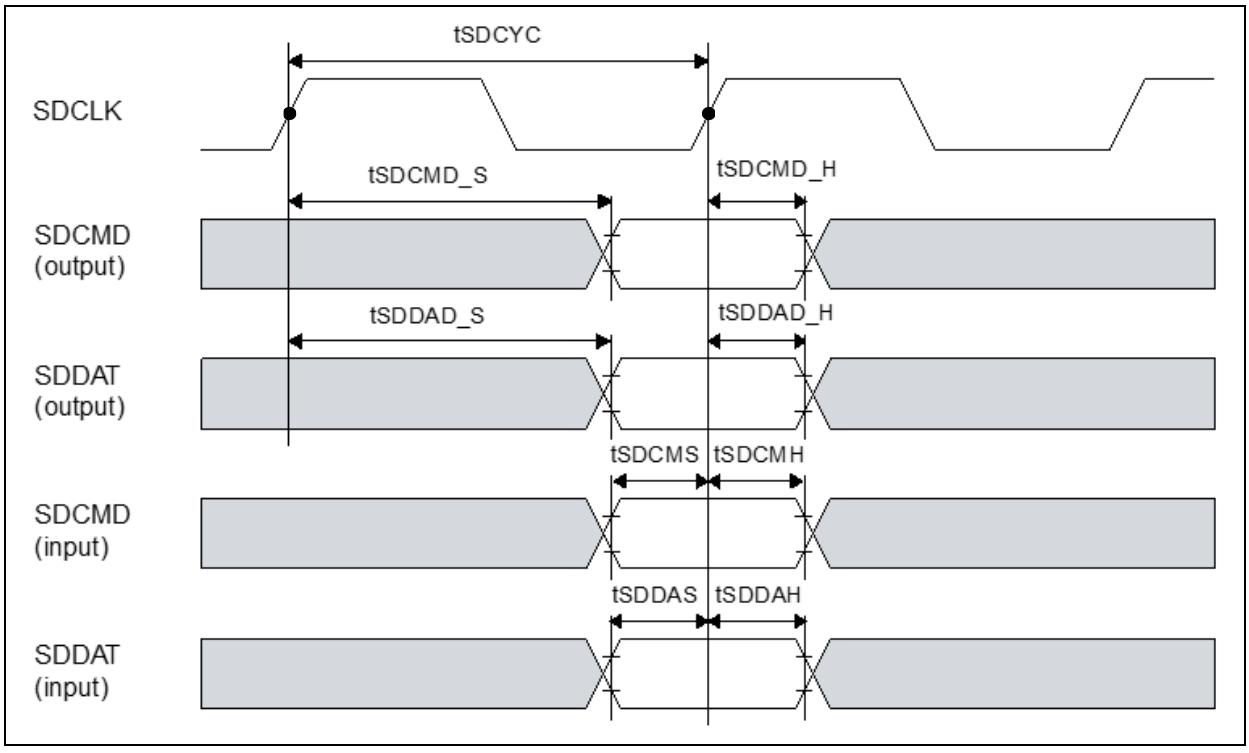



**Figure 200.48 SDHI Signal Timing (Default mode)**

### (2) SDHI(3.3V) Signal Timing (High Speed mode)

**Table 200.65 SDHI(3.3V) Signal Timing (High Speed mode)**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**
CL = 40 pF

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| SDCLK clock cycle time | $t_{SDCYC}$ | 20.0 | — | — | ns | Figure 200.49 |
| SDCMD output data delay time | $t_{SDCMD}$ | 3.0 | — | 0.5 × $t_{SDCYC}$ + 3.0 | ns | |
| SDCMD output data hold time | $t_{SDCMDOH}$ | 3.0 | — | — | ns | |
| SDDAT output data delay time | $t_{SDDAD}$ | 3.0 | — | 0.5 × $t_{SDCYC}$ + 3.0 | ns | |
| SDDAT output data hold time | $t_{SDDATOH}$ | 3.0 | — | — | ns | |
| SDCMD input data setup time | $t_{SDCMS}$ | 3.0 | — | — | ns | |
| SDCMD input data hold time | $t_{SDCMH}$ | 2.0 | — | — | ns | |
| SDDAT input data setup time | $t_{SDDAS}$ | 3.0 | — | — | ns | |
| SDDAT input data hold time | $t_{SDDAH}$ | 2.0 | — | — | ns | |

Note: This interface can be used for debug purpose. Electrical characteristics described in this table is a target specification but not guaranteed in V4H and V4M officially.

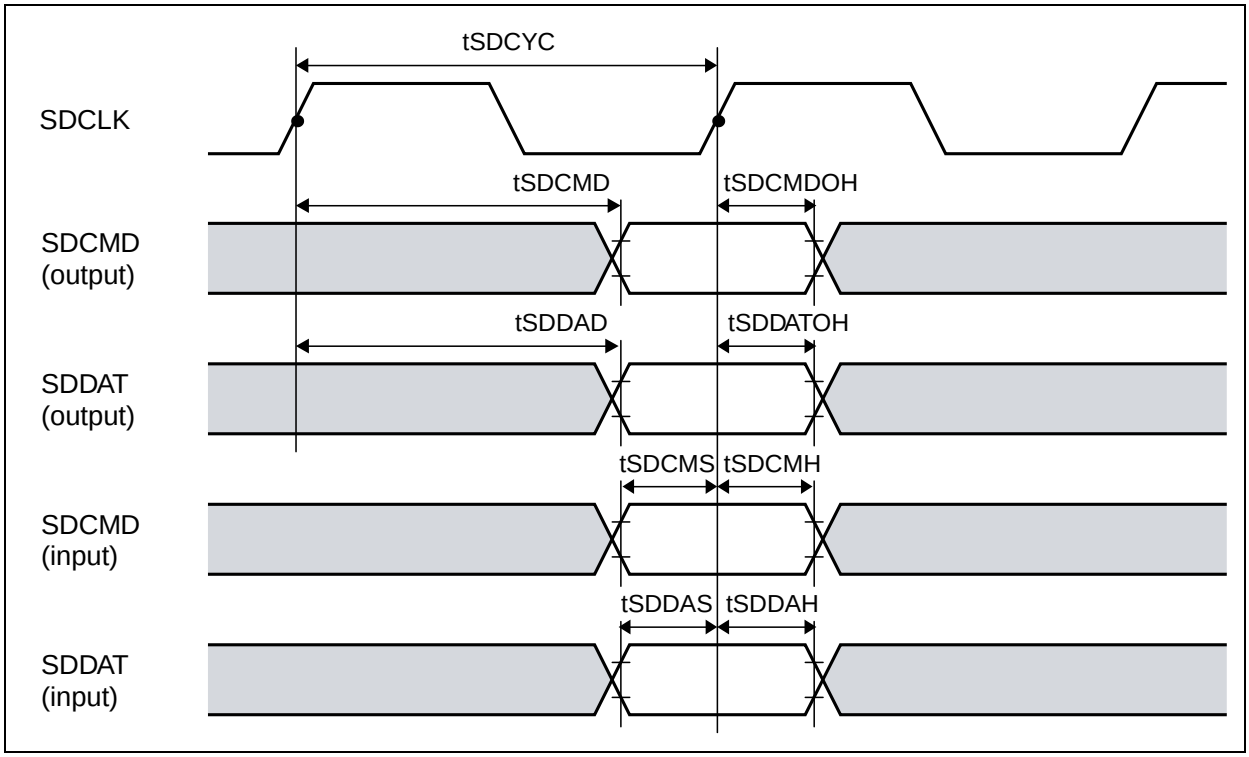



**Figure 200.49 SDHI Signal Timing (High Speed mode)**

### 200.7.14 MMC Interface

#### (1) MMC(3.3V) Signal Timing (High Speed mode)

**Table 200.66 MMC(3.3V) Signal Timing (High Speed mode)**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| MMCCLK clock cycle time | tMMCCYC | 20.0 | — | — | ns | Figure 200.50 |
| MMCCMD output data delay time | tMMCCMD | 4.0 | — | 0.5 × tMMCCYC + 3.0 | ns | |
| MMCCMD output data hold time | tMMCCMDOH | 4.0 | — | — | ns | |
| MMCDAT output data delay time | tMMCDAD | 4.0 | — | 0.5 × tMMCCYC + 3.0 | ns | |
| MMCDAT output data hold time | tMMCDATOH | 4.0 | — | — | ns | |
| MMCCMD input data setup time | tMMCCMS | 3.0 | — | — | ns | |
| MMCCMD input data hold time | tMMCCMH | 2.0 | — | — | ns | |
| MMCDAT input data setup time | tMMCDAS | 3.0 | — | — | ns | |
| MMCDAT input data hold time | tMMCDAH | 2.0 | — | — | ns | |

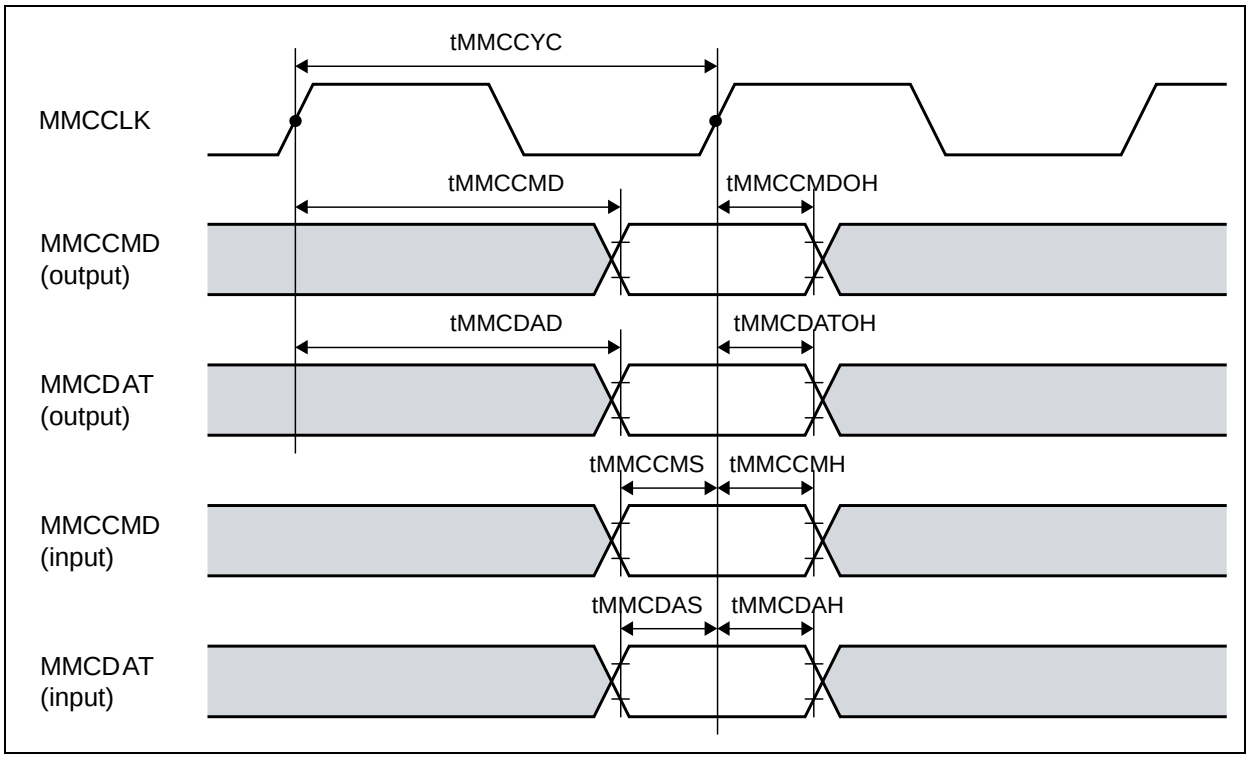



**Figure 200.50 MMC Signal Timing (High Speed mode)**

### (2) MMC(1.8V) Signal Timing (High Speed mode)

**Table 200.67 MMC(1.8V) Signal Timing (High Speed mode)**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**
**See 200.5.6 for measurement points of bar notation**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| MMCCLK clock cycle time | $t_{MMCCYC}$ | 20.0 | — | — | ns | Figure 200.51 |
| MMCCMD output data delay time | $t_{MMCCMD}$ | 4.0 | — | 0.5 × $t_{MMCCYC}$ + 3.0 | ns | |
| MMCCMD output data hold time | $t_{MMCCMDOH}$ | 4.0 | — | — | ns | |
| MMCDAT output data delay time | $t_{MMCDAD}$ | 4.0 | — | 0.5 × $t_{MMCCYC}$ + 3.0 | ns | |
| MMCDAT output data hold time | $t_{MMCDATOH}$ | 4.0 | — | — | ns | |
| MMCCMD input data setup time | $t_{MMCCMS}$ | 3.0 | — | — | ns | |
| MMCCMD input data hold time | $t_{MMCCMH}$ | 2.0 | — | — | ns | |
| MMCDAT input data setup time | $t_{MMCDAS}$ | 3.0 | — | — | ns | |
| MMCDAT input data hold time | $t_{MMCDAH}$ | 2.0 | — | — | ns | |

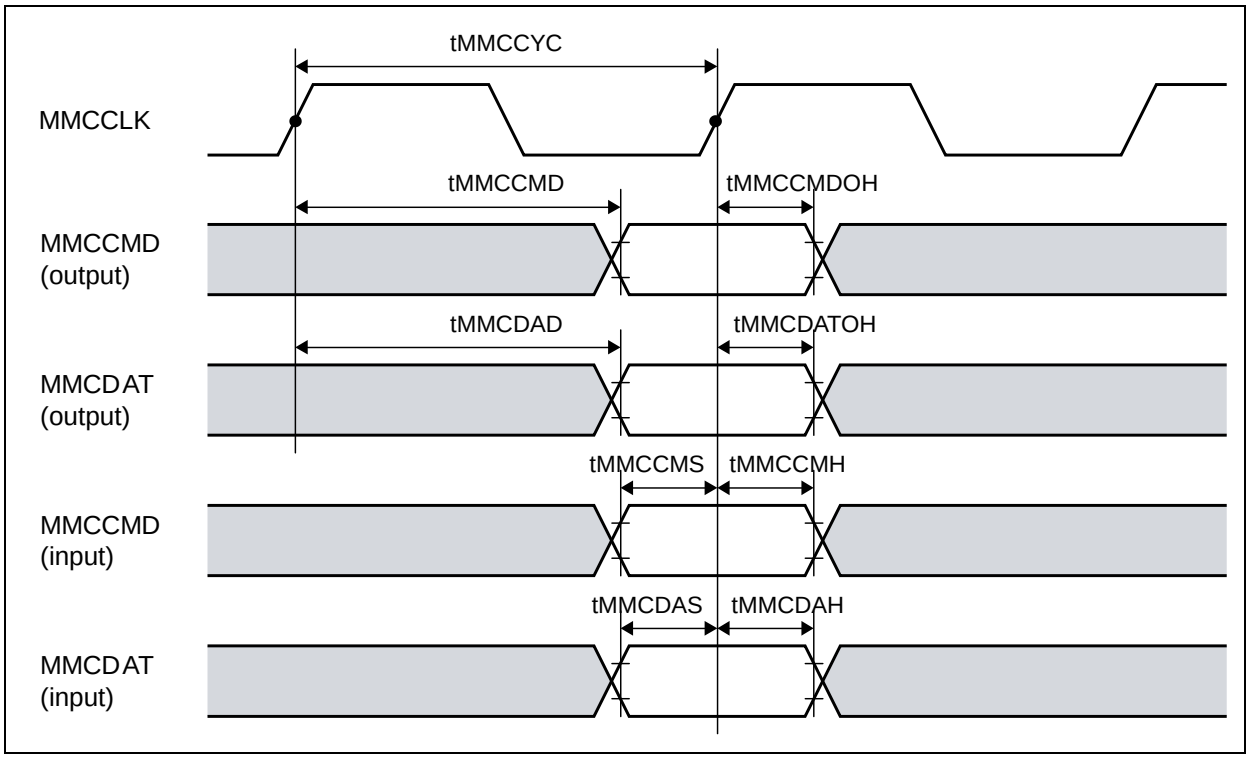



**Figure 200.51 MMC Signal Timing**
**(See 200.5.6 for measurement points of bar notation)**

### (3) MMC(1.8V) Signal Timing (HS200 mode)

**Table 200.68 MMC(1.8V) Signal Timing (HS200 mode)**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**
**See 200.5.6 for measurement points of bar notation**
CL = 10 pF

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| MMCCLK clock cycle time | tMMCCYC | 5.0 | — | — | ns | Figure 200.52 |
| MMCCMD output data delay time | tMMCCMD | 1.30 | — | 3.50 | ns | |
| MMCDAT output data delay time | tMMCDAD | 1.30 | — | 3.50 | ns | |
| MMCCMD input data width (for tuning) | tMMCICW | 2.57 | 3.07 | — | ns | |
| MMCCMD input data width (after tuning) | tMMCICW | 2.57 | - | - | ns | |
| MMCDAT input data width (for tuning) | tMMCIDW | 2.57 | 3.07 | — | ns | |
| MMCDAT input data width (after tuning) | tMMCIDW | 2.57 | - | - | ns | |

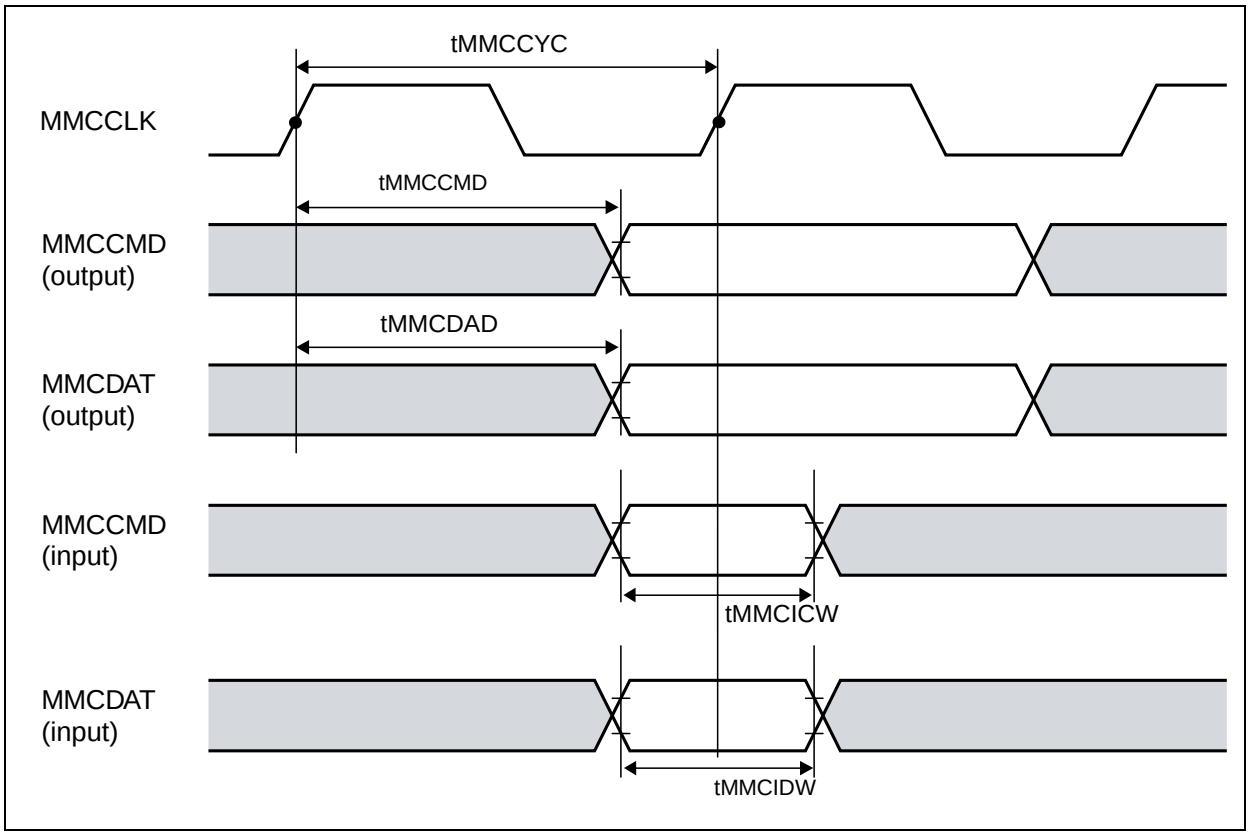



**Figure 200.52 MMC Signal Timing (HS200 mode)**
**(See 200.5.6 for measurement points of bar notation)**

**(4) MMC(1.8V) Signal Timing (HS400 mode)**

**Table 200.69 MMC(1.8V) Signal Timing (HS400 mode)**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**
**See 200.5.6 for measurement points of bar notation**
CL = 4 pF

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| MMCCLK clock cycle time | $t_{MMCCYC}$ | 5.0 | — | — | ns | Figure 200.53 |
| MMCDS data strobe cycle time | $t_{MMCDCYC}$ | 5.0 | — | — | ns | |
| MMCDS minimum pulse with | $t_{DSW}$ | 2.0 | — | — | ns | |
| Input Slew rate with MMCDS | — | 1.125 | — | — | V/ns | |
| Input Slew rate with MMCDAT | — | 1.125 | — | — | V/ns | |
| MMCDAT input data setup time | $t_{MMCDIDS}$ | — | — | 0.6 | ns | |
| MMCDAT input data hold time | $t_{MMCDIDH}$ | 0.5*tMMCDCYC-1.0 | — | — | ns | |
| MMCDAT output data setup time | $t_{MMCDWDS}$ | 0.6 | — | — | ns | |
| MMCDAT output data hold time | $t_{MMCDWDH}$ | 0.6 | — | — | ns | |

The MMCCMD input timing for HS400 mode is the same as CMD timing for HS200 mode.

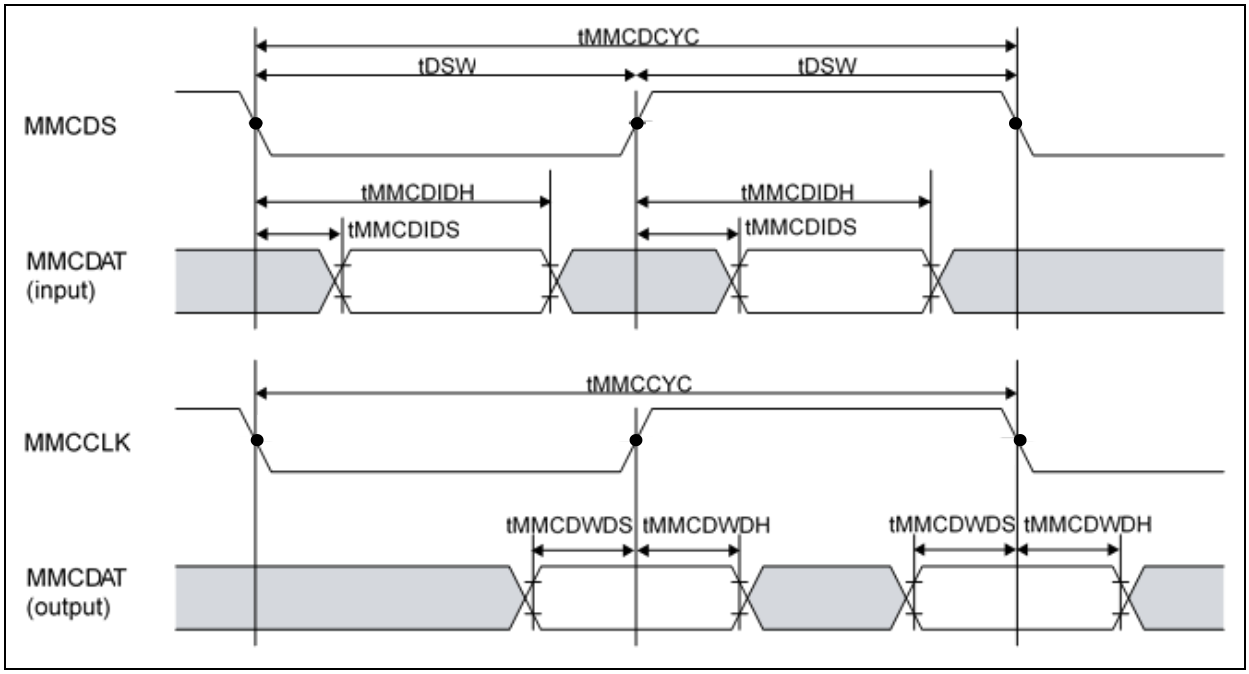



**Figure 200.53 MMC Signal Timing (HS400 mode)**
**(See 200.5.6 for measurement points of bar notation)**

### 200.7.15 TMU

**Table 200.70 TMU Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figure |
|---|---|---|---|---|---|---|
| TCLK clock cycle | tTCLKCY | 4 | — | 16.37 | tCYC | |
| Input clock pulse width | tTCLKW | 0.4 | — | 0.6 | tTCLKCY | |

Note: tCYC is for one cycle of the SASYNCPERD2φ clock.

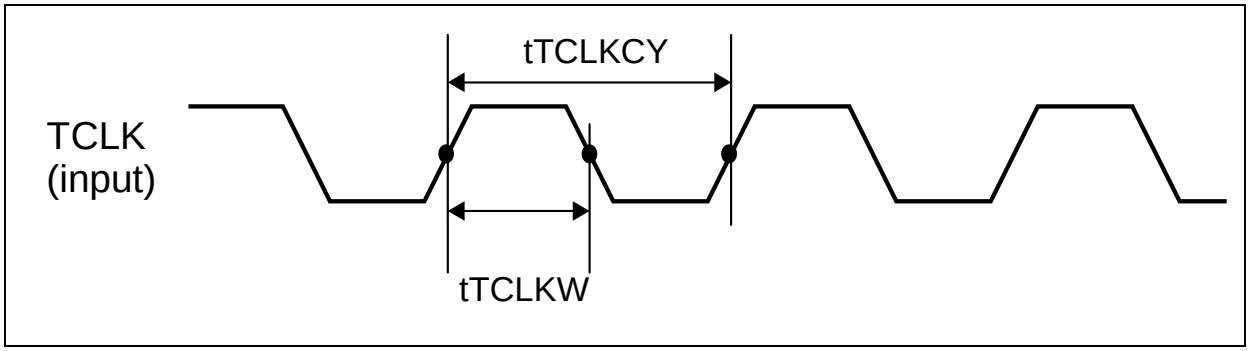



**Figure 200.54 TMU Signal Timing**

### 200.7.16 Reserved

### 200.7.17 DBSC5 Access Timing (LPDDR4X)

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

**Table 200.71 DBSC5 Access Timing (LPDDR4X)**

| Item | Symbol | Min. | Max. | Unit | Figure |
|---|---|---|---|---|---|
| MCK average clock period | tCK(avg) | 0.468 | 20 | ns | Figure 200.55 |
| MCK absolute high pulse width | tCH(abs) | 0.45 | 0.55 | tCK(avg) | |
| MCK absolute low pulse width | tCL(abs) | 0.45 | 0.55 | tCK(avg) | |
| Command, Address and Control output setup time to MCK, MCK# | tOS(1T) | 168 | — | ps | Figure 200.56 |
| Command, Address and Control output hold time from MCK, MCK# | tOH(1T) | 168 | — | ps | |
| Command, Address and Control pulse width for each output | tOPW(1T) | 349 | — | ps | |
| MDQS, MDQS# rising edge to MCK, MCK# rising edge (write) | tWDQSS | 0.77 | 1.23 | tCK(avg) | Figure 200.57 |
| MDQS, MDQS# falling edge setup time to MCK, MCK# rising edge (write) | tWDSS | 0.22 | — | tCK(avg) | |
| MDQS, MDQS# falling edge hold time from MCK, MCK# rising edge (write) | tWDSH | 0.22 | — | tCK(avg) | |
| MDQS, MDQS# differential high pulse width (write) | tWDQSH | 0.4 | — | tCK(avg) | Figure 200.58 |
| MDQS, MDQS# differential low pulse width (write) | tWDQSL | 0.4 | — | tCK(avg) | |
| MDQS, MDQS# differential WRITE Preamble (write) | tWPRE | 1.8 | — | tCK(avg) | |
| MDQS, MDQS# differential WRITE Postamble (write) | tWPST | 0.4 | — | tCK(avg) | |
| MDQ and MDM output setup time to MDQS, MDQS# (write) | tWDS | 64 | — | ps | Figure 200.59 |
| MDQ and MDM output hold time from DQS, DQS# (write) | tWDH | 64 | — | ps | |
| MDQ and MDM output pulse width for each output (write) | tWDIPW | 128 | — | ps | |
| MDQS, MDQS# rising edge input access time from rising MCK, MCK# (read) | tRDQSCK | 1500 | 4600 | ps | Figure 200.60 |
| MDQS, MDQS# differential input high pulse width (read) | tRQSH | 0.38 | — | tCK(avg) | Figure 200.61 |
| MDQS, MDQS# differential input low pulse width (read) | tRQSL | 0.38 | — | tCK(avg) | |
| MDQS, MDQS# differential READ Preamble (read) | tRPRE | 1.8 | — | tCK(avg) | |
| MDQS, MDQS# differential READ Postamble (read) | tRPST | 0.4 | — | tCK(avg) | |
| Read Data Eye Mask (read) | TdIVW_total | — | 0.25 | UI | Figure 200.62 |
| | VdIVW_total | 120 | — | mV | |

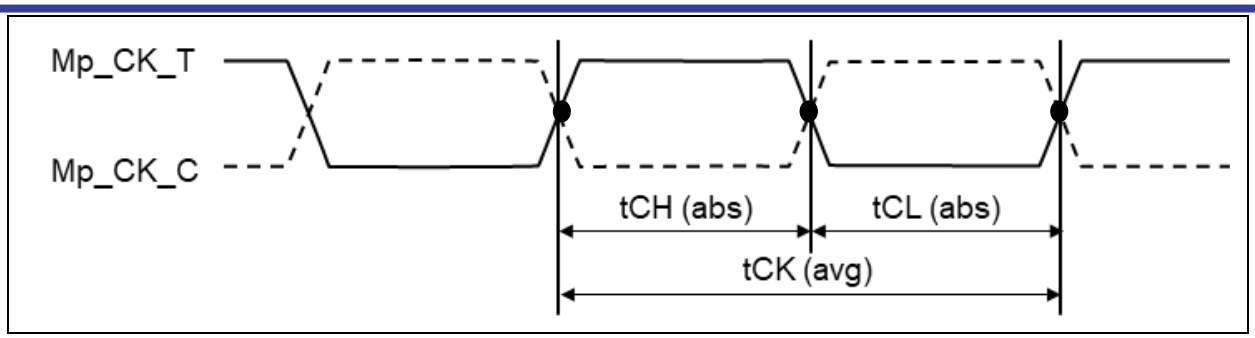



**Figure 200.55 MCK Clock Output(p=0,1)**

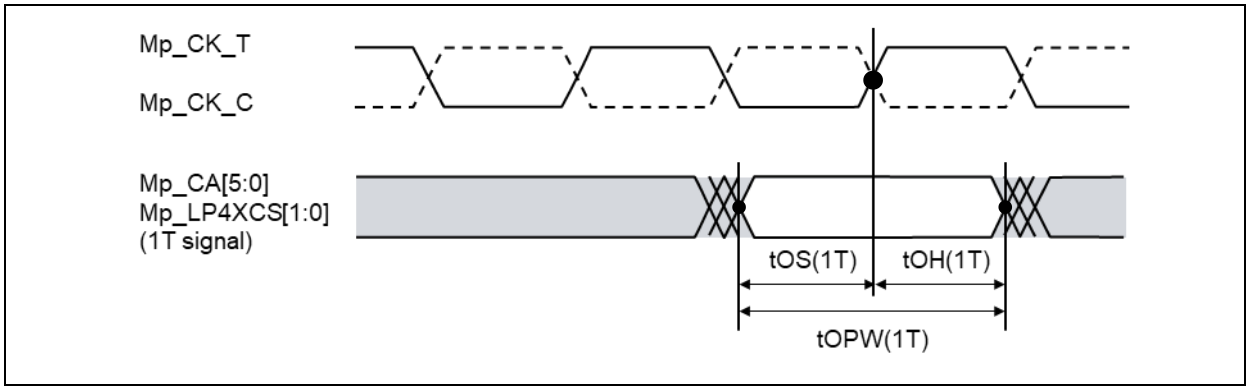



**Figure 200.56 Address and Command Output Timing relative to MCK Output(p=0,1)**

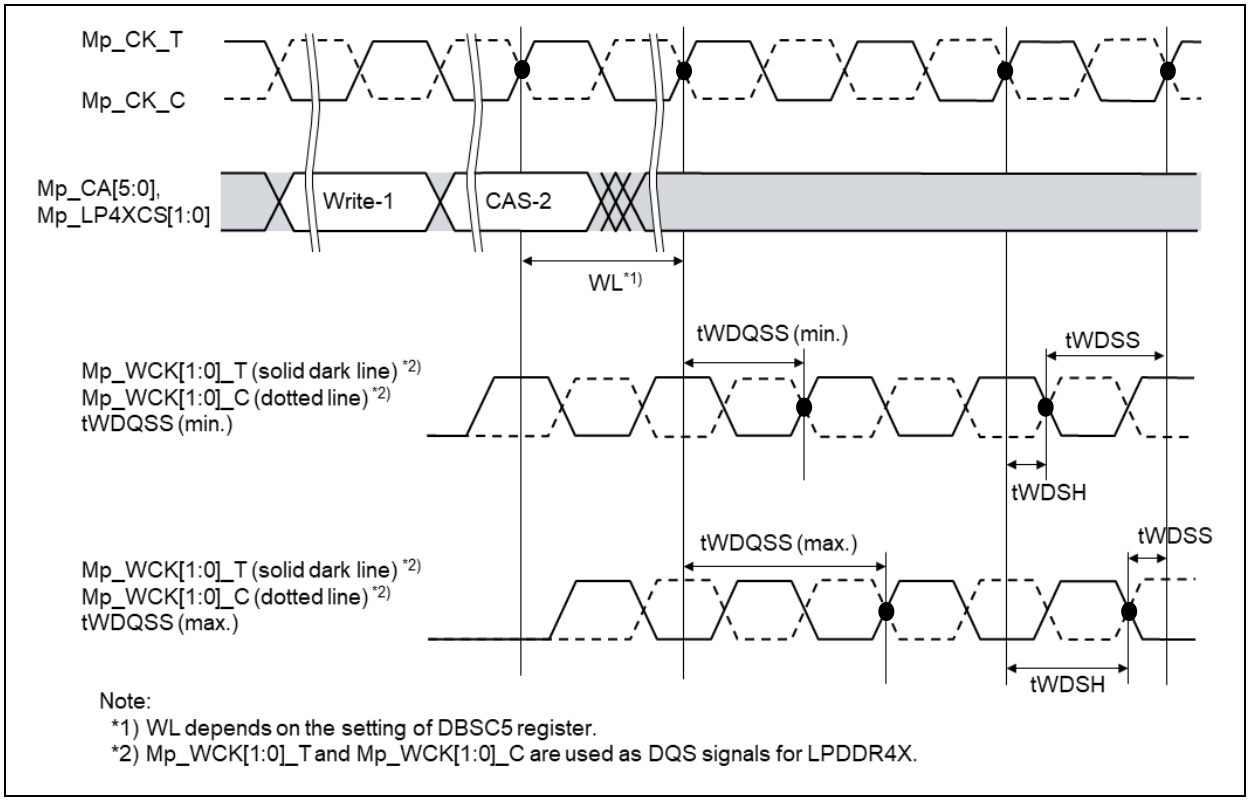



**Figure 200.57 MDQS Output Timing relative to MCK Output (write) (p=0,1)**

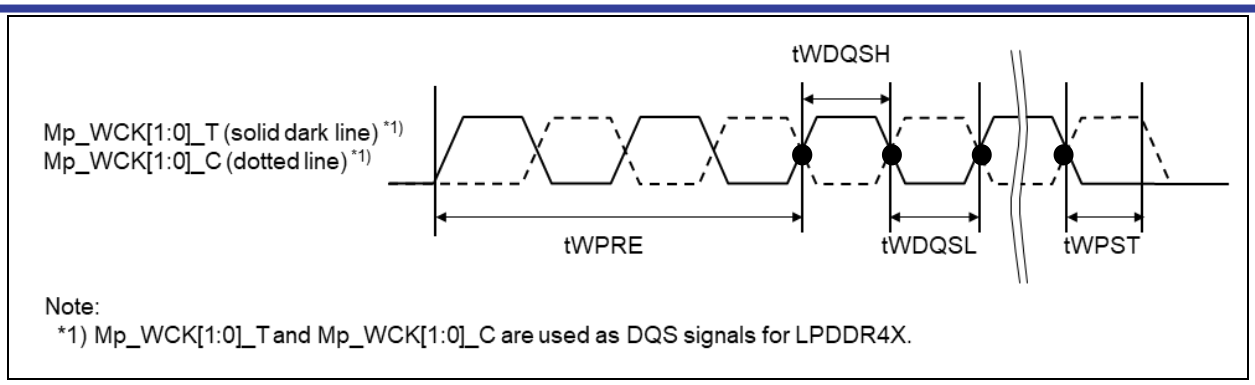



**Figure 200.58 MDQS Output Timing (write) (p=0,1)**

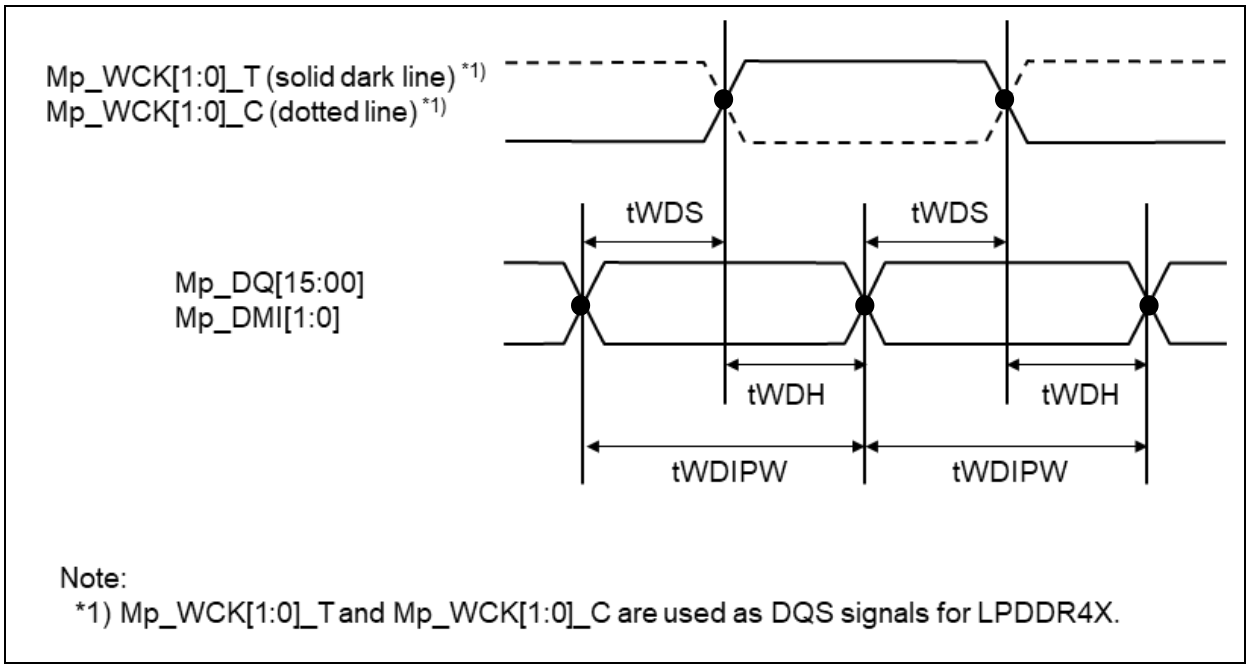



**Figure 200.59 MDQ/MDM Output Timing relative to MDQ (write) (p=0,1)**

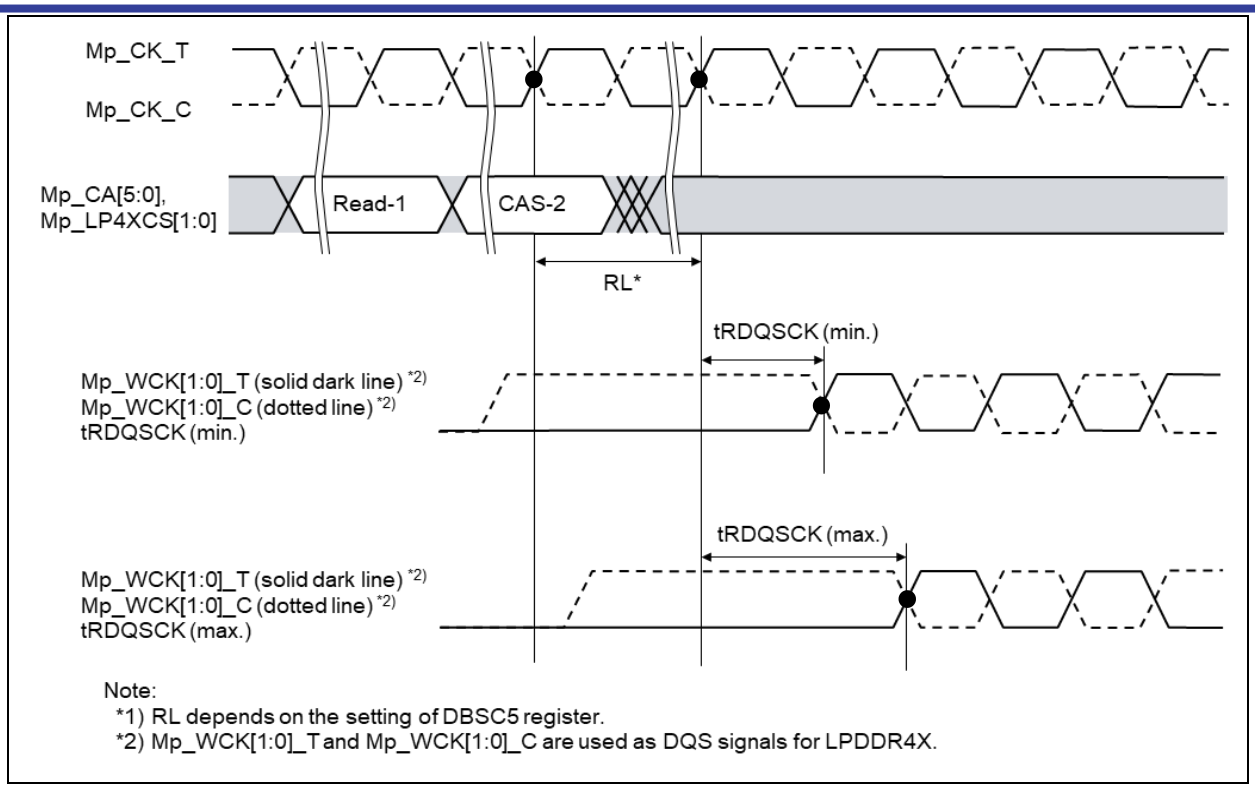



**Figure 200.60 MDQS Input Timing relative to MCK Output (read) (p=0,1)**

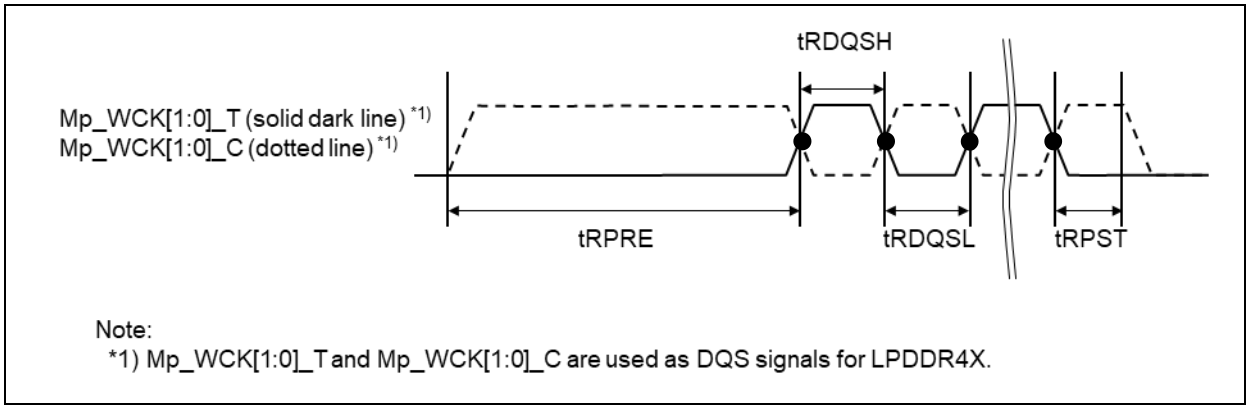



**Figure 200.61 MDQS Input Timing (read) (p=0,1)**

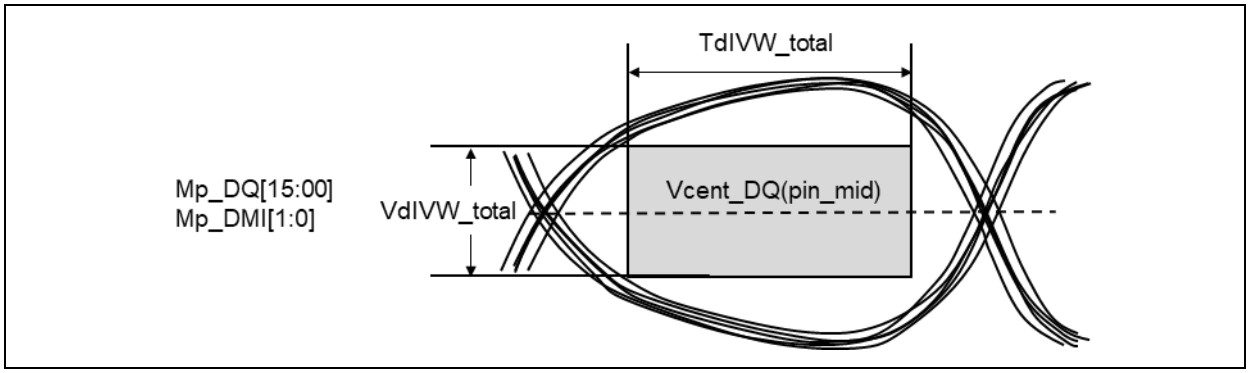



**Figure 200.62 Read Data Eye Mask (read) (p=0,1)**

### 200.7.18 DBSC5 Access Timing (LPDDR5)

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

**Table 200.72 DBSC5 Access Timing (LPDDR5)**

| Item | Symbol | Min. | Max. | Unit | Figure |
|---|---|---|---|---|---|
| MCK average clock period | tCK(avg) | 1.25 | 20 | ns | Figure 200.63 |
| MCK absolute high pulse width | tCH(abs) | 0.44 | 0.56 | tCK(avg) | |
| MCK absolute low pulse width | tCL(abs) | 0.44 | 0.56 | tCK(avg) | |
| Command and Address pulse width for each output | tCAOPW | 505 | — | ps | Figure 200.64 |
| Control pulse width for each output | tCSOPW | 1005 | — | ps | Figure 200.65 |
| MWCK to MCK phase offset | tWCK2CK | -0.42 | 0.42 | tWCK(avg) | Figure 200.66 |
| MWCK absolute high Write clock pulse width | tWCKH(abs) | 0.43 | 0.57 | tWCK(avg) | Figure 200.67 |
| MWCK absolute low Write clock pulse width | tWCKL(abs) | 0.43 | 0.57 | tWCK(avg) | |
| MWCK, MWCK# differential Preamble (Static) | tWCKPRE_Static | 3.8 | — | tCK(avg) | |
| MWCK, MWCK# differential Postamble | tWCKPST | 2.2 | — | tWCK(avg) | |
| MDQ and MDMI output pulse width for each output | tWDOPW | 122 | — | ps | Figure 200.68 |
| MRDQS, MRDQS# differential input high pulse width | tRDQSH(abs) | 0.42 | 0.58 | tWCK(avg) | Figure 200.69 |
| MRDQS, MRDQS# differential input low pulse width | tRDQSL(abs) | 0.42 | 0.58 | tWCK(avg) | |
| MRDQS, MRDQS# differential Preamble | tRPRE | 3.6 | — | tWCK(avg) | |
| MRDQS, MRDQS# differential Postamble | tRPST | 0.4 | — | tWCK(avg) | |
| Read Data Eye Mask (at VREF) | TdIVW1 | — | 0.35 | UI | Figure 200.70 |
| Read Data Eye Mask (at VDIVW) | TdIVW2 | — | 0.18 | UI | |
| Read Data Eye Mask | VdIVW | 100 | — | mV | |

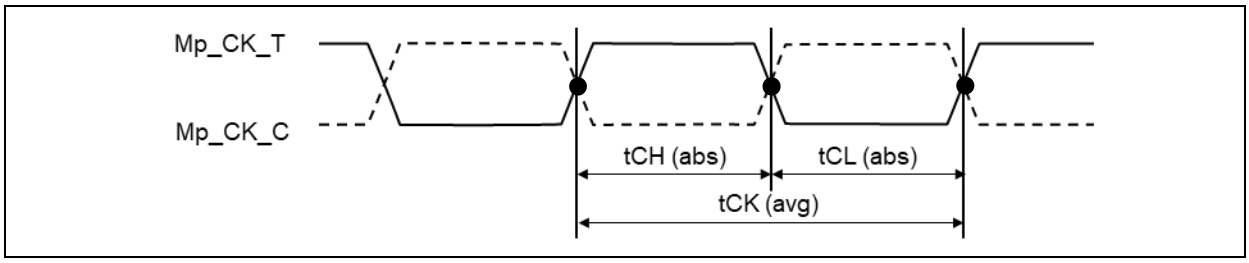



**Figure 200.63 MCK Clock Output(p=0,1)**

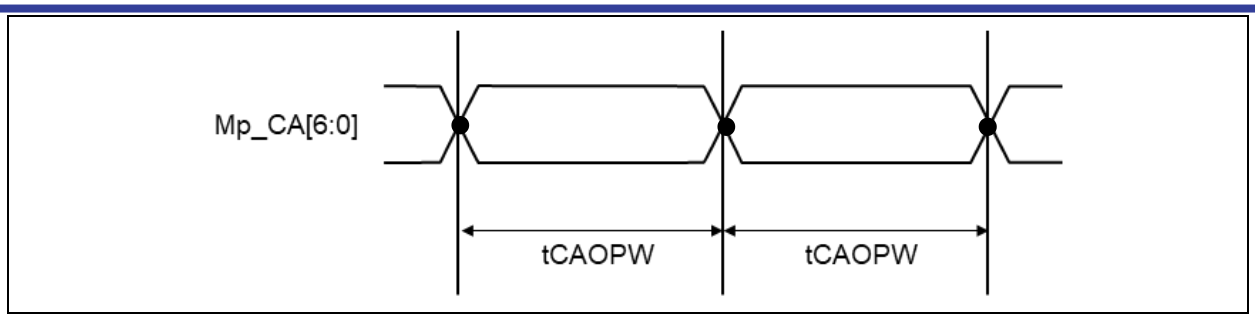



**Figure 200.64 Command and Address Pulse width for each output(p=0,1)**

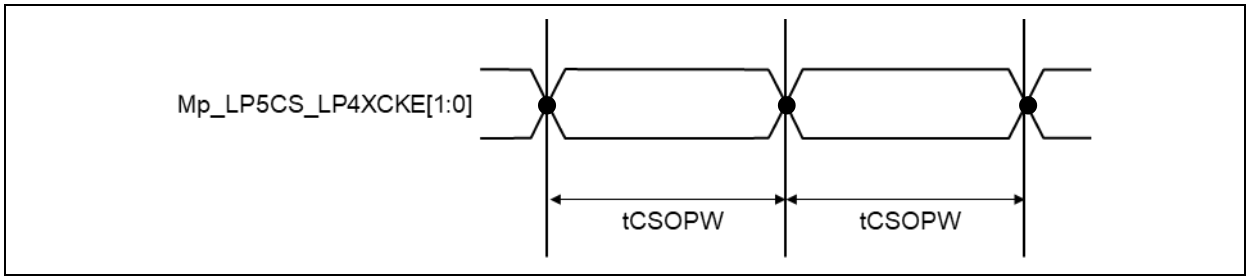



**Figure 200.65 Control Pulse width for each output(p=0,1)**

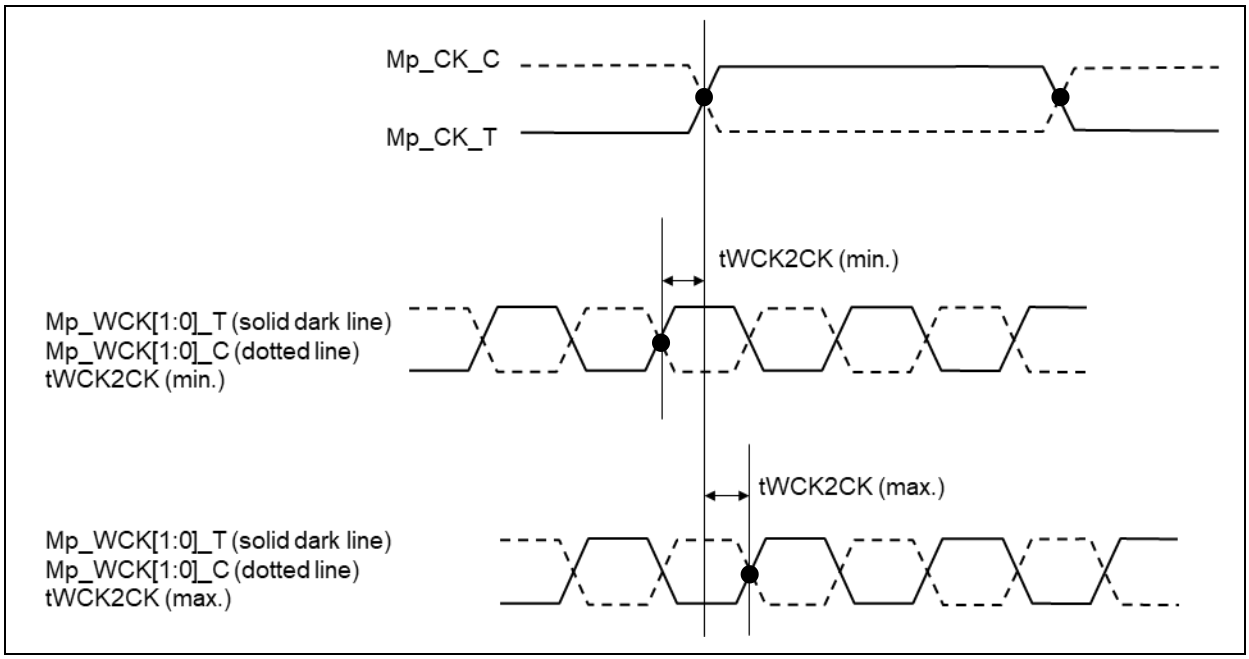



**Figure 200.66 MWCK to MCK phase offset(p=0,1)**

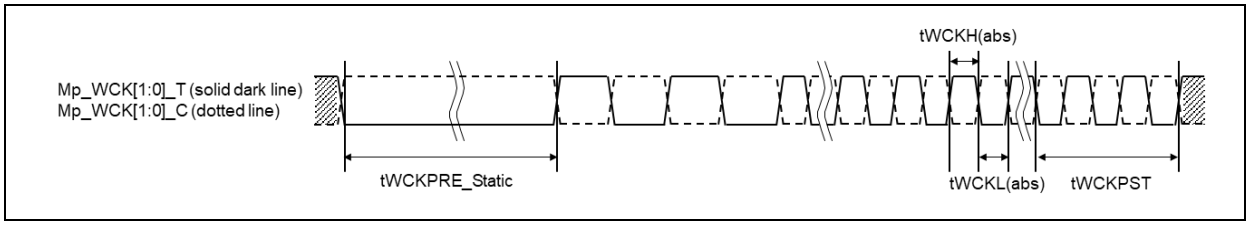



**Figure 200.67 MWCK Output Timing (write and read) (p=0,1)**

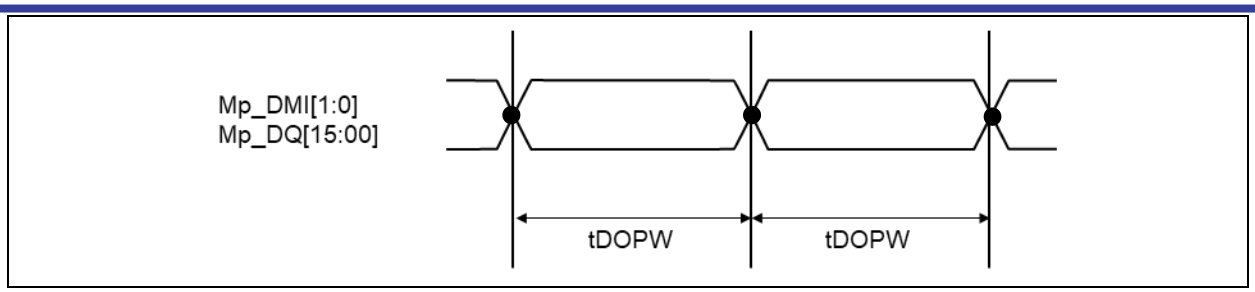



**Figure 200.68 MDQ and MDMI output pulse width for each output(p=0,1)**

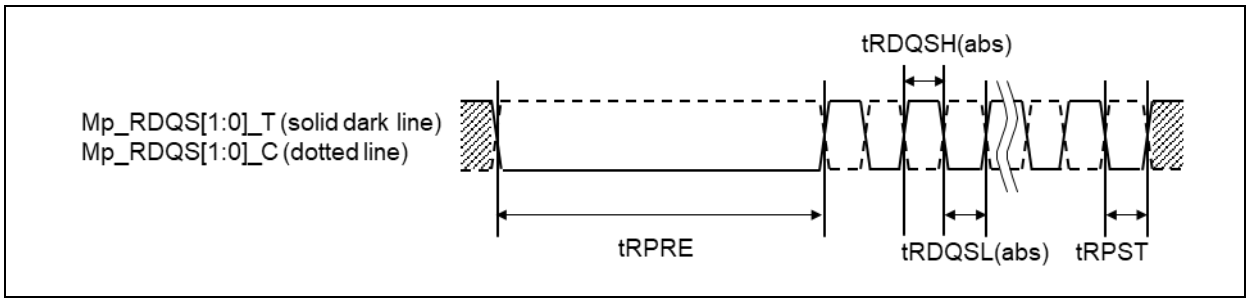



**Figure 200.69 MRDQS Input Timing (read) (p=0,1)**

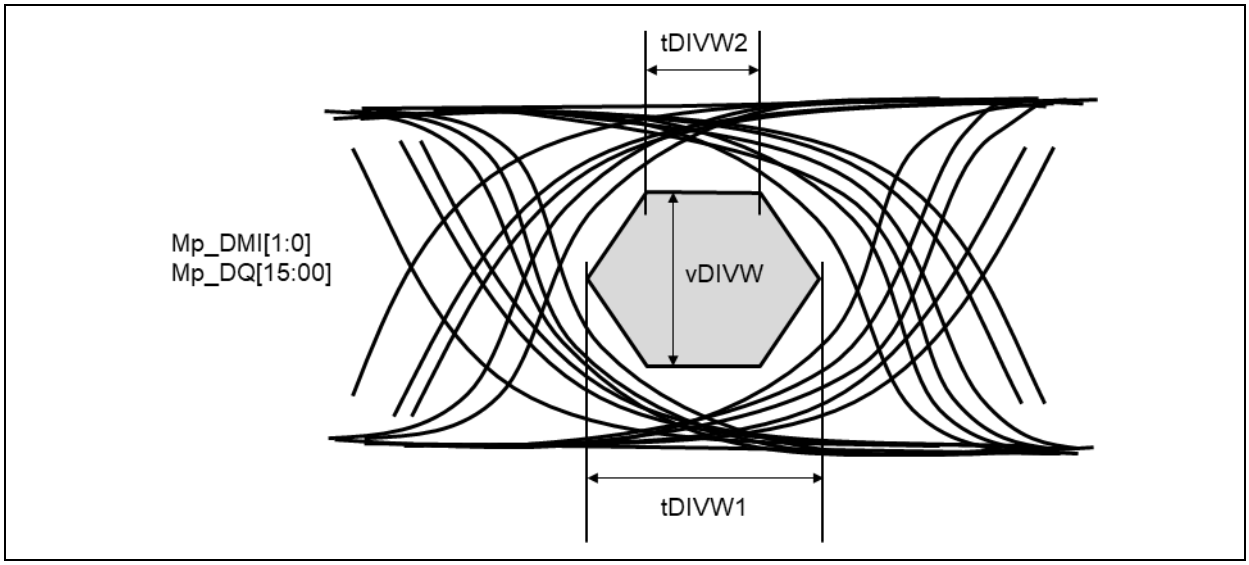



**Figure 200.70 Read Data Eye Mask(p=0,1)**

### 200.7.19 Debug and Trace

#### (1) JTAG IF Signal Timing for Arm CoreSight

**Table 200.73 JTAG IF Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**
[Main-JTAG] , [Sub-JTAG]

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| TCK Input clock cycle | tTCKcyc | 50* | — | — | ns | |
| TCK Input clock pulse width (high level) | tTCKH | 20 | — | — | ns | |
| TCK Input clock pulse width (low level) | tTCKL | 20 | — | — | ns | |
| TDI/TMS/SWDIO setup time | tTDIS | 15 | — | — | ns | |
| TDI/TMS/SWDIO hold time | tTDIH | 15 | — | — | ns | |
| SWDIO output delay time | tSWDO | 0 | — | 14 | ns | |
| TDO output delay time | tTDO | 0 | — | 14 | ns | |

Note: * The cycle is 500 ns (2 MHz) during boundary scan operation.

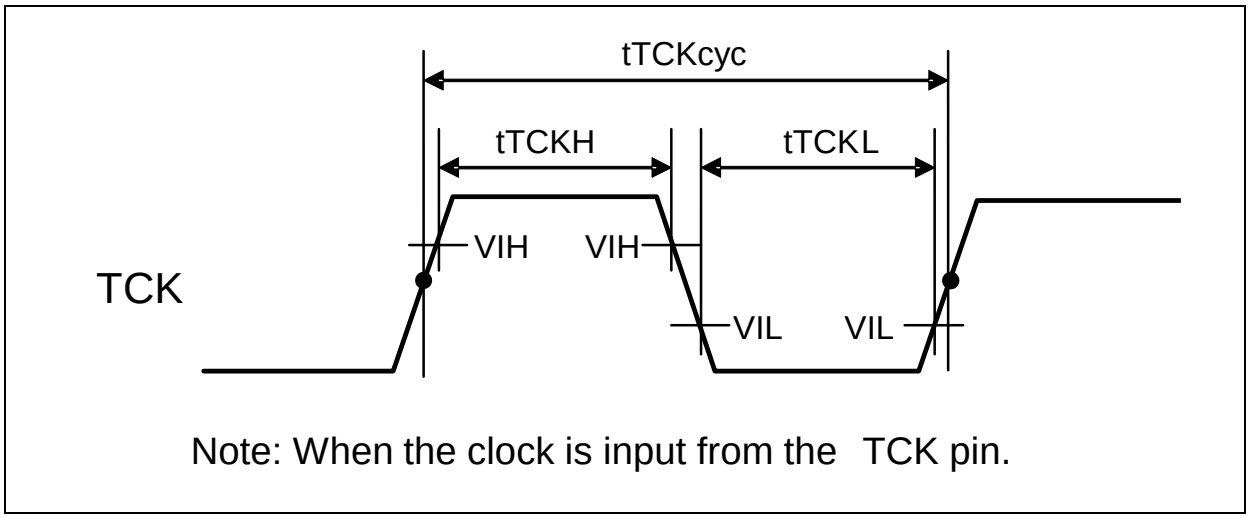



**Figure 200.71 TCK Input Timing**

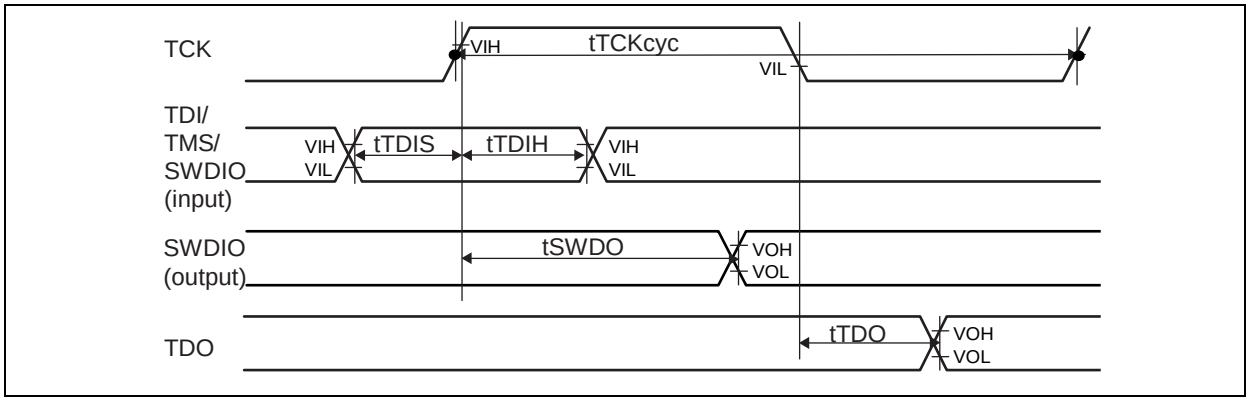



**Figure 200.72 Data Transfer Timing**

**(2) JTAG IF for ICUMXB**

**Table 200.74 JTAG IF for ICUMXB Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**
[Main-JTAG]

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| DCUTCK Input clock cycle | tTCKcyc | 50 | — | — | ns | |
| DCUTCK Input clock pulse width (high level) | tTCKH | 20 | — | — | ns | |
| DCUTCK Input clock pulse width (low level) | tTCKL | 20 | — | — | ns | |
| DCUTDI/DCUTMS setup time | tTDIS | 15 | — | — | ns | |
| DCUTDI/DCUTMS hold time | tTDIH | 15 | — | — | ns | |
| DCUTDO/DCURDY# output delay time | tTDO | 0 | — | 14 | ns | |

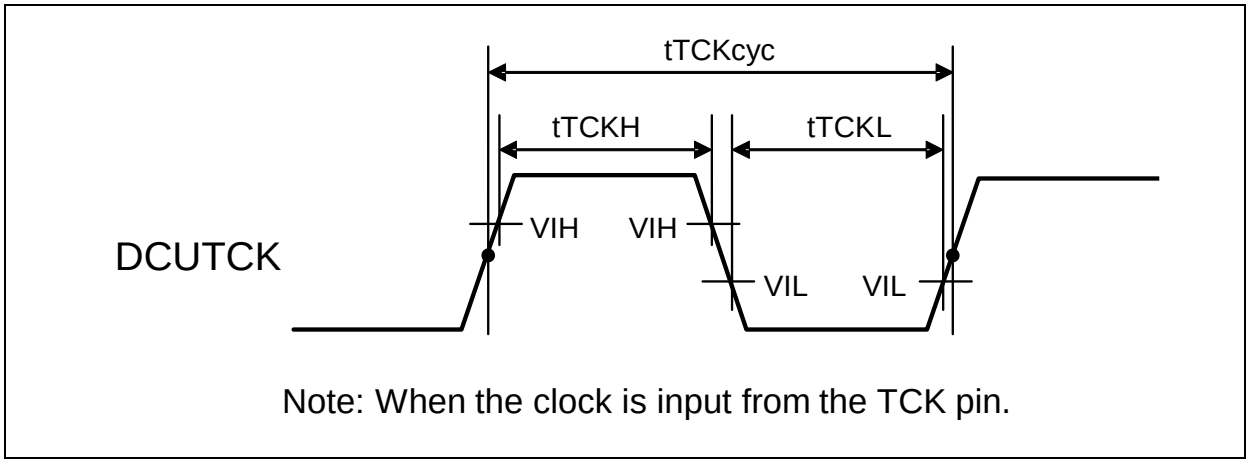



**Figure 200.73 DCUTCK Input Timing**

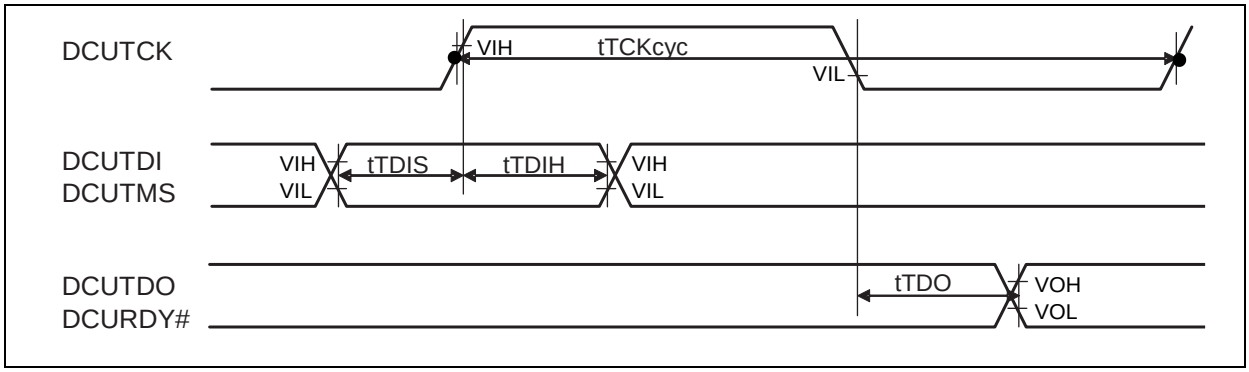



**Figure 200.74 Data Transfer Timing**

**(3) LPD IF for ICUMXB**

**Table 200.75 LPD IF for ICUMXB Signal Timing**

Conditions: **See 200.7.1 AC Characteristic Measurement Condition**

| Item | Symbol | Min. | Typ. | Max. | Unit | Figures |
|---|---|---|---|---|---|---|
| LPDCLK cycle time | tLPDCKW | 50 | — | — | ns | Figure 200.75 |
| LPDCLK high level width | tLPDCKWH | 20 | — | — | ns | |
| LPDCLK low level width | tLPDCKWL | 20 | — | — | ns | |
| LPDI setup time (vs LPDCLK↑) | tLPDS | 15 | — | — | ns | |
| LPDI hold time (vs LPDCLK↑) | tLPDH | 15 | — | — | ns | |
| LPDCLKOUT cycle time | tLPDCKOW | 50 | — | — | ns | |
| LPDCLKOUT high level width | tLPDCKOWH | 15 | — | — | ns | |
| LPDCLKOUT low level width | tLPDCKOWL | 15 | — | — | ns | |
| LPDO output delay (vs LPDCLKOUT↑) | tLPDOD | -8 | — | 10 | ns | |

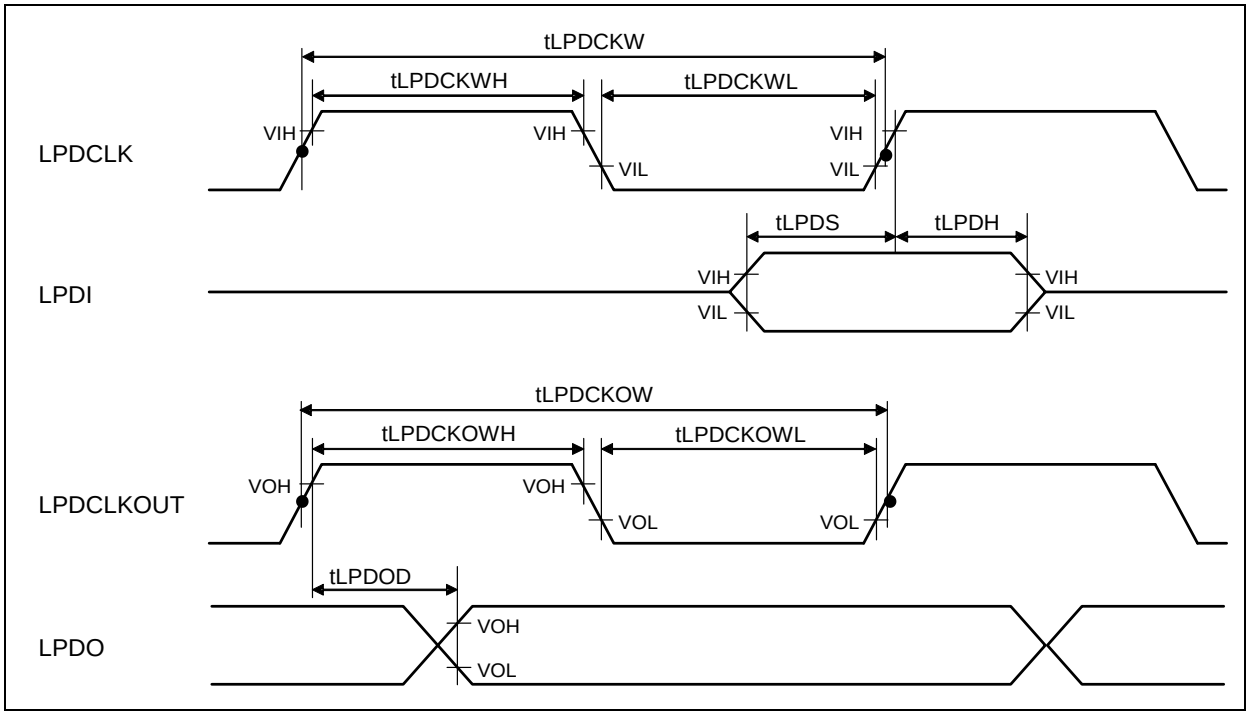



**Figure 200.75 LPD IF for ICUMXB Signal Timing**

# U10.1. E-FUSE

## U10.1.1 Overview

The E-FUSE is a type of non-volatile memory fabricated in standard CMOS logic process. This E-FUSE macro is used for hardware macro trimming, recording Chip information, disabling certain features, etc. Data read from E-Fuse macro is latched in Buffer and output to other modules.

## U10.1.2 Register description

The E-FUSE does not have a register.

## U10.1.3 Operation

See U10.1.1 Overview.
The FUSE output value is automatically read from the E-FUSE macro and latched into buffer when the chip is powered on, and it is output from that buffer to each module using E- FUSE signal.

## U10.1.4 Usage Notes

There is no attention of the usage.

## U10.1.5 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### U10.1.5.1 FUSE Monitoring Register

#### U10.1.5.1.1 Overview

E-FUSE bits of enable/disable switch can be monitored by FUSE monitor register in OTP in order to detect mismatching to expected FUSE bit. FUSE monitor register shall be read by software periodically to confirm each bit condition.

#### U10.1.5.1.2 Hardware Description

The FUSE status, for which the signal is the same as that sent to other modules, can be read from the register in the OTP module.

**Figure U10.1.1** shows the relations between the OTP, E-FUSE and other modules.

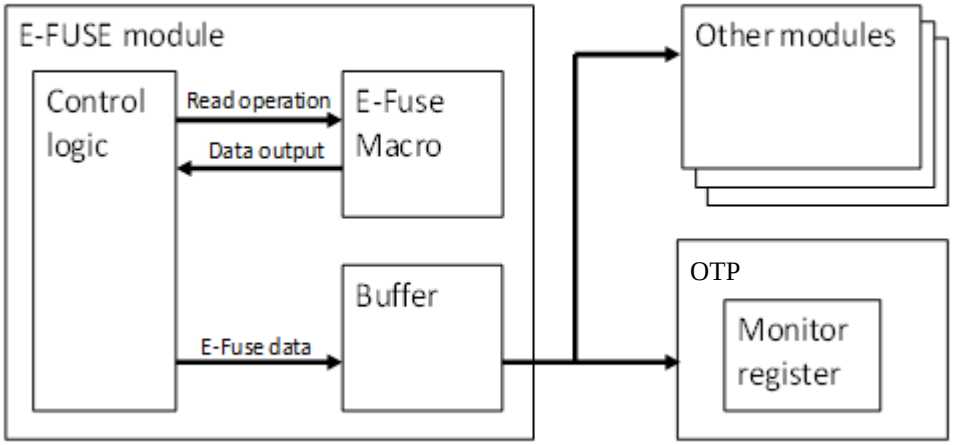



**Figure U10.1.1 Relations between OTP, E-FUSE and other modules**

**Table U10.1.1 Register Configuration**

| Register Name | Abbreviation | R/W | Base Address | Initial Value | Offset Address [lower 9bits] | Access Size |
|---|---|---|---|---|---|---|
| OTP_MEM Monitor Register 0 | OTPMONITOR0 | R | H'E61B_F000 | - | H'100 | 32 |
| OTP_MEM Monitor Register 3 | OTPMONITOR3 | R | H'E61B_F000 | - | H'10C | 32 |

### (1) OTP_MEM Monitor Register 0 (OTPMONITOR0)

Function: OTPMONITOR0 can read fuse bits for safety mechanisms.
OTP can read some fuse bits of some modules, because they cannot read fuse bits itself.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | otpmonitor0[9] | otpmonitor0[8] | otpmonitor0[7] | otpmonitor0[6] | - | otpmonitor0[4] | - | - | otpmonitor0[1] | otpmonitor0[0] |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

This register is functional safety use only. Keep initial value.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | - | - | R | Reserved. |
| 9 | otpmonitor0[9] | - | R | This bit is always read as 0. |
| 8 | otpmonitor0[8] | - | R | This bit is always read as 0. |
| 7 | otpmonitor0[7] | - | R | This bit is always read as 0. |
| 6 | otpmonitor0[6] | - | R | GPU disable<br>0: GPU function operation<br>1: GPU function stop |
| 5 | - | - | R | Reserved. |
| 4 | otpmonitor0[4] | - | R | This bit is always read as 0. |
| 3 to 2 | - | - | R | Reserved. |
| 1 | otpmonitor0[1] | - | R | VDSP1 disabling<br>0: VDSP1 function operation<br>1: VDSP1 function stop |
| 0 | otpmonitor0[0] | - | R | This bit is always read as 0. |

### (2) OTP_MEM Monitor Register 3 (OTPMONITOR3)

Function: OTPMONITOR3 can read fuse bits for safety mechanisms.
OTP can read some fuse bits of some modules, because they cannot read fuse bits itself.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - | - | - | otpmonitor3[5] | otpmonitor3[4] | otpmonitor3[3] | otpmonitor3[2] | otpmonitor3[1] | otpmonitor3[0] |
| Initial value: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

This register is functional safety use only. Keep initial value.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | - | - | R | Reserved. |
| 5 | otpmonitor3[5] | - | R | Control signal of external power supply IC. (copy_bit)<br>AVS0 and AVS1 pins are connected to PMIC.<br>Select the Pull function of AVS1 LSI pin.<br>0 : Pull-down<br>1 : Pull-up |
| 4 | otpmonitor3[4] | - | R | Control signal of external power supply IC.<br>AVS0 and AVS1 pins are connected to PMIC.<br>Select the Pull function of AVS1 LSI pin.<br>0 : Pull-down<br>1 : Pull-up |
| 3 | otpmonitor3[3] | - | R | Control signal of external power supply IC. (copy_bit)<br>AVS0 and AVS1 pins are connected to PMIC.<br>Select the Pull function of AVS0 LSI pin.<br>0 : Pull-down<br>1 : Pull-up |
| 2 | otpmonitor3[2] | - | R | Control signal of external power supply IC.<br>AVS0 and AVS1 pins are connected to PMIC.<br>Select the Pull function of AVS0 LSI pin.<br>0 : Pull-down<br>1 : Pull-up |
| 1 | otpmonitor3[1] | - | R | This bit is always read as 0. |
| 0 | otpmonitor3[0] | - | R | This bit is always read as 0. |

### U10.1.5.2 Hardware Redundancy for E-FUSE

#### U10.1.5.2.1 Overview

Dual circuit is applied to E-FUSE in order to detect mismatching expected FUSE bit of E-FUSE.

#### U10.1.5.2.2 Hardware Description

Figure U10.1.2 shows a block diagram of this safety mechanisms.

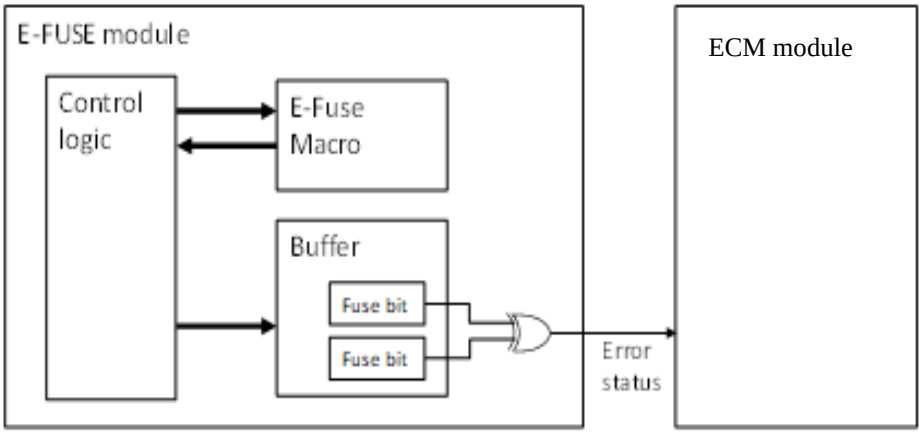



**Figure U10.1.2 Block Diagram of Hardware Redundancy for E-FUSE**

There are plural error notification signals to ECM (shown as Error status) and their implementations for the products are different from each other.

# U10.2. OTP_MEM

## U10.2.1 Overview

The OTP_MEM is a type of non-volatile memory fabricated in standard CMOS logic process. This OTP_MEM macro is used for hardware macro trimming, recording Chip information, disabling certain features, etc. Data read from OTP-MEM is latched in Buffer and output to other modules.

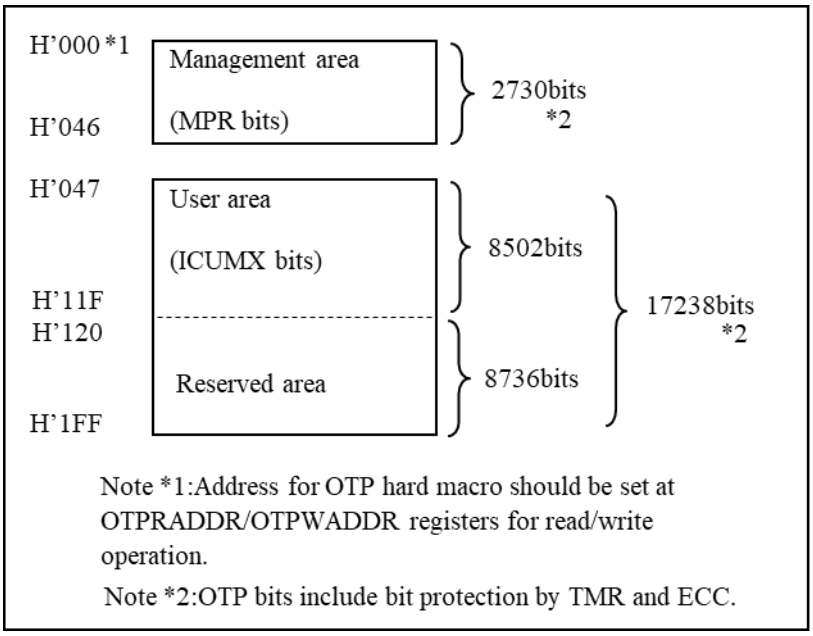



**Figure U10.2.1 OTP bit map area and size**

### U10.2.1.1 Feature

OTP(One Time Programmable device) hard macro uses antifuse based technology. Customers writes/reads OTP at run time.

- OTP hard macro : 512address x 39bits x 1macro
- Read OTP after POR(Power On Reset by use of external reset pin (PRESET#)).
- Write OTP by CPU(Controlled by APB master side for security)
- Read OTP by CPU(Controlled by APB master side for security)
- Read/Write Protection by MPR (Protect one address with MPR one bit))
- ECC for OTP bits (FuSa requirement (ECC shall be checked by POST))
- Redundancy bits for OTP bits (for safety and security)
- Monitoring registers for eFuse and OTP bits.
- DCLS (FuSa requirement)
- ID authentication function from eFUSE to OTP
- Register Access Protection by Region ID(Two 4KB areas, Secure and Non-Secure)

  (OTP control register for Read/Write OTP and Test function is a Secure area.

  FuSa, eFUSE monitor and OTP monitor register is a Non-Secure area.)
- Control registers for eFuse and LTM.
- Each product ID can be referenced from OTPMONITOR17

#### U10.2.1.2 Register Configuration

Refer to R-CarV4M_UM_U10_2_OTP_MEM_G_r0p81.xlsx

## U10.2.2 Register Description

The base-address of OTP is H'E61B_E000 and H'E61B_F000.

OTP has register access protection by Region ID. The register can only access to permitted protection base-address.

There are 1 protection base-address (H'E61B_E000).

Registers related to OTP control function.

- Accessible by ICUMX for security:
  - The base-address is H'E61B_E000 registers.

Registers related to FuSa function, eFUSE monitor, OTP monitor.

- All error control registers are writable.
  - The base-address is H'E61B_F000 registers.

Do not write to addresses other than those listed below, otherwise normal operation cannot be guaranteed. Values read from other addresses are undefined.

Access size are all 32bits.

Legend for Register Configuration

Initial value : Register value after a reset

– : Undefined value

R/W : Readable/writable. The written value can be read.

R : Read-only.

W : Write-only. The read value is undefined.

Refer to R-CarV4M_UM_U10_2_OTP_MEM_G_r0p81.xlsx

## U10.2.3 Operation

### U10.2.3.1 Initial read OTP hard macro

The OTP output value is automatically read from the OTP hard macro and latched into buffer when the chip is powered on, and it is output from that buffer to each module using OTP signal.

### U10.2.3.2 Program OTP hard macro by registers

1 Write address to the register (OTPWADDR).

2 Write data to the register (OTPWDATA).

3 Write keycode 0xA7141222 to the register (OTPKEYCODE).

4 Write 1 to the write start bit (OTPWCONF[0]).

5 Wait for the write end bit (OTPWSTATUS[1]) to be 1.

(Wait from 4195 cycles to maximum 13204 cycles of CL16Mϕ.

It increases by 231 cycles according to the number of bits to be written from 4195 cycles.)

6 Write 0 to the register (OTPKEYCODE).

7 Check the write status register (OTPWSTATUS) is 0x00000002.

8 Check the error status register (OTPERRSTATUS) is 0x00000000.

### U10.2.3.3 Read OTP hard macro by registers

1 Write address to the register (OTPRADDR).

2 Write 1 to the read start bit (OTPRCONF[0]).

3 Wait for the read end bit (OTPRSTATUS[1]) to be 1. (Wait 1361 cycles of CL16Mϕ).

4 Check the read status register (OTPRSTATUS) is 0x00000002.

5 Check the error status register (OTPERRSTATUS) is 0x00000000.

6 Check the read data register (OTPRDATA).

### U10.2.3.4 Initial/PGM Margin Read OTP hard macro by registers

1. Write address to the register (OTPRADDR).
2. Write 0x00000108 to the register (OTPTESTCONF).

   Set 1 to the margin read mode bit (OTPTESTCONF [3]).

   Set 1 to the margin read mode select bit (OTPTESTCONF[11:8]). *1
3. Write 1 to the read start bit (OTPRCONF[0]).
4. Wait for the read end bit (OTPRSTATUS[1]) to be 1. (Wait 1361 cycles of CL16Mϕ).
5. Check the read status register (OTPRSTATUS) is 0x00000002.
6. Check the error status register (OTPERRSTATUS) is 0x00000000.
7. Check the read data register (OTPRDATA).
8. Write 0x00000000 to the register (OTPTESTCONF).

Note *1: If you need other margin read mode, change register value of OTPTESTCONF [11:8]. Refer to register description. OTP has 5 kinds of margin read mode. (Initial Margin Read Mode (for LT*2 and RT*3), HT*4 initial Margin Read Mode (for HT), PGM*5 Margin Read Mode (for RT), HT PGM Margin Read Mode (for HT) and LT PGM Margin Read Mode (for LT))

Note *2: LT is an abbreviation for Low Temperature. LT is a test mode for CP sorting at low temp (<0degC).

Note *3: RT is an abbreviation for Room Temperature.

Note *4: HT is an abbreviation for High Temperature. The HT is a test mode for CP sorting at high temp (>85degC).

Note *5: PGM is the abbreviation for program.

### U10.2.3.5 Read OTP hard macro by test function registers

1. Write address to the register (OTPTESTRADDR).
2. Write 0x00000002 to the register (OTPTESTCONF).

   Set 1 to the test mode bit (OTPTESTCONF [1]).
3. Write 1 to the read start bit (OTPRCONF[0]).
4. Wait for the read end bit (OTPRSTATUS[1]) to be 1. (Wait 1361 cycles of CL16Mϕ).
5. Check the read status register (OTPRSTATUS) is 0x00000002.
6. Check the error status register (OTPERRSTATUS) is 0x00000000.
7. Check the test status register (OTPTESTSTATUS) is 0x00000000.
8. Check the read data register (OTPTESTRUDATA[6:0], OTPTESTRLDATA[31:0]).
9. Write 0x00000000 to the register (OTPTESTCONF).

### U10.2.3.6 Initial/PGM Margin Read OTP hard macro by test function registers

1. Write address to the register (OTPTESTRADDR).
2. Write 0x0000010A to the register (OTPTESTCONF).

   Set 1 to the test mode bit (OTPTESTCONF [1]).

   Set 1 to the margin read mode bit (OTPTESTCONF [3]).

   Set 1 to the margin read mode select bit (OTPTESTCONF[11:8]). *1
3. Write 1 to the read start bit (OTPRCONF[0]).
4. Wait for the read end bit (OTPRSTATUS[1]) to be 1. (Wait 1361 cycles of CL16Mϕ).
5. Check the read status register (OTPRSTATUS) is 0x00000002.
6. Check the error status register (OTPERRSTATUS) is 0x00000000.
7. Check the test status register (OTPTESTSTATUS) is 0x00000000.
8. Check the read data register (OTPTESTRUDATA[6:0], OTPTESTRLDATA[31:0]).
9. Write 0x00000000 to the register (OTPTESTCONF).

Note *1: If you need other margin read mode, change register value of OTPTESTCONF [11:8]. Refer to register description. OTP has 5 kinds of margin read mode. (Initial Margin Read Mode (for LT*2 and RT*3), HT*4 initial Margin Read Mode (for HT), PGM*5 Margin Read Mode (for RT), HT PGM Margin Read Mode (for HT) and LT PGM Margin Read Mode (for LT))

Note *2: LT is an abbreviation for Low Temperature. LT is a test mode for CP sorting at low temp (<0degC).

Note *3: RT is an abbreviation for Room Temperature.

Note *4: HT is an abbreviation for High Temperature. The HT is a test mode for CP sorting at high temp (>85degC).

Note *5: PGM is the abbreviation for program.

## U10.2.4 User Programming Guide

### U10.2.4.1 For customer

This guide provides guidelines for OTP programming. The OTP has the bits including in user area and management (write/read protection) area. Refer to the attached "U10_2_R-CarV4M_OTP_Bit_Map_r0p70.xlsx". Figure U10.2.2 User Programming Guide for customer show a flowchart of the User Programming Guide for customer.
Refer to Table 200.2 Power Supply and set VDD and VDD18_OTP within specification.
Refer to Table 200.1 Absolute Maximum Ratings and set operating temperature within specification.
Please refer to the security manual for details of OTP bit information.

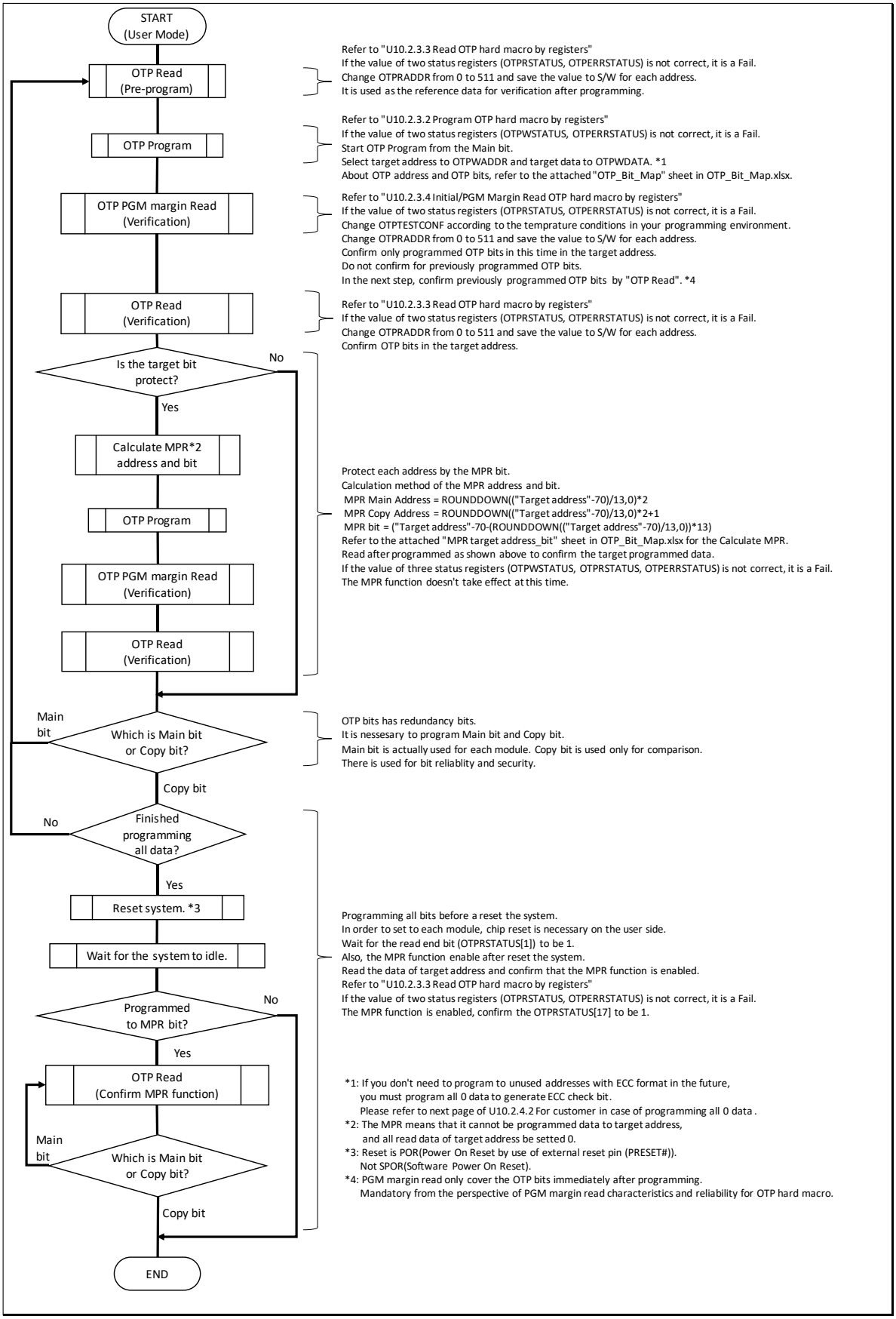



**Figure U10.2.2 User Programming Guide for customer**

### U10.2.4.2 For customer in case of programming all 0 data

This guide provides guidelines for OTP in case of programming all 0 data. If you don't need to program to unused addresses with ECC format in the future, you must program all 0 data to generate ECC check bit. The OTP has the bits including in user area and management (write/read protection) area.

Refer to the attached "U10_2_R-CarV4M_OTP_Bit_Map_r0p70.xlsx". Figure U10.2.3 User Programming Guide for customer in case of programming all 0 data show a flowchart of the User Programming Guide for customer in case of programming all 0 data.

Refer to Table 200.2 Power Supply and set VDD and VDD18_OTP within specification.

Refer to Table 200.1 Absolute Maximum Ratings and set operating temperature within specification.

Please refer to the security manual for details of OTP bit information.

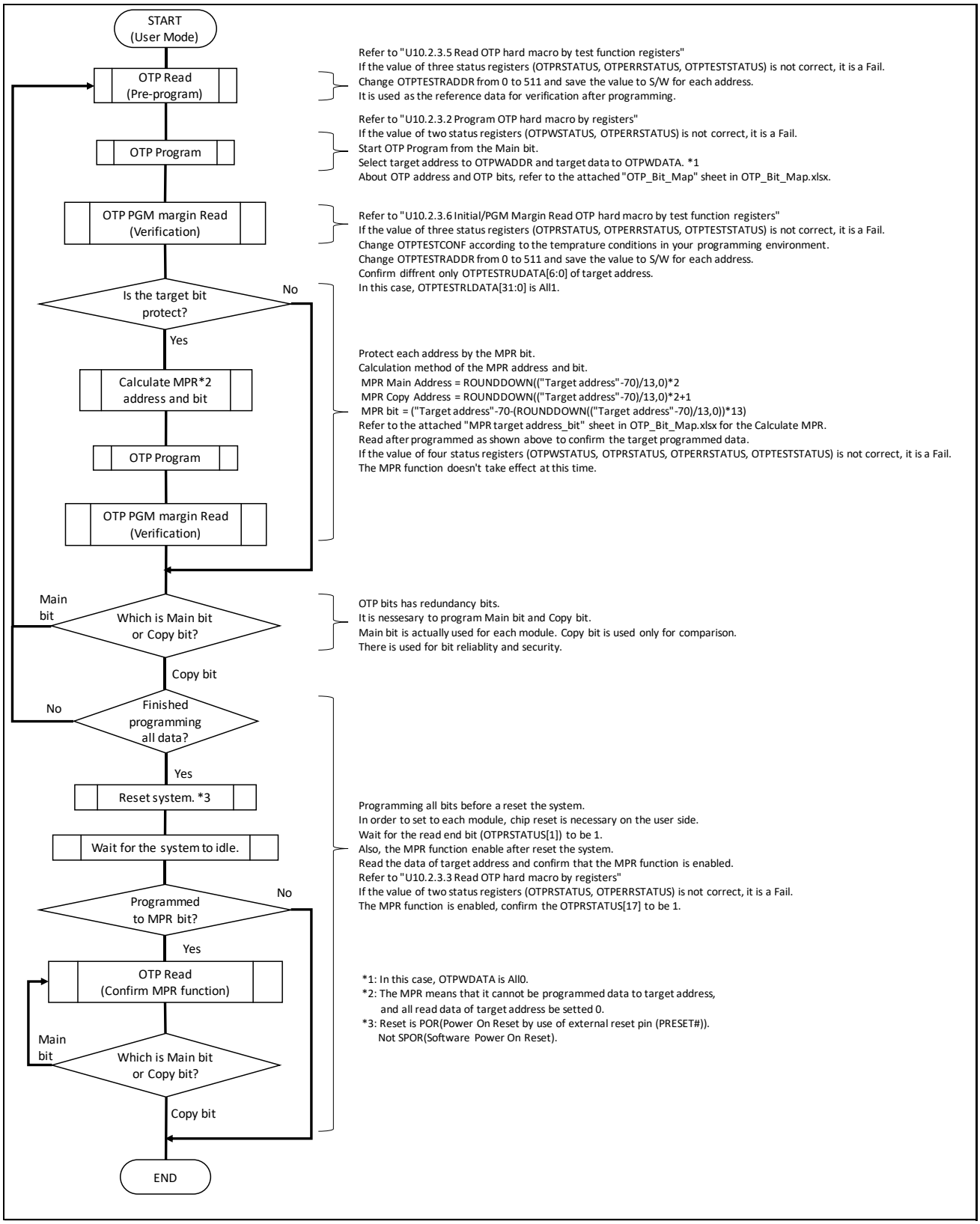



**Figure U10.2.3 User Programming Guide for customer in case of programming all 0 data**

## U10.2.5 Usage Notes

### U10.2.5.1 OTP Bit Map

Refer to the attached "U10_2_R-CarV4M_OTP_Bit_Map_r0p70.xlsx".

Note:

- The TMR is the majority logic format. (Write bits one by one)

| | [38:26] | [25:13] | [12:0] |
|---|---|---|---|
| address 2n | copy of main data | copy of main data | Main data |
| address 2n+1 | copy of copy data | copy of copy data | Copy data |

n < 255

OTP uses output data of majority logics of [38:26], [25:13], and [12:0].

OTP compares output data of majority logics of address 2n and address 2n+1.

- The ECC is the ECC format. (Write 32bit at only one time)

| | [38:32] | [31:0] |
|---|---|---|
| address 2n | ECC check bit (Generated by address 9bit + data 32bit) | Main data |
| address 2n+1 | ECC check bit (Generated by address 9bit + data 32bit) | Copy data |

n < 255

OTP uses data corrected by ECC check bits.

OTP compares corrected data of address 2n and address 2n+1.

- The MPR is used to mask the bit of the target address for security.

  If MPR bit 0 to 1, it will not be able to write to the target address. And it all 0 will be read.

- The OTP bit is like an eFUSE bit. It is connected to each module.

### U10.2.5.2 Cautionary point for programming operation

- Shall not be programming operation at frequencies lower than the CL16Mφ specifications (OTP clock).
- Shall not be turning off the power during programming.
- The ECC format cannot be written twice.
- The TMR format only the bit to be written is set to 1.
- Shall be programming the same data for the Main/Copy bits and then resets the system.
- If there are bits to protect, the MPR bit of the target address is set to 1.

- The MPR bit protection is enabled by system reset after programming.

### U10.2.6 Safety Mechanisms

- DCLS
- OTP Data format (ECC or Majority logics)
- ECC check bits(7bits) are generated by 41bit {address 9bit, data 32bit}
- Redundancy bits
  - address 2n : data is connected to other modules
  - address 2n+1 : Redundancy bits (data shall be the same to data of address 2n)
  - n < 255
- Error injection
- OTP Monitor Register
- eFUSE Monitor Register

# U11. ROM

## U11.1 Overview

This module is treated as safety-related function in case of the products supporting functional safety. For details, refer to the functional safety documents.

The boot ROM is used for booting processor codes. R-Car V4M supports Cortex-R52 and ICUMX booting.

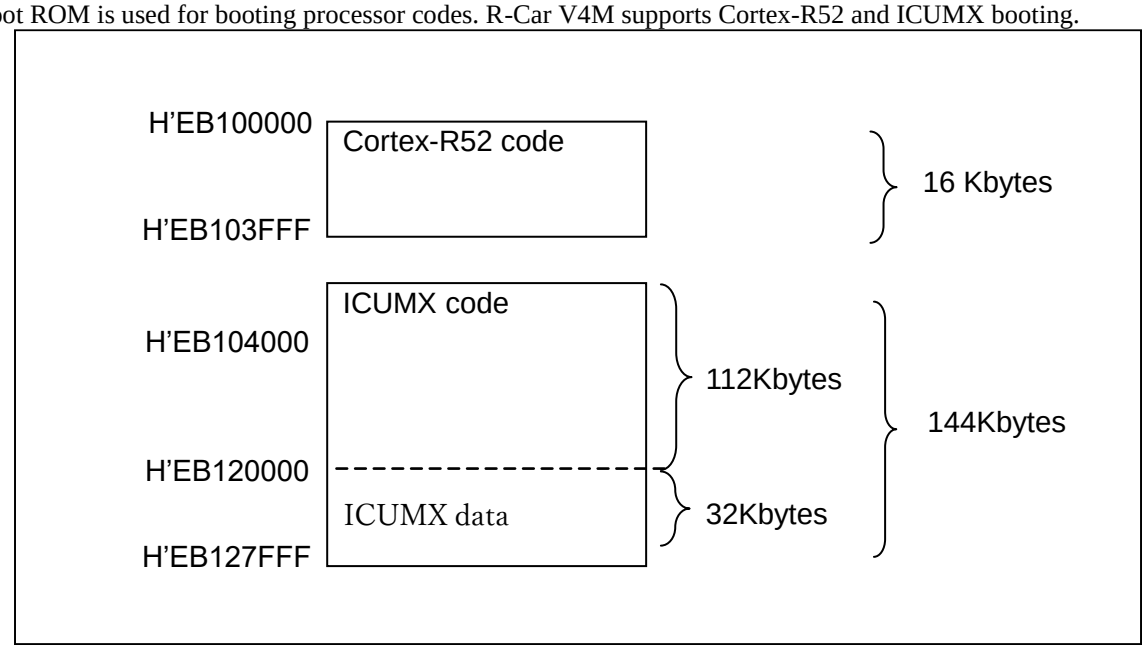



Figure U11.1 Boot ROM memory map

### U11.1.1 Features

•EDC protection for ROM code loading.

• Capacity 160KB

• ROM code repair function

•64-b AXI interface

•Return AXI error response if EDC error occurs.

•ICUMX code can only read by fetch access from ICUMX.
(The ICUMX code cannot be read from other master (Cortex-R52, Debugger, etc.).)

•Error notification if read to the ICUMX code area other than instruction fetch access from ICUMX occurs.

### U11.1.2 Register configuration

| Base Address Name | Base Address |
|---|---|
| ROM | H'EB12 0000 |

| Module Name | Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|---|
| ROM | Access error status 1 | ACCERRSTS1 | RC | <ROM>+H'D000 | H'0000 0000 | 32 |
| ROM | Access error status 2 | ACCERRSTS2 | RC | <ROM>+H'D004 | H'0000 0000 | 32 |
| ROM | Single bit error2 injection | INJ_ERR_S2 | R | <ROM>+H'E000 | H'0000 0000 | 32 |
| ROM | Multi bits error2 injection | INJ_ERR_M2 | R | <ROM>+H'E004 | H'0000 0000 | 32 |
| ROM | Single bit error injection | INJ_ERR_S | R | <ROM>+H'E008 | H'0000 0000 | 32 |
| ROM | Multi bits error injection | INJ_ERR_M | R | <ROM>+H'E00C | H'0000 0000 | 32 |
| ROM | Error injection enable control 0 | INJ_ERR_CTR0 | R | <ROM>+H'F000 | H'0000 0000 | 32 |
| ROM | Error injection enable control 1 | INJ_ERR_CTR1 | R | <ROM>+H'F800 | H'0000 0000 | 32 |
| ROM | Error injection enable control 2 | INJ_ERR_CTR2 | R | <ROM>+H'F100 | H'0000 0000 | 32 |
| ROM | Error injection enable control 3 | INJ_ERR_CTR3 | R | <ROM>+H'F300 | H'0000 0000 | 32 |

## U11.2 Register description

### U11.2.1 Access error status 1(ACCERRSTS1)

This register indicates the status of access error to ICUMX code area from ICUP code access I/F in ICUMX.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | err |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | H'0 | R | Reserved<br>The read value of these bits are always 0. |
| 0 | err | H'0 | RC | 0 : No access error from ICUP code access I/F in ICUMX.<br>1 : Access error occurs from ICUP code access I/F in ICUMX.<br>This bit can be cleared by read access. |

### U11.2.2 Access error status 2(ACCERRSTS2)

This register indicates the status of access error to ICUMX code area fromother than ICUP code access I/F.

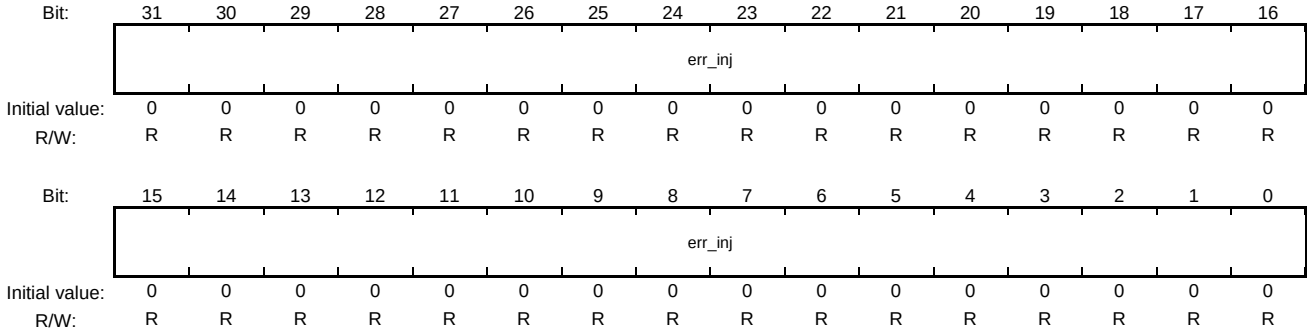
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | err |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | RC |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | H'0 | R | Reserved<br>The read value of these bits are always 0. |
| 0 | err | H'0 | RC | 0 : No access error from other than ICUP code access I/F in ICUMX.<br>1 : Access error occurs from other than ICUP code access I/F in ICUMX.<br>This bit can be cleared by read access. |

### U11.2.3 Single bit error injection(INJ_ERR_S)

Access read to this register will trigger error injection for EDC single error




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | err_inj | H'0 | R | trigger of error injection for EDC single bit error.<br>The read value of these bits are always 0. |

### U11.2.4 Multi bits error injection(INJ_ERR_M)

Access read to this register will trigger error injection for EDC multiple error

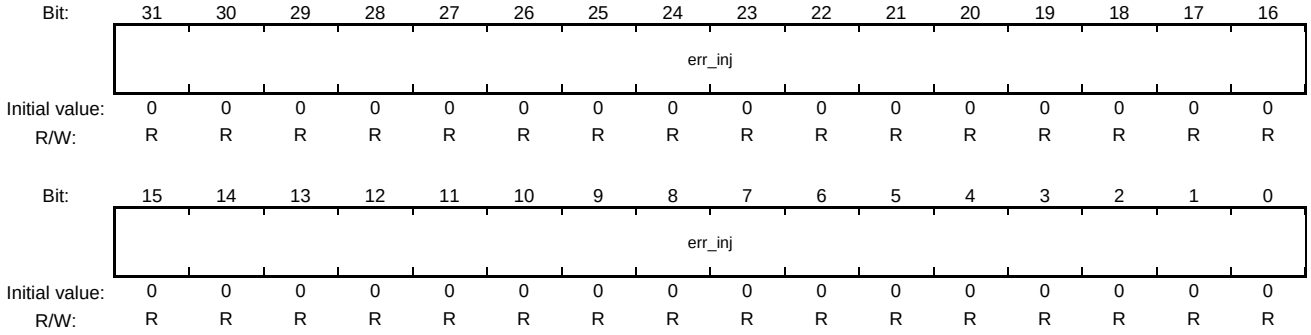
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | err_inj | H'0 | R | trigger of error injection for EDC single and multi bit error.<br>The read value of these bits are always 0. |

### U11.2.5 Single bit error2 injection(INJ_ERR_S2)

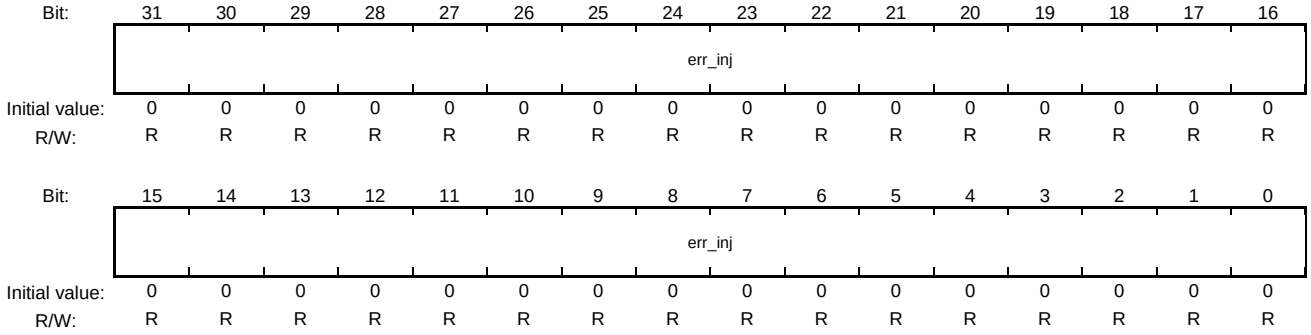
Access read to this register will trigger error injection for EDC single bit error2




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | err_inj | H'0 | R | trigger of error injection for EDC single bit error2.<br>The read value of these bits are always 0. |

### U11.2.6 Multi bits error2 injection(INJ_ERR_M2)

Access read to this register will trigger error injection for EDC multiple bit error2




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | err_inj | H'0 | R | trigger of error injection for EDC single and multi bit error2.<br>The read value of these bits are always 0. |

### U11.2.7 Error injection enable control 0(INJ_ERR_CTR0)

These registers control the enable of error injection.
See U11.4.1 for error injection procedure.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj_ctr | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj_ctr | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | err_inj_ctr | H'0 | R | error injection enable control.<br>The read value of these bits are always 0. |

### U11.2.8 Error injection enable control 1(INJ_ERR_CTR1)

These registers control the enable of error injection.
See U11.4.1 for error injection procedure.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj_ctr | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj_ctr | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | err_inj_ctr | H'0 | R | error injection enable control.<br>The read value of these bits are always 0. |

### U11.2.9 Error injection enable control 2(INJ_ERR_CTR2)

These registers control the enable of error injection.
See U11.4.1 for error injection procedure.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj_ctr | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj_ctr | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | err_inj_ctr | H'0 | R | error injection enable control.<br>The read value of these bits are always 0. |

### U11.2.10 Error injection enable control 3(INJ_ERR_CTR3)

These registers control the enable of error injection.
See U11.4.1 for error injection procedure.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj_ctr | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | err_inj_ctr | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | err_inj_ctr | H'0 | R | error injection enable control.<br>The read value of these bits are always 0. |

## U11.3 Operation

No description on this section.

## U11.4 Usage Notes

### U11.4.1 EDC Error Injection

To use this function, it must be access by the following procedure.

1. Read Access to INJ_ERR_CTR0
2. Read Access to INJ_ERR_CTR1
3. Read Access to INJ_ERR_CTR2
4. Read Access to INJ_ERR_CTR3
5. Read Access to INJ_ERR_S or INJ_ERR_M or INJ_ERR_S2 or INJ_ERR_M2
   -> Error injecting (1 pulse)

## U11.5 Safety Mechanisms

### U11.5.1 Overview

EDC protection is applied to BootROM.

### U11.5.2 Hardware Description

Figure U11.2 shows block diagram of EDC for Boot ROM.

EDC check is always enabled.

When an EDC error is detected, the error notified to ECM module shall be reported to the external systems through ERROROUT# pin.

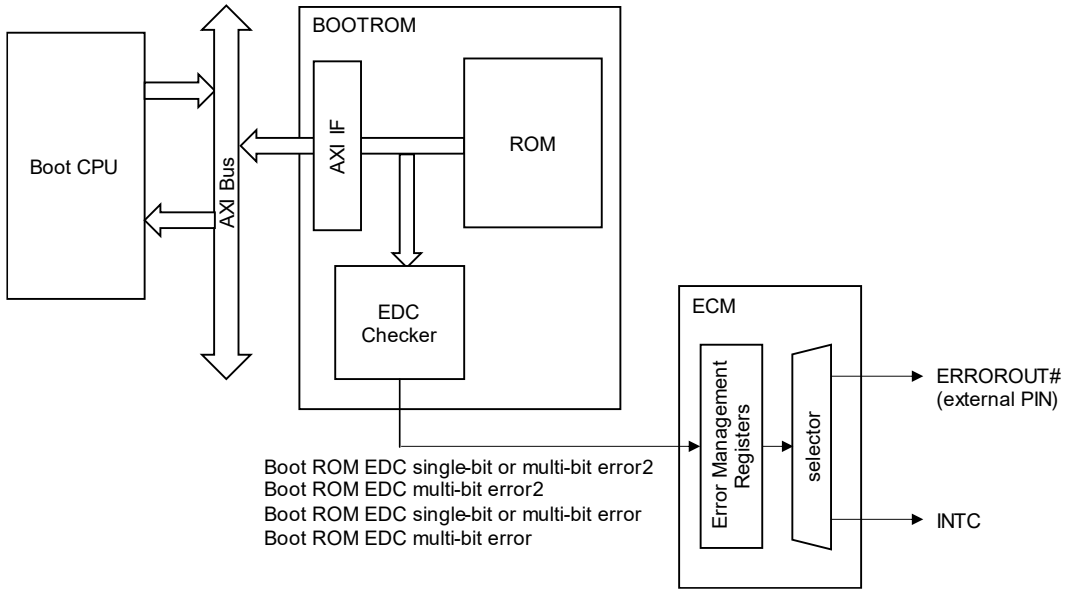



Figure U11.2 Block Diagram of EDC for Boot ROM

# U20. Measurement Trace Interface (MTI)

## U20.1 Overview

The measurement transmission interface (MTI) module is intended for use in debugging. Specifically, this module produces an output of the internal image transfer data being processed. To output the data, paths for connection of the module to an image signal processor (ISP), and to a module functionally equivalent to CoreSight (CSD), PCI express(PCIe) module are formed.

### U20.1.1 Features

The MTI module consists of three blocks; the MTI block itself, the MTI retiming (MTIRT) block, and the RT interface (MTIRI).

(1) The MTI block receives monitored signals from the ISPCS and rearranges each kind of signal into the specified data transfer format. After that, it transmits the rearranged data as image data to the MTIRT block.

ISP monitor channel number :

ISPCS -> ISPCORE: ispin x 1ch

ISPCS -> VIN: ispout x 8ch

The MTI block monitors data with the desired tag value as determined by the setting in a register (selecting all tag values is also possible).

When multiple monitors are selected, the MTI block divides the data into those from each monitor and repeats the output of the divided data while changing the monitor number.

To restrain the amount of data to be transmitted, the MTI block transmits a single cycle of synchronization signals (vsync and hsync) with the valid image data.

(2) The MTIRT block transmits the data to the MTIRI block where they are passed through two sets of flip-flop stages.

(3) The MTIRI block converts the image data to an AXI4 protocol stream and transmits the converted data to the CSD.

(4) CSD transmits data to PCIe .

(5) When the buffer memory within the MTI or MTIRI blocks overflows, the MTI block notifies to the CSD At the same time, it reports to cpu by interrupt.

### U20.1.2 Block Diagram

Figure U20.1 shows the block diagram of MTI.

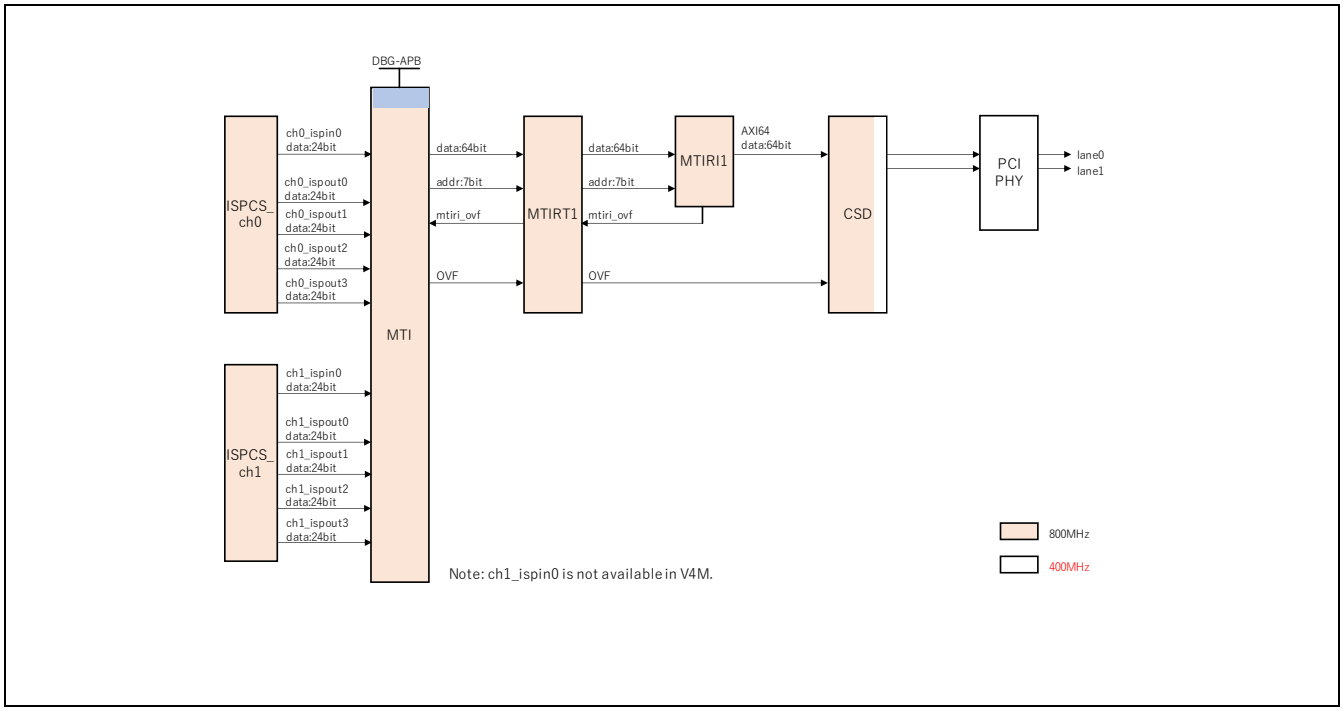



Figure U20.1 MTI Block Diagram

### U20.1.3 External Pins

MTI does not have external pins

### U20.1.4 Register Configuration

MTI instance added, base address for new instance is prepared.

On the CPU side : 0x00_F820_0XXX

On the debugger side : 0x8020_0XXX

Attempted write access to unused addresses is ignored, and read access returns all 0s.

The MTI registers are accessed through the APB interface for debugging (DEBUG-APB), which the CSD module controls. Registers at address H'FB0 and subsequent addresses (DEBUG-APB control registers) are required in control of the DEBUG-APB.

Table U20.1 shows the Register Configuration

**Table U20.1 Register Configuration**

| Register Name | Abbreviation | R/W | Initial value | APB Address Offset | Access Size |
|---|---|---|---|---|---|
| ispin monitoring enable Register for ISP_ch0 | MONI_ENABLE0_0 | R/W | H'0000_0000 | H'000 | 32 |
| ispin monitoring mode setting register for ISP _ch0 | ISPIN_MONI_MODE0 | R/W | H'0000_0000 | H'004 | 32 |
| isp monitoring mode setting register for ISP _ch0 | ISP_MONI_MODE0 | R/W | H'0000_0000 | H'008 | 32 |
| isp monitoring enable Register for ISP_ch0 | MONI_ENABLE0_1 | R/W | H'0000_0000 | H'00C | 32 |
| ispin monitoring enable Register for ISP _ch1 | MONI_ENABLE1_0 | R/W | H'0000_0000 | H'010 | 32 |
| ispin monitoring mode setting register for ISP _ch1 | ISPIN_MONI_MODE1 | R/W | H'0000_0000 | H'014 | 32 |
| isp monitoring mode setting register for ISP _ch1 | ISP_MONI_MODE1 | R/W | H'0000_0000 | H'018 | 32 |
| isp monitoring enable Register for ISP_ch1 | MONI_ENABLE1_1 | R/W | H'0000_0000 | H'01C | 32 |
| Interrupt status register | INT_STATUS | R | H'0000_0000 | H'080 | 32 |
| Interrupt enable register | INT_ENABLE | R/W | H'0000_0000 | H'084 | 32 |
| Interrupt clearing register | INT_CLEAR | W | H'0000_0000 | H'088 | 32 |
| MTIRI AXI address setting register for sync | AWADDR_SET_S | R/W | H'2B00_0000 | H'100 | 32 |
| MTIRI AXI addressing setting register for data | AWADDR_SET_D | R/W | H'2B00_0018 | H'104 | 32 |
| Reserved register for MTIRI | MTIRI_REG_RESERVE | R/W | H'0000_0000 | H'204 | 32 |
| Image data storage size setting register 1 | MONI_LEN1 | R/W | H'001E_001E | H'208 | 32 |
| Image data storage size setting register 2 | MONI_LEN2 | R/W | H'001E_001E | H'20C | 32 |
| Image data storage size setting register 3 | MONI_LEN3 | R/W | H'001E_001E | H'210 | 32 |
| Image data storage size setting register 4 | MONI_LEN4 | R/W | H'001E_001E | H'214 | 32 |
| Image data storage size setting register 5 | MONI_LEN5 | R/W | H'001E_001E | H'218 | 32 |
| Image data storage size setting register 6 | MONI_LEN6 | R/W | H'001E_001E | H'21C | 32 |
| Image data storage size setting register 7 | MONI_LEN7 | R/W | H'001E_001E | H'220 | 32 |

| Register Name | Abbreviation | R/W | Initial value | APB Address Offset | Access Size |
|---|---|---|---|---|---|
| Image data storage size setting register 8 | MONI_LEN8 | R/W | H'001E_001E | H'224 | 32 |
| Signal counter 1 | SIG_COUNT1 | R | H'0000_0000 | H'300 | 32 |
| Signal counter 2 | SIG_COUNT2 | R | H'0000_0000 | H'304 | 32 |
| Signal counter 3 | SIG_COUNT3 | R | H'0000_0000 | H'308 | 32 |
| Signal counter 4 | SIG_COUNT4 | R | H'0000_0000 | H'30C | 32 |
| Signal counter 5 | SIG_COUNT5 | R | H'0000_0000 | H'310 | 32 |
| Signal counter 6 | SIG_COUNT6 | R | H'0000_0000 | H'314 | 32 |
| Signal counter 7 | SIG_COUNT7 | R | H'0000_0000 | H'318 | 32 |
| Signal counter 8 | SIG_COUNT8 | R | H'0000_0000 | H'31C | 32 |
| Signal counter 9 | SIG_COUNT9 | R | H'0000_0000 | H'320 | 32 |
| Signal counter 10 | SIG_COUNT10 | R | H'0000_0000 | H'324 | 32 |
| Signal counter 11 | SIG_COUNT11 | R | H'0000_0000 | H'328 | 32 |
| Signal counter 12 | SIG_COUNT12 | R | H'0000_0000 | H'32C | 32 |
| Signal counter 13 | SIG_COUNT13 | R | H'0000_0000 | H'330 | 32 |
| Signal counter 14 | SIG_COUNT14 | R | H'0000_0000 | H'334 | 32 |
| Signal counter 15 | SIG_COUNT15 | R | H'0000_0000 | H'338 | 32 |
| Signal counter 16 | SIG_COUNT16 | R | H'0000_0000 | H'33C | 32 |
| Signal counter 17 | SIG_COUNT17 | R | H'0000_0000 | H'340 | 32 |
| Signal counter 18 | SIG_COUNT18 | R | H'0000_0000 | H'344 | 32 |
| Signal counter 19 | SIG_COUNT19 | R | H'0000_0000 | H'348 | 32 |
| Signal counter 20 | SIG_COUNT20 | R | H'0000_0000 | H'34C | 32 |
| Signal counter hold | SIG_COUNT_HOLD | R/W | H'0000_0000 | H'380 | 32 |
| Debugging mode register | DEBUG_MODE | R/W | H'0000_0000 | H'400 | 32 |
| Debugging mode data1 | DEBUG_DATA1 | R/W | H'0000_0000 | H'404 | 32 |
| Debugging mode data2 | DEBUG_DATA2 | R/W | H'0000_0000 | H'408 | 32 |
| Debugging mode addr | DEBUG_ADDR | R/W | H'0000_0000 | H'40C | 32 |
| Lock access register | LOCKACCESS | W | H'0000_0000 | H'FB0 | 32 |
| Lock status register | LOCKSTATUS | R | H'0000_0003 | H'FB4 | 32 |
| Peripheral identification register 4 | Peripheral_ID4 | R | H'0000_0004 | H'FD0 | 32 |
| Peripheral identification register 5 | Peripheral_ID5 | R | H'0000_0000 | H'FD4 | 32 |
| Peripheral identification register 6 | Peripheral_ID6 | R | H'0000_0000 | H'FD8 | 32 |
| Peripheral identification register 7 | Peripheral_ID7 | R | H'0000_0000 | H'FDC | 32 |
| Peripheral identification register 0 | Peripheral_ID0 | R | H'0000_0017 | H'FE0 | 32 |

| Register Name | Abbreviation | R/W | Initial value | APB Address Offset | Access Size |
|---|---|---|---|---|---|
| Peripheral identification register 1 | Peripheral_ID1 | R | H'0000_0030 | H'FE4 | 32 |
| Peripheral identification register 2 | Peripheral_ID2 | R | H'0000_000A | H'FE8 | 32 |
| Peripheral identification register 3 | Peripheral_ID3 | R | H'0000_0000 | H'FEC | 32 |
| Component identification register 0 | Component _ID0 | R | H'0000_000D | H'FF0 | 32 |
| Component identification register 1 | Component _ID1 | R | H'0000_00F0 | H'FF4 | 32 |
| Component identifcation register 2 | Component _ID2 | R | H'0000_0005 | H'FF8 | 32 |
| Component identification register 3 | Component _ID3 | R | H'0000_00B1 | H'FFC | 32 |

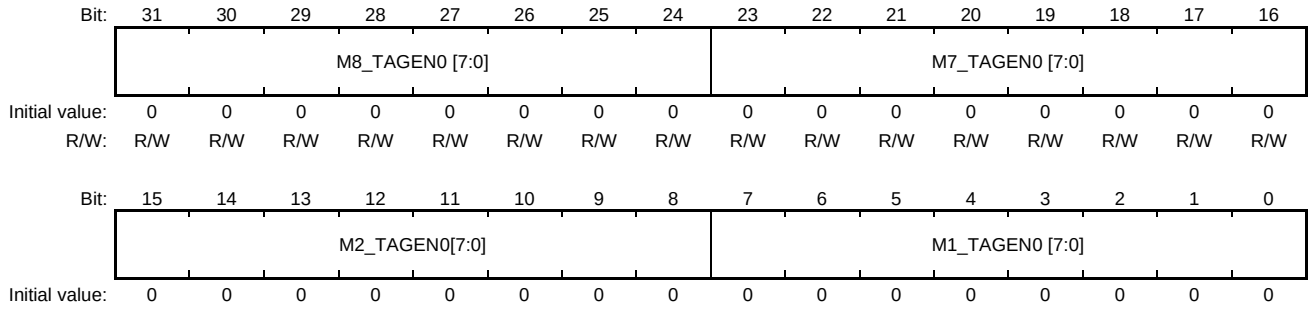
### U20.1.5 Connected Module

The interface signals for acquiring image data connected from the ISP to the MTI are shown below.

| Category | Signals | Number | Description |
|---|---|---|---|
| ISP_ch0_ispin*<br>ISP_ch1_ispin*<br>ISP_ch2_ispin*<br>ISP_ch3_ispin*<br>(*:0～1) | data[23:0] | 24 | Image data itself.<br>Specify the sampling range with ISPIN_MONI_MODE0 or 1 register. |
| | tag[2:0] | 3 | This is information attached to image data. |
| | fmt[5:0] | 6 | This is information attached to image data. |
| | field | 1 | This is information attached to image data. |
| | vsync | 1 | |
| | hsync | 1 | |

## U20.2 Register Description

### U20.2.1 ispin monitoring Enable Register for ISP_ch0 (MONI_ENABLE0_0)



| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | M8_TAGEN0 [7] | 0 | R/W | ISP_ch0_ispin3 Monitoring Enable<br>1: Data from ispin3 with 7 as the tag value are monitored. |
| 30 | M8_TAGEN0 [6] | 0 | R/W | ISP_ch0_ispin3 Monitoring Enable<br>1: Data from ispin3 with 6 as the tag value are monitored. |
| 29 | M8_TAGEN0 [5] | 0 | R/W | ISP_ch0_ispin3 Monitoring Enable<br>1: Data from ispin3 with 5 as the tag value are monitored. |
| 28 | M8_TAGEN0 [4] | 0 | R/W | ISP_ch0_ispin3 Monitoring Enable<br>1: Data from ispin3 with 4 as the tag value are monitored. |
| 27 | M8_TAGEN0 [3] | 0 | R/W | ISP_ch0_ispin3 Monitoring Enable<br>1: Data from ispin3 with 3 as the tag value are monitored. |
| 26 | M8_TAGEN0 [2] | 0 | R/W | ISP_ch0_ispin3 Monitoring Enable<br>1: Data from ispin3 with 2 as the tag value are monitored. |
| 25 | M8_TAGEN0 [1] | 0 | R/W | ISP_ch0_ispin3 Monitoring Enable<br>1: Data from ispin3 with 1 as the tag value are monitored. |
| 24 | M8_TAGEN0 [0] | 0 | R/W | ISP_ch0_ispin3 Monitoring Enable<br>1: Data from ispin3 with 0 as the tag value are monitored. |
| 23 | M7_TAGEN0 [7] | 0 | R/W | ISP_ch0_ispin2 Monitoring Enable<br>1: Data from ispin2 with 7 as the tag value are monitored. |
| 22 | M7_TAGEN0 [6] | 0 | R/W | ISP_ch0_ispin2 Monitoring Enable<br>1: Data from ispin2 with 6 as the tag value are monitored. |
| 21 | M7_TAGEN0 [5] | 0 | R/W | ISP_ch0_ispin2 Monitoring Enable<br>1: Data from ispin2 with 5 as the tag value are monitored. |
| 20 | M7_TAGEN0 [4] | 0 | R/W | ISP_ch0_ispin2 Monitoring Enable<br>1: Data from ispin2 with 4 as the tag value are monitored. |
| 19 | M7_TAGEN0 [3] | 0 | R/W | ISP_ch0_ispin2 Monitoring Enable<br>1: Data from ispin2 with 3 as the tag value are monitored. |
| 18 | M7_TAGEN0 [2] | 0 | R/W | ISP_ch0_ispin2 Monitoring Enable<br>1: Data from ispin2 with 2 as the tag value are monitored. |
| 17 | M7_TAGEN0 [1] | 0 | R/W | ISP_ch0_ispin2 Monitoring Enable<br>1: Data from ispin2 with 1 as the tag value are monitored. |
| 16 | M7_TAGEN0 [0] | 0 | R/W | ISP_ch0_ispin2 Monitoring Enable<br>1: Data from ispin2 with 0 as the tag value are monitored. |
| 15 | M2_TAGEN0[7] | 0 | R/W | ISP_ch0_ispin1 Monitoring Enable<br>1: Data from ispin1 with 7 as the tag value are monitored. |
| 14 | M2_TAGEN0 [6] | 0 | R/W | ISP_ch0_ispin1 Monitoring Enable<br>1: Data from ispin1 with 6 as the tag value are monitored. |
| 13 | M2_TAGEN0 [5] | 0 | R/W | ISP_ch0_ispin1 Monitoring Enable<br>1: Data from ispin1 with 5 as the tag value are monitored. |
| 12 | M2_TAGEN0 [4] | 0 | R/W | ISP_ch0_ispin1 Monitoring Enable<br>1: Data from ispin1 with 4 as the tag value are monitored. |
| 11 | M2_TAGEN0 [3] | 0 | R/W | ISP_ch0_ispin1 Monitoring Enable<br>1: Data from ispin1 with 3 as the tag value are monitored. |
| 10 | M2_TAGEN0 [2] | 0 | R/W | ISP_ch0_ispin1 Monitoring Enable<br>1: Data from ispin1 with 2 as the tag value are monitored. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | M2_TAGEN0 [1] | 0 | R/W | ISP_ch0_ispin1 Monitoring Enable<br>1: Data from ispin1 with 1 as the tag value are monitored. |
| 8 | M2_TAGEN0 [0] | 0 | R/W | ISP_ch0_ispin1 Monitoring Enable<br>1: Data from ispin 1 with 0 as the tag value are monitored. |
| 7 | M1_TAGEN0 [7] | 0 | R/W | ISP_ch0_ispin0 Monitoring Enable<br>1: Data from ispin0 with 7 as the tag value are monitored. |
| 6 | M1_TAGEN0 [6] | 0 | R/W | ISP_ch0_ispin0 Monitoring Enable<br>1: Data from ispin0 with 6 as the tag value are monitored. |
| 5 | M1_TAGEN0 [5] | 0 | R/W | ISP_ch0_ispin0 Monitoring Enable<br>1: Data from ispin0 with 5 as the tag value are monitored. |
| 4 | M1_TAGEN0 [4] | 0 | R/W | ISP_ch0_ispin0 Monitoring Enable<br>1: Data from ispin0 with 4 as the tag value are monitored. |
| 3 | M1_TAGEN0 [3] | 0 | R/W | ISP_ch0_ispin0 Monitoring Enable<br>1: Data from ispin0 with 3 as the tag value are monitored. |
| 2 | M1_TAGEN0 [2] | 0 | R/W | ISP_ch0_ispin0 Monitoring Enable<br>1: Data from ispin0 with 2 as the tag value are monitored. |
| 1 | M1_TAGEN0 [1] | 0 | R/W | ISP_ch0_ispin0 Monitoring Enable<br>1: Data from ispin0 with 1 as the tag value are monitored. |
| 0 | M1_TAGEN0 [0] | 0 | R/W | ISP_ch0_ispin0 Monitoring Enable<br>1: Data from ispin0 with 0 as the tag value are monitored. |

### U20.2.2 ispin Monitoring Mode Setting Register for ISP_ch0 (ISPIN_MONI_MODE0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | M8_MONI_MODE0 | | — | — | — | — | — | — | M7_MONI_MODE0 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | M2_MONI_MODE0 | | — | — | — | — | — | — | M1_MONI_MODE0 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R/W | Reserved |
| 25 to 24 | M8_MONI_MODE0 | 0 | R/W | ISP_ch0_ispin3 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch0 ispin3<br>1: 12-bit mode for data from ISP_ch0 ispin3<br>2: 20-bit mode for data from ISP_ch0 ispin3<br>3: 8-bit made for data from ISP_ch0 ispin3 |
| 23 to 18 | — | All 0 | R/W | Reserved |
| 17 to 16 | M7_MONI_MODE0 | 0 | R/W | ISP_ch0_ispin2 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch0 ispin2<br>1: 12-bit mode for data from ISP_ch0 ispin2<br>2: 20-bit mode for data from ISP_ch0 ispin2<br>3: 8-bit made for data from ISP_ch0 ispin2 |
| 15 to 10 | — | All 0 | R/W | Reserved |
| 9 to 8 | M2_MONI_MODE0 | 0 | R/W | ISP_ch0_ispin1 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch0 ispin1<br>1: 12-bit mode for data from ISP_ch0 ispin1<br>2: 20-bit mode for data from ISP_ch0 ispin1<br>3: 8-bit made for data from ISP_ch0 ispin1 |
| 7 to 2 | — | All 0 | R/W | Reserved |
| 1 to 0 | M1_MONI_MODE0 | 0 | R/W | ISP_ch0_ispin0 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch0 ispin0<br>1: 12-bit mode for data from ISP_ch0 ispin0<br>2: 20-bit mode for data from ISP_ch0 ispin0<br>3: 8-bit mode for data from ISP_ch0 ispin0 |

### U20.2.3 isp Monitoring Mode Setting Register for ISP_ch0 (ISP_MONI_MODE0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | M6_MONI_MODE0 | | — | — | — | — | — | — | M5_MONI_MODE0 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | M4_MONI_MODE0 | | — | — | — | — | — | — | M3_MONI_MODE0 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R/W | Reserved |
| 25 to 24 | M6_MONI_MODE0 | 0 | R/W | ISP_ch0_ispout3 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch0 ispout3<br>1: 12-bit mode for data from ISP_ch0 ispout3<br>2: 20-bit mode for data from ISP_ch0 ispout3<br>3: 8-bit made for data from ISP_ch0 ispout3 |
| 23 to 18 | — | All 0 | R/W | Reserved |
| 17 to 16 | M5_MONI_MODE0 | 0 | R/W | ISP_ch0_ispout2 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch0 ispout2<br>1: 12-bit mode for data from ISP_ch0 ispout2<br>2: 20-bit mode for data from ISP_ch0 ispout2<br>3: 8-bit made for data from ISP_ch0 ispout2 |
| 15 to 10 | — | All 0 | R/W | Reserved |
| 9 to 8 | M4_MONI_MODE0 | 0 | R/W | ISP_ch0_ispout1 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch0 ispout1<br>1: 12-bit mode for data from ISP_ch0 ispout1<br>2: 20-bit mode for data from ISP_ch0 ispout1<br>3: 8-bit made for data from ISP_ch0 ispout1 |
| 7 to 2 | — | All 0 | R/W | Reserved |
| 1 to 0 | M3_MONI_MODE0 | 0 | R/W | ISP_ch0_ispout0 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch0 ispout0<br>1: 12-bit mode for data from ISP_ch0 ispout0<br>2: 20-bit mode for data from ISP_ch0 ispout0<br>3: 8-bit mode for data from ISP_ch0 ispout0 |

### U20.2.4 ispout monitoring Enable Register for ISP_ch0 (MONI_ENABLE0_1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M6_TAGEN0 [7:0] | | | | | | | | M5_TAGEN0 [7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M4_TAGEN0[7:0] | | | | | | | | M3_TAGEN0 [7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | M6_TAGEN0[7] | 0 | R/W | ISP_ch0_ispout3 Monitoring Enable<br>1: Data from ispout3 with 7 as the tag value are monitored. |
| 30 | M6_TAGEN0 [6] | 0 | R/W | ISP_ch0_ispout3 Monitoring Enable<br>1: Data from ispout3 with 6 as the tag value are monitored. |
| 29 | M6_TAGEN0 [5] | 0 | R/W | ISP_ch0_ispout3 Monitoring Enable<br>1: Data from ispout3 with 5 as the tag value are monitored. |
| 28 | M6_TAGEN0 [4] | 0 | R/W | ISP_ch0_ispout3 Monitoring Enable<br>1: Data from ispout3 with 4 as the tag value are monitored. |
| 27 | M6_TAGEN0 [3] | 0 | R/W | ISP_ch0_ispout3 Monitoring Enable<br>1: Data from ispout3 with 3 as the tag value are monitored. |
| 26 | M6_TAGEN0 [2] | 0 | R/W | ISP_ch0_ispout3 Monitoring Enable<br>1: Data from ispout3 with 2 as the tag value are monitored. |
| 25 | M6_TAGEN0 [1] | 0 | R/W | ISP_ch0_ispout3 Monitoring Enable<br>1: Data from ispout3 with 1 as the tag value are monitored. |
| 24 | M6_TAGEN0 [0] | 0 | R/W | ISP_ch0_ispout3 Monitoring Enable<br>1: Data from ispout3 with 0 as the tag value are monitored. |
| 23 | M5_TAGEN0[7] | 0 | R/W | ISP_ch0_ispout2 Monitoring Enable<br>1: Data from ispout2 with 7 as the tag value are monitored. |
| 22 | M5_TAGEN0 [6] | 0 | R/W | ISP_ch0_ispout2 Monitoring Enable<br>1: Data from ispout2 with 6 as the tag value are monitored. |
| 21 | M5_TAGEN0 [5] | 0 | R/W | ISP_ch0_ispout2 Monitoring Enable<br>1: Data from ispout2 with 5 as the tag value are monitored. |
| 20 | M5_TAGEN0 [4] | 0 | R/W | ISP_ch0_ispout2 Monitoring Enable<br>1: Data from ispout2 with 4 as the tag value are monitored. |
| 19 | M5_TAGEN0 [3] | 0 | R/W | ISP_ch0_ispout2 Monitoring Enable<br>1: Data from ispout2 with 3 as the tag value are monitored. |
| 18 | M5_TAGEN0 [2] | 0 | R/W | ISP_ch0_ispout2 Monitoring Enable<br>1: Data from ispout2 with 2 as the tag value are monitored. |
| 17 | M5_TAGEN0 [1] | 0 | R/W | ISP_ch0_ispout2 Monitoring Enable<br>1: Data from ispout2 with 1 as the tag value are monitored. |
| 16 | M5_TAGEN0 [0] | 0 | R/W | ISP_ch0_ispout2 Monitoring Enable<br>1: Data from ispout2 with 0 as the tag value are monitored. |
| 15 | M4_TAGEN0[7] | 0 | R/W | ISP_ch0_ispout1 Monitoring Enable<br>1: Data from ispin1 with 7 as the tag value are monitored. |
| 14 | M4_TAGEN0 [6] | 0 | R/W | ISP_ch0_ispout1 Monitoring Enable<br>1: Data from ispout1 with 6 as the tag value are monitored. |
| 13 | M4_TAGEN0 [5] | 0 | R/W | ISP_ch0_ispout1 Monitoring Enable<br>1: Data from ispout1 with 5 as the tag value are monitored. |
| 12 | M4_TAGEN0 [4] | 0 | R/W | ISP_ch0_ispout1 Monitoring Enable<br>1: Data from ispout1 with 4 as the tag value are monitored. |
| 11 | M4_TAGEN0 [3] | 0 | R/W | ISP_ch0_ispout1 Monitoring Enable<br>1: Data from ispout1 with 3 as the tag value are monitored. |
| 10 | M4_TAGEN0 [2] | 0 | R/W | ISP_ch0_ispout1 Monitoring Enable<br>1: Data from ispout1 with 2 as the tag value are monitored. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 9 | M4_TAGEN0 [1] | 0 | R/W | ISP_ch0_ispout1 Monitoring Enable<br>1: Data from ispout1 with 1 as the tag value are monitored. |
| 8 | M4_TAGEN0 [0] | 0 | R/W | ISP_ch0_ispout1 Monitoring Enable<br>1: Data from ispout1 with 0 as the tag value are monitored. |
| 7 | M3_TAGEN0 [7] | 0 | R/W | ISP_ch0_ispout0 Monitoring Enable<br>1: Data from ispout0 with 7 as the tag value are monitored. |
| 6 | M3_TAGEN0 [6] | 0 | R/W | ISP_ch0_ispout0 Monitoring Enable<br>1: Data from ispout0 with 6 as the tag value are monitored. |
| 5 | M3_TAGEN0 [5] | 0 | R/W | ISP_ch0_ispout0 Monitoring Enable<br>1: Data from ispout0 with 5 as the tag value are monitored. |
| 4 | M3_TAGEN0 [4] | 0 | R/W | ISP_ch0_ispout0 Monitoring Enable<br>1: Data from ispout0 with 4 as the tag value are monitored. |
| 3 | M3_TAGEN0 [3] | 0 | R/W | ISP_ch0_ispout0 Monitoring Enable<br>1: Data from ispout0 with 3 as the tag value are monitored. |
| 2 | M3_TAGEN0 [2] | 0 | R/W | ISP_ch0_ispout0 Monitoring Enable<br>1: Data from ispout0 with 2 as the tag value are monitored. |
| 1 | M3_TAGEN0 [1] | 0 | R/W | ISP_ch0_ispout0 Monitoring Enable<br>1: Data from ispout0 with 1 as the tag value are monitored. |
| 0 | M3_TAGEN0 [0] | 0 | R/W | ISP_ch0_ispout0 Monitoring Enable<br>1: Data from ispout0 with 0 as the tag value are monitored. |

### U20.2.5 ispin monitoring Enable Register for ISP_ch1 (MONI_ENABLE1_0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M8_TAGEN1[7:0] | | | | | | | | M7_TAGEN1[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M2_TAGEN1[7:0] | | | | | | | | M1_TAGEN1[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | M8_TAGEN1 [7] | 0 | R/W | ISP_ch1_ispin3 Monitoring Enable<br>1: Data from ispin3 with 7 as the tag value are monitored. |
| 30 | M8_TAGEN1 [6] | 0 | R/W | ISP_ch1_ispin3 Monitoring Enable<br>1: Data from ispin3 with 6 as the tag value are monitored. |
| 29 | M8_TAGEN1 [5] | 0 | R/W | ISP_ch1_ispin3 Monitoring Enable<br>1: Data from ispin3 with 5 as the tag value are monitored. |
| 28 | M8_TAGEN1 [4] | 0 | R/W | ISP_ch1_ispin3 Monitoring Enable<br>1: Data from ispin3 with 4 as the tag value are monitored. |
| 27 | M8_TAGEN1 [3] | 0 | R/W | ISP_ch1_ispin3 Monitoring Enable<br>1: Data from ispin3 with 3 as the tag value are monitored. |
| 26 | M8_TAGEN1 [2] | 0 | R/W | ISP_ch1_ispin3 Monitoring Enable<br>1: Data from ispin3 with 2 as the tag value are monitored. |
| 25 | M8_TAGEN1 [1] | 0 | R/W | ISP_ch1_ispin3 Monitoring Enable<br>1: Data from ispin3 with 1 as the tag value are monitored. |
| 24 | M8_TAGEN1 [0] | 0 | R/W | ISP_ch1_ispin3 Monitoring Enable<br>1: Data from ispin3 with 0 as the tag value are monitored. |
| 23 | M7_TAGEN1 [7] | 0 | R/W | ISP_ch1_ispin2 Monitoring Enable<br>1: Data from ispin2 with 7 as the tag value are monitored. |
| 22 | M7_TAGEN1 [6] | 0 | R/W | ISP_ch1_ispin2 Monitoring Enable<br>1: Data from ispin2 with 6 as the tag value are monitored. |
| 21 | M7_TAGEN1 [5] | 0 | R/W | ISP_ch1_ispin2 Monitoring Enable<br>1: Data from ispin2 with 5 as the tag value are monitored. |
| 20 | M7_TAGEN1 [4] | 0 | R/W | ISP_ch1_ispin2 Monitoring Enable<br>1: Data from ispin2 with 4 as the tag value are monitored. |
| 19 | M7_TAGEN1 [3] | 0 | R/W | ISP_ch1_ispin2 Monitoring Enable<br>1: Data from ispin2 with 3 as the tag value are monitored. |
| 18 | M7_TAGEN1 [2] | 0 | R/W | ISP_ch1_ispin2 Monitoring Enable<br>1: Data from ispin2 with 2 as the tag value are monitored. |
| 17 | M7_TAGEN1 [1] | 0 | R/W | ISP_ch1_ispin2 Monitoring Enable<br>1: Data from ispin2 with 1 as the tag value are monitored. |
| 16 | M7_TAGEN1 [0] | 0 | R/W | ISP_ch1_ispin2 Monitoring Enable<br>1: Data from ispin2 with 0 as the tag value are monitored. |
| 15 | M2_TAGEN1[7] | 0 | R/W | ISP_ch1_ispin1 Monitoring Enable<br>1: Data from ispin1 with 7 as the tag value are monitored. |
| 14 | M2_TAGEN1 [6] | 0 | R/W | ISP_ch1_ispin1 Monitoring Enable<br>1: Data from ispin1 with 6 as the tag value are monitored. |
| 13 | M2_TAGEN1 [5] | 0 | R/W | ISP_ch1_ispin1 Monitoring Enable<br>1: Data from ispin1 with 5 as the tag value are monitored. |
| 12 | M2_TAGEN1 [4] | 0 | R/W | ISP_ch1_ispin1 Monitoring Enable<br>1: Data from ispin1 with 4 as the tag value are monitored. |
| 11 | M2_TAGEN1 [3] | 0 | R/W | ISP_ch1_ispin1 Monitoring Enable<br>1: Data from ispin  1 with 3 as the tag value are monitored. |
| 10 | M2_TAGEN1 [2] | 0 | R/W | ISP_ch1_ispin1 Monitoring Enable<br>1: Data from ispin1 with 2 as the tag value are monitored. |

| | | | | |
|---|---|---|---|---|
| 9 | M2_TAGEN1 [1] | 0 | R/W | ISP_ch1_ispin1 Monitoring Enable<br>1: Data from ispin1 with 1 as the tag value are monitored. |
| 8 | M2_TAGEN1 [0] | 0 | R/W | ISP_ch1_ispin1 Monitoring Enable<br>1: Data from ispin1 with 0 as the tag value are monitored. |
| 7 | M1_TAGEN1 [7] | 0 | R/W | ISP_ch1_ispin0 Monitoring Enable<br>1: Data from ispin0 with 7 as the tag value are monitored. |
| 6 | M1_TAGEN1 [6] | 0 | R/W | ISP_ch1_ispin0 Monitoring Enable<br>1: Data from ispin0 with 6 as the tag value are monitored. |
| 5 | M1_TAGEN1 [5] | 0 | R/W | ISP_ch1_ispin0 Monitoring Enable<br>1: Data from ispin0 with 5 as the tag value are monitored. |
| 4 | M1_TAGEN1 [4] | 0 | R/W | ISP_ch1_ispin0 Monitoring Enable<br>1: Data from ispin0 with 4 as the tag value are monitored. |
| 3 | M1_TAGEN1 [3] | 0 | R/W | ISP_ch1_ispin0 Monitoring Enable<br>1: Data from ispin0 with 3 as the tag value are monitored. |
| 2 | M1_TAGEN1 [2] | 0 | R/W | ISP_ch1_ispin0 Monitoring Enable<br>1: Data from ispin0 with 2 as the tag value are monitored. |
| 1 | M1_TAGEN1 [1] | 0 | R/W | ISP_ch1_ispin0 Monitoring Enable<br>1: Data from ispin0 with 1 as the tag value are monitored. |
| 0 | M1_TAGEN1 [0] | 0 | R/W | ISP_ch1_ispin0 Monitoring Enable<br>1: Data from ispin0 with 0 as the tag value are monitored. |

### U20.2.6 ispin Monitoring Mode Setting Register for ISP_ch1 (ISPIN_MONI_MODE1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | M8_MONI_MODE1 | | — | — | — | — | — | — | M7_MONI_MODE1 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | M2_MONI_MODE1 | | — | — | — | — | — | — | M1_MONI_MODE1 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 26 | — | All 0 | R/W | Reserved |
| 25 to 24 | M8_MONI_MODE1 | 0 | R/W | ISP_ch1_ispin3 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch1 ispin3<br>1: 12-bit mode for data from ISP_ch1 ispin3<br>2: 20-bit mode for data from ISP_ch1 ispin3<br>3: 8-bit made for data from ISP_ch1 ispin3 |
| 23 to 18 | — | All 0 | R/W | Reserved |
| 17 to 16 | M7_MONI_MODE1 | 0 | R/W | ISP_ch1_ispin2 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch1 ispin2<br>1: 12-bit mode for data from ISP_ch1 ispin2<br>2: 20-bit mode for data from ISP_ch1 ispin2<br>3: 8-bit made for data from ISP_ch1 ispin2 |
| 15 to 10 | — | All 0 | R/W | Reserved |
| 9 to 8 | M2_MONI_MODE1 | 0 | R/W | ISP_ch1_ispin1 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch1 ispin1<br>1: 12-bit mode for data from ISP_ch1 ispin1<br>2: 20-bit mode for data from ISP_ch1 ispin1<br>3: 8-bit made for data from ISP_ch1 ispin1 |
| 7 to 2 | — | All 0 | R/W | Reserved |
| 1 to 0 | M1_MONI_MODE1 | 0 | R/W | ISP_ch1_ispin0 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch1 ispin0<br>1: 12-bit mode for data from ISP_ch1 ispin0<br>2: 20-bit mode for data from ISP_ch1 ispin0<br>3: 8-bit mode for data from ISP_ch1 ispin0 |

### U20.2.7 isp Monitoring Mode Setting Register for ISP_ch1 (ISP_MONI_MODE1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | M6_MONI_MODE1 | | — | — | — | — | — | — | M5_MONI_MODE1 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | M4_MONI_MODE1 | | — | — | — | — | — | — | M3_MONI_MODE1 | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R/W | Reserved |
| 25 to 24 | M6_MONI_MODE1 | 0 | R/W | ISP_ch1_ispout3 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch1 ispout3<br>1: 12-bit mode for data from ISP_ch1 ispout3<br>2: 20-bit mode for data from ISP_ch1 ispout3<br>3: 8-bit made for data from ISP_ch1 ispout3 |
| 23 to 18 | — | All 0 | R/W | Reserved |
| 17 to 16 | M5_MONI_MODE1 | 0 | R/W | ISP_ch1_ispout2 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch1 ispout2<br>1: 12-bit mode for data from ISP_ch1 ispout2<br>2: 20-bit mode for data from ISP_ch1 ispout2<br>3: 8-bit made for data from ISP_ch1 ispout2 |
| 15 to 10 | — | All 0 | R/W | Reserved |
| 9 to 8 | M4_MONI_MODE1 | 0 | R/W | ISP_ch1_ispout1 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch1 ispout1<br>1: 12-bit mode for data from ISP_ch1 ispout1<br>2: 20-bit mode for data from ISP_ch1 ispout1<br>3: 8-bit made for data from ISP_ch1 ispout1 |
| 7 to 2 | — | All 0 | R/W | Reserved |
| 1 to 0 | M3_MONI_MODE1 | 0 | R/W | ISP_ch1_ispout0 Monitoring Mode Setting<br>0: 24-bit mode for data from ISP_ch1 ispout0<br>1: 12-bit mode for data from ISP_ch1 ispout0<br>2: 20-bit mode for data from ISP_ch1 ispout0<br>3: 8-bit mode for data from ISP_ch1 ispout0 |

### U20.2.8 ispout monitoring Enable Register for ISP_ch1 (MONI_ENABLE1_1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M6_TAGEN1[7:0] | | | | | | | | M5_TAGEN1[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | M4_TAGEN1[7:0] | | | | | | | | M3_TAGEN1[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | M6_TAGEN1 [7] | 0 | R/W | ISP_ch1_ispout3 Monitoring Enable<br>1: Data from ispout3 with 7 as the tag value are monitored. |
| 30 | M6_TAGEN1 [6] | 0 | R/W | ISP_ch1_ispout3 Monitoring Enable<br>1: Data from ispout3 with 6 as the tag value are monitored. |
| 29 | M6_TAGEN1 [5] | 0 | R/W | ISP_ch1_ispout3 Monitoring Enable<br>1: Data from ispout3 with 5 as the tag value are monitored. |
| 28 | M6_TAGEN1 [4] | 0 | R/W | ISP_ch1_ispiout3 Monitoring Enable<br>1: Data from ispout3 with 4 as the tag value are monitored. |
| 27 | M6_TAGEN1 [3] | 0 | R/W | ISP_ch1_ispout3 Monitoring Enable<br>1: Data from ispout3 with 3 as the tag value are monitored. |
| 26 | M6_TAGEN1 [2] | 0 | R/W | ISP_ch1_ispout3 Monitoring Enable<br>1: Data from isput3 with 2 as the tag value are monitored. |
| 25 | M6_TAGEN1 [1] | 0 | R/W | ISP_ch1_ispout3 Monitoring Enable<br>1: Data from ispout3 with 1 as the tag value are monitored. |
| 24 | M6_TAGEN1 [0] | 0 | R/W | ISP_ch1_ispout3 Monitoring Enable<br>1: Data from ispout3 with 0 as the tag value are monitored. |
| 23 | M5_TAGEN1 [7] | 0 | R/W | ISP_ch1_ispout2 Monitoring Enable<br>1: Data from ispout2 with 7 as the tag value are monitored. |
| 22 | M5_TAGEN1 [6] | 0 | R/W | ISP_ch1_ispout2 Monitoring Enable<br>1: Data from ispout2 with 6 as the tag value are monitored. |
| 21 | M5_TAGEN1 [5] | 0 | R/W | ISP_ch1_ispout2 Monitoring Enable<br>1: Data from ispout2 with 5 as the tag value are monitored. |
| 20 | M5_TAGEN1 [4] | 0 | R/W | ISP_ch1_ispiout2 Monitoring Enable<br>1: Data from ispout2 with 4 as the tag value are monitored. |
| 19 | M5_TAGEN1 [3] | 0 | R/W | ISP_ch1_ispout2 Monitoring Enable<br>1: Data from ispout2 with 3 as the tag value are monitored. |
| 18 | M5_TAGEN1 [2] | 0 | R/W | ISP_ch1_ispout2 Monitoring Enable<br>1: Data from isput2 with 2 as the tag value are monitored. |
| 17 | M5_TAGEN1 [1] | 0 | R/W | ISP_ch1_ispout2 Monitoring Enable<br>1: Data from ispout2 with 1 as the tag value are monitored. |
| 16 | M5_TAGEN1 [0] | 0 | R/W | ISP_ch1_ispout2 Monitoring Enable<br>1: Data from ispout2 with 0 as the tag value are monitored. |
| 15 | M4_TAGEN1[7] | 0 | R/W | ISP_ch1_ispout1 Monitoring Enable<br>1: Data from ispout1 with 7 as the tag value are monitored. |
| 14 | M4_TAGEN1 [6] | 0 | R/W | ISP_ch1_ispout1 Monitoring Enable<br>1: Data from ispout1 with 6 as the tag value are monitored. |
| 13 | M4_TAGEN1 [5] | 0 | R/W | ISP_ch1_ispout1 Monitoring Enable<br>1: Data from ispout1 with 5 as the tag value are monitored. |
| 12 | M4_TAGEN1 [4] | 0 | R/W | ISP_ch1_ispout1 Monitoring Enable<br>1: Data from ispout1 with 4 as the tag value are monitored. |
| 11 | M4_TAGEN1 [3] | 0 | R/W | ISP_ch1_ispout1 Monitoring Enable<br>1: Data from ispout1 with 3 as the tag value are monitored. |
| 10 | M4_TAGEN1 [2] | 0 | R/W | ISP_ch1_ispout1 Monitoring Enable<br>1: Data from ispout1 with 2 as the tag value are monitored. |

| | | | | |
|---|---|---|---|---|
| 9 | M4_TAGEN1 [1] | 0 | R/W | ISP_ch1_ispout1 Monitoring Enable<br>1: Data from ispout1 with 1 as the tag value are monitored. |
| 8 | M4_TAGEN1 [0] | 0 | R/W | ISP_ch1_ispout1 Monitoring Enable<br>1: Data from ispout1 with 0 as the tag value are monitored. |
| 7 | M3_TAGEN1 [7] | 0 | R/W | ISP_ch1_ispout0 Monitoring Enable<br>1: Data from ispout0 with 7 as the tag value are monitored. |
| 6 | M3_TAGEN1 [6] | 0 | R/W | ISP_ch1_ispout0 Monitoring Enable<br>1: Data from ispout0 with 6 as the tag value are monitored. |
| 5 | M3_TAGEN1 [5] | 0 | R/W | ISP_ch1_ispout0 Monitoring Enable<br>1: Data from ispout0 with 5 as the tag value are monitored. |
| 4 | M3_TAGEN1 [4] | 0 | R/W | ISP_ch1_ispiout0 Monitoring Enable<br>1: Data from ispout0 with 4 as the tag value are monitored. |
| 3 | M3_TAGEN1 [3] | 0 | R/W | ISP_ch1_ispout0 Monitoring Enable<br>1: Data from ispout0 with 3 as the tag value are monitored. |
| 2 | M3_TAGEN1 [2] | 0 | R/W | ISP_ch1_ispout0 Monitoring Enable<br>1: Data from isput0 with 2 as the tag value are monitored. |
| 1 | M3_TAGEN1 [1] | 0 | R/W | ISP_ch1_ispout0 Monitoring Enable<br>1: Data from ispout0 with 1 as the tag value are monitored. |
| 0 | M3_TAGEN1 [0] | 0 | R/W | ISP_ch1_ispout0 Monitoring Enable<br>1: Data from ispout0 with 0 as the tag value are monitored. |

### U20.2.9 Interrupt Status Register (INT_STATUS)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INT_STATUS[17:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R | Reserved |
| 17 | INT_STATUS[17] | 0 | R | Overflow of buffer holding data from ISP_ch1_ispin3<br>0: No overflow.<br>1: The buffer has overflowed. |
| 16 | INT_STATUS[16] | 0 | R | Overflow of buffer holding data from ISP_ch1_ispin2<br>0: No overflow.<br>1: The buffer has overflowed. |
| 15 | INT_STATUS[15] | 0 | R | Overflow of buffer holding data from ISP_ch0_ispin3<br>0: No overflow.<br>1: The buffer has overflowed. |
| 14 | INT_STATUS[14] | 0 | R | Overflow of buffer holding data from ISP_ch0_ispin2<br>0: No overflow.<br>1: The buffer has overflowed. |
| 13 | INT_STATUS[13] | 0 | R | Overflow of buffer holding data from ISP_ch1_ispout3<br>0: No overflow.<br>1: The buffer has overflowed. |
| 12 | INT_STATUS[12] | 0 | R | Overflow of buffer holding data from ISP_ch1_ispout2<br>0: No overflow.<br>1: The buffer has overflowed. |
| 11 | INT_STATUS[11] | 0 | R | Overflow of buffer holding data from ISP_ch1_ispout1<br>0: No overflow.<br>1: The buffer has overflowed. |
| 10 | INT_STATUS[10] | 0 | R | Overflow of buffer holding data from ISP_ch1_ispout0<br>0: No overflow.<br>1: The buffer has overflowed. |
| 9 | INT_STATUS[9] | 0 | R | Overflow of buffer holding data from ISP_ch1_ispin1<br>0: No overflow.<br>1: The buffer has overflowed. |
| 8 | INT_STATUS[8] | 0 | R | Overflow of buffer holding data from ISP_ch1_ispin0<br>0: No overflow.<br>1: The buffer has overflowed. |
| 7 | INT_STATUS[7] | 0 | R | Overflow of buffer holding data from ISP_ch0_ispout3<br>0: No overflow.<br>1: The buffer has overflowed. |
| 6 | INT_STATUS[6] | 0 | R | Overflow of buffer holding data from ISP_ch0_ispout2<br>0: No overflow.<br>1: The buffer has overflowed. |
| 5 | INT_STATUS[5] | 0 | R | Overflow of buffer holding data from ISP_ch0_ispout1<br>0: No overflow.<br>1: The buffer has overflowed. |
| 4 | INT_STATUS[4] | 0 | R | Overflow of buffer holding data from ISP_ch0_ispout0<br>0: No overflow.<br>1: The buffer has overflowed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | INT_STATUS[3] | 0 | R | Overflow of buffer holding data from ISP_ch0_ispin1<br>0: No overflow.<br>1: The buffer has overflowed. |
| 2 | INT_STATUS[2] | 0 | R | Overflow of buffer holding data from ISP_ch0_ispin0<br>0: No overflow.<br>1: The buffer has overflowed. |
| 1 | INT_STATUS[1] | 0 | R | Overflow of buffer in the MTIRI<br>0: No overflow.<br>1: The buffer has overflowed. |
| 0 | INT_STATUS[0] | 0 | R | bresp error (AXI slave error or AXI decoding error) in the MTIRI<br>0: No error.<br>1: An error has occurred. |

### U20.2.10 Interrupt Enable Register (INT_ENABLE)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INT_ENABLE[17:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | R/W | Reserved |
| 17 | INT_ENABLE[17] | 0 | R/W | Overflow of buffer holding data from ISP_ch1_ispin3<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 16 | INT_ENABLE[16] | 0 | R/W | Overflow of buffer holding data from ISP_ch1_ispin2<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 15 | INT_ENABLE[15] | 0 | R/W | Overflow of buffer holding data from ISP_ch0_ispin3<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 14 | INT_ENABLE[14] | 0 | R/W | Overflow of buffer holding data from ISP_ch0_ispin2<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 13 | INT_ENABLE[13] | 0 | R/W | Overflow of buffer holding data from ISP_ch1_ispout3<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 12 | INT_ENABLE[12] | 0 | R/W | Overflow of buffer holding data from ISP_ch1_ispout2<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 11 | INT_ENABLE[11] | 0 | R/W | Overflow of buffer holding data from ISP_ch1_ispout1<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 10 | INT_ENABLE[10] | 0 | R/W | Overflow of buffer holding data from ISP_ch1_ispout0<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 9 | INT_ENABLE[9] | 0 | R/W | Overflow of buffer holding data from ISP_ch1_ispin1<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 8 | INT_ENABLE[8] | 0 | R/W | Overflow of buffer holding data from ISP_ch1_ispin0<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 7 | INT_ENABLE[7] | 0 | R/W | Overflow of buffer holding data from ISP_ch0_ispout3<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 6 | INT_ENABLE[6] | 0 | R/W | Overflow of buffer holding data from ISP_ch0_ispout2<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 5 | INT_ENABLE[5] | 0 | R/W | Overflow of buffer holding data from ISP_ch0_ispout1<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 4 | INT_ENABLE[4] | 0 | R/W | Overflow of buffer holding data from ISP_ch0_ispout0<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | INT_ENABLE[3] | 0 | R/W | Overflow of buffer holding data from ISP_ch0_ispin1<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 2 | INT_ENABLE[2] | 0 | R/W | Overflow of buffer holding data from ISP_ch0_ispin0<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 1 | INT_ENABLE[1] | 0 | R/W | Overflow of buffer in the MTIRI<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |
| 0 | INT_ENABLE[0] | 0 | R/W | bresp error in the MTIRI<br>0: The interrupt request is masked.<br>1: The interrupt request is not masked. |

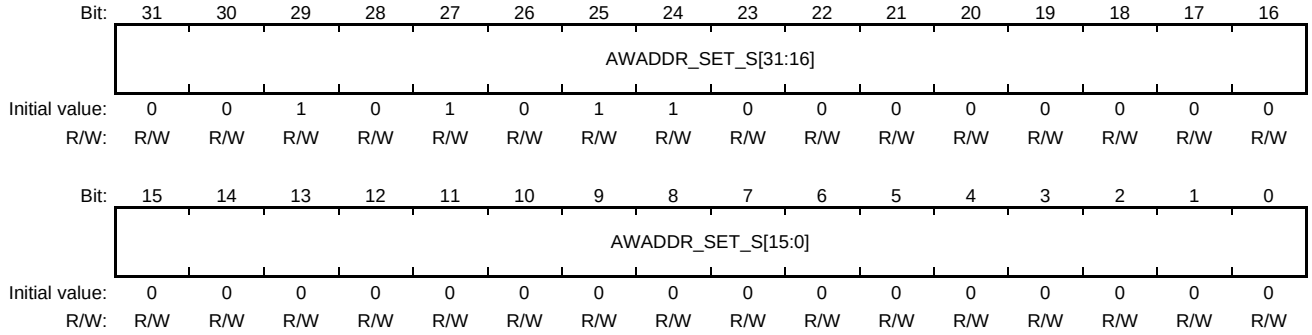
### U20.2.11 Interrupt Clearing Register (INT_CLEAR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INT_CLEAR[17:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

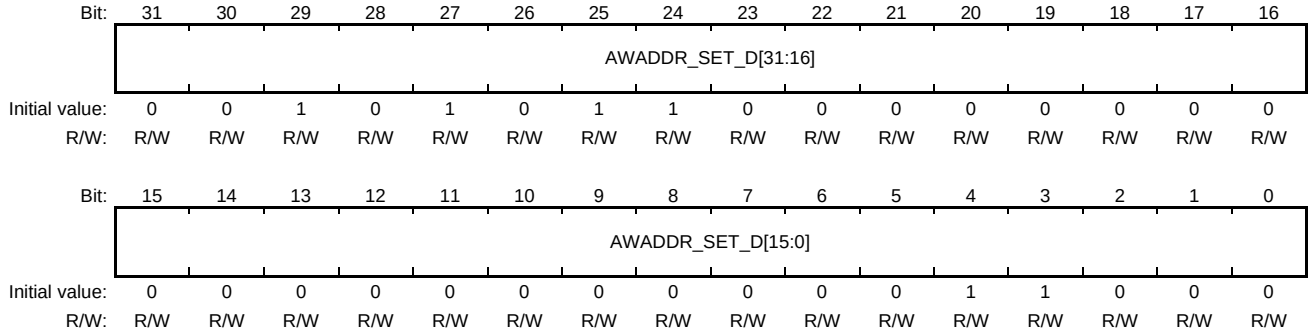
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 18 | — | All 0 | W | Reserved |
| 17 | INT_CLEAR[17] | 0 | W | Overflow of buffer holding data from ISP_ch1_ispin3<br>1: The corresponding bit in the interrupt status register is cleared. |
| 16 | INT_CLEAR[16] | 0 | W | Overflow of buffer holding data from ISP_ch1_ispin2<br>1: The corresponding bit in the interrupt status register is cleared. |
| 15 | INT_CLEAR[15] | 0 | W | Overflow of buffer holding data from ISP_ch0_ispin3<br>1: The corresponding bit in the interrupt status register is cleared. |
| 14 | INT_CLEAR[14] | 0 | W | Overflow of buffer holding data from ISP_ch0_ispin2<br>1: The corresponding bit in the interrupt status register is cleared. |
| 13 | INT_CLEAR[13] | 0 | W | Overflow of buffer holding data from ISP_ch1_ispout3<br>1: The corresponding bit in the interrupt status register is cleared. |
| 12 | INT_CLEAR[12] | 0 | W | Overflow of buffer holding data from ISP_ch1_ispout2<br>1: The corresponding bit in the interrupt status register is cleared. |
| 11 | INT_CLEAR[11] | 0 | W | Overflow of buffer holding data from ISP_ch1_ispout1<br>1: The corresponding bit in the interrupt status register is cleared. |
| 10 | INT_CLEAR[10] | 0 | W | Overflow of buffer holding data from ISP_ch1_ispout0<br>1: The corresponding bit in the interrupt status register is cleared. |
| 9 | INT_CLEAR[9] | 0 | W | Overflow of buffer holding data from ISP_ch1_ispin1<br>1: The corresponding bit in the interrupt status register is cleared. |
| 8 | INT_CLEAR[8] | 0 | W | Overflow of buffer holding data from ISP_ch1_ispin0<br>1: The corresponding bit in the interrupt status register is cleared. |
| 7 | INT_CLEAR[7] | 0 | W | Overflow of buffer holding data from ISP_ch0_ispout3<br>1: The corresponding bit in the interrupt status register is cleared. |
| 6 | INT_CLEAR[6] | 0 | W | Overflow of buffer holding data from ISP_ch0_ispout2<br>1: The corresponding bit in the interrupt status register is cleared. |
| 5 | INT_CLEAR[5] | 0 | W | Overflow of buffer holding data from ISP_ch0_ispout1<br>1: The corresponding bit in the interrupt status register is cleared. |
| 4 | INT_CLEAR[4] | 0 | W | Overflow of buffer holding data from ISP_ch0_ispout0<br>1: The corresponding bit in the interrupt status register is cleared. |
| 3 | INT_CLEAR[3] | 0 | W | Overflow of buffer holding data from ISP_ch0_ispin1<br>1: The corresponding bit in the interrupt status register is cleared. |
| 2 | INT_CLEAR[2] | 0 | W | Overflow of buffer holding data from ISP_ch0_ispin0<br>1: The corresponding bit in the interrupt status register is cleared. |
| 1 | INT_CLEAR[1] | 0 | W | Overflow of buffer in the MTIRI1<br>1: The corresponding bit in the interrupt status register is cleared. |
| 0 | INT_CLEAR[0] | 0 | W | bresp error in the MTIRI1<br>1: The corresponding bit in the interrupt status register is cleared. |

### U20.2.12 MTIRI AXI Address Setting Register for sync (AWADDR_SET_S)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | AWADDR_SET_S [31:0] | H'2B00 0000 | R/W | AWADDR address for sync(VSYNC, HSYNC) |

### U20.2.13 MTIRI AXI Address Setting Register for data (AWADDR_SET_D)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | AWADDR_SET_D [31:0] | H'2B00 0018 | R/W | AWADDR address for data |

### U20.2.14 Reserved Register for MTIRI (MTIRI_REG_RESERVE)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R/W | Reserved |

### U20.2.15 Image Data Storage Size Setting Register 1 (MONI_LEN1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH0_MON2_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH0_MON1_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R/W | Reserved |
| 22 to 16 | CH0_MON2_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch0_ispin1<br>The amount of data stored in the RAM is the setting of these bits plus 1. |
| 15 to 7 | — | All 0 | R/W | Reserved |
| 6 to 0 | CH0_MON1_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch0_ispin0<br>The amount of data stored in the RAM is the setting of these bits plus 1. |

### U20.2.16 Image Data Storage Size Setting Register 2 (MONI_LEN2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH0_MON4_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH0_MON3_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R/W | Reserved |
| 22 to 16 | CH0_MON4_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch0_ispout1<br>The amount of data stored in the RAM is the setting of these bits plus 1. |
| 15 to 7 | — | All 0 | R/W | Reserved |
| 6 to 0 | CH0_MON3_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch0_ispout0<br>The amount of data stored in the RAM is the setting of these bits plus 1. |

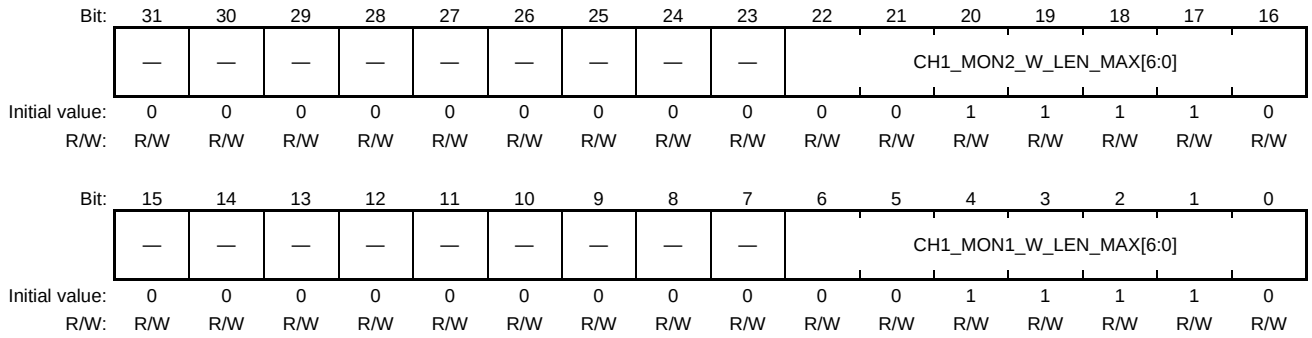
### U20.2.17 Image Data Storage Size Setting Register 3 (MONI_LEN3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH0_MON6_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH0_MON5_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

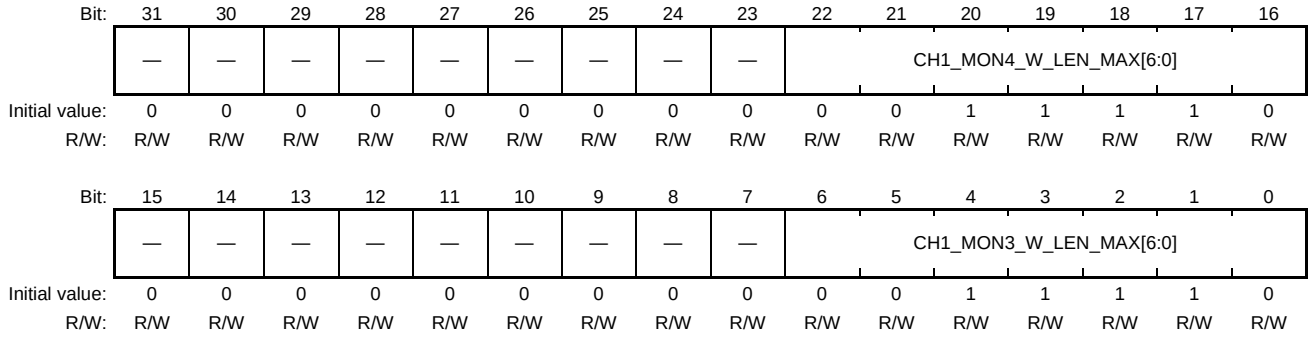
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R/W | Reserved |
| 22 to 16 | CH0_MON6_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch0_ispout3<br>The amount of data stored in the RAM is the setting of these bits plus 1. |
| 15 to 7 | — | All 0 | R/W | Reserved |
| 6 to 0 | CH0_MON5_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch0_ispout2<br>The amount of data stored in the RAM is the setting of these bits plus 1. |

### U20.2.18 Image Data Storage Size Setting Register 4 (MONI_LEN4)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R/W | Reserved |
| 22 to 16 | CH1_MON2_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch1_ispin1<br>The amount of data stored in the RAM is the setting of these bits plus 1. |
| 15 to 7 | — | All 0 | R/W | Reserved |
| 6 to 0 | CH1_MON1_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch1_ispin0<br>The amount of data stored in the RAM is the setting of these bits plus 1. |

### U20.2.19 Image Data Storage Size Setting Register 5 (MONI_LEN5)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R/W | Reserved |
| 22 to 16 | CH1_MON4_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch1_ispout1<br>The amount of data stored in the RAM is the setting of these bits plus 1. |
| 15 to 7 | — | All 0 | R/W | Reserved |
| 6 to 0 | CH1_MON3_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch1_ispout0<br>The amount of data stored in the RAM is the setting of these bits plus 1. |

### U20.2.20 Image Data Storage Size Setting Register 6 (MONI_LEN6)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH1_MON6_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH1_MON5_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R/W | Reserved |
| 22 to 16 | CH1_MON6_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch1_ispout3<br>The amount of data stored in the RAM is the setting of these bits plus 1. |
| 15 to 7 | — | All 0 | R/W | Reserved |
| 6 to 0 | CH1_MON5_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch1_ispout2<br>The amount of data stored in the RAM is the setting of these bits plus 1. |

### U20.2.21 Image Data Storage Size Setting Register 7 (MONI_LEN7)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH0_MON8_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH0_MON7_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R/W | Reserved |
| 22 to 16 | CH0_MON8_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch0_ispin3<br>The amount of data stored in the RAM is the setting of these bits plus 1. |
| 15 to 7 | — | All 0 | R/W | Reserved |
| 6 to 0 | CH0_MON7_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch0_ispin2<br>The amount of data stored in the RAM is the setting of these bits plus 1. |

### U20.2.22 Image Data Storage Size Setting Register 8 (MONI_LEN8)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH1_MON8_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | CH1_MON7_W_LEN_MAX[6:0] | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 23 | — | All 0 | R/W | Reserved |
| 22 to 16 | CH1_MON8_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch1_ispin3<br>The amount of data stored in the RAM is the setting of these bits plus 1. |
| 15 to 7 | — | All 0 | R/W | Reserved |
| 6 to 0 | CH1_MON7_W_LEN_MAX[6:0] | H'1E | R/W | Data Storage Size Setting for ISP_ch1_ispin2<br>The amount of data stored in the RAM is the setting of these bits plus 1. |

### U20.2.23 Signal Counter 1 (SIG_COUNT1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 1[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT1[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch0_ispin0 |

### U20.2.24 Signal Counter 2 (SIG_COUNT2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 2[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT2[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch0_ispin1 |

### U20.2.25 Signal Counter 3 (SIG_COUNT3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 3[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT3[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch0_ispout0 |

### U20.2.26 Signal Counter 4 (SIG_COUNT4)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 4[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT4[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch0_ispout1 |

### U20.2.27 Signal Counter 5 (SIG_COUNT5)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 5[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT5[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch0_ispout2 |

### U20.2.28 Signal Counter 6 (SIG_COUNT6)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 6[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT6[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch0_ispout3 |

### U20.2.29 Signal Counter 7 (SIG_COUNT7)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 7[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT7[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch1_ispin0 |

### U20.2.30 Signal Counter 8 (SIG_COUNT8)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 8[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT8[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch1_ispin1 |

### U20.2.31 Signal Counter 9 (SIG_COUNT9)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 9[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT9[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch1_ispout0 |

### U20.2.32 Signal Counter 10 (SIG_COUNT10)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 10[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT10[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch1_ispout1 |

### U20.2.33 Signal Counter 11 (SIG_COUNT11)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 11[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT11[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch1_ispout2 |

### U20.2.34 Signal Counter 12 (SIG_COUNT12)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 12[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT12[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch1_ispout3 |

### U20.2.35 Signal Counter 13 (SIG_COUNT13)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 13[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT13[15:0] | H'0000 | R | Total number of times data were transferred to the MTI block |

### U20.2.36 Signal Counter 14 (SIG_COUNT14)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 14[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT14[15:0] | H'0000 | R | Number of times the MTIRI1 received data |

### U20.2.37 Signal Counter 15 (SIG_COUNT15)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 15[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT15[15:0] | H'0000 | R | Number of times the MTIRI1 transmitted data |

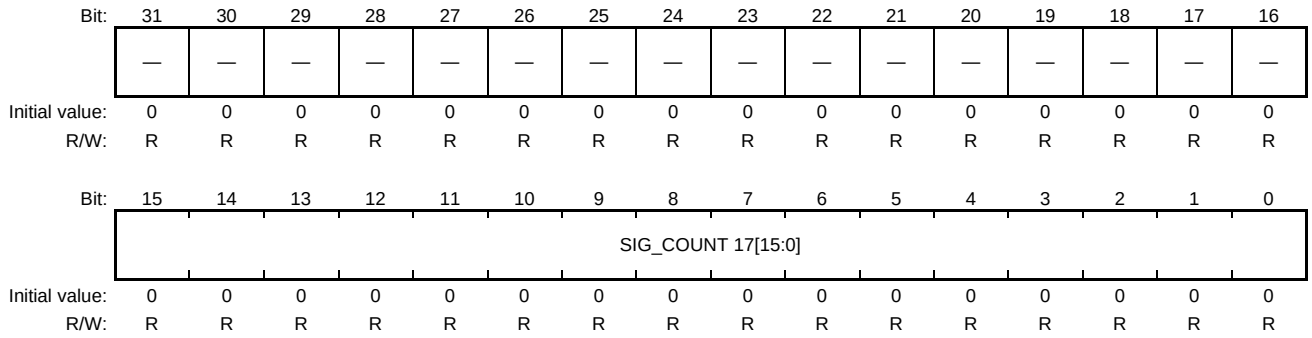
### U20.2.38 Signal Counter 16 (SIG_COUNT16)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

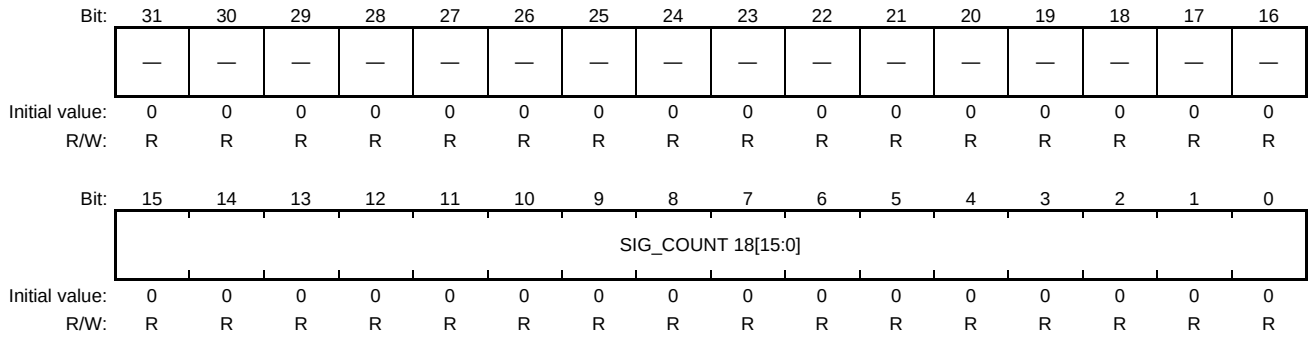
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R | Reserved |

### U20.2.39 Signal Counter 17 (SIG_COUNT17)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT17[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch0_ispin2 |

### U20.2.40 Signal Counter 18 (SIG_COUNT18)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT18[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch0_ispin3 |

### U20.2.41 Signal Counter 19 (SIG_COUNT19)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 19[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT19[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch1_ispin2 |

### U20.2.42 Signal Counter 20 (SIG_COUNT20)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SIG_COUNT 20[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved |
| 15 to 0 | SIG_COUNT20[15:0] | H'0000 | R | Number of times data were transferred from ISP_ch1_ispin3 |

### U20.2.43 Signal Counter hold (SIG_COUNT_HOLD)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | HOLD |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R/W | Reserved |
| 0 | HOLD | 0 | R/W | 0:read value SIG_COUNT1-15 register<br>1:hold value SIG_COUNT1-15 register |

### U20.2.44 Debugging Mode Register (DEBUG_MODE)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TVALID_FORCE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | OVF_FORCE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R/W | Reserved |
| 16 | TVALID_FORCE | 0 | R/W | 1: Sets the mti_tvalid1 signal to 1. |
| 15 to 1 | — | All 0 | R/W | Reserved |
| 0 | OVF_FORCE | 0 | R/W | 1: Sets the mti_common_ovf_err1 signal to 1. |

### U20.2.45 Debugging Mode Data 1 (DEBUG_DATA1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEBUG_DATA1 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEBUG_DATA1 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DEBUG_DATA1 | 0 | R/W | Sets the mti_tdata1[31:0] signal.<br>This setting is valid<br>when register DEBUG_MODE.TVALID_FORCE = 1. |

### U20.2.46 Debugging Mode Data 2 (DEBUG_DATA2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEBUG_DATA2 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEBUG_DATA2 | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DEBUG_DATA2 | 0 | R/W | Sets the mti_tdata1[63:32] signal.<br>This setting is valid<br>when register DEBUG_MODE.TVALID_FORCE = 1. |

### U20.2.47 Debugging Mode Addr (DEBUG_ADDR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | DEBUG_ADDR[7:0] | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R/W | Reserved |
| 7 to 0 | DEBUG_ADDR [7:0] | 0 | R/W | Sets the mti_taddr1[7:0] signal.<br>This setting is valid<br>when register DEBUG_MODE.TVALID_FORCE = 1. |

### U20.2.48 Lock Access Register (LOCKACCESS)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | | All 0 | W | |

### U20.2.49 Lock Status Register (LOCKSTATUS)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | | H'0000 0003 | R | |

### U20.2.50 Peripheral Identification Register 4 (Peripheral_ID4)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | | H'0000 0004 | R | |

### U20.2.51 Peripheral Identification Register 5 (Peripheral_ID5)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R | Reserved |

### U20.2.52 Peripheral Identification Register 6 (Peripheral_ID6)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R | Reserved |

### U20.2.53 Peripheral Identification Register 7 (Peripheral_ID7)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | — | All 0 | R | Reserved |

### U20.2.54 Peripheral Identification Register 0 (Peripheral_ID0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Part No.0 | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved |
| 7 to 0 | Part_No0 | H'17 | R | |

### U20.2.55 Peripheral Identification Register 1 (Peripheral_ID1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | JEP106 ID code[3:0] | | | | Part No.1 | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved |
| 7 to 4 | JEP106_ID_code[3:0] | H'3 | R | |
| 3 to 0 | Part_No1 | H'0 | R | |

### U20.2.56 Peripheral Identification Register 2 (Peripheral_ID2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Rev | | | | | JEP106 ID code[6:4] | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved |
| 7 to 4 | Rev | H'0 | R | |
| 3 | ID2_Bit3 | 1 | R | |
| 2 to 0 | JEP106_ID_code[6:4] | 010 | R | |

### U20.2.57 Peripheral Identification Register 3 (Peripheral_ID3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | RevAnd | | | | CustomerModified | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved |
| 7 to 4 | RevAnd | H'0 | R | |
| 3 to 0 | CustomerModified | H'0 | R | |

### U20.2.58 Component Identification Register 0 (Component_ID0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Preamble | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved |
| 7 to 0 | Preamble | H'0D | R | |

### U20.2.59 Component Identification Register 1 (Component_ID1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Component class | | | | Preamble | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved |
| 7 to 4 | Component_class | H'F | R | |
| 3 to 0 | Preamble | H'0 | R | |

### U20.2.60 Component Identification Register 2 (Component_ID2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Preamble | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved |
| 7 to 0 | Preamble | H'05 | R | |

### U20.2.61 Component Identification Register 3 (Component_ID3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Preamble | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 8 | — | All 0 | R | Reserved |
| 7 to 0 | Preamble | H'B1 | R | |

## U20.3 Operation

### U20.3.1 initial setting

In order to correctly output ISP's image data to LVDS or PCIe, it is necessary to set CSD, AUR, LVDS, PCIe and ISP as well as MTI settings. Please follow the setting order as follows.

(1) CSD,AUR,LVDS,PCIe setting

Refer to U10 HSSTP for the setting contents.

(2) MTI setting

See U20.5.2 MTI setting. If (3) ISP setting is done before this setting, monitoring of the image data of the ISP is started halfway, there is a possibility that the monitor data can not be generated correctly.

(3) ISP setting

Refer to 25B ISP for the setting contents.

### U20.3.2 MTI setting

In order to make MTI operate correctly, perform the setting in the following procedure.

(1) MTI standby release

MTI setting

| procedure | contents | Register [Address] | Setting value |
|---|---|---|---|
| 1 | Set Soft Reset | CPGWPR [H'E6150000] | Bit4 = 0 |
| | | SFTRST13[H'E6152C34] | Bit4 = 1 |
| 2 | Clear Module stop | CPGWPR [H'E6150000] | Bit4 = 1 |
| | | MSTPCR13 [H'E6152D34] | Bit4 = 0 |
| | | MSTPSR13 [H'E6152E34] | Bit4 = 0 |
| 3 | Clear Soft Reset | CPGWPR [H'E6150000] | Bit4 = 0 |
| | | SFTRSTCLR13 [H'E6152CB4] | Bit4= 1 |

MTI register access permission setting

After this setting is forgotten, subsequent register access can not be performed.

| procedure | contents | MTI Register | Setting value |
|---|---|---|---|
| 1 | Register access permission | LOCKACCESS | H'C5ACCE55 |

(2) Set target of image data collection

a) Specification target for ispin

Specify the collection target with ISPIN_MONI_MODE0 and 1 register.

| MTI Register | bit | Setting value | Contents |
|---|---|---|---|
| ISPIN_MONI_MODE0 | 1:0 | 0 | Target all bit(24bit) image data of ISP_ch0_ispin0. |
| | | 1 | Target lower 12bit image data of ISP_ch0_ispin0. |
| | | 2 | Target lower 20bit image data of ISP_ch0_ispin0. |
| | | 3 | Target lower 8bit image data of ISP_ch0_ispin0. |
| | 9:8 | 0 | Target all bit(24bit) image data of ISP_ch0_ispin1. |
| | | 1 | Target lower 12bit image data of ISP_ch0_ispin1. |
| | | 2 | Target lower 20bit image data of ISP_ch0_ispin1. |
| | | 3 | Target lower 8bit image data of ISP_ch0_ispin1. |
| ISPIN_MONI_MODE1 | 1:0 | 0 | Target all bit(24bit) image data of ISP_ch1_ispin0. |
| | | 1 | Target lower 12bit image data of ISP_ch1_ispin0. |
| | | 2 | Target lower 20bit image data of ISP_ch1_ispin0. |
| | | 3 | Target lower 8bit image data of ISP_ch1_ispin0. |
| | 9:8 | 0 | Target all bit(24bit) image data of ISP_ch1_ispin1. |
| | | 1 | Target lower 12bit image data of ISP_ch1_ispin1. |
| | | 2 | Target lower 20bit image data of ISP_ch1_ispin1. |
| | | 3 | Target lower 8bit image data of ISP_ch1_ispin1. |
| ISPIN_MONI_MODE2 | 1:0 | 0 | Target all bit(24bit) image data of ISP_ch2_ispin0. |
| | | 1 | Target lower 12bit image data of ISP_ch2_ispin0. |

| | | | |
|---|---|---|---|
| | | 2 | Target lower 20bit image data of ISP_ch2_ispin0. |
| | | 3 | Target lower 8bit image data of ISP_ch2_ispin0. |
| | 9:8 | 0 | Target all bit(24bit) image data of ISP_ch2_ispin1. |
| | | 1 | Target lower 12bit image data of ISP_ch2_ispin1. |
| | | 2 | Target lower 20bit image data of ISP_ch2_ispin1. |
| | | 3 | Target lower 8bit image data of ISP_ch2_ispin1. |
| ISPIN_MONI_MODE3 | 1:0 | 0 | Target all bit(24bit) image data of ISP_ch3_ispin0. |
| | | 1 | Target lower 12bit image data of ISP_ch3_ispin0. |
| | | 2 | Target lower 20bit image data of ISP_ch3_ispin0. |
| | | 3 | Target lower 8bit image data of ISP_ch3_ispin0. |
| | 9:8 | 0 | Target all bit(24bit) image data of ISP_ch3_ispin1. |
| | | 1 | Target lower 12bit image data of ISP_ch3_ispin1. |
| | | 2 | Target lower 20bit image data of ISP_ch3_ispin1. |
| | | 3 | Target lower 8bit image data of ISP_ch3_ispin1. |

b) Specification target for ispout

Specify the collection target with ISP_MONI_MODE0 and 1 register.

| MTI Register | bit | Setting value | Contents |
|---|---|---|---|
| ISP_MONI_MODE0 | 1:0 | 0 | Target all bit(24bit) image data of ISP_ch0_ispout0. |
| | | 1 | Target lower 12bit image data of ISP_ch0_ispout0. |
| | | 2 | Target lower 20bit image data of ISP_ch0_ispout0. |
| | | 3 | Target lower 8bit image data of ISP_ch0_ispout0. |
| | 9:8 | 0 | Target all bit(24bit) image data of ISP_ch0_ispout1. |
| | | 1 | Target lower 12bit image data of ISP_ch0_ispout1. |
| | | 2 | Target lower 20bit image data of ISP_ch0_ispout1. |
| | | 3 | Target lower 8bit image data of ISP_ch0_ispout1. |
| | 17:16 | 0 | Target all bit(24bit) image data of ISP_ch0_ispout2. |
| | | 1 | Target lower 12bit image data of ISP_ch0_ispout2. |
| | | 2 | Target lower 20bit image data of ISP_ch0_ispout2. |
| | | 3 | Target lower 8bit image data of ISP_ch0_ispout2. |
| | 25:24 | 0 | Target all bit(24bit) image data of ISP_ch0_ispout3. |
| | | 1 | Target lower 12bit image data of ISP_ch0_ispout3. |
| | | 2 | Target lower 20bit image data of ISP_ch0_ispout3. |
| | | 3 | Target lower 8bit image data of ISP_ch0_ispout3. |
| ISP_MONI_MODE1 | 1:0 | 0 | Target all bit(24bit) image data of ISP_ch1_ispout0. |
| | | 1 | Target lower 12bit image data of ISP_ch1_ispout0. |
| | | 2 | Target lower 20bit image data of ISP_ch1_ispout0. |
| | | 3 | Target lower 8bit image data of ISP_ch1_ispout0. |
| | 9:8 | 0 | Target all bit(24bit) image data of ISP_ch1_ispout1. |
| | | 1 | Target lower 12bit image data of ISP_ch1_ispout1. |
| | | 2 | Target lower 20bit image data of ISP_ch1_ispout1. |
| | | 3 | Target lower 8bit image data of ISP_ch1_ispout1. |

(3) Setting interrupt enable

To report the error interrupt detected by the MTI to the CPU, set the corresponding bit in the INT_ENABLE register to 1.

(4) Start MTI

MTI is started with the MONI_ENABLE0 and 1 register.
Set the bit to be activated to 1.

| MTI Register | bit | Setting value | Contents |
|---|---|---|---|
| MONI_MODE0 | 0 | 1 | Allow collecting image data of TAG number ”0” of ISP_ch0_ispin0. |
| | 1 | 1 | Allow collecting image data of TAG number ”1” of ISP_ch0_ispin0. |
| | 2 | 1 | Allow collecting image data of TAG number ”2” of ISP_ch0_ispin0. |
| | 3 | 1 | Allow collecting image data of TAG number ”3” of ISP_ch0_ispin0. |
| | 4 | 1 | Allow collecting image data of TAG number ”4” of ISP_ch0_ispin0. |
| | 5 | 1 | Allow collecting image data of TAG number ”5” of ISP_ch0_ispin0. |
| | 6 | 1 | Allow collecting image data of TAG number ”6” of ISP_ch0_ispin0. |
| | 7 | 1 | Allow collecting image data of TAG number ”7” of ISP_ch0_ispin0. |
| | 15:8 | 1 | Allow collecting image data of TAG number ”X” (x:0-7)of ISP_ch0_ispin1.<br>Bit assignment is the same as bit7:0. |
| | 16 | 1 | Allow collecting image data of ISP_ch0_ispout0. |
| | 17 | 1 | Allow collecting image data of ISP_ch0_ispout1. |
| | 18 | 1 | Allow collecting image data of ISP_ch0_ispout2. |
| | 19 | 1 | Allow collecting image data of ISP_ch0_ispout3. |
| MONI_MODE1 | 19:0 | 1 | Allow collecting image data of ISP_ch1_ispin0-1,ispout0-3.<br>Bit assignment is the same as MONI_MODE0 register. |

### U20.3.3 Data Format

The image data of the ISP is notified to the CSD via the AXI4 bus. Transfer format will be below.

#### U20.3.3.1 AWID

The format differs between ispin* and ispout*.

For ispin*, the TAG number is stored.
AXI4 issues an outstanding write request for each TAG number.

For ispout*, the ispout_no(0-3) is stored.
AXI4 issues an outstanding write request for each ispout_no.

### U20.3.3.2 AWADDR

| Contents of AWADDR | | |
|---|---|---|
| Bit[31:24] | Bit[23:17] | Bit[16:0] |
| (*2) | (*1) | (*2) |

(*1) When the content WDATA is image data of ispin

| Bit | Contents |
|---|---|
| 23 | ISP_ch number(0-1) |
| 22 | Fixed to 0 |
| 21:20 | ispin number (0-3) |
| 19:17 | tag number(0-7) |

When the content WDATA is image data of ispout

| Bit | Contents |
|---|---|
| 23 | ISP_ch number(0-1) |
| 22 | Fixed to 1 |
| 21:20 | ispout number (0-3) |
| 19:17 | tag number(0-7) |

(*2) When the content of WDATA is VSYNC or HSYNC, AWADDR_SET_S register
When the content of WDATA is image data, the AWADDR_SET_D register is reflected.

### U20.3.3 WDATA

(1) Transfer of VSYNC

a) When ispin is to be monitored, data is transferred under the following conditions.
At the rising edge of the VSYNC signal

[Transfer format]

| VSYNC | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | 1 | 1 | fmt[5:0] (H'00-3F) | | | | | |
| | 1-7 | Fixed to 0 | | | | | | | |

b) When ispout is to be monitored, data is transferred under the following conditions.
At the rising edge of the VSYNC signal

[Transfer format]

| VSYNC | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | 1 | 1 | Fixed to 0 | | | | | |
| | 1-7 | Fixed to 0 | | | | | | | |

(2) Transfer of HSYNC

a) When ispin is to be monitored, data is transferred under the following conditions.
At the rising edge of the HSYNC signal
However, in order to minimize data overhead,
data transfer at the rise of HSYNC following the rise of VSYNC is omitted.

[Transfer format]

| HSYNC | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | |
|---|---|---|---|---|
| Byte | 0 | 0 | 1 | fmt[5:0] (H'00-3F) |
| | 1-7 | Fixed to 0 | | |

b) When ispout is to be monitored, data is transferred under the following conditions.
At the rising edge of the HSYNC signal
However, in order to minimize data overhead,
data transfer at the rise of HSYNC following the rise of VSYNC is omitted.

[Transfer format]

| HSYNC | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | 0 | 1 | Fixed to 0 | | | | | |
| | 1-7 | Fixed to 0 | | | | | | | |

(3) Transfer of image data
0 is stored in the invalid data area.

a) When transferring image data of ispin to the lower 8bit

[Transfer format]

| ispin 8bit lsb | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | data[7:0] (cycle n) | | | | | | | |
| | 1 | data[7:0] (cycle n+1) | | | | | | | |
| | 2 | data[7:0] (cycle n+2) | | | | | | | |
| | 3 | data[7:0] (cycle n+3) | | | | | | | |
| | 4 | data[7:0] (cycle n+4) | | | | | | | |
| | 5 | data[7:0] (cycle n+5) | | | | | | | |
| | 6 | data[7:0] (cycle n+6) | | | | | | | |
| | 7 | data[7:0] (cycle n+7) | | | | | | | |

b) When transferring image data of ispin to the lower 12bit, three WDATA's become one set

[Transfer format]
First WDATA format

| ispin 12bit lsb | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | data[7:0] (cycle n) | | | | | | | |
| | 1 | data[3:0] (cycle n+1) | | | | data[11:8] (cycle n) | | | |
| | 2 | data[11:4] (cycle n+1) | | | | | | | |
| | 3 | data[7:0] (cycle n+2) | | | | | | | |
| | 4 | data[3:0] (cycle n+3) | | | | data[11:8] (cycle n+2) | | | |
| | 5 | data[11:4] (cycle n+3) | | | | | | | |
| | 6 | data[7:0] (cycle n+4) | | | | | | | |
| | 7 | data[3:0] (cycle n+5) | | | | data[11:8] (cycle n+4) | | | |

Second WDATA format

| ispin 12bit lsb | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | data[11:4] (cycle n+5) | | | | | | | |
| | 1 | data[7:0] (cycle n+6) | | | | | | | |

| | | | |
|---|---|---|---|
| | 2 | data[3:0] (cycle n+7) | data[11:8] (cycle n+6) |
| | 3 | data[11:4] (cycle n+7) | |
| | 4 | data[7:0] (cycle n+8) | |
| | 5 | data[3:0] (cycle n+9) | data[11:8] (cycle n+8) |
| | 6 | data[11:4] (cycle n+9) | |
| | 7 | data[7:0] (cycle n+10) | |

Third WDATA format

| ispin 12bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[3:0] (cycle n+11) | | | | data[11:8] (cycle n+10) | | | |
| | 1 | data[11:4] (cycle n+11) | | | | | | | |
| | 2 | data[7:0] (cycle n+12) | | | | | | | |
| | 3 | data[3:0] (cycle n+13) | | | | data[11:8] (cycle n+12) | | | |
| | 4 | data[11:4] (cycle n+13) | | | | | | | |
| | 5 | data[7:0] (cycle n+14) | | | | | | | |
| | 6 | data[3:0] (cycle n+15) | | | | data[11:8] (cycle n+14) | | | |
| | 7 | data[11:4] (cycle n+15) | | | | | | | |

c) When transferring image data of ispin to the lower 20bit, five WDATA's become one set

[Transfer format]

First WDATA format

| ispin 20bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[7:0] (cycle n) | | | | | | | |
| | 1 | data[15:8] (cycle n) | | | | | | | |
| | 2 | data[3:0] (cycle n+1) | | | | data[19:16] (cycle n) | | | |
| | 3 | data[11:4] (cycle n+1) | | | | | | | |
| | 4 | data[19:12] (cycle n+1) | | | | | | | |
| | 5 | data[7:0] (cycle n+2) | | | | | | | |
| | 6 | data[15:8] (cycle n+2) | | | | | | | |
| | 7 | data[3:0] (cycle n+3) | | | | data[19:16] (cycle n+2) | | | |

Second WDATA format

| ispin 20bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[11:4] (cycle n+3) | | | | | | | |
| | 1 | data[19:12] (cycle n+3) | | | | | | | |
| | 2 | data[7:0] (cycle n+4) | | | | | | | |
| | 3 | data[15:8] (cycle n+4) | | | | | | | |
| | 4 | data[3:0] (cycle n+5) | | | | data[19:16] (cycle n+4) | | | |
| | 5 | data[11:4] (cycle n+5) | | | | | | | |
| | 6 | data[19:12] (cycle n+5) | | | | | | | |
| | 7 | data[7:0] (cycle n+6) | | | | | | | |

Third WDATA format

| ispin 20bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[15:8] (cycle n+6) | | | | | | | |
| | 1 | data[3:0] (cycle n+7) | | | | data[19:16] (cycle n+6) | | | |
| | 2 | data[11:4] (cycle n+7) | | | | | | | |

| | | |
|---|---|---|
| | 3 | data[19:12] (cycle n+7) |
| | 4 | data[7:0] (cycle n+8) |
| | 5 | data[15:8] (cycle n+8) |
| | 6 | data[3:0] (cycle n+9) \| data[19:16] (cycle n+8) |
| | 7 | data[11:4] (cycle n+9) |

Fourth WDATA format

| ispin 20bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[19:12] (cycle n+9) | | | | | | | |
| | 1 | data[7:0] (cycle n+10) | | | | | | | |
| | 2 | data[15:8] (cycle n+10) | | | | | | | |
| | 3 | data[3:0] (cycle n+11) | | | | data[19:16] (cycle n+10) | | | |
| | 4 | data[11:4] (cycle n+11) | | | | | | | |
| | 5 | data[19:12] (cycle n+11) | | | | | | | |
| | 6 | data[7:0] (cycle n+12) | | | | | | | |
| | 7 | data[15:8] (cycle n+12) | | | | | | | |

Fifth WDATA format

| ispin 20bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[3:0] (cycle n+13) | | | | data[19:16] (cycle n+12) | | | |
| | 1 | data[11:4] (cycle n+13) | | | | | | | |
| | 2 | data[19:12] (cycle n+13) | | | | | | | |
| | 3 | data[7:0] (cycle n+14) | | | | | | | |
| | 4 | data[15:8] (cycle n+14) | | | | | | | |
| | 5 | data[3:0] (cycle n+15) | | | | data[19:16] (cycle n+14) | | | |
| | 6 | data[11:4] (cycle n+15) | | | | | | | |
| | 7 | data[19:12] (cycle n+15) | | | | | | | |

d) When transferring image data of ispin to the all bit(24bit), three WDATA's become one set

[Transfer format]
First WDATA format

| ispin 24bit | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[7:0] (cycle n) | | | | | | | |
| | 1 | data[15:8] (cycle n) | | | | | | | |
| | 2 | data[23:16] (cycle n) | | | | | | | |
| | 3 | data[7:0] (cycle n+1) | | | | | | | |
| | 4 | data[15:8] (cycle n+1) | | | | | | | |
| | 5 | data[23:16] (cycle n+1) | | | | | | | |
| | 6 | data[7:0] (cycle n+2) | | | | | | | |
| | 7 | data[15:8] (cycle n+2) | | | | | | | |

Second WDATA format

| ispin 24bit | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[23:16] (cycle n+2) | | | | | | | |
| | 1 | data[7:0] (cycle n+3) | | | | | | | |
| | 2 | data[15:8] (cycle n+3) | | | | | | | |

| | | |
|---|---|---|
| | 3 | data[23:16] (cycle n+3) |
| | 4 | data[7:0] (cycle n+4) |
| | 5 | data[15:8] (cycle n+4) |
| | 6 | data[23:16] (cycle n+4) |
| | 7 | data[7:0] (cycle n+5) |

Third WDATA format

| ispin 24bit | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | data[15:8] (cycle n+5) | | | | | | | |
| | 1 | data[23:16] (cycle n+5) | | | | | | | |
| | 2 | data[7:0] (cycle n+6) | | | | | | | |
| | 3 | data[15:8] (cycle n+6) | | | | | | | |
| | 4 | data[23:16] (cycle n+6) | | | | | | | |
| | 5 | data[7:0] (cycle n+7) | | | | | | | |
| | 6 | data[15:8] (cycle n+7) | | | | | | | |
| | 7 | data[23:16] (cycle n+7) | | | | | | | |

When transferring image data of ispout to the lower 8bit

[Transfer format]

| ispout 8bit lsb | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | data[7:0] (cycle n) | | | | | | | |
| | 1 | data[7:0] (cycle n+1) | | | | | | | |
| | 2 | data[7:0] (cycle n+2) | | | | | | | |
| | 3 | data[7:0] (cycle n+3) | | | | | | | |
| | 4 | data[7:0] (cycle n+4) | | | | | | | |
| | 5 | data[7:0] (cycle n+5) | | | | | | | |
| | 6 | data[7:0] (cycle n+6) | | | | | | | |
| | 7 | data[7:0] (cycle n+7) | | | | | | | |

e) When transferring image data of ispout to the upper 8bit

[Transfer format]

| ispout 8bit msb | | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte | 0 | data[11:4] (cycle n) | | | | | | | |
| | 1 | data[11:4] (cycle n+1) | | | | | | | |
| | 2 | data[11:4] (cycle n+2) | | | | | | | |
| | 3 | data[11:4] (cycle n+3) | | | | | | | |
| | 4 | data[11:4] (cycle n+4) | | | | | | | |
| | 5 | data[11:4] (cycle n+5) | | | | | | | |
| | 6 | data[11:4] (cycle n+6) | | | | | | | |
| | 7 | data[11:4] (cycle n+7) | | | | | | | |

f) When transferring image data of ispout to the all bit(12bit), three WDATA's become one set

[Transfer format]
First WDATA format

| ispout 12bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[7:0] (cycle n) | | | | | | | |
| | 1 | data[3:0] (cycle n+1) | | | | data[11:8] (cycle n) | | | |
| | 2 | data[11:4] (cycle n+1) | | | | | | | |
| | 3 | data[7:0] (cycle n+2) | | | | | | | |
| | 4 | data[3:0] (cycle n+3) | | | | data[11:8] (cycle n+2) | | | |
| | 5 | data[11:4] (cycle n+3) | | | | | | | |
| | 6 | data[7:0] (cycle n+4) | | | | | | | |
| | 7 | data[3:0] (cycle n+5) | | | | data[11:8] (cycle n+4) | | | |

Second WDATA format

| ispout 12bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[11:4] (cycle n+5) | | | | | | | |
| | 1 | data[7:0] (cycle n+6) | | | | | | | |
| | 2 | data[3:0] (cycle n+7) | | | | data[11:8] (cycle n+6) | | | |
| | 3 | data[11:4] (cycle n+7) | | | | | | | |
| | 4 | data[7:0] (cycle n+8) | | | | | | | |
| | 5 | data[3:0] (cycle n+9) | | | | data[11:8] (cycle n+8) | | | |
| | 6 | data[11:4] (cycle n+9) | | | | | | | |
| | 7 | data[7:0] (cycle n+10) | | | | | | | |

Third WDATA format

| ispout 12bit lsb | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | data[3:0] (cycle n+11) | | | | data[11:8] (cycle n+10) | | | |
| | 1 | data[11:4] (cycle n+11) | | | | | | | |
| | 2 | data[7:0] (cycle n+12) | | | | | | | |
| | 3 | data[3:0] (cycle n+13) | | | | data[11:8] (cycle n+12) | | | |
| | 4 | data[11:4] (cycle n+13) | | | | | | | |
| | 5 | data[7:0] (cycle n+14) | | | | | | | |
| | 6 | data[3:0] (cycle n+15) | | | | data[11:8] (cycle n+14) | | | |
| | 7 | data[11:4] (cycle n+15) | | | | | | | |

## U20.4 Usage Notes

(1) Restriction on the number of collected channels
Monitoring of multiple image data can be specified by setting the MONI_ENABLE register.
If you do not use it within the band limit of the transfer path,
overflow will occur and correct data transfer will not be possible

**When the output destination is PCIe**

**Bandwidth limit formula =**
**[(Amount of data per channel[bit]×ch number)÷(1000/Pixel clock[MHz])÷8] ≦ 1.0GB/s**

[Example of combination]

| Pixel clock [MHz] | ispin/ispout | Data width [bit] | Ch number | Used bandwidth [GB/s] |
|---|---|---|---|---|
| 166.7 | ispin | 24 | 1 | 0.5 |
| 166.7 | ispout | 8 | 3 | 0.5 |
| | | | total | 1.0 |

| Pixel clock [MHz] | ispin/ispout | Data width [bit] | Ch number | Used bandwidth [GB/s] |
|---|---|---|---|---|
| 166.7 | ispin | 12 | 2 | 0.5 |
| 166.7 | ispout | 12 | 2 | 0.5 |
| | | | total | 1.0 |

| Pixel clock [MHz] | ispin/ispout | Data width [bit] | Ch number | Used bandwidth [GB/s] |
|---|---|---|---|---|
| 166.7 | ispin | 8 | 2 | 0.5 |
| 166.7 | ispout | 8 | 4 | 0.5 |
| | | | total | 1.0 |

## U20.5 Safety Mechanisms

Not supported any safety mechanisms

# U25. Vision-IP

## U25.1 Overview

The Vision-IP captures image data from the DDR-SDRAM and processes images. As image processing algorithms, it supports optical flow (optical flow vectors generation from a pair of images), and SfM engine (Structure from motion).

It's sometimes indicated on this document by another name, so please paraphrase.

- DOF or UMF : Dense type of Optical Flow vectors generation.
- SMD-PS/SMD-POST or SMPS/SMPO : Structure from motion engines.

### U25.1.1 Features

The Vision-IP which includes the DOF and SMD-* cores, and DMAC engines integrated in each core which can accelerate as an accelerator for image processing at high speed. These cores realize computer vision algorithm such as optical flow, structure from motion with co-operation with software.

**(1) DOF core**

The DOF core is inherently running through a sequence of three elementary generations of flow vectors; each one is matching a pyramid level. The input for each pyramid level is not built inside the DOF core and is assumed to be prepared by another unit in the system and ready in system memory once DOF core needs them. The detail information is described in licensee guide.

**(2) SMD-* core**

The SMD cores are a hardware unit to support structure from motion. The detail information is described in licensee guide.

**(3) Wrapper function**

All wrapper modules have several functions such as AXI-BUS bridge, software reset function and so on. Only UMF wrapper has a format conversion function (OTFDCR) that is converted the format of input image data on the fly. In addition, there is a register block to control functions in the wrapper. The description of this register in the wrapper is specify in Register Description section in this document. On the other hand, the register description in IP cores are in each licensee guide.

### U25.1.2 Block diagram

Figure U25.1 shows the block diagram of UMF module. UMF module have some kind of function in the wrapper module such as software reset control (rst_ctl/is_ctl), on the fly data format converter (OTFDCR), AXI-BUS bridge which is converting bus width from 64bits to 256bits (X2X), and register block in order to control each function in “umf_wrap”. And also, there is “fcprm” which is data compression function in “UMF_TOP”.

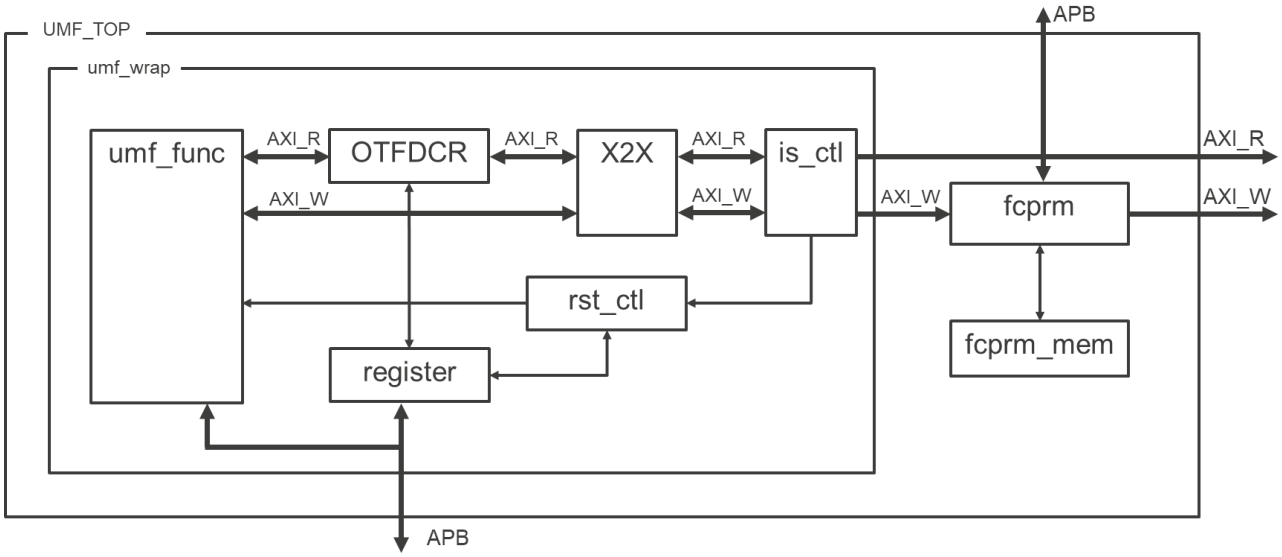



**Figure U25.1 UMF block diagram**

Figure U25.2 shows the block diagram of SMD_PS and SMD_POST module. These module have some kind of function in the wrapper module such as software reset control (rst_ctl/is_ctl), AXI-BUS bridge which is converting bus width from 64bits to 256bits (X2X), and register block in order to control each function.

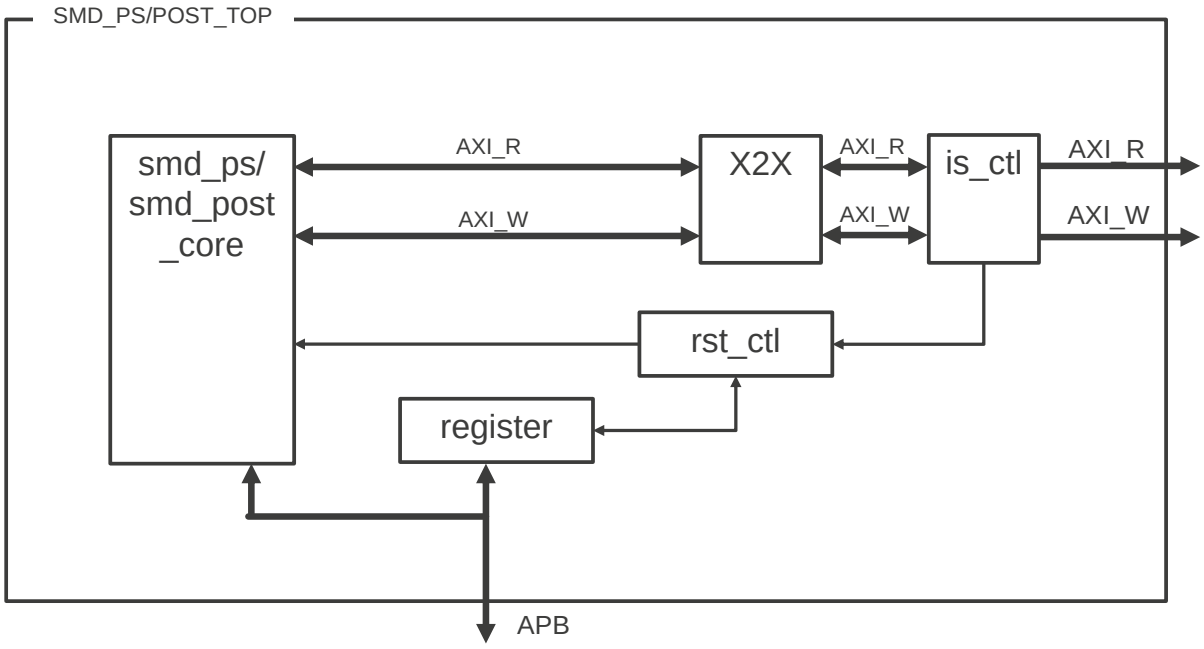



**Figure U25.2 SMD_PS and SMD_POST block diagram**

### U25.1.3 Specifications of the Vision-IP by Products

**Table U25.1** Specifications of the Vision-IP

| | AXI master | APB slave | DOF core | SMD-PS core | SMD-POST core |
|---|---|---|---|---|---|
| **Bus Width** | 256bits (*) | 32bits | - | - | - |
| **Clock** | SVD1_VIPϕ | SVD2_VIPϕ | SVD1_VIPϕ | SVD2_VIPϕ | SVD2_VIPϕ |
| **Number of cores** | 1 for each core | 1 for each core | 1 | 1 | 1 |

(*) DMAC engine's AXI interface is 64 bits, but it connects with external BUS by 256bits AXI interface.

### U25.1.4 Register Configuration

| Module | Wrap register base Address |
|---|---|
| UMF wrap | H'00 E7B3 B000 |
| SMD_PS wrap | H'00 E7B3 B400 |
| SMD_POST wrap | H'00 E7B3 BC00 |

**Table U25.2** Register configuration in the wrapper

| Name | Abbreviation | R/W | Address | Access Size | Module (*) |
|---|---|---|---|---|---|
| Reset control register | RST_CTRL | R/W | Base Address + H'00 | 32 | All |
| Software reset status register | SWRST_STA | R | Base Address + H'04 | 32 | All |
| Channel control register1 | CH_CTRL1 | R/W | Base Address + H'40 | 32 | UMF |
| Input picture horizontal size register1 | SRC_HSIZE1 | R/W | Base Address + H'44 | 32 | UMF |
| Input picture vertical size register1 | SRC_VSIZE1 | R/W | Base Address + H'48 | 32 | UMF |
| Input picture format register1 | SRC_FMT1 | R/W | Base Address + H'4C | 32 | UMF |
| Input picture address register1 | SRC_ADR1 | R/W | Base Address + H'50 | 32 | UMF |
| Input picture stride register1 | SRC_PSTRIDE1 | R/W | Base Address + H'54 | 32 | UMF |
| Output picture format register1 | DST_FMT1 | R/W | Base Address + H'58 | 32 | UMF |
| Output padding pixel value register1 | PAD_VALUE1 | R/W | Base Address + H'5C | 32 | UMF |
| Channel control register2 | CH_CTRL2 | R/W | Base Address + H'70 | 32 | UMF |
| Input picture horizontal size register2 | SRC_HSIZE2 | R/W | Base Address + H'74 | 32 | UMF |
| Input picture vertical size register2 | SRC_VSIZE2 | R/W | Base Address + H'78 | 32 | UMF |
| Input picture format register2 | SRC_FMT2 | R/W | Base Address + H'7C | 32 | UMF |
| Input picture address register2 | SRC_ADR2 | R/W | Base Address + H'80 | 32 | UMF |
| Input picture stride register2 | SRC_PSTRIDE2 | R/W | Base Address + H'84 | 32 | UMF |
| Output picture format register2 | DST_FMT2 | R/W | Base Address + H'88 | 32 | UMF |

| Name | Abbreviation | R/W | Address | Access Size | Module (*) |
|---|---|---|---|---|---|
| Output padding pixel value register2 | PAD_VALUE2 | R/W | Base Address + H'8C | 32 | UMF |
| Channel control register3 | CH_CTRL3 | R/W | Base Address + H'A0 | 32 | UMF |
| Input picture horizontal size register3 | SRC_HSIZE3 | R/W | Base Address + H'A4 | 32 | UMF |
| Input picture vertical size register3 | SRC_VSIZE3 | R/W | Base Address + H'A8 | 32 | UMF |
| Input picture format register3 | SRC_FMT3 | R/W | Base Address + H'AC | 32 | UMF |
| Input picture address register3 | SRC_ADR3 | R/W | Base Address + H'B0 | 32 | UMF |
| Input picture stride register3 | SRC_PSTRIDE3 | R/W | Base Address + H'B4 | 32 | UMF |
| Output picture format register3 | DST_FMT3 | R/W | Base Address + H'B8 | 32 | UMF |
| Output padding pixel value register3 | PAD_VALUE3 | R/W | Base Address + H'BC | 32 | UMF |

(*) ALL: UMF, SMD_PS and SMD_POST have these registers. UMF: Only UMF has these registers.

### U25.1.5 Connected Module

**Table U25.3** Vision-IP connected Module

| Module | Connected module | Description |
|---|---|---|
| UMF (DOF) | AXI-BUS | Address map (core) to H'00 E7A1 0000 to H'00 E7A1 FFFF (*) |
| SMD-PS | AXI-BUS | Address map (core) to H'00 E7A2 0000 to H'00 E7A2 FFFF (*) |
| SMD-POST | AXI-BUS | Address map (core) to H'00 E7A4 0000 to H'00 E7A4 FFFF (*) |
| UMF (DOF) | Reset | S/W reset signals comes from CPG, for details refer to register description in Module Standby Software Reset section. |
| SMD-PS | Reset | S/W reset signals comes from CPG, for details refer to register description in Module Standby Software Reset section. |
| SMD-POST | Reset | S/W reset signals comes from CPG, for details refer to register description in Module Standby Software Reset section. |
| UMF (DOF) | Clock | Clock to SVD1_VIPϕ<br>Clock to SVD2_VIPϕ |
| SMD-PS | Clock | Clock to SVD1_VIP ϕ<br>Clock to SVD2_VIP ϕ |
| SMD-POST | Clock | Clock to SVD1_VIP ϕ<br>Clock to SVD2_VIP ϕ |
| UMF (DOF) | Interrupt | Status interrupt signals connect to INTC module and Error interrupt signal connect to ECM module, for details refer to INTC and ECM section. |
| SMD-PS | Interrupt | Status interrupt signals connect to INTC module and Error interrupt signal connect to ECM module, for details refer to INTC and ECM section. |
| SMD-POST | Interrupt | Status interrupt signals connect to INTC module and Error interrupt signal connect to ECM module, for details refer to INTC and ECM section. |
| UMF (DOF) | IPMMU | uTLB information is described in IPMMU section. |
| SMD-PS | IPMMU | uTLB information is described in IPMMU section. |
| SMD-POST | IPMMU | uTLB information is described in IPMMU section. |

(*) Note: Do not access undefined register.

## U25.2 Register Description

### U25.2.1 Reset control register (RST_CTRL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWRST_EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | SWRST_EN | 0 | R/W | Software reset enable<br>This is the enable bit of software reset for AXI-BUS bridges, OTFDCR in Wrapper, and the AXI interface of IP cores.<br>Write 1 to this bit to enable software reset, right after being set to 1, this bit will be autonomously clear to 0. The read value should be 0.<br>Note: The usage of this register shall be limited as same as IP errata: IP SW resets usage forbidden in product. Some sub-modules in the IP provide a SW reset: IP Core, AXI_DMA_AEI, CSC, AEI_MUX, PCHK.When one of these sub-modules SW reset is asserted, its interface may become unstable. So this software reset is only for AXI-BUS bridges, OTFDCR in Wrapper, and the AXI interface of IP cores |

### U25.2.2 Software reset status register (SWRST_STA)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWRST_STA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | SWRST_STA | 0 | R | Software reset status<br>This bit shows the status of software reset, after SWRST_EN is set to 1, this bit is immediately set to 1. After the end of software reset sequence, this status bit returns to 0.<br>0: Software reset procedure has been completed.<br>1: Software reset procedure has not been competed yet. |

### U25.2.3 Channel control register1 (CH_CTRL1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ACT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | ACT | 0 | R/W | Channel activation.<br>0: Inactive, transaction's address belongs to inactive channel will be assigned to "Bypass" channel.<br>1: Channel activated. |

### U25.2.4 Input picture horizontal size register1 (SRC_HSIZE1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | HSIZE[26:16] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSIZE[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 26 to 0 | HSIZE | 0 | R/W | Input picture Horizontal size<br>Value range: 1~8191 pixels (2D) or 1~8192*8192 (1D) |

### U25.2.5 Input picture vertical size register1 (SRC_VSIZE1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | VSIZE[13:0] | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 0 | VSIZE | 0 | R/W | Input picture vertical size<br>Value range: 1~8192 pixels (2D) or 0 (1D) |

### U25.2.6 Input picture format register1 (SRC_FMT1)

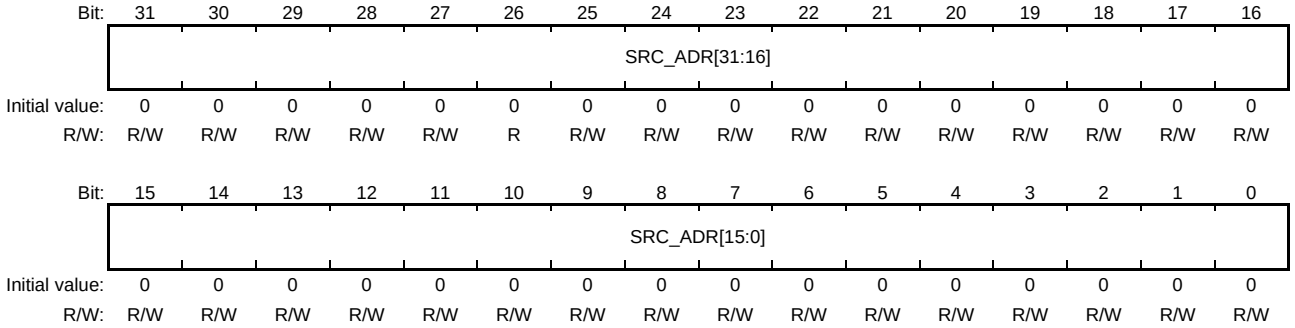
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSBOFS | | | | LSBOFS | | | | — | — | — | BPP | — | — | — | TBPPCE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 12 | MSBOFS | 0 | R/W | MSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 11 to 8 | LSBOFS | 0 | R/W | LSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | BPP | 0 | R/W | Byte per pixels<br>0: 2 Bytes/pixel; 1: 1 Byte/pixel |
| 3 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

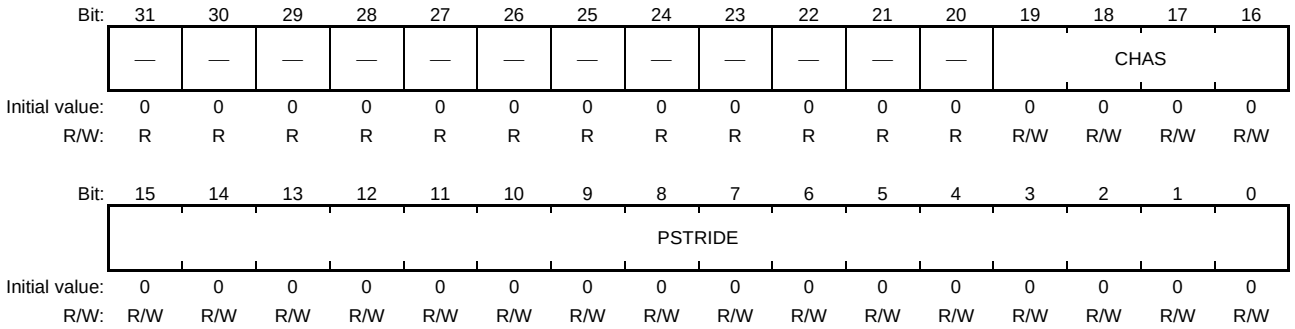
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | TBPPCE | 0 | R/W | 2BPP conversion enabled<br>0: Disabled; 1: Enabled |

### U25.2.7 Input picture address register1 (SRC_ADR1)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRC_ADR | 0 | R/W | Source picture address (always align 8Bytes)<br>Bit [31:16] and SRC_PSTRIDE{n}.CHAS are used to identify channel ID |

### U25.2.8 Input picture stride register1 (SRC_PSTRIDE1)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 16 | CHAS | 0 | R/W | Channel address space<br>Specify how many Least Significant Bit of SRC_ADR are used<br>Source picture address range = SRC_ADR[31-(CHAS+1):0]<br>when D'2 <= CHAS <= D'15<br>Source picture address range = SRC_ADR[29:0]<br>when D'0 <= CHAS <= D'1 |
| 15 to 0 | PSTRIDE | 0 | R/W | Source picture memory stride (always align 8Bytes) |

### U25.2.9 Output picture format register1 (DST_FMT1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSBOFS | | | | LSBOFS | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 12 | MSBOFS | 0 | R/W | MSB offset bit.<br>Range: 0~15. User shall fix 0 in 1 Byte/pixel |
| 11 to 8 | LSBOFS | 0 | R/W | LSB offset bit.<br>Range: 0~15. User shall fix 0 in 1 Byte/pixel |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### U25.2.10 Output padding pixel value register1 (PAD_VALUE1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSB_PAD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LSB_PAD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | MSB_PAD | 0 | R/W | Padding value for MSB. Example if MSBOFS is 4, MSB_PAD[3:0] are filled to 4 MSB output pixel. |
| 15 to 0 | LSB_PAD | 0 | R/W | Padding value for LSB. Example if LSBOFS is 4, LSB_PAD[3:0] are filled to 4 LSB output pixel. |

### U25.2.11 Channel control register2 (CH_CTRL2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ACT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | ACT | 0 | R/W | Channel activation.<br>0: Inactive, transaction's address belongs to inactive channel will be assigned to "Bypass" channel.<br>1: Channel activated. |

### U25.2.12 Input picture horizontal size register2 (SRC_HSIZE2)

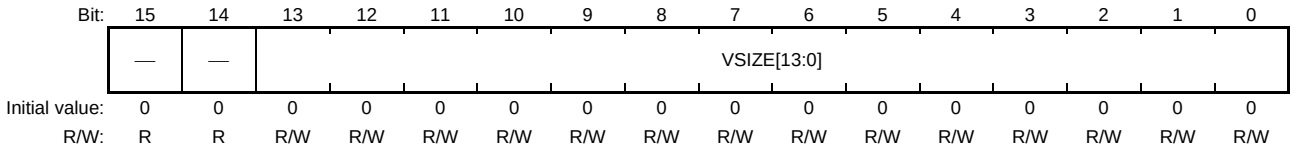
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | HSIZE[26:16] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSIZE[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 26 to 0 | HSIZE | 0 | R/W | Input picture Horizontal size<br>Value range: 1~8191 pixels (2D) or 1~8192*8192 (1D) |

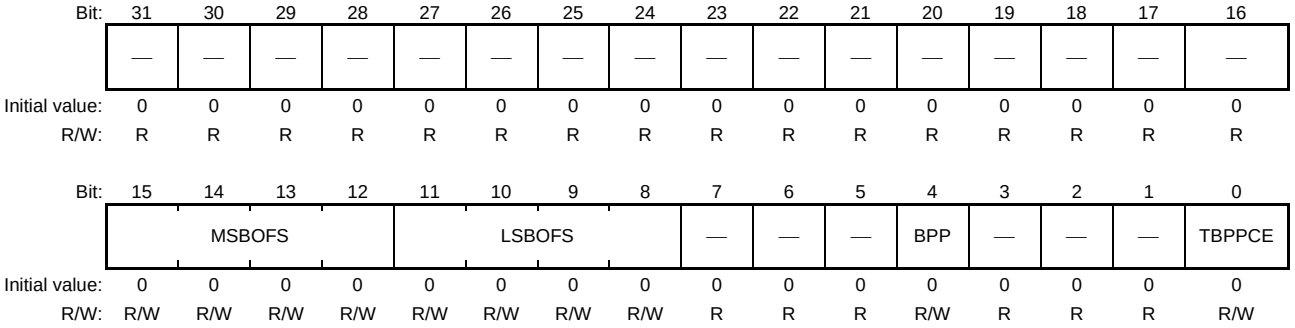
### U25.2.13 Input picture vertical size register2 (SRC_VSIZE2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 0 | VSIZE | 0 | R/W | Input picture vertical size<br>Value range: 1~8192 pixels (2D) or 0 (1D) |

### U25.2.14 Input picture format register2 (SRC_FMT2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 12 | MSBOFS | 0 | R/W | MSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 11 to 8 | LSBOFS | 0 | R/W | LSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | BPP | 0 | R/W | Byte per pixels<br>0: 2 Bytes/pixel; 1: 1 Byte/pixel |
| 3 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | TBPPCE | 0 | R/W | 2BPP conversion enabled<br>0: Disabled; 1: Enabled |

### U25.2.15 Input picture address register2 (SRC_ADR2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRC_ADR[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SRC_ADR[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRC_ADR | 0 | R/W | Source picture address (always align 8Bytes)<br>Bit [31:16] and SRC_PSTRIDE{n}.CHAS are used to identify channel ID |

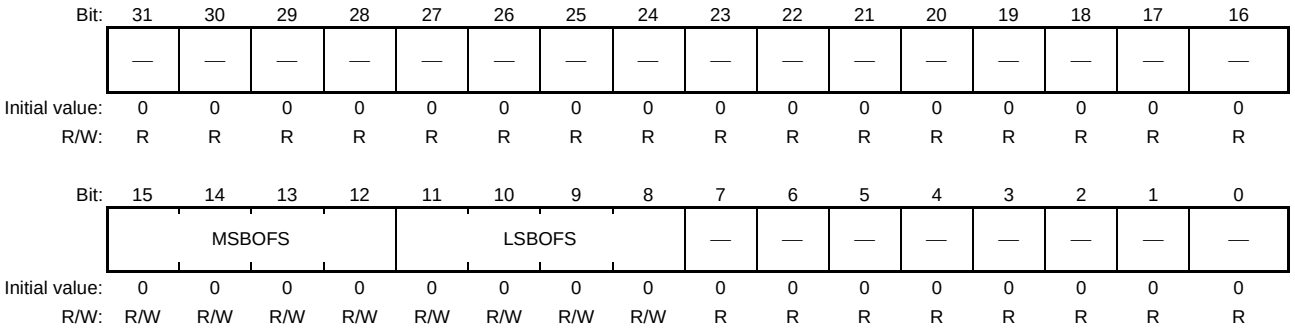
### U25.2.16 Input picture stride register2 (SRC_PSTRIDE2)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | CHAS | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSTRIDE | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

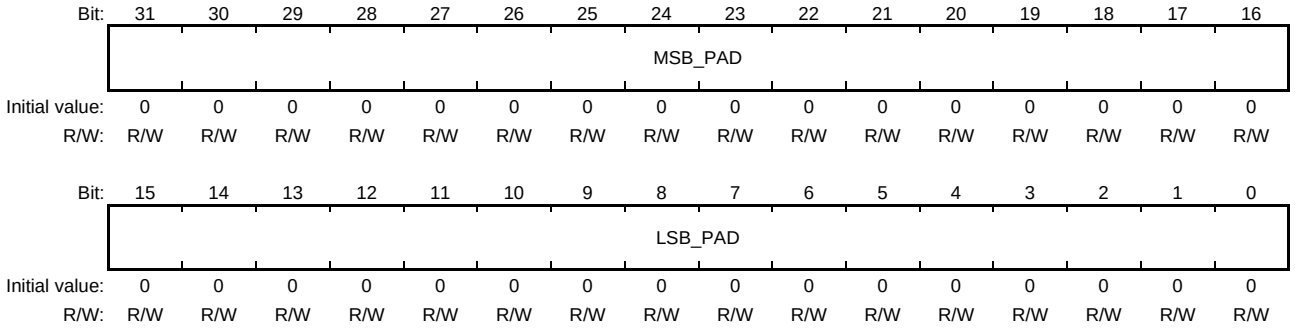
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 16 | CHAS | 0 | R/W | Channel address space<br>Specify how many Least Significant Bit of SRC_ADR are used<br>Source picture address range = SRC_ADR[31-(CHAS+1):0]<br>when D'2 <= CHAS <= D'15<br>Source picture address range = SRC_ADR[29:0]<br>when D'0 <= CHAS <= D'1 |
| 15 to 0 | PSTRIDE | 0 | R/W | Source picture memory stride (always align 8Bytes) |

### U25.2.17 Output picture format register2 (DST_FMT2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 12 | MSBOFS | 0 | R/W | LSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 11 to 8 | LSBOFS | 0 | R/W | MSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### U25.2.18 Output padding pixel value register2 (PAD_VALUE2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | MSB_PAD | 0 | R/W | Padding value for MSB. Example if MSBOFS is 4, MSB_PAD[3:0] are filled to 4 MSB output pixel. |
| 15 to 0 | LSB_PAD | 0 | R/W | Padding value for LSB. Example if LSBOFS is 4, LSB_PAD[3:0] are filled to 4 LSB output pixel. |

### U25.2.19 Channel control register3 (CH_CTRL3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | ACT |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 0 | ACT | 0 | R/W | Channel activation.<br>0: Inactive, transaction's address belongs to inactive channel will be assigned to "Bypass" channel.<br>1: Channel activated. |

### U25.2.20 Input picture horizontal size register3 (SRC_HSIZE3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | HSIZE[26:16] | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSIZE[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 26 to 0 | HSIZE | 0 | R/W | Input picture Horizontal size<br>Value range: 1~8191 pixels (2D) or 1~8192*8192 (1D) |

### U25.2.21 Input picture vertical size register3 (SRC_VSIZE3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | VSIZE[13:0] | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 13 to 0 | VSIZE | 0 | R/W | Input picture vertical size<br>Value range: 1~8192 pixels (2D) or 0 (1D) |

### U25.2.22 Input picture format register3 (SRC_FMT3)

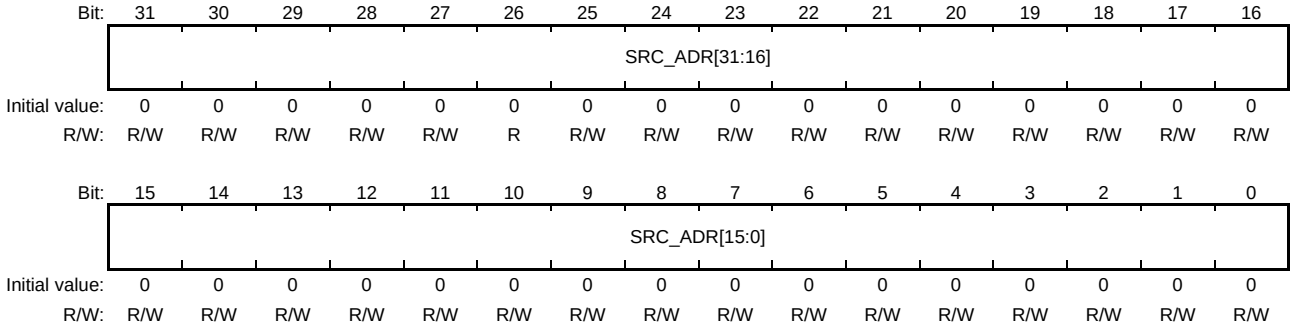
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSBOFS | | | | LSBOFS | | | | — | — | — | BPP | — | — | — | TBPPCE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 12 | MSBOFS | 0 | R/W | MSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 11 to 8 | LSBOFS | 0 | R/W | LSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 4 | BPP | 0 | R/W | Byte per pixels<br>0: 2 Bytes/pixel; 1: 1 Byte/pixel |
| 3 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

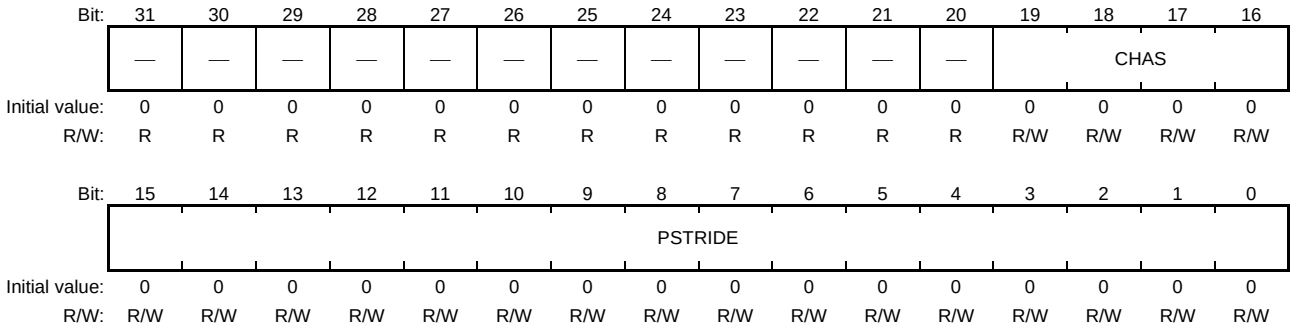
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | TBPPCE | 0 | R/W | 2BPP conversion enabled<br>0: Disabled; 1: Enabled |

### U25.2.23 Input picture address register3 (SRC_ADR3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SRC_ADR | 0 | R/W | Source picture address (always align 8Bytes)<br>Bit [31:16] and SRC_PSTRIDE{n}.CHAS are used to identify channel ID |

### U25.2.24 Input picture stride register3 (SRC_PSTRIDE3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 19 to 16 | CHAS | 0 | R/W | Channel address space<br>Specify how many Least Significant Bit of SRC_ADR are used<br>Source picture address range = SRC_ADR[31-(CHAS+1):0]<br>when D'2 <= CHAS <= D'15<br>Source picture address range = SRC_ADR[29:0]<br>when D'0 <= CHAS <= D'1 |
| 15 to 0 | PSTRIDE | 0 | R/W | Source picture memory stride (always align 8Bytes) |

### U25.2.25 Output picture format register3 (DST_FMT3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSBOFS | | | | LSBOFS | | | | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |
| 15 to 12 | MSBOFS | 0 | R/W | MSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 11 to 8 | LSBOFS | 0 | R/W | LSB offset bit.<br>Range: 0~15. It shall be fixed 0 in 1 Byte/pixel |
| 7 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0. The write value should always be 0. |

### U25.2.26 Output padding pixel value register3 (PAD_VALUE3)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSB_PAD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LSB_PAD | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | MSB_PAD | 0 | R/W | Padding value for MSB. Example if MSBOFS is 4, MSB_PAD[3:0] are filled to 4 MSB output pixel. |
| 15 to 0 | LSB_PAD | 0 | R/W | Padding value for LSB. Example if LSBOFS is 4, LSB_PAD[3:0] are filled to 4 LSB output pixel. |

## U25.3 Register relationship

Because OTFDCR function in UMF wrapper works in cooperation with UMF DMA function, the settings of OTFDCR shall match the some UMF DMA settings. The below table shows the relationship with each register.

**Table U25.4** Register relationship with UMF core and OTFDCR

| DMA normal entry | OTFDCR Register | Note |
|---|---|---|
| DMA Address | SRC_ADR | Always 8 bytes align |
| XSize | roundup_8( (SRC_ADR & 0x7) + SRC_HSIZE*BPP) | BPP = (SRC_FMT.BPP == 0) ? 2 : 1<br>roundup_8 : round up a number to nearest multiple of 8 |
| XOffset | SRC_PSTRIDE.PSTRIDE | Always 8 bytes align |
| YSize | SRC_VSIZE | – |

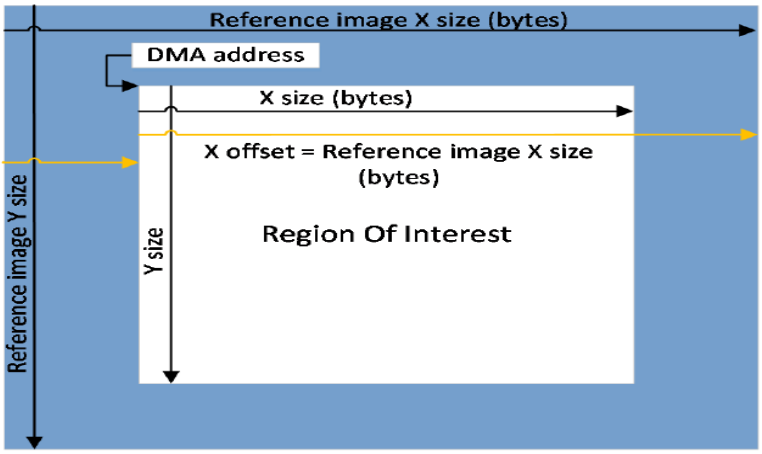



**Figure U25.3 2D transfer regarding a 2D image definition**

## U25.4 Operation

- Detail information is described in IP licensee guide.

## U25.5 Usage Notes

### U25.5.1 Module Standby and power-shutoff

- Enter Module Standby (stop clock supply) by CPG
- Enter power-shutoff status (stop power supply) by SYSC

### U25.5.2 Usage notes for IP core

- Detail information is described in IP licensee guide.

### U25.5.3 Usage notes for OTFDCR

#### U25.5.3.1 Pixel conversion setting guide

Set according to the following register setting guide.

- For 16bit/pixel input (conversion from 12bit/pixel or 14bit/pixel to 16bit/pixel), set CH_CTRL{n}.ACT=1 and set the following.

  SRC_FMT{n}.TBPPCE = 1

- For 8bit/pixel input (conversion from 8bit/pixel to 16bit/pixel), set CH_CTRL{n}.ACT=1 and set the following.

  SRC_FMT{n}.MSBOFS = 0

  DST_FMT{n}.MSBOFS = 8

  SRC_FMT{n}.LSBOFS = 0

  DST_FMT{n}.LSBOFS = 0

- When padding for 8bit/pixel input, set the value to be stored in the lower 8 bits in the following registers.

  PAD_VALUE{n}.MSB_PAD[15:8]

Registers can be set independently for each channel.

#### U25.5.3.2 Channel address space usage notes

Because OTFDCR uses SRC_PSTRIDE{n}.CHAS and 16 high-order bits of source address to identify channel's ID and address space, following conditions shall be followed:

- 64K-Byte align of source image is necessary to use multiple channels.
- Source picture address range = SRC_ADR[31-(CHAS+1):0] must not overlap and evaluation priority. If you use overlapping of masked area in CHAS, image that has bigger mask should be set to smaller index channel register.

## U25.6 Safety Mechanism

### U25.6.1 Overview

All memories in Vision IP are protected by EDC or parity.

### U25.6.2 Hardware Description

#### U25.6.2.1 Block Diagrams

Figure U25.4 shows a block diagram of the UMF protection for functional safety.

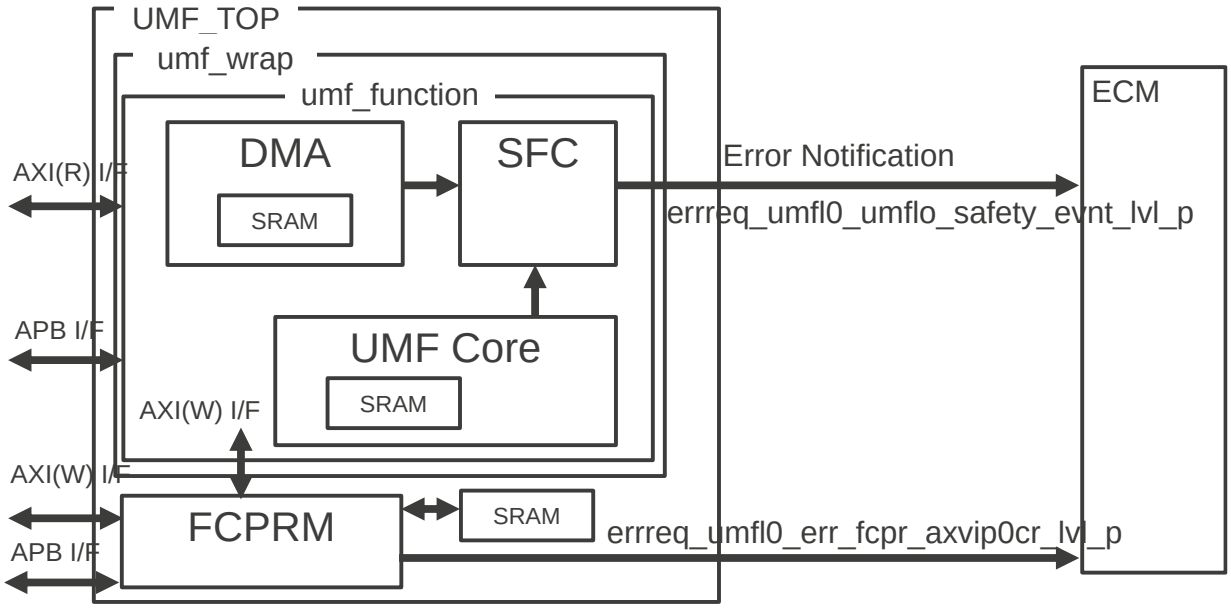



**Figure U25.4 EDC check for Vision IP (UMF)**

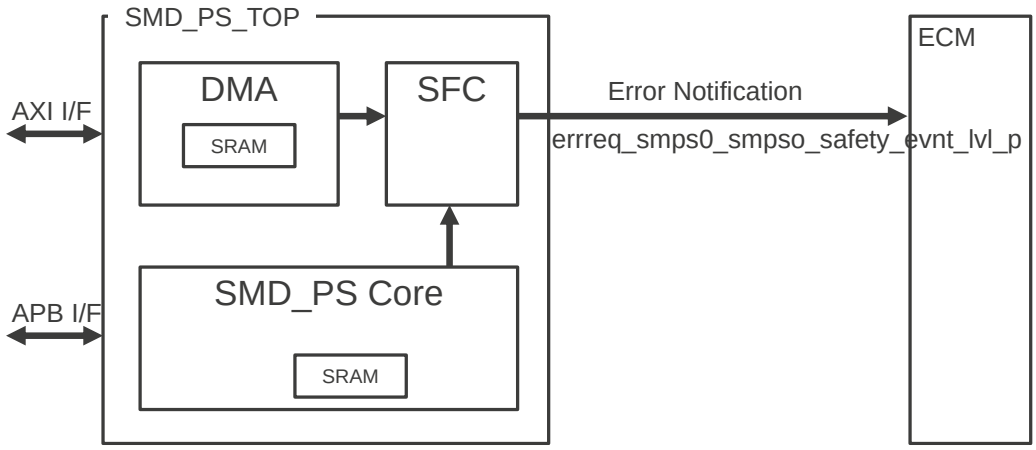



**Figure U25.5 EDC check for Vision IP (SMD-PS)**

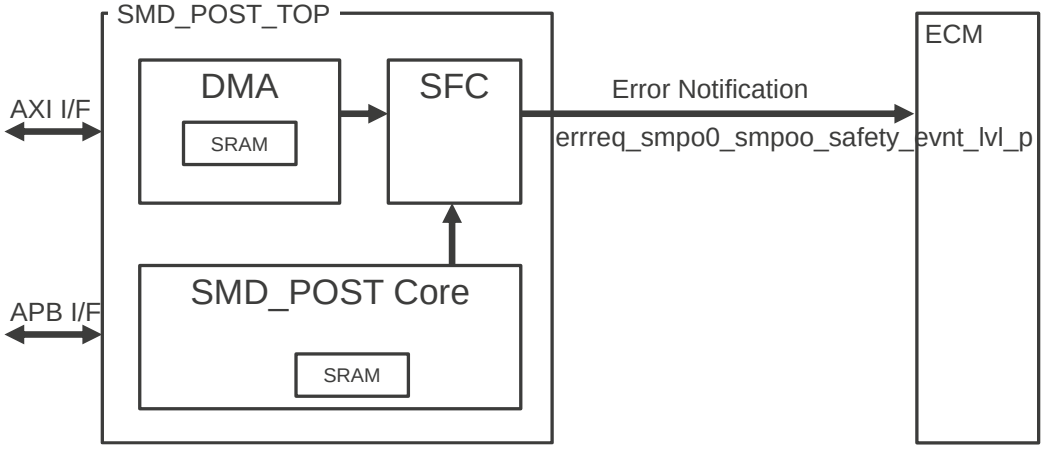



**Figure U25.6 EDC check for Vision IP (SMD-POST)**

#### U25.6.2.2 Register Descriptions

Refer to Software interface section in each attached licensee guide.

| Module | Attached licensee guide file name |
|---|---|
| DOF (UMF) | U25_umf_IPSpec_v1p2.pdf |
| SMD_PS | U25_smdps_IPSpec_v1p1.pdf |
| SMD_POST | U25_smdpost_IPSpec_v1p1.pdf |

### U25.6.3 Error Injection

#### U25.6.3.1 Register setting flow

- Write 0x1 to register SFC_EVTC_GRP1GLB_FORCE to activate error event output port safety_evnt
- Write 0x0 to register SFC_EVTC_GRP1GLB_FORCE to de-activate error event
- Write 0x1 to register SFC_EVTC_GRP1GLB_CLEAR to negate output port safety_evnt

#### U25.6.3.2 Register address

| Module | Register name | Register's address |
|---|---|---|
| DOF (UMF) | SFC_EVTC_GRP1GLB_FORCE | H'00 E7A1 1028 |
| | SFC_EVTC_GRP1GLB_CLEAR | H'00 E7A1 102C |
| SMD_PS | SFC_EVTC_GRP1GLB_FORCE | H'00 E7A2 1028 |
| | SFC_EVTC_GRP1GLB_CLEAR | H'00 E7A2 102C |
| SMD_POST | SFC_EVTC_GRP1GLB_FORCE | H'00 E7A4 1028 |
| | SFC_EVTC_GRP1GLB_CLEAR | H'00 E7A4 102C |

# U30. IMP DMAC

## U30.1 Overview

The IMP DMAC/Slim-IMP DMAC is one of the modules that make up IMP-X7. IMP DMAC/Slim-IMP DMAC performs data transfer between external memory and IMP dedicated internal memory at high speed.

The IMP DMAC logically operates two input sources (Source 0, Source 1) at maximum, and outputs them to one destination (Destination 0).

The Slim-IMP DMAC operates one input source (Source0), and output it to one destination (Destination0)

The IMP DMAC/Slim-IMP DMAC process uses little endian.

Note: The actual modules are referred to as the IMP DMAC and Slim-IMP DMAC in the R-Car Gen4.

**Table 30.1 Base Address of IMP DMAC**

| Channel | Base Address | Number of Threads *1 |
|---|---|---|
| 0 | H'FFA8 0000 (Thread0)<br>H'FFA8 1000 (Thread1) | 2 |
| 1 | H'FFB8 0000 (Thread0)<br>H'FFB8 1000 (Thread1) | 2 |

**Table 30.2 Base Address of Slim-IMP DMAC**

| Channel | Base Address | Number of Threads *1 |
|---|---|---|
| 0 | H'FFA9 0000 (Thread0)<br>H'FFA9 1000 (Thread1) | 2 |
| 1 | H'FFA9 4000 (Thread0)<br>H'FFA9 5000 (Thread1) | 2 |

Notes: 1. Thread implements registers that are necessary for the instruction fetch/decode circuit and data transmission.
Each Thread shares a transfer circuit, and each Thread is arbitrated for each data transfer unit. If the data transmission of Thread0 and Thread1 starts at the same time, that of Thread0 will take precedence.

### U30.1.1 Features

- Command List (CL) system
  - No Operation Instruction (NOP)
  - CL Close Instruction (TRAP)
  - Register Write Instruction (WPR)
  - Interrupt Generation Instruction (INT)
  - Data Transmission Completion Synchronization Instruction (SYNCS)
  - Instruction string control (GOSUB, RET), Relative and Absolute addressing, Conditional GOSUB
  - JUMP Instruction (JUMP), Relative and Absolute addressing
  - Synchronization Instruction with other modules (WUP, SLP)
  - CL break(debug) function.

- **IMP DMAC/ Slim-IMP DMAC**
  - Support transfer of 8, 16, 32bpp data
  - Perform various logical operations (LOP) between data of 2 sources Source0 and Source1, then transfer to Destination *2
  - Support for constant write to the destination memory
  - Support BitCount function that counts the number of 1 after logical operation*3
  - Support tile addressing mode*1*3.
  - Support the data swap function (SWAP) that can be used in endian conversion
  - Support alignment of 1 pixel unit in linear addressing mode
  - Support shifting, rounding and saturation for the output of various logical operations. *3
  - Support Source and destination address wraparound. *4

Notes:

1. Only supported by Source0, Destination. Source1 supports linear addressing only.
2. In Slim-IMP DMAC, the source data of the Source0 is transfer to the Destination and does not support LOP. Slim-IMP DMAC does not support Source1, and source buffer is half of IMPDMAC.
3. These features are not supported in Slim-IMP DMAC.
4. This feature is supported in Slim-IMP DMAC only.

### U30.1.2 Block diagram

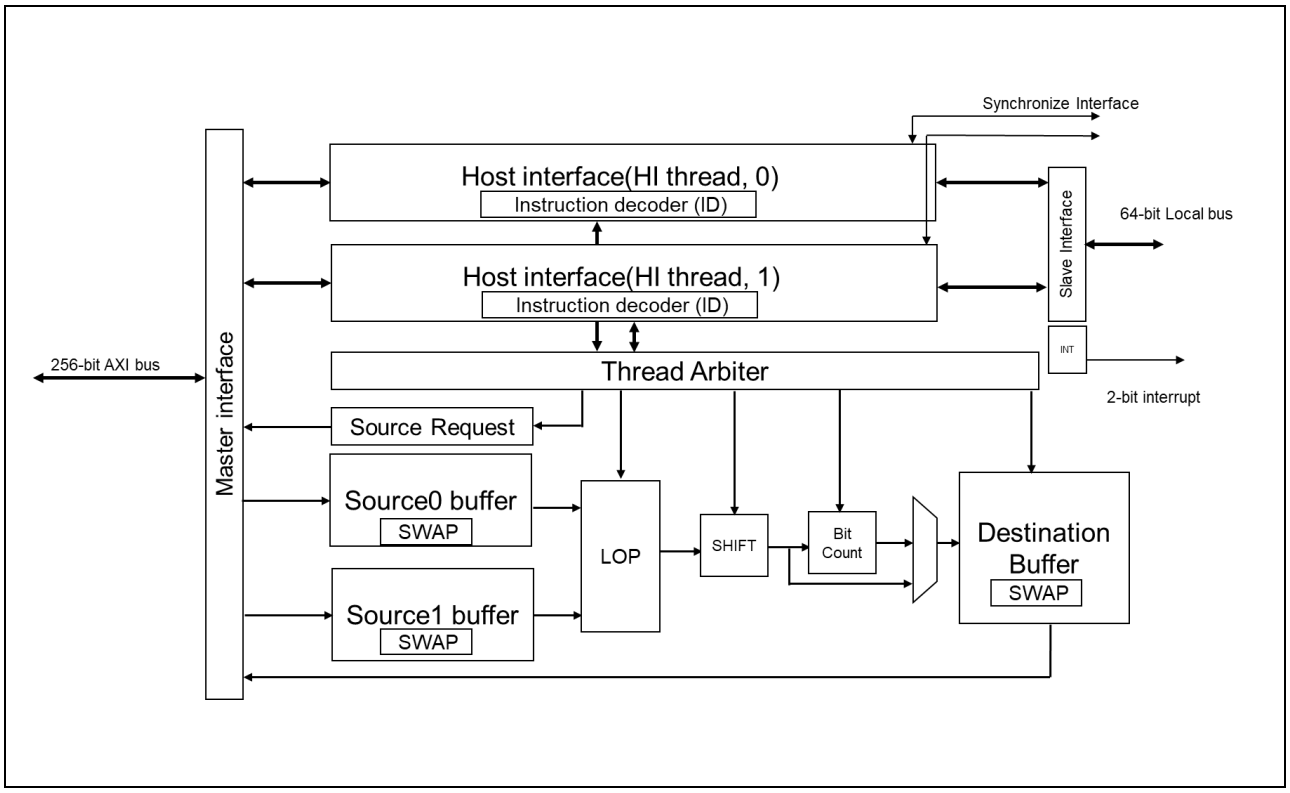



**Figure U30.1 Block Diagram [IMP DMAC]**

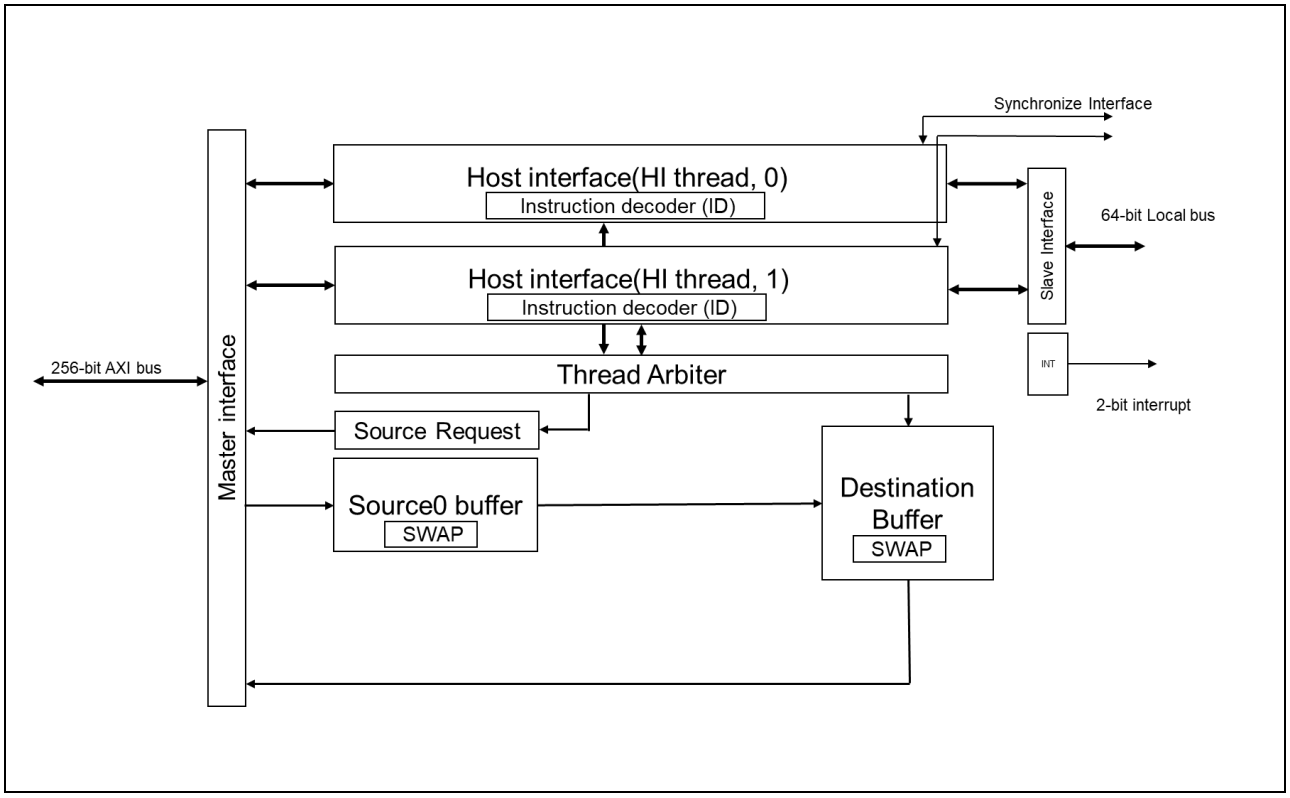



**Figure U30.2 Block Diagram [Slim-IMP DMAC]**

### U30.1.3 Register Configuration

Accessing to addresses not listed in this register list is prohibited. Operation cannot be guaranteed when accessing to an address, which is not listed. Access size is 32-bit. Operation cannot be guaranteed when accessing by other than 32-bit. Operation cannot be guaranteed when writing to a register, which can't be accessed during IMP DMAC execution *1.

**Table 30.3 Register Configuration**

| Name | Symbol | R/W | Offset | CPU Write Access during dmac execution *2 | Initial value | Reset*3 | WPR support |
|---|---|---|---|---|---|---|---|
| Version Control Register | VCR | R | H'0000 | — | *6 | — | — |
| System Control Register 0 | SCTLR0 | R/W | H'0004 | — | H'00000000 | H | √ |
| Status Register 0 | SR0 | R | H'0008 | — | H'00000000 | S/H | — |
| Status Clear Register | SCR | -/W | H'000C | √ | H'00000000 | S/H | — |
| Status Enable Register | SER | R/W | H'0010 | √ | H'0001F3F1 | S/H | — |
| Interrupt Mask Register | IMR | R/W | H'0014 | √ | H'0001F7F1 | S/H | — |
| System Control Register1 | SCTLR1 | R/W | H'0018 | — | H'00000000 | S/H | — |
| Transaction Size Control Register | TSCR | R/W | H'0020 | — | H'00220002*5 | S/H | √ |
| Command List Start Address Register | CLSAR | R/W | H'0028 | — | H'00000000 | S/H | — |
| Status Register1 | SR1 | R | H'0030 | — | H'00000000 | S/H | — |
| Data Swap Register | DSWPR | R/W | H'0038 | — | H'00000000 | S/H | √ |
| Source0 Start Address Register | S0SAR | R/W | H'0040 | — | H'00000000 | S/H | √ |
| Source0 Stride Register | S0STR | R/W | H'0044 | — | H'00000000 | S/H | √ |
| Source0 Data Register | S0DATAR | R/W | H'004C | — | H'00000000 | S/H | √ |
| Source0 Control Register | S0CR | R/W | H'0050 | — | H'00000003 | S/H | √ |
| Source1 Start Address Register*4 | S1SAR | R/W | H'0060 | — | H'00000000 | S/H | √ |
| Source1 Stride Register*4 | S1STR | R/W | H'0064 | — | H'00000000 | S/H | √ |
| Source1 Data Register*4 | S1DATAR | R/W | H'006C | — | H'00000000 | S/H | √ |
| Source1 Control Register*4 | S1CR | R/W | H'0070 | — | H'00000003 | S/H | √ |
| Transaction control Register 0 | TCR0 | R/W | H'007C | — | H'00000000 | S/H | √ |
| Destination0 Start Address Register | D0SAR | R/W | H'0080 | — | H'00000000 | S/H | √ |
| Destination0 Stride Register | D0STR | R/W | H'0084 | — | H'00000000 | S/H | √ |
| Destination0 Control Register | D0CR | R/W | H'0090 | — | H'00000003 | S/H | √ |
| Image Size Register | IMGSIZER | R/W | H'0094 | — | H'00000000 | S/H | √ |
| Function Control Register0 | FCR0 | R/W | H'00A0 | — | H'000000CC | S/H | √ |
| Function Control Register1 | FCR1 | R/W | H'00AC | — | H'10000000 | S/H | √ |
| Command List Fetch Address Register | CLFAR | R | H'00C8 | — | Undefined | — | — |
| Command List Execute Instruction Register | CLEIR | R | H'00CC | — | H'00000000 | S/H | — |
| RAM Test Register | RAMTSTR | R/W | H'00D0 | — | H'00000000 | S/H | √ |
| DMAC Performance Control Register | DPCR | R/W | H'00DC | — | H'00000000 | SH | √ |
| DMAC Performance Busy Counter Register | DPBSYCNTR | R/W | H'00E0 | — | H'00000000 | SH | √ |

| Name | Symbol | R/W | Offset | CPU Write Access during dmac execution *2 | Initial value | Reset*3 | WPR support |
|---|---|---|---|---|---|---|---|
| Synchronization Control Register 0 | SYNCCR0 | R/W | H'00E8 | — | *7 | S/H | √ |
| Synchronization Control Register 1 | SYNCCR1 | R/W | H'00EC | — | *7 | S/H | √ |
| Synchronization Control Register 2 | SYNCCR2 | R/W | H'00F0 | — | *7 | S/H | √ |
| Synchronization Control Register 3 | SYNCCR3 | R/W | H'00F4 | — | *7 | S/H | √ |
| Index Register0 | INDEX0 | R | H'0260 | — | H'00000000 | S/H | — |
| Index Register (1-7) | INDEX (1-7) | R/W | H'0264 to H'027C | — | H'00000000 | S/H | — |
| Wraparound enable register | WRAPER | R/W | H'0298 | — | H'00000000 | SH | √ |
| SRC minimum address wraparound register | WRAPS_MIN | R/W | H'029C | — | H'00000000 | SH | √ |
| SRC maximum address wraparound register | WRAPS_MAX | R/W | H'02A0 | — | H'00000000 | SH | √ |
| DST minimum address wraparound register | WRAPD_MIN | R/W | H'02A4 | — | H'00000000 | SH | √ |
| DST maximum address wraparound register | WRAPD_MAX | R/W | H'02A8 | — | H'00000000 | SH | √ |
| Transaction control Register 1 | TCR1 | R/W | H'0344 | — | H'00000000 | S/H | √ |
| Source0 Start Address Offset Register | S0SAR_OFS | W | H'0350 | — | H'00000000 | S/H | √ |
| Source1 Start Address Offset Register | S1SAR_OFS | W | H'0354 | — | H'00000000 | S/H | √ |
| Destination0 Start Address Offset Register | D0SAR_OFS | W | H'0358 | — | H'00000000 | S/H | √ |
| CL Break Address Register | CLBRKADDRR | R/W | H'03F4 | √ | H'00000000 | S/H | — |
| CL Conditinal GOSUB Register | CLCNDGSBR | R/W | H'03FC | √ | H'00000000 | S/H | — |

Notes: 1. It includes the time during CL execution.
2. DMAC execution is the state that DMAEN bit of SCTLR0 register = 1, or CLE bit of SCTLR1 register = 1.
3. H: Power on reset S: Software reset by SWRST bit of SCTLR0 register.

4. These registers are not support for Slim-IMP DMAC**.**
5. For the IMP DMAC, initial value is H'00220002. For Slim-IMP DMAC initial value is H'00030003**.**

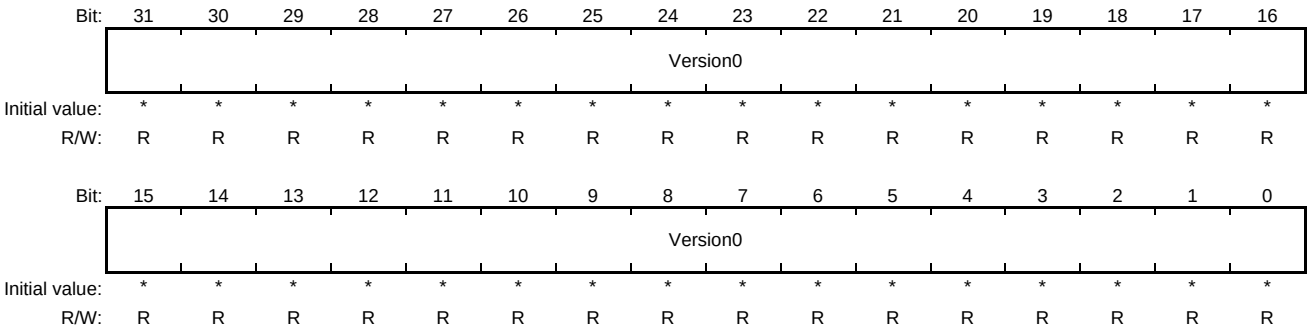
6. The value to be read varies depending on the product. Refer register description for the initial value.
7. Initial value is varies depending on the product. Refer register description for the initial value.

## U30.2 Register Description

**[Legend]**

Initial value: Register value after a reset
—: Undefined value
R/W: Readable/writable. The written value can be read.
R: Read-only. The write value should be the same as the read value. Only modify the target bit (Read Modify Write) because the initial value should not be changed.
-/W: Write-only. The read value is undefined.

### U30.2.1 Version Control Register (VCR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Version0 | * | R | These bits indicate the version of this module. |

Note: The value to be read varies depending on the product.
IMP DMAC/ Slim-IMP DMAC: H’21111616

### U30.2.2 System Control Register 0 (SCTLR0)

This is the register to control the reset (software), and the start of DMA.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SWRST | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DMAEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | SWRST | 0 | R/W | Software reset<br>Writing 1 to this bit applies a software reset and writing 0 cancels a software reset.<br>0: Cancels a software reset for this module.<br>1: Applies a software reset to this module.<br>When setting to 1, the value of the register with the notation 'S' written in the "Reset" column of Table 30.3 Register Configuration will be initialized.<br>When the DMAEN bit of SCTLR0 register is 1 or the CLE bit of CTLR1 register is 1, setting to 1 is prohibited. In that case, the operation cannot be guaranteed.<br>Setting this bit and the DMAEN bit to 1 at the same time is prohibited. In that case, the operation cannot be guaranteed.<br>In order to cancel the reset, write 0 to this bit, then read this bit to check if that the value is 0.<br>Setting this bit to 1 with WPR is prohibited. In that case, the operation cannot be guaranteed. |
| 30 to 1 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 0 | DMAEN | 0 | R/W | DMA transmission start bit<br>[Write]<br>0: NOP<br>1: DMA transmission starts<br>[Read]<br>0: DMA transmission has not been performed<br>1: DMA transmission is in progress<br>After DMA transmission is completed, this bit automatically becomes 0. At this time, if the TENDE bit of SER register is 1, the TEND bit of SR0 register will become 1.<br>Setting this bit to 0 or 1 while it is 1 is prohibited. In that case, the operation cannot be guaranteed.<br>Setting this bit to 1 from the CPU is prohibited while the CLE bit of SCTLR1 register is 1. In that case, the operation cannot be guaranteed. |

### U30.2.3 Status Register 0 (SR0)

This register displays every status generated by IMP DMAC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLBRK | — | — | WUPOVF | INT | IER | TRA | — | — | — | TEND |
| Initial value: | * | * | * | * | * | 0 | * | * | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | Undefined | R | Reserved<br>The read value is undefined.<br>Operation cannot be guaranteed if a value other than the read value is written. |
| 10 | CLBRK | 0 | R | CL break status.<br>0: CL does not break<br>1: CL break<br>CL break can not be used if CLBRKENB of SER register is 0. Set CLBRKENB of SER register to 1 to use CL break function.<br>Command list continue execution after clearing this bit using CLBRKC bit of SCR register. |
| 9 to 8 | — | Undefined | R | Reserved<br>The read value is undefined.<br>Operation cannot be guaranteed if a value other than the read value is written. |
| 7 | WUPOVF | 0 | R | WUP counter overflow<br>This bit indicates the overflow of WUP instruction.<br>0: WUP overflow has not been occured<br>1: WUP overflow has been occured<br>Note: When WUPOVFE bit in SER is 0, this bit is not set to 1 even if the WUP overflow occured.<br>When WUPOVFE bit in SER is 1, this bit is set to 1 if the WUP overflow is occured. |
| 6 | INT | 0 | R | INT instruction decoding<br>This bit indicates the decoding of INT instruction.<br>Once this bit becomes 1, the next instruction will not start until this bit is cleared to 0.<br>0: INT instruction has not been decoded<br>1: INT instruction has been decoded<br>Note: When INTE bit in SER is 0, this bit is not set to 1 even if the INT instruction is decoded.<br>When INTE bit in SER is 1, this bit is set to 1 if the INT instruction is decoded. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 5 | IER | 0 | R | Illegal instruction decoding or GOSUB overflow/underflow.<br>This bit indicates the decoding of the illegal instruction.<br>Once this bit becomes 1, the CL execution will end.<br>Illegal instructions are instructions that are not listed in Table 30.4.<br>0: Illegal instruction has not been decoded<br>1: Illegal instruction has been decoded<br>Note: When IERE bit in SER is 0, this bit is not set to 1 even if illegal instructions are decoded.<br>When IERE bit in SER is 1, this bit is set to 1 if illegal instructions are decoded.<br>Reset IMP DMAC by using SWRST bit of SCTLR0 after the IER error. |
| 4 | TRA | 0 | R | TRAP instruction decoding<br>This bit indicates the decoding of TRAP instruction.<br>After TRAP instruction is decoded, the CL execution will end.<br>0: TRAP instruction has not been decoded<br>1: TRAP instruction has been decoded<br>Note: When TRAE bit in SER is 0, this bit is not set to 1 even if the TRAP instruction is decoded.<br>When TRAE bit in SER is 1, this bit is set to 1 if the TRAP instruction is decoded. |
| 3 to1 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 0 | TEND | 0 | R | Transmission completion<br>When DMA transmission ends, this bit is set to 1.<br>[Load]<br>0: Transmission has not been started or it is in progress<br>1: Transmission has ended<br>Note: When TENDE bit in SER is 0, this bit is not set to 1 even if the DMA transmission has ended.<br>When TENDE bit in SER is 1, this bit is set to 1 if the DMA transmission has ended.<br>When using CL, set the TENDE bit of SER register to 0 and do not use this bit. |

### U30.2.4 Status Clear Register (SCR)

This register clears every status displayed in SR0. Write 1 to clear, writing 0 is disabled.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLBRKC | — | — | WUPOVFC | INTC | IERC | TRAC | — | — | — | TENDC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | -/W | R | R | -/W | -/W | -/W | -/W | R | R | R | -/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 11 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 10 | CLBRKC | 0 | -/W | CL break status clear.<br>By writing 1, the 0 in the CLBRK bit of SR0 register will be cleared.<br>This bit is always read as 0.<br>Command list continues execution after clearing CLBRK bit of SR0 register. |
| 9 to 8 | — | 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than the read value is written. |
| 7 | WUPOVFC | 0 | -/W | WUPOVF bit clearing<br>By writing 1, the 0 in the WUPOVF bit of SR0 register will be cleared.<br>This bit is always read as 0. |
| 6 | INTC | 0 | -/W | INT bit clearing<br>By writing 1, the 0 in the INT bit of SR0 register will be cleared.<br>This bit is always read as 0. |
| 5 | IERC | 0 | -/W | IER bit clearing<br>By writing 1, the 0 in the IER bit of SR0 register will be cleared.<br>This bit is always read as 0. |
| 4 | TRAC | 0 | -/W | TRA bit clearing<br>By writing 1, the 0 in the TRA bit of SR0 register will be cleared.<br>This bit is always read as 0. |
| 3 to1 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 0 | TENDC | 0 | -/W | TEND bit clearing<br>By writing 1, the TEND bit of SR0 register will be cleared.<br>This bit is always read as 0. |

### U30.2.5 Status Enable Register (SER)

This register is used to set whether to reflect status in each bit of SR0 register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLBRK ENB | — | — | WUPO VFE | INTE | IERE | TRAE | — | — | — | TENDE |
| Initial value: | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 16 | — | 1 | R | Reserved<br>The read value should always be 1.<br>Operation cannot be guaranteed if a value other than 1 is written. |
| 15 to 12 | — | B’1111 | R | Reserved<br>The read value should always be 1.<br>Operation cannot be guaranteed if a value other than 1 is written. |
| 11 | — | 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 10 | CLBRKENB | 0 | R/W | CL break status/CL break enable.<br>0: CL break disable<br>When CLBRK is not detected, 1 is not set to CLBRK bit of SR0 register.<br>1: CL break enable<br>When CLBRK is detected, 1 is set to CLBRK bit of SR0 register.<br>When this bit is 0, CL break operation is not be possible. |
| 9 to 8 | — | B’11 | R | Reserved<br>The read value should always be 1.<br>Operation cannot be guaranteed if a value other than 1 is written. |
| 7 | WUPOVFE | 1 | R/W | WUPOVF bit set enable<br>0: When WUP overflow is detected, 1 is not set to WUPOVF bit of SR0 register.<br>1: When WUP overflow is detected, 1 is set to WUPOVF bit of SR0 register.<br>Refer to the **U34.IMP-X7_system, U34.10.3 Inter Module Synchronization** when setting this bit. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | INTE | 1 | R/W | INT bit set enable<br>0: When the INT instruction is decoded, 1 is not set to the INT bit of SR0 register.<br>1: When the INT instruction is decoded, 1 is set to the INT bit of SR0 register |
| 5 | IERE | 1 | R/W | IER bit set enable<br>0: When the illegal instruction is decoded, 1 is not set to the IER bit of SR0 register<br>1: When the illegal instruction is decoded, 1 is set to the IER bit of SR0 register |
| 4 | TRAE | 1 | R/W | TRA bit set enable<br>0: When the TRAP instruction is decoded, 1 is not set to the TRA bit of SR0 register.<br>1: When the TRAP instruction is decoded, 1 is set to the TRA bit of SR0 register |
| 3 to1 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 0 | TENDE | 1 | R/W | TEND bit set enable<br>0: Even if DMA transmission ends, 1 is not set to the TEND bit of SR0 register.<br>1: If DMA transmission ends, 1 is set to the TEND bit of SR0 register. |

### U30.2.6 Interrupt Mask Register (IMR)

This register is used to set whether to generate an interrupt or mask an interrupt according to occurrence of each status.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLBRKM | — | — | WUPOVFM | INTM | IERM | TRAM | — | — | — | TENDM |
| Initial value: | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R/W | R | R | R/W | R/W | R/W | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 16 | — | 1 | R | The read value should always be 1.<br>Operation cannot be guaranteed if a value other than 1 is written. |
| 15 to 12 | — | B’1111 | R | Reserved<br>The read value should always be 1.<br>Operation cannot be guaranteed if a value other than 1 is written. |
| 11 | — | 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 10 | CLBRKM | 1 | R/W | Mask setting of interrupt by CLBRK status<br>0: The generation of CLBRK interrupt is not masked.<br>1: The generation of CLBRK interrupt is masked. |
| 9 to 8 | — | B’11 | R | Reserved<br>The read value should always be 1.<br>Operation cannot be guaranteed if a value other than 1 is written. |
| 7 | WUPOVFM | 1 | R/W | Mask setting of interrupt by WUPOVF status<br>0: The generation of WUPOVF interrupt is not masked.<br>1: The generation of WUPOVF interrupt is masked. |
| 6 | INTM | 1 | R/W | Mask setting of interrupt by INT status<br>0: The generation of INT interrupt is not masked.<br>1: The generation of INT interrupt is masked. |
| 5 | IERM | 1 | R/W | Mask setting of interrupt by IER status<br>0: The generation of IER interrupt is not masked.<br>1: The generation of IER interrupt is masked. |
| 4 | TRAM | 1 | R/W | Mask setting of interrupt by TRA status<br>0: The generation of TRA interrupt is not masked.<br>1: The generation of TRA interrupt is masked. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 to1 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 0 | TENDM | 1 | R/W | Mask setting of interrupt by TEND status<br>0: The generation of TEND interrupt is not masked.<br>1: The generation of TEND interrupt is masked. |

### U30.2.7 System Control Register1 (SCTLR1)

This is the register to control the start of CL.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLE | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | CLE | 0 | R/W | IMP DMAC's CL starting bit<br>[Write]<br>0: - (Prohibited)<br>1: Start CL<br>[Read]<br>0: CL is not in progress.<br>1: CL is in progress.<br>When 1 is written to this bit, IMP DMAC fetches the CL from the address which set CL to CLSAR, and executes the instruction.<br>This bit is read as 1 during the CL execution and read as 0 when execution is completed. Writing 1 to this bit during the CL execution is prohibited. If it is written, the operation cannot be guaranteed. |
| 30 to 0 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |

### U30.2.8 Transaction Size Control Register (TSCR)

**[IMP DMAC]**

Source0, Source1, Destination specifies the transaction size to access for the bus.

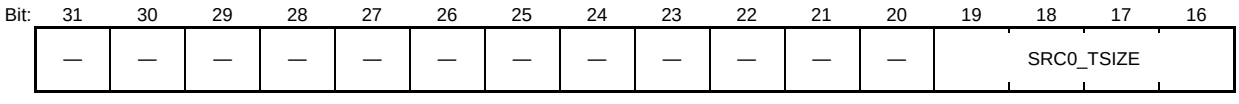
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SRC1_TSIZE | | | | SRC0_TSIZE | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | DST0_TSIZE | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 23 to 20 | SRC1_TSIZE | 0010 | R/W | These bits define the transmission size of Source1.<br>2: The read request of Source1 is performed in 64-byte unit.<br>3: The read request of Source1 is performed in 128-byte unit.<br>4: The read request of Source1 is performed in 256-byte unit.<br>Others: Setting prohibited<br>When the PXALIGN bit of FCR1 register is 0:<br>Setting this bit to other than 2 and 3 is prohibited.<br>Set the same value as SRC0_TSIZE bit and DST0_TSIZE bit.<br>When the PXALIGN bit of FCR1 register is 1:<br>Setting this bit to other than 3 and 4 is prohibited.<br>Set the same value as SRC0_TSIZE bit. |
| 19 to 16 | SRC0_TSIZE | 0010 | R/W | These bits define the transmission size of Source0.<br>2: The read request of Source0 is performed in 64-byte unit.<br>3: The read request of Source0 is performed in 128-byte unit.<br>4: The read request of Source0 is performed in 256-byte unit.<br>Others: Setting prohibited<br>When the PXALIGN bit of FCR1 register is 0:<br>Setting this bit to other than 2 and 3 is prohibited.<br>Set the same value as SRC1_TSIZE bit and DST0_TSIZE bit.<br>When the PXALIGN bit of FCR1 register is 1:<br>Setting this bit to other than 3 and 4 is prohibited.<br>Set the same value as SRC1_TSIZE bit. |
| 15 to 4 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 3 to 0 | DST0_TSIZE | 0010 | R/W | These bits define the transmission size of Destination0.<br>0: The write request of Destination0 is performed in 4-byte unit.<br>2: The write request of Destination0 is performed in 64-byte unit.<br>3: The write request of Destination0 is performed in 128-byte unit.<br>4: The write request of Destination0 is performed in 256-byte unit.<br>Others: Setting prohibited<br>When the PXALIGN bit of FCR1 register is 0:<br>Setting this bit to other than 0, 2 and 3 is prohibited.<br>Set the same value as SRC0_TSIZE and SRC1_TSIZE.<br>When the PXALIGN bit of FCR1 register is 1:<br>Setting this bit to other than 0, 3 and 4 is prohibited. |

**[Slim-IMP DMAC]**

This register is used to set the transaction size in bus access from each of the source0 and destination buffers.



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | DST0_TSIZE | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 20 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 19 to 16 | SRC0_TSIZE | 0011 | R/W | 3: The read request of Source0 is performed in 128-byte unit.<br>4: The read request of Source0 is performed in 256-byte unit.<br>Setting other than above is prohibited for this bit. |
| 15 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 3 to 0 | DST0_TSIZE | 0011 | R/W | 3: The write request of Destination0 is performed in 128-byte unit.<br>4: The write request of Destination0 is performed in 256-byte unit.<br>Setting other than above is prohibited for this bit. |

### U30.2.9 Command List Start Address Register (CLSAR)

This register is used to set the first address when fetching CL.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLSAR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLSAR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CLSAR | H'0000 0000 | R/W | First address of CL<br>These bits set the first address of CL execution in this register.<br>Set it in 4-byte unit. |

### U30.2.10 Status Register1 (SR1)

This register displays the code value when IMP DMAC executes INT and TRAP instructions.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCODE | | | | | | | | TRAPCODE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | INTCODE | H'00 | R | INT interrupt code<br>If the INT instruction is executed, the value specified in the CODE field (bit[7:0]) of this instruction will be reflected in this field.<br>This field is updated independently of the setting of INTE bit in SER register. |
| 23 to 16 | TRAPCODE | H'00 | R | TRAP interrupt code<br>If the TRAP instruction is executed, the value specified in the CODE field (bit[7:0]) of this instruction will be reflected in this field.<br>This field is updated independently of the setting of TRAE bit in SER register. |
| 15 to 0 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |

### U30.2.11 Data Swap Register (DSWPR)

This register is used to set the swap of data in Source0, Source1, Destination0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | S1SWP 128BIT | S1SWP 64BIT | S1SWP 32BIT | S1SWP 16BIT | S1SWP 8BIT | — | — | — | S0SWP 128BIT | S0SWP 64BIT | S0SWP 32BIT | S0SWP 16BIT | S0SWP 8BIT | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R/W | R/W | R | R | R | R/W | R/W | R/W | R/W | R/W | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | D0SWP 128BIT | D0SWP 64BIT | D0SWP 32BIT | D0SWP 16BIT | D0SWP 8BIT | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 30 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 29 | S1SWP128BIT | 0 | R/W | Source1 128-bit swap setting<br>This bit swaps 256 bits of Source1 data in 128-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1, set S1SAR, S1STR, and IWIDTH in 256-bit unit.<br>When the S1PF bit of S1CR register is not 32bpp, 16bpp, 8bpp, setting this bit to 1 is prohibited.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 28 | S1SWP64BIT | 0 | R/W | Source1 64-bit swap setting<br>This bit swaps 128 bits of Source1 data in 64-bit unit.<br>When this bit is set to 1 by S1SWP128BIT=0, set S1SAR, S1STR, and IWIDTH in 128-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When the S1PF bit of S1CR register is not 32bpp, 16bpp, 8bpp, setting this bit to 1 is prohibited.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than 0 is written here. |

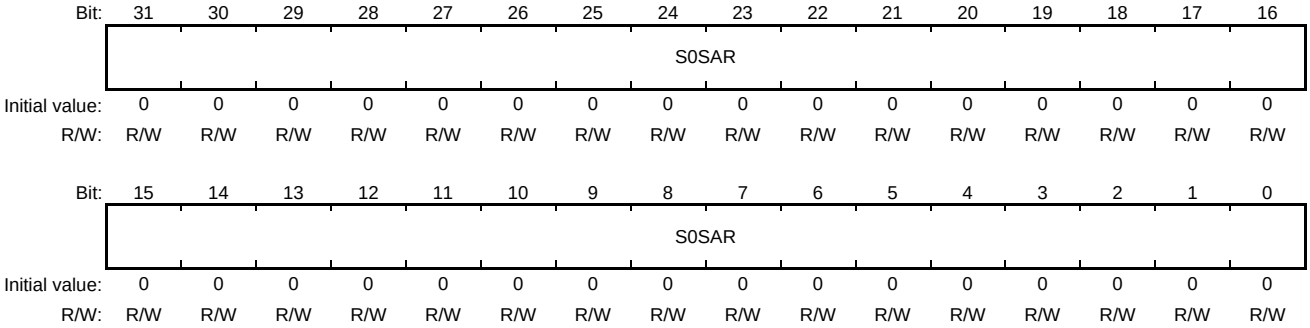
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 27 | S1SWP32BIT | 0 | R/W | Source1 32-bit swap setting<br>This bit swaps 64 bits of Source1 data in 32-bit unit.<br>When this bit is set to 1 by S1SWP128BIT=S1SWP64BIT=0, set S1SAR, S1STR, and IWIDTH in 64-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When the S1PF bit of S1CR register is not 32bpp, 16bpp, 8bppsetting this bit to 1 is prohibited.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 26 | S1SWP16BIT | 0 | R/W | Source1 16-bit swap setting<br>This bit swaps 32 bits of Source1 data in 16-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by S1SWP128BIT=S1SWP64BIT=S1SWAP32BIT=0, set S1SAR, S1STR, and IWIDTH in 32-bit unit.<br>When the S1PF bit of S1CR register is not 16bpp, 8bppsetting this bit to 1 is prohibited.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 25 | S1SWP8BIT | 0 | R/W | Source1 8-bit swap setting<br>This bit swaps 16 bits of Source1 data in 8-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by S1SWP128BIT=S1SWP64BIT=S1SWAP32BIT=S1SWAP16BIT=0, set S1SAR, S1STR, and IWIDTH in 16-bit unit.<br>When the S1PF bit of S1CR register is not 8bpp, setting this bit to 1 is prohibited.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 24 to 22 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 21 | S0SWP128BIT | 0 | R/W | Source0 128-bit swap setting<br>This bit swaps 256 bits of Source0 data in 128-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1, set S0SAR, S0STR, and IWIDTH in 256-bit unit.<br>When the S0PF bit of S0CR register is not 32bpp, 16bpp, 8bppsetting this bit to 1 is prohibited. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 20 | S0SWP64BIT | 0 | R/W | Source0 64-bit swap setting<br>This bit swaps 128 bits of Source0 data in 64-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by S0SWP128BIT=0, set S0SAR, S0STR, and IWIDTH in 128-bit unit.<br>When the S0PF bit of S0CR register is not 32bpp, 16bpp, 8bpp, setting this bit to 1 is prohibited. |
| 19 | S0SWP32BIT | 0 | R/W | Source0 32-bit swap setting<br>This bit swaps 64 bits of Source0 data in 32-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by S0SWP128BIT=S0SWP64BIT=0, set S0SAR, S0STR, and IWIDTH in 64-bit unit.<br>When the S0PF bit of S0CR register is not 32bpp, 16bpp, 8bppsetting this bit to 1 is prohibited. |
| 18 | S0SWP16BIT | 0 | R/W | Source0 16-bit swap setting<br>This bit swaps 32 bits of Source0 data in 16-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by S0SWP128BIT=S0SWP64BIT=S0SWAP32BIT=0, set S0SAR, S0STR, and IWIDTH in 32-bit unit.<br>When the S0PF bit of S0CR register is not 16bpp, 8bppsetting this bit to 1 is prohibited. |
| 17 | S0SWP8BIT | 0 | R/W | Source0 8-bit swap setting<br>This bit swaps 16 bits of Source0 data in 8-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by S0SWP128BIT=S0SWP64BIT=S0SWAP32BIT=S0SWAP16BIT=0, set S0SAR, S0STR, and IWIDTH in 16-bit unit.<br>When the S0PF bit of S0CR register is not 8bpp, setting this bit to 1 is prohibited. |
| 16to 6 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 5 | D0SWP128BIT | 0 | R/W | Destination0 128-bit swap setting<br>This bit swaps 256 bits of Destination0 data in 128-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1, set D0SAR, D0STR, and IWIDTH in 256-bit unit.<br>When the D0PF bit of D0CR register is not 32bpp, 16bpp, 8bpp, setting this bit to 1 is prohibited. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 4 | D0SWP64BIT | 0 | R/W | Destination0 64-bit swap setting<br>This bit swaps 128 bits of Destination0 data in 64-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by D0SWP128BIT=0, set D0SAR, D0STR, and IWIDTH in 128-bit unit.<br>When the D0PF bit of D0CR register is not 32bpp, 16bpp, 8bpp, setting this bit to 1 is prohibited. |
| 3 | D0SWP32BIT | 0 | R/W | Destination0 32-bit swap setting<br>This bit swaps 64 bits of Destination0 data in 32-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by D0SWP128BIT=D0SWP64BIT=0, set D0SAR, D0STR, and IWIDTH in 64-bit unit.<br>When the D0PF bit of D0CR register is not 32bpp, 16bpp, 8bpp, setting this bit to 1 is prohibited. |
| 2 | D0SWP16BIT | 0 | R/W | Destination0 16-bit swap setting<br>This bit swaps 32 bits of Destination0 data in 16-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by D0SWP128BIT=D0SWP64BIT=D0SWAP32BIT=0, set D0SAR, D0STR, and IWIDTH in 32-bit unit.<br>When the D0PF bit of D0CR register is not 16bpp, 8bppsetting this bit to 1 is prohibited. |
| 1 | D0SWP8BIT | 0 | R/W | Destination0 8-bit swap setting<br>This bit swaps 16 bits of Destination0 data in 8-bit unit.<br>0: It is not swapped<br>1: It is swapped<br>When this bit is set to 1 by D0SWP128BIT=D0SWP64BIT=D0SWAP32BIT=D0SWAP16BIT=0, set D0SAR, D0STR, and IWIDTH in 16-bit unit.<br>When the D0PF bit of D0CR register is not 8bpp, setting this bit to 1 is prohibited. |
| 0 | — | 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |

### U30.2.12 Source0 Start Address Register (S0SAR)

This register is used to set the start address of Source0 to read from external memory or internal memory.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | S0SAR | H'0000 0000 | R/W | Set the start address of Source0 in byte unit.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If SRC0_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If SRC0_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If S0PF bits of S0CR register are 3, set this register in 1-byte unit.<br>・If S0PF bits of S0CR register are 4, set this register in 2-byte unit.<br>・If S0PF bits of S0CR register are 5, set this register in 4-byte unit. |

### U30.2.13 Source0 Stride Register (S0STR)

This register is used to set the memory width of Source0 to read from external memory or internal memory.

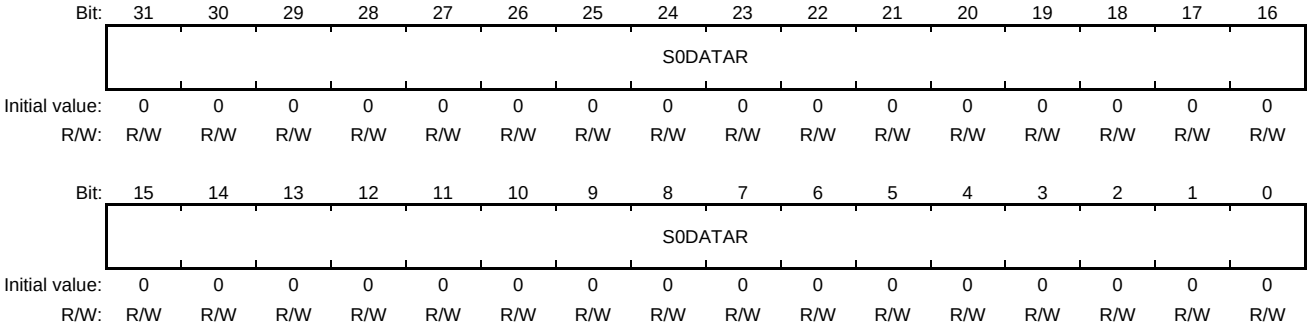
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | S0STR | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 15 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 14 to 0 | S0STR | H'0000 | R/W | These bits set the memory width of Source0 in byte unit. The setting range is 0~32767.<br>Be sure to set the number of pixels, which is set in the IWIDTH bit of IMGSIZER register, to be equal to or greater than the byte converted value.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If SRC0_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If SRC0_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If S0PF bits of S0CR register are 3, set this register in 1-byte unit.<br>・If S0PF bits of S0CR register are 4, set this register in 2-byte unit.<br>・If S0PF bits of S0CR register are 5, set this register in 4-byte unit. |

### U30.2.14 Source0 Data Register (S0DATAR)

This is the data register of Source0.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | S0DATAR | H'0000 0000 | R/W | This is the data register of Source0.<br>When the S0REN bit of S0CR register is 0, this register is used.<br>If the pixel format is less than 32bpp, specify this data repeatedly until it reaches 32 bits in this register.<br>For example, if it is 8 bpp, set four 8-bit data D8, D8, D8, D8.<br>(A D8 represents one 8-bit data.) |

### U30.2.15 Source0 Control Register (S0CR)

This register is used to control Source0.

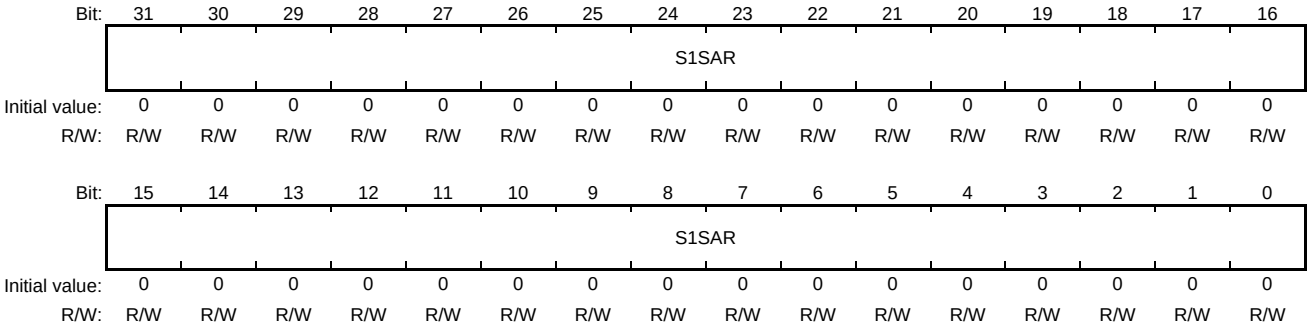
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S0REN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S0TAMODE | | | | — | — | — | — | — | — | — | — | S0PF | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | S0REN | 0 | R/W | Source0 read enable<br>0: Do not read data from external memory or internal memory, use the value of S0DATAR register as the data of Source0.<br>1: Read data of Source0 from external memory or internal memory.<br>If this bit is 0, setting the S1REN bit of S1CR register to 1 is prohibited. |
| 30 to 16 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 15 to 12 | S0TAMODE | 0000 | R/W | Source0 tile addressing mode<br>0: Read the data of Source0 by the linear addressing.<br>1: Read the data of Source0 by the IMP original tile addressing.<br>For details of tile addressing, refer to chapter Operation.<br>Settings other than above are prohibited. If they are set, the operation cannot be guaranteed.<br>If this bit is 1, set S0PF bits in this register to 3 or 4.<br>If this bit is 1, setting the S1REN bit of S1CR register to 1 is prohibited.<br>If this bit is 1, setting the BITCNTEN bit of FCR0 register to 1 is prohibited.<br>When PXALIGN of FCR1 register is 1, setting this bit to 1 is prohibited. |
| 11 to 4 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 3 to 0 | S0PF | 0011 | R/W | Source0 pixel format<br>When the BITCNTEN bit of FCR0 register is 0, specify the same pixel format as the D0PF bits of D0CR register.<br>3: 8bit/pixel<br>4: 16bit/pixel<br>5: 32bit/pixel<br>Settings other than above are prohibited. |

### U30.2.16 Source1 Start Address Register (S1SAR)

This register is used to set the start address of Source1 to read from external memory or internal memory.
Note: This register is not support for Slim-IMP DMAC. Do not set this register for Slim-IMP DMAC operation.
This register is always read as 0x00000000.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | S1SAR | H'0000 0000 | R/W | Set the start address of Source1 in byte unit.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If SRC1_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If SRC1_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If S1PF bits of S1CR register are 3, set this register in 1-byte unit.<br>・If S1PF bits of S1CR register are 4, set this register in 2-byte unit.<br>・If S1PF bits of S1CR register are 5, set this register in 4-byte unit. |

### U30.2.17 Source1 Stride Register (S1STR)

This register is used to set the memory width of Source1 to read from external memory or internal memory.
Note: This register is not support for Slim-IMP DMAC. Do not set this register for Slim-IMP DMAC operation.
This register is always read as 0x00000000.

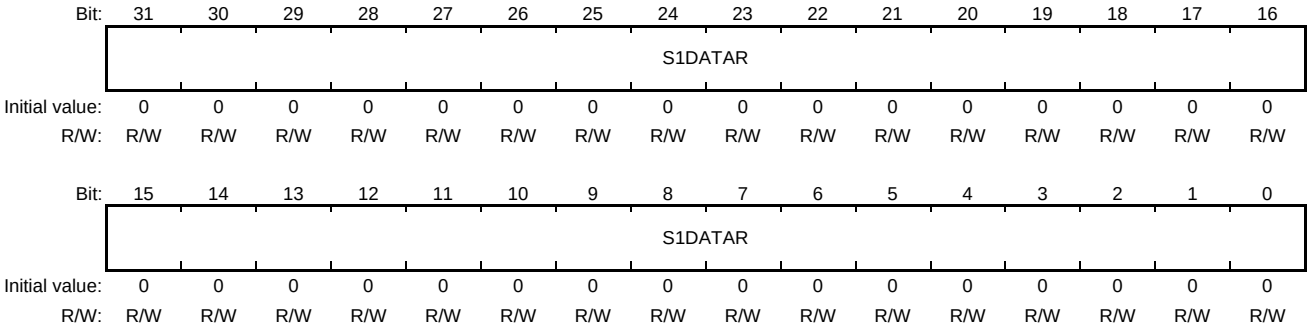
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | S1STR | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 15 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 14 to 0 | S1STR | H'0000 | R/W | These bits set the memory width of Source1 in byte unit. The setting range is 0~32767.<br>Be sure to set the number of pixels, which is set in the IWIDTH bit of IMGSIZER register, to be equal to or greater than the byte converted value.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If SRC1_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If SRC1_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If S1PF bits of S1CR register are 3, set this register in 1-byte unit.<br>・If S1PF bits of S1CR register are 4, set this register in 2-byte unit.<br>・If S1PF bits of S1CR register are 5, set this register in 4-byte unit. |

### U30.2.18 Source1 Data Register (S1DATAR)

This is the data register of Source1.
Note: This register is not support for Slim-IMP DMAC**.** Do not set this register for Slim-IMP DMAC operation. This register is always read as 0x00000000.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | S1DATAR | H'0000 0000 | R/W | This is the data register of Source1.<br>When the S1REN bit of S1CR register is 0, this register is used.<br>If the pixel format is less than 32bpp, specify this data repeatedly until it reaches 32 bits in this register.<br>For example, if it is 8 bpp, set four 8-bit data D8, D8, D8, D8.<br>(A D8 represents one 8-bit data.) |

### U30.2.19 Source1 Control Register (S1CR)

This register is used to control Source1.
Note: This register is not support for Slim-IMP DMAC**.** Do not set this register for Slim-IMP DMAC operation.
This register is always read as 0x00000000.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1REN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

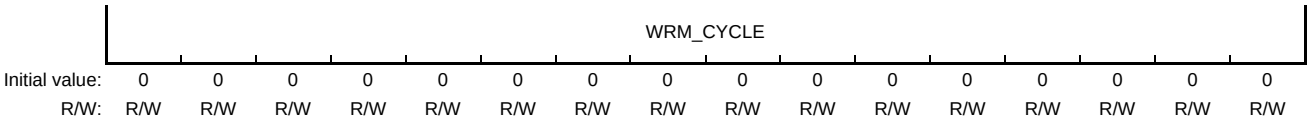
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | S1PF | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R | R | R | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | S1REN | 0 | R/W | Source1 read enable<br>0: Do not read data from external memory or internal memory, use the value of S1DATAR register as the data of Source1.<br>1: Read data of Source1 from external memory or internal memory.<br>[Note]<br>Do not use when (S0REN bit of S0CR register, S1REN bit of S1CR register)=(0,1).<br>It is possible to set this bit to 1 when both S0TAMODE bits of S0CR register and D0TAMODE bits of D0CR register are 0.<br>Otherwise, set this bit to 0. |
| 30 to 4 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 3 to 0 | S1PF | 0011 | R/W | Source1 pixel format<br>Set the same value as the S0PF field of S0CR register.<br>3: 8bit/pixel<br>4: 16bit/pixel<br>5: 32bit/pixel<br>Settings other than above are prohibited. |

### U30.2.20 Transaction control Register 0 (TCR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | WRM_EN | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R/W | R | R | R | R | R | R | R | R | R | R | R |

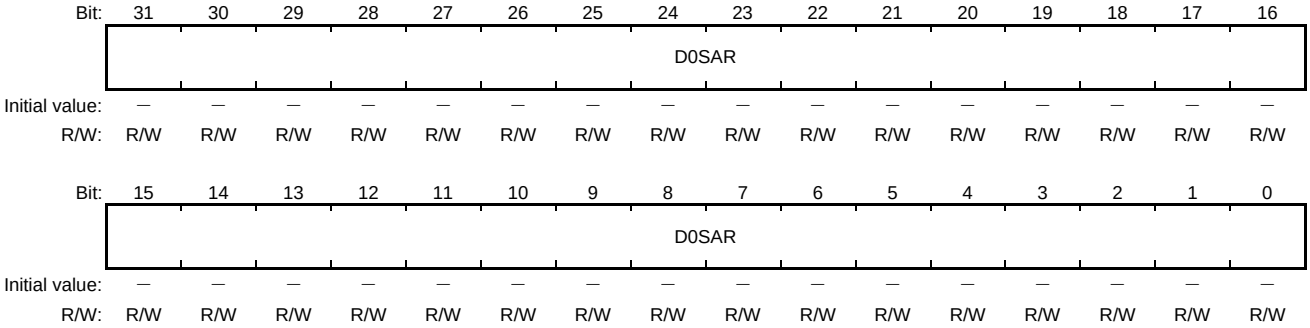
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | — | H'0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 27 | WRM_EN | 0 | R/W | Enabling minimum interval control of write request issuance<br>This bit limits the interval of write request issuance by 1. |
| 26 to 16 | — | H'000 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 15 to 0 | WRM_CYCLE | H'0000 | R/W | Setting minimum interval of write request issuance<br>When WRM_EN=1, the write request is issued with the write request issuance interval greater than the value set in this bit.<br>When writing to the internal memory, set an appropriate value of 32 or more according to the number of IPs using internal memory. |

### U30.2.21 Destination0 Start Address Register (D0SAR)

This register is used to set the start address of Destination0 to write to external memory or internal memory.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | D0SAR | H'0000 0000 | R/W | Set the start address of Destination0 in byte unit.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If DST0_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If DST0_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If D0PF bits of D0CR register are 3, set this register in 1-byte unit.<br>・If D0PF bits of D0CR register are 4, set this register in 2-byte unit.<br>・If D0PF bits of D0CR register are 5, set this register in 4-byte unit. |

### U30.2.22 Destination0 Stride Register (D0STR)

This register is used to set the memory width of Destination0 to write to external memory or internal memory.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | D0STR | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 15 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 14 to 0 | D0STR | H'0000 | R/W | Memory width of Destination0 is set in byte unit. The setting range is 1~32767.<br>Be sure to set the number of pixels, which is set in the IWIDTH bit of IMGSIZER register, to be equal to or greater than the byte converted value.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If DST0_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If DST0_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If D0PF bits of D0CR register are 3, set this register in 1-byte unit.<br>・If D0PF bits of D0CR register are 4, set this register in 2-byte unit.<br>・If D0PF bits of D0CR register are 5, set this register in 4-byte unit. |

### U30.2.23 Destination0 Control Register (D0CR)

This register is used to control Destination0.

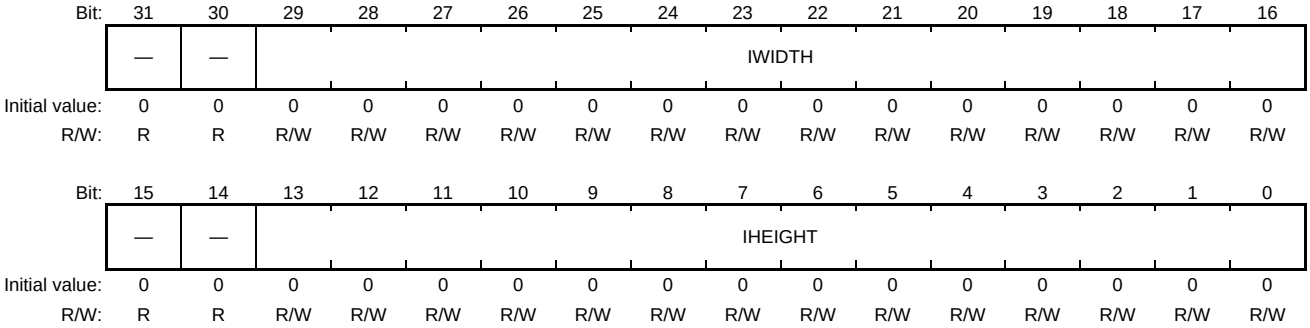
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | D0TAMODE | | | | — | — | — | — | — | — | — | — | D0PF | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 15 to 12 | D0TAMODE | 0000 | R/W | Destination0 tile addressing mode<br>0: Write Destination0 by the linear addressing.<br>1: Write Destination0 by the IMP original tile addressing.<br>For details of tile addressing, refer to chapter Operation.<br>Settings other than above are prohibited. If they are set, the operation cannot be guaranteed.<br>If this bit is 1, set D0PF bits in this register to 3 or 4.<br>If this bit is 1, setting the S1REN bit of S1CR register to 1 is prohibited.<br>If this bit is 1, setting the BITCNTEN bit of FCR0 register to 1 is prohibited.<br>When PXALIGN of FCR1 register is 1, setting this bit to 1 is prohibited.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than 0000 is written here. |
| 11 to 4 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 3 to 0 | D0PF | 0011 | R/W | Destination0 pixel format<br>When FCR0.BITCNTEN = 0, specify the same pixel format as the S0PF bit of S0CR register.<br>If BITCNTEN bit of FCR0 register = 1 and S0PF, S1PF are 8bpp, 16bpp, 32bpp, set this field to 8bit/pixel.<br>3: 8bit/pixel<br>4: 16bit/pixel<br>5: 32bit/pixel<br>Settings other than above are prohibited. |

### U30.2.24 Image Size Register (IMGSIZER)

This register is used to set the processing size (width and height) of Source and Destination. It is set in pixel unit.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31, 30 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 29 to 16 | IWIDTH | H'0000 | R/W | These bits specify the width of the processed image.<br>The horizontal size of Source and Destination is specified in pixel unit.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>If SRC0_TSIZE bit, SRC1_TSIZE bit, and DST0_TSIZE bit of TSCR register are 2, set the number of pixels so that it falls within the following ranges.<br>・If S0PF bit of S0CR register is 3, set within the range from 64 to 8192 pixels in 64-pixel unit.<br>・If S0PF bit of S0CR register is 4, set within the range from 32 to 4096 pixels in 32-pixel unit.<br>・If S0PF bit of S0CR register is 5, set within the range from 16 to 2048 pixels in 16-pixel unit.<br>If SRC0_TSIZE bit, SRC1_TSIZE bit, and DST0_TSIZE bit of TSCR register are 3, set the number of pixels so that it falls within the following ranges.<br>・If S0PF bit of S0CR register is 3, set within the range from 128 to 8192 pixels in 128-pixel unit.<br>・If S0PF bit of S0CR register is 4, set within the range from 64 to 4096 pixels in 64-pixel unit.<br>・If S0PF bit of S0CR register is 5, set within the range from 32 to 2048 pixels in 32-pixel unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If S0PF bit of S0CR register is 3, set within the range from 1 to 8192 pixels in 1-pixel unit.<br>・If S0PF bit of S0CR register is 4, set within the range from 1 to 4096 pixels in 1-pixel unit.<br>・If S0PF bit of S0CR register is 5, set within the range from 1 to 2048 pixels in 1-pixel unit. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15, 14 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 13 to 0 | IHEIGHT | H'0000 | R/W | These bits specify the height of the processed image.<br>The vertical size of Source and Destination is specified in line unit.<br>Set the number of lines so that it is within the following ranges.<br>・If S0PF bit of S0CR register is 3, set within the range from 1 to 8192 lines.<br>・If S0PF bit of S0CR register is 4, set within the range from 1 to 4096 lines.<br>・If S0PF bit of S0CR register is 5, set within the range from 1 to 2048 lines. |

### U30.2.25 Function Control Register 0 (FCR0)

This register is used to control functions of IMP DMAC

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BITCNTEN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | LOP | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | BITCNTEN | 0 | R/W | Bit count enable setting<br>0: Do not operate bit count<br>1: Operate bit count<br>During bit count operation, the number of 1 after logical operation is counted.<br>If this bit is 1, set LOP bits of FCR0 register to H'66 or H'99.<br>Set this bit to 0 when S0TAMODE bits of S0CR register are 1 or when D0TAMODE bits of D0CR register is 1.<br>When PXALIGN of FCR1 register is 1, set this bit to 0.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 30 to 8 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 0 | LOP | H'CC | R/W | Logical operation setting<br>When Source0 is s0 and Source1 is s1,<br>the following logical operations will be performed.<br>H'00 : All 0<br>H'11 : ~(s0 \| s1)<br>H'22 : (~s0) & s1<br>H'33 : ~s0<br>H'44 : s0 & (~s1)<br>H'66 : s0 ^ s1<br>H'77 : ~(s0 & s1)<br>H'88 : s0 & s1<br>H'99 : ~(s0^ s1)<br>H'BB : (~s0) \| s1<br>H'CC : s0<br>H'DD : s0 \| (~s1)<br>H'EE : s0 \| s1<br>H'FF : All 1<br>Settings other than above are prohibited.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than H'CC is written here. |

### U30.2.26 Function Control Register 1 (FCR1)

**[IMP DMAC]**

This register is used to control functions of IMP DMAC

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PXALIGN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | SHIFT | | | | | RNDMD | | | | SATEN | — | SATW | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | PXALIGN | 0 | R/W | Pixel alignment mode enable<br>0 : 64-byte or 128-byte alignment mode<br>1 : Pixel alignment mode<br>This bit specifies the units that can be set to S0SAR, S0STR, S1SAR, S1STR, D0SAR, and DSTR registers. For the settable units, refer to each register. |
| 30 to 28 | — | 001 | R | Reserved<br>The read value should always be 1.<br>Operation cannot be guaranteed if a value other than 1 is written. |
| 27 to 13 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 12 to 8 | SHIFT | 00000 | R/W | Setting of shift amount after LOP<br>These bits set the amount of shift to be performed for LOP output.<br>-7~-1 : Logical right-shift<br>0 : No shift<br>1~7: Logical left-shift<br>Set it by two's complement.<br>The shift result is output in 16 bits.<br>Set this bit to 0 unless all the following register settings are satisfied.<br>・PXALIGN bit of FCR1 register = 1<br>・S0PF bits of S0CR register = 4<br>・S1PF bits of S1CR register = 4<br>・D0PF bits of D0CR register = 4 |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 4 | RNDMD | 0000 | R/W | Setting rounding mode after shift<br>These bits set the rounding mode for right-shift.<br>0: Towards neg.infinity (Round to negative infinity)<br>9: Towards nearest・Towards pos.infinity (Round to nearest/positive infinity)<br>Settings other than above are prohibited.<br>Set this bit to 0 except when SHIFT bits are from -7 to -1.<br>Set this bit to 0 unless all the following register settings are satisfied.<br>・PXALIGN bit of FCR1 register = 1<br>・S0PF bits of S0CR register = 4<br>・S1PF bits of S1CR register = 4<br>・D0PF bits of D0CR register = 4 |
| 3 | SATEN | 0 | R/W | Saturation process enable<br>This bit specifies whether or not saturation process is performed for the rounded 16-bit data.<br>0: Do not process saturation.<br>1: Process saturation.<br>Set the bit width to process saturation by SATW bits.<br>Set this bit to 0 unless all the following register settings are satisfied.<br>・PXALIGN bit of FCR1 register = 1<br>・S0PF bits of S0CR register = 4<br>・S1PF bits of S1CR register = 4<br>・D0PF bits of D0CR register = 4 |
| 2 | — | 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 1, 0 | SATW | 00 | R/W | Saturation bit width selection<br>These bits set the bit width for saturation process when SATEN = "1".<br>Saturation is processed as unsigned data.<br>1: Saturate to 10 bits<br>2: Saturate to 12 bits<br>3: Saturate to 14 bits<br>If SATEN = 0, set this bit to 0.<br>If SATEN = 1, setting this bit to 0 is prohibited.<br>Set this bit to 0 unless all the following register settings are satisfied.<br>・PXALIGN bit of FCR1 register = 1<br>・S0PF bits of S0CR register = 4<br>・S1PF bits of S1CR register = 4<br>・D0PF bits of D0CR register = 4 |

**[Slim-IMP DMAC]**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PXALIGN | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | PXALIGN | 1 | R | Reserved<br>This bit always read as 1.<br>Operation cannot be guaranteed if a value other than 1 is written here. |
| 30 to 0 | — | H'10000000 | R | Reserved<br>These bits are always read as H'10000000.<br>Operation cannot be guaranteed if a value other initial value is written here. |

### U30.2.27 Command List Fetch Address Register (CLFAR)

This register is used to display the address of CL fetch destination.

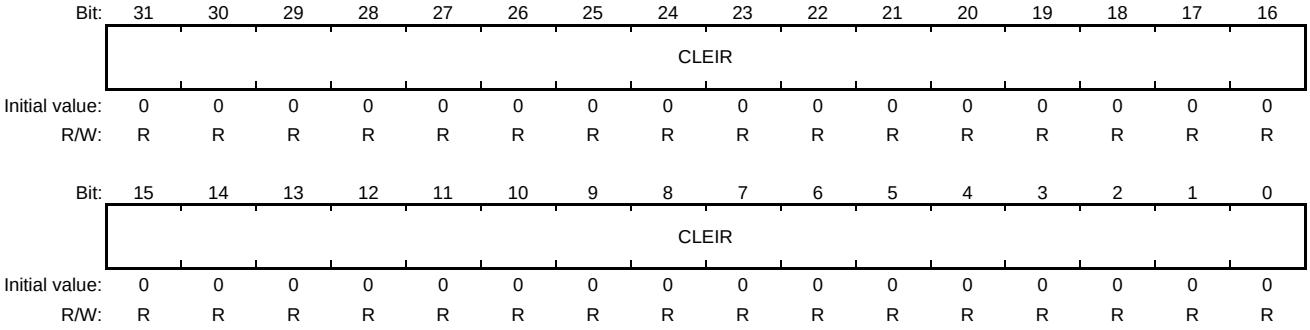
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLFAR | | | | | | | | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLFAR | | | | | | | | | | | | | | — | — |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | CLFAR | Undefined | R | These bits display bit31~bit2 of the address of CL fetch destination (during execution). |
| 1, 0 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |

### U30.2.28 Command List Execute Instruction Register (CLEIR)

This register is used to display the instruction during CL execution.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CLEIR | H'0000 0000 | R | These bits display the instruction during CL execution. |

### U30.2.29 RAM Test Register (RAMTSTR)

This is safety related register. Do not set this register during DMAC operation.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MEMINIT | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 30 to 16 | — | All 0 | R | Reserved.<br>These bits are always read as 0.<br>The write value should always be 0. |
| 15 | MEMINIT | 0 | R/W | DMAC Memory Initialize Bit<br>If this bit is set to "1", DMAC's SRAM area is initialized to "0". After setting this bit to "1", SYNCS command insert (waiting initialize complete) or CPU polling to this status bit is needed for data coherency.<br>Read:<br>0: Not under initialization<br>1: Under initialization<br>Write:<br>0: Invalid<br>1: Initiate initialization<br>*Do not write 1, when this bit is 1. |
| 14 to 0 | — | All 0 | R | Reserved.<br>These bits are always read as 0.<br>The write value should always be 0. |

### U30.2.30 DMAC Performance Control Register (DPCR)

This is the register to control the DPBSYCNT register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IFBSYCBTENB | DBSYCNTENB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 1 | IFBSYCNTENB | 0 | R/W | IFID operation count enable<br>When this bit is 1, CYCLE is counted while CL is running.<br>Setting both IFBSYCNTENB and DBSYCNTENB bit to 1 at same time is prohibited. |
| 0 | DBSYCNTENB | 0 | R/W | Data transmission operation count enable<br>When this bit is 1, CYCLE is counted while DMAEN = 1.<br>Setting both IFBSYCNTENB and DBSYCNTENB bit to 1 at same time is prohibited. |

### U30.2.31 DMAC Performance Busy Counter Register (DPBSYCNTR)

This register count cycles during operation.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPBSYCNTR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DPBSYCNTR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DPBSYCNTR | H'0000 0000 | R/W | This register count cycles during operation<br>This reads the value counted in the condition set in DMAC Performance Control Register (DPCR). Counted value of this register is not guarenteed .<br>When this register is written, the counter is set to the written value, and starts counts from set value onwards.<br>Writing this register during the counting is prohibited. |

### U30.2.32 Synchronization Control Register 0 (SYNCCR0)

This register specifies the module corresponding to bits 0 to 3 in the SYNCC enable field of WUP and SLP instructions. Set the number of the module you want to specify in this register. For module number, refer to **U34.IMP-X7_system, U34.10.5 Inter Module Synchronization**. Set this register in CPU before setting the CLE bit of SCTLR1 register to 1. Rewriting this register is prohibited while CLE bit is 1.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNCC3 | | | | | | | | SYNCC2 | | | | | | | |
| Initial value: | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SYNCC1 | | | | | | | | SYNCC0 | | | | | | | |
| Initial value: | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

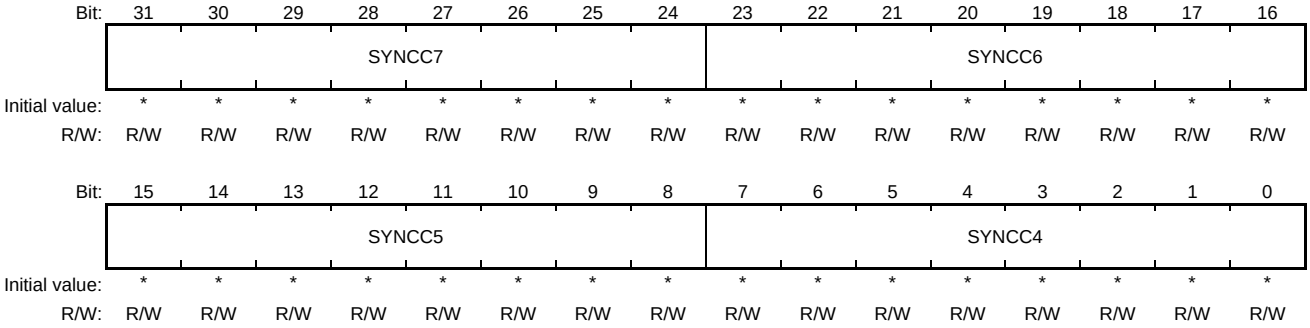
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC3 | * | R/W | This register specifies the module corresponding to bit 3 in the SYNCC enable field of WUP and SLP instructions. |
| 23 to 16 | SYNCC2 | * | R/W | This register specifies the module corresponding to bit 2 in the SYNCC enable field of WUP and SLP instructions. |
| 15 to 8 | SYNCC1 | * | R/W | This register specifies the module corresponding to bit 1 in the SYNCC enable field of WUP and SLP instructions. |
| 7 to 0 | SYNCC0 | * | R/W | This register specifies the module corresponding to bit 0 in the SYNCC enable field of WUP and SLP instructions. |

Note: The initial value to be read varies depending on the product.

Initial value is H'FFFFFFFF

### U30.2.33 Synchronization Control Register 1 (SYNCCR1)

This register specifies the module corresponding to bits 4 to 7 in the SYNCC enable field of WUP and SLP instructions. Set the number of the module you want to specify in this register. For module number, refer to **U34.IMP-X7_system, U34.10.5 Inter Module Synchronization**. Set this register in CPU before setting the CLE bit of SCTLR1 register to 1. Rewriting this register is prohibited while CLE bit is 1.

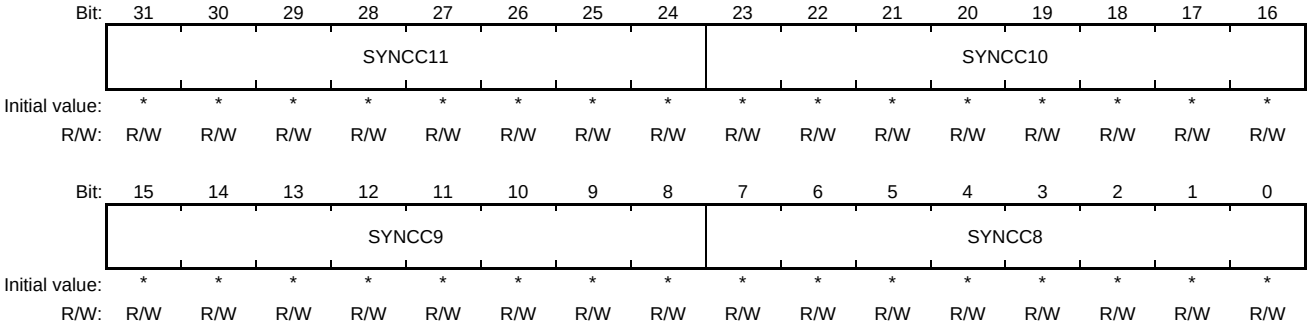



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC7 | * | R/W | This register specifies the module corresponding to bit 7 in the SYNCC enable field of WUP and SLP instructions. |
| 23 to 16 | SYNCC6 | * | R/W | This register specifies the module corresponding to bit 6 in the SYNCC enable field of WUP and SLP instructions. |
| 15 to 8 | SYNCC5 | * | R/W | This register specifies the module corresponding to bit 5 in the SYNCC enable field of WUP and SLP instructions. |
| 7 to 0 | SYNCC4 | * | R/W | This register specifies the module corresponding to bit 4 in the SYNCC enable field of WUP and SLP instructions. |

Note: The initial value to be read varies depending on the product.
Initial value is H'FFFFFFFF

### U30.2.34 Synchronization Control Register 2 (SYNCCR2)

This register specifies the module corresponding to bits 8 to 11 in the SYNCC enable field of WUP and SLP instructions. Set the number of the module you want to specify in this register. For module number, refer to **U34.IMP-X7_system, U34.10.5 Inter Module Synchronization**. Set this register in CPU before setting the CLE bit of SCTLR1 register to 1. Rewriting this register is prohibited while CLE bit is 1.

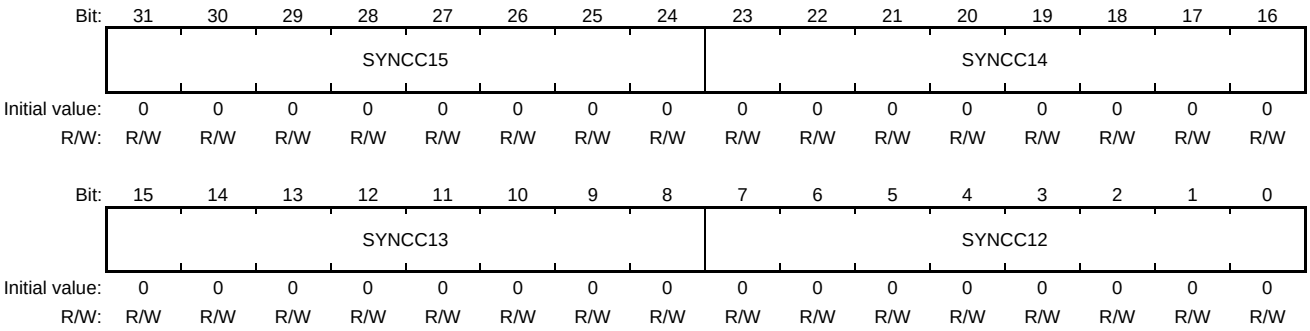



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC11 | * | R/W | This register specifies the module corresponding to bit 11 in the SYNCC enable field of WUP and SLP instructions. |
| 23 to 16 | SYNCC10 | * | R/W | This register specifies the module corresponding to bit 10 in the SYNCC enable field of WUP and SLP instructions. |
| 15 to 8 | SYNCC9 | * | R/W | This register specifies the module corresponding to bit 9 in the SYNCC enable field of WUP and SLP instructions. |
| 7 to 0 | SYNCC8 | * | R/W | This register specifies the module corresponding to bit 8 in the SYNCC enable field of WUP and SLP instructions. |

Note: The initial value to be read varies depending on the product.
Initial value is H'FFFFFFFF

### U30.2.35 Synchronization Control Register 3 (SYNCCR3)

This register specifies the module corresponding to bits 12 to 15 of the SYNCC enable field of WUP and SLP instruction. Set the number of the module you want to specify in this register. For module number, refer to **U34.IMP-X7_system, U34.10.5 Inter Module Synchronization**. Set this register in CPU before setting the CLE bit of SCTLR1 register to 1. Rewriting this register is prohibited while CLE bit is 1.

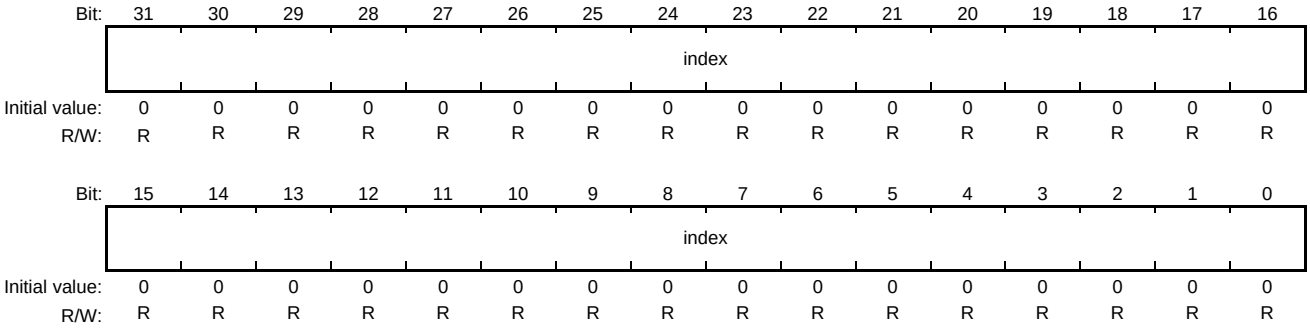



| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC15 | * | R/W | This register specifies the module corresponding to bit 15 in the SYNCC enable field of WUP and SLP instructions. |
| 23 to 16 | SYNCC14 | * | R/W | This register specifies the module corresponding to bit 14 in the SYNCC enable field of WUP and SLP instructions. |
| 15 to 8 | SYNCC13 | * | R/W | This register specifies the module corresponding to bit 13 in the SYNCC enable field of WUP and SLP instructions. |
| 7 to 0 | SYNCC12 | * | R/W | This register specifies the module corresponding to bit 12 in the SYNCC enable field of WUP and SLP instructions. |

Note: The initial value to be read varies depending on the product.
Initial value is H'FFFFFFFF

### U30.2.36 Index Register 0

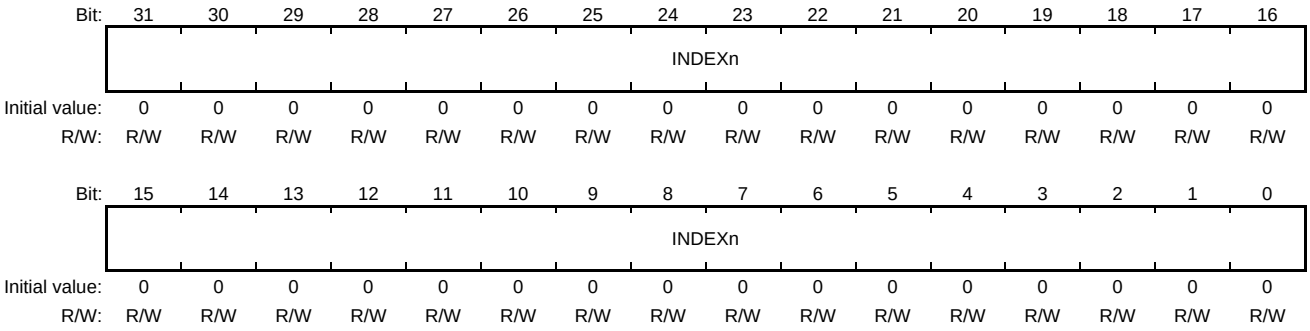
This is the register to control the source and destination addresses




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | index | H'0000 0000 | R | This is the index register.<br>Read value is always 0. |

### U30.2.37 Index Register (INDEXn) (n=1 to 7)

These are the registers to control the source and destination addresses.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | INDEXn | H'0000 0000 | R/W | This is the index register.<br>Value written in these register is added to the S0SAR, S1SAR and D0SAR according to the setting in Index Number in **U30A.4.3 WPR** command. Be sure to follow the alignment of S0SAR, S1SAR and D0SAR when setting these registers. Note that Slim-IMP DMAC does not use S1SAR. |

### U30.2.38 Wraparound enable register (WRAPER)

This is the register to control the SRC/DST wraparound min/max register register. This register is implemented for Slim-IMP DMAC only.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DST_WRAP_ENB | SRC_WRAP_ENB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 1 | DST_WRAP_ENB | 0 | R/W | Dst address wraparound enable<br>0: Not wraparound Dst address<br>1: Wraparound Dst address |
| 0 | SRC_WRAP_ENB | 0 | R/W | Src address wraparound enable<br>0: Not wraparound Src address<br>1: Wraparound Src address |

### U30.2.39 SRC minimum address wraparound register (WRAPS_MIN)

This is the register to control the SRC min wraparound address. This register is implemented for Slim-IMP DMAC only.

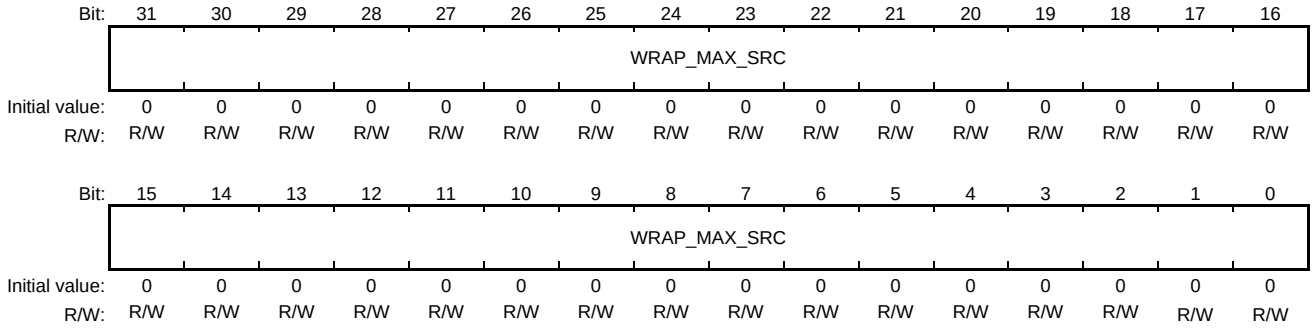
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WRAP_MIN_SRC | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WRAP_MIN_SRC | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | WRAP_MIN_SRC | All 0 | R/W | Minimum address of Src wraparound.<br>If src_wrap_enb of WRAPER register is 0, setting in this register is not used.<br>Start address of source (S0SAR) shall be in between the WRAPS_MIN and WRAPS_MAX address.<br>WRAPS_MIN < WRAPS_MAX<br>set this register in 512-byte alignment. (Bits 0~8 shall be 0). |

### U30.2.40 SRC maximum address wraparound register (WRAPS_MAX)

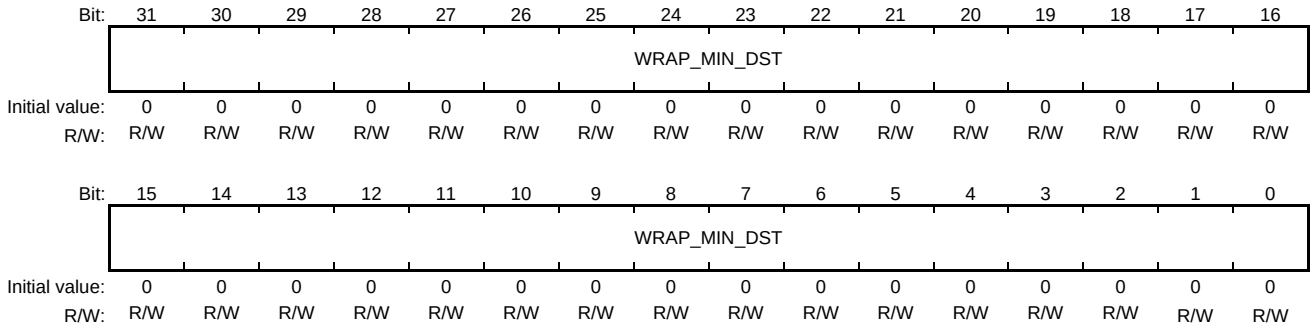
This is the register to control the SRC min wraparound address. This register is implemented for Slim-IMP DMAC only.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | WRAP_MAX_SRC | All 0 | R/W | Maximum address of Src wraparound.<br>If src_wrap_enb of WRAPER register is 0, setting in this register is not used.<br>Start address of source (S0SAR) shall be in between the WRAPS_MIN and WRAPS_MAX address.<br>WRAPS_MIN < WRAPS_MAX<br>(WRAPS_MAX-WRAPS_MIN+1) shall be in 512-byte alignment.<br>For ex: When wraparound between 512-byte and 1024-byte.<br>WRAPS_MIN = 0x00000200 (512-byte).<br>(WRAPS_MAX+1 ) shall be in 512 byte alignment.<br>(WRAPS_MAX+1 )= (0x000003FF+1) =0x00000400. |

### U30.2.41 DST minimum address wraparound register (WRAPD_MIN)

This is the register to control the SRC min wraparound address. This register is implemented for Slim-IMP DMAC only.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | WRAP_MIN_DST | All 0 | R/W | Minimum address of Dst wraparound.<br>If dst_wrap_enb of WRAPER register is 0, setting in this register is not used.<br>Start address of destination (D0SAR) shall be in between the WRAPD_MIN and WRAPD_MAX address.<br>WRAPD_MIN < WRAPD_MAX<br>set this register in 512-byte alignment. (Bits 0~8 shall be 0). |

### U30.2.42 DST maximum address wraparound register (WRAPD_MAX)

This is the register to control the SRC min wraparound address. This register is implemented for Slim-IMP DMAC only.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WRAP_MAX_DST | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WRAP_MAX_DST | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | WRAP_MAX_DST | All 0 | R/W | Maximum address of Dst wraparound.<br>If dst_wrap_enb of WRAPER register is 0, setting in this register is not used.<br>Start address of destination (D0SAR) shall be in between the WRAPD_MIN and WRAPD_MAX address.<br>WRAPD_MIN < WRAPD_MAX<br>(WRAPD_MAX-WRAPD_MIN+1) shall be in 512-byte alignment.<br>For ex: When wraparound between 512-byte and 1024-byte.<br>WRAPD_MIN = 0x00000200 (512-byte).<br>(WRAPD_MAX+1 ) shall be in 512 byte alignment.<br>(WRAPD_MAX+1 )= (0x000003FF+1) =0x00000400. |

### U30.2.43 Transaction control Register 1 (TCR1)

This controls the issue cycle of Src0,1 read request.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

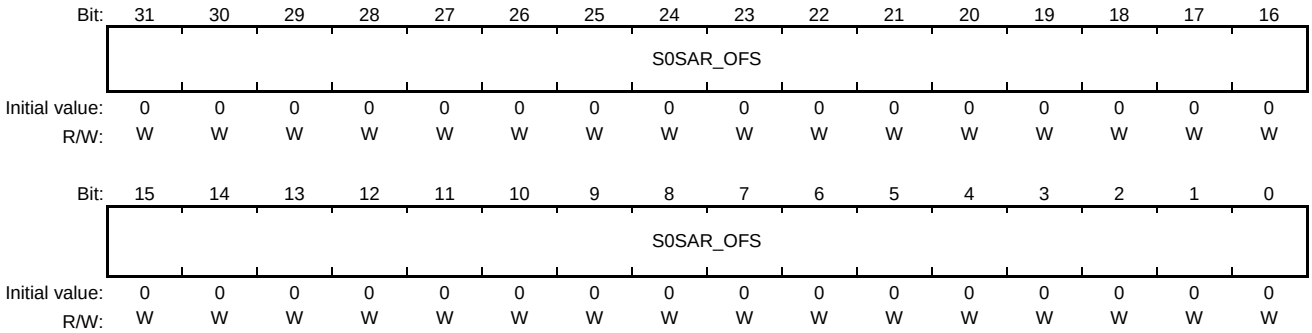
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1RM_CYCLE | | | | | | | | S0RM_CYCLE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 8 | S1RM_CYCLE | H'00 | R/W | Setting minimum interval to issue the read request of Src1<br>The read request is issued with the read request issuance interval greater than the value set in this bit.<br>When reading from the internal memory, set an appropriate value of 32 or more according to the number of IPs using internal memory.<br>**[Slim-IMP DMAC]**<br>Reserved.<br>Operation cannot be guaranteed if a value other than H'00 is written here. |
| 7 to 0 | S0RM_CYCLE | H'00 | R/W | Setting minimum interval to issue the read request of Src0<br>The read request is issued with the read request issuance interval greater than the value set in this bit.<br>When reading from the internal memory, set an appropriate value of 32 or more according to the number of IPs using internal memory. |

### U30.2.44 Source0 Start Address Offset Register (S0SAR_OFS)

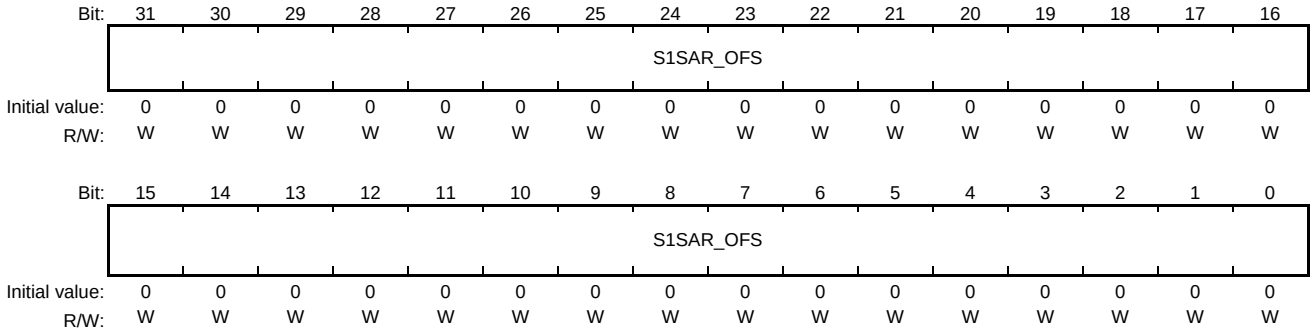
This register is used to set the offset to be added to the start address of Source0.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | S0SAR_OFS | H'0000 0000 | W | Set the offset value to be added to the start address of Source0 in byte unit.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If SRC0_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If SRC0_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If S0PF bits of S0CR register are 3, set this register in 1-byte unit.<br>・If S0PF bits of S0CR register are 4, set this register in 2-byte unit.<br>・If S0PF bits of S0CR register are 5, set this register in 4-byte unit. |

### U30.2.45 Source1 Start Address Offset Register (S1SAR_OFS)

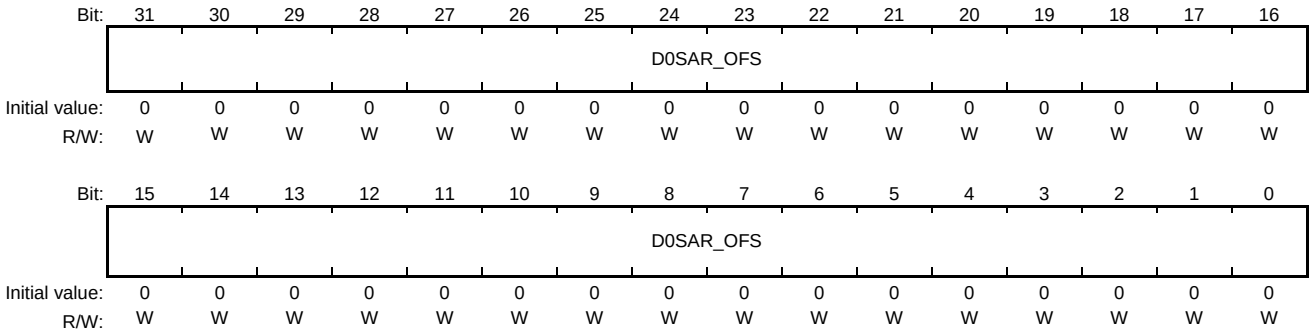
This register is used to set the offset to be added to the start address of Source1.
Note: This register is not support for Slim-IMP DMAC. Do not set this register for Slim-IMP DMAC operation.
This register is always read as 0x00000000.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | S1SAR_OFS | H'0000 0000 | W | Set the offset value to be added to the start address of Source1 in byte unit.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If SRC1_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If SRC1_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If S1PF bits of S1CR register are 3, set this register in 1-byte unit.<br>・If S1PF bits of S1CR register are 4, set this register in 2-byte unit.<br>・If S1PF bits of S1CR register are 5, set this register in 4-byte unit. |

### U30.2.46 Destination0 Start Address Offset Register (D0SAR_OFS)

This register is used to set the offset to be added to the start address of Destination0.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | D0SAR_OFS | H'0000 0000 | W | Set the offset value to be added to the start address of Destination0 in byte unit.<br>When PXALIGN of FCR1 register is 0, set according to the following.<br>・If DST0_TSIZE bits of TSCR register are 2, set this register in 64-byte unit.<br>・If DST0_TSIZE bits of TSCR register are 3, set this register in 128-byte unit.<br>When PXALIGN of FCR1 register is 1, set according to the following.<br>・If D0PF bits of D0CR register are 3, set this register in 1-byte unit.<br>・If D0PF bits of D0CR register are 4, set this register in 2-byte unit.<br>・If D0PF bits of D0CR register are 5, set this register in 4-byte unit. |

### U30.2.47 CL Break Address Register (CLBRKADDRR)

This is the register to control the CL break.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLBRKADDRR[31:16] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLBRKADDRR[15:0] | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CLBRKADDR | All 0 | R/W | CL break address. Specify the address(break point) of command list.<br>Specify a 32 bit (4-byte) command list address.<br>Data address is prohibited. |

### U30.2.48 CL Conditional GOSUB Register(CLCNDGSBR)

his is the register to control the GOSUB command.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | | — | — | — | — | — | — | — | CLCND GSBR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 0 | CLCNDGSBR | 0 | R/W | Condition for conditional GOSUB instruction:<br>1: GOSUB is executed<br>0: GOSUB is not executed, NOP is instead executed<br>(Bit 22 of the GOSUB command should be 1 for NOP operation) |

## U30.3 Operation

### U30.3.1 Data-Path configuration

The configuration of Data-Path built in IMP DMAC is shown below.

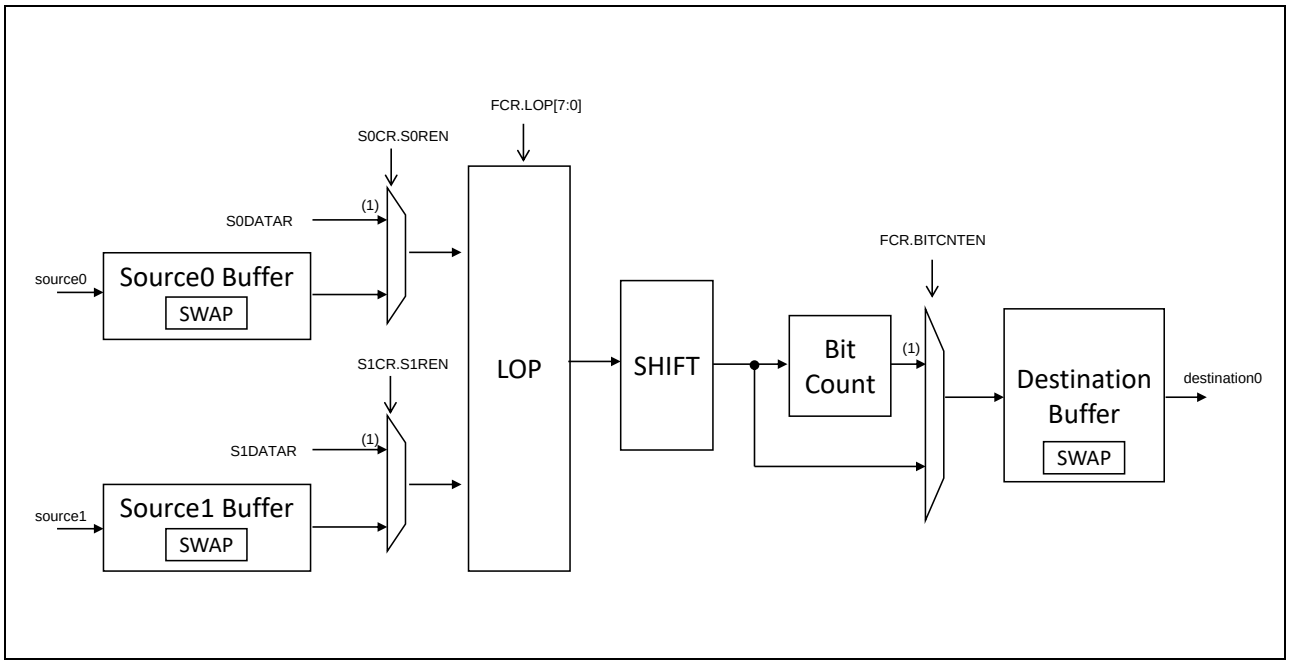



**Figure U30.3 Data-Path block diagram [IMP DMAC]**

**[IMP DMAC]**

The source input (Source0, Source1) and the register setting fixed value (S0DATAR, S1DATAR) are selected by S0REN, S1REN bits of S0CR, S1CR registers.
The data is inverted by INVS0D and INVS1D bits of FCR0 register, then Logic Operation (LOP) is performed according to the set value of LOP bits in FCR0 register. The result of LOP is shifted according to the setting of FCR1 register and then output to the destination buffer. By enabling the BITCNTEN bit of FCR0 register, it is possible to output the number of 1 resulting from the shift process to the destination buffer.

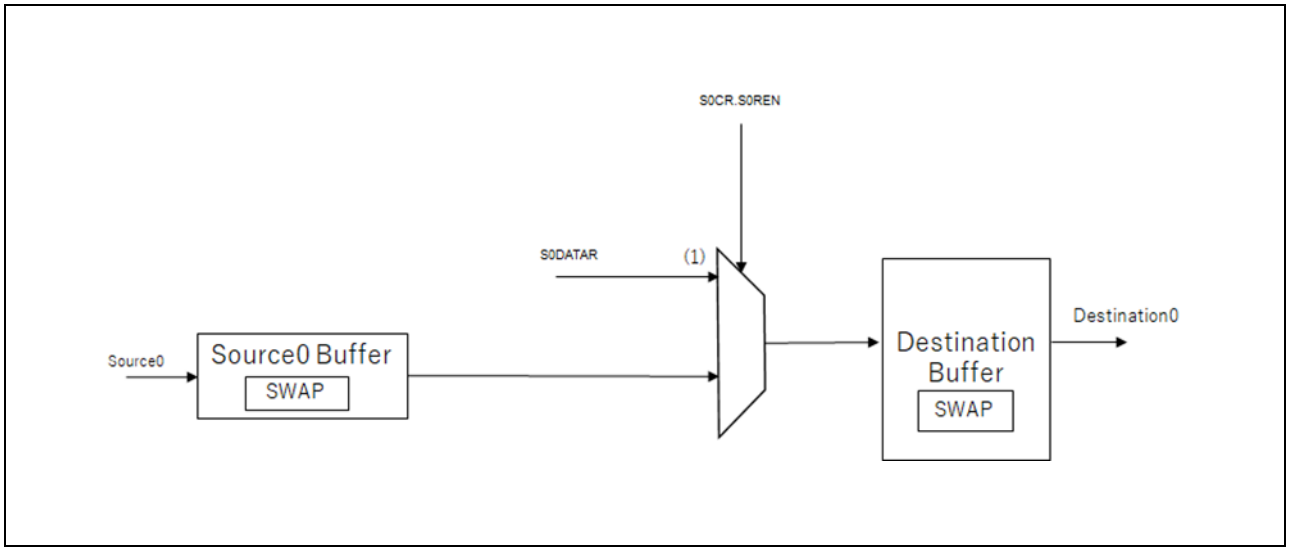



**Figure U30.4 Data-Path block diagram [Slim-IMP DMAC]**

**[Slim-IMP DMAC]**

The S0REN bits of the S0CR register is used to select whether memory or the S0DATAR is to be used as sources 0.The logical operation (LOP), BITCNTEN and shifting feature is not implemented in Slim-IMP DMAC.

### U30.3.2 Shift

Note: This feature is not support for Slim-IMP DMAC**.**

The shift is valid when PXALIGN bit of FCR1 register = 1, S0PF bits of S0CR register = 4, S1PF bits of S1CR register = 4, and D0PF bits of D0CR register = 4. When shift is enabled, shifting, rounding, and saturation are performed for the output data of LOP according to the setting of FCR1 register.
When left shifting, 0 is put in the lower bit without holding the sign. When right shifting, the sign or 0 is put in the upper level according to the data format. When right shifting, it is possible to round the shift output.
The shift result is output in 16 bits. When rounding with right shift, the result of rounding to positive infinity is output as 16 bits. If the left shift or shift amount is 0, set the rounding mode to rounding down.
When saturation is enabled, the shifted or rounded 16 bits data is regarded as unsigned 16 bits data, and saturated to either unsigned 10 bits, unsigned 12 bits or unsigned 14 bits.
When it is saturated with unsigned 10 bits/ unsigned 12 bits/ unsigned 14 bit, the upper bits are filled with 0 to output as 16 bits.
Refer to FCR1 register for shift amount setting, rounding setting and saturation setting.

### U30.3.3 Tile addressing

Note: This feature is not supported for Slim-IMP DMAC.

This module supports not only linear addressing where the two-dimensional image data is arranged in memory as a whole in raster scan order, but also a tile addressing mode where the data is arranged tile wise. In tile addressing mode, continuous data in x direction is interleaved with data from other rows, thus improving the cache-hit ratio for vertical access to the image data (e.g., for rotation of an image) and also improving processing performance.

In tile addressing mode, the data are stored with the data incremented in 16-byte units within 64-byte sections. An example where 8-bpp data are exclusively output is shown below. (x, y) in the figure below indicate the x and y coordinates of pixels.

The tile addressing mode can be set up individually for source and destination images.

To set up the tile addressing mode for the source0 image, set the S0TAMODE bit in S0CR to 1.

To set up the tile addressing mode for the destination0 image, set the D0TAMODE bit in D0CR to 1.

To set the stride in tile addressing mode, set the corresponding bits in D0STR and S0STRto a value equal to one quarter of the actual memory stride.

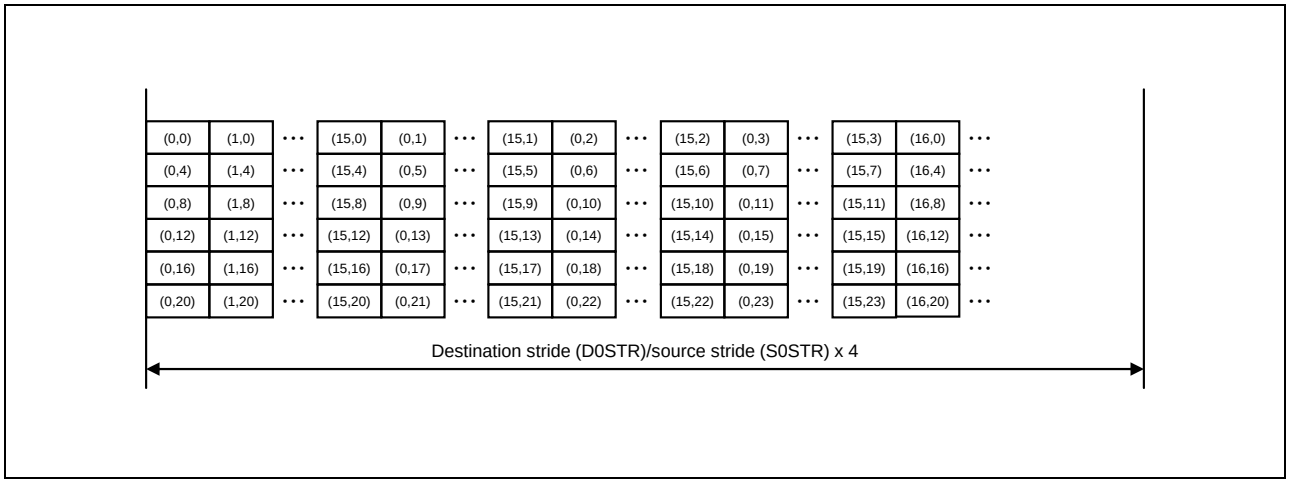



**Figure U30.5 Image of data arrangement on memory in tile addressing mode**

### U30.3.4 Inter Module Synchronization System

The synchronization system between modules is the system that synchronizes the control of each CL Controller by specifying the target module and executing the WUP instruction and the SLP instruction for IMP core, CVe, IMP DMAC, IMP PSC, IMP CNN, slim-IMP DMAC, VDSP and IMPDTA.

For the WUP instruction and the SLP instruction in IMP DMAC, refer to “U30.4.9 WUP” and “U30.4.10 SLP”. For target module specification, refer to the next section.

#### U30.3.4.1 Target module specification

Refer to the **U34.IMP-X7_system, U34.10.5 Inter Module Synchronization** chapter for the details.

### U30.3.5 Data compression

For the DATA compression, refer to 63. FCPR HWM for the details. Note that transmission size(TSCR register) of the IMPDMAC/Slim-IMP DMAC should be 256 byte when using DATA compression.

Note: It is prohibited to compress the Command List (CL), so place the CL only to the uncompressed region. IMPDMAC can pre-fetch the CL of maximum 640-byte. Therefore, do not use last 640-byte of uncompressed region before the compressed region.

### U30.3.6 Addition of offset to start address

Note: This feature is not supported for Slim-IMP DMAC.

IMP DMAC can add an offset to the start address of Source0, Source1, Destination0.
By writing the offset value to S0SAR_OFS, S1SAR_OFS*, and D0SAR_OFS, the value obtained by adding the offset specified in S0SAR, S1SAR, and D0SAR is written. If offset is not added, write the value directly to S0SAR, S1SAR, and D0SAR. In addition, the read values of S0SAR_OFS, S1SAR_OFS*, and D0SAR_OFS are always 0.

### U30.3.7 CL break

To use the CL break function, following setting is required.

1. Specify the break point address of command list. Be sure that break point is command address of command list.

2. Set CLBRKENB of SER register to 1 and if interrupt is required, set CLBRKM bit in IMR bit to use CL break function.
3. Check the CLBRK status in SR0 register.

4. To continue execution after CL break, Set the CLBRKC bit of SCR register to 1.

## U30.4 Command List(CL)

It is prohibited from using instructions other than those described in this section.

Table 30.4 **Instruction list**

| Instruction | OP CODE | Description |
|---|---|---|
| NOP | H'80 | No Operation |
| TRAP | H'88 | Close the CL |
| WPR | H'55 | Write to register |
| INT | H'81 | Generate an interrupt cause and wait until the interrupt is cleared |
| SYNCS | H'8B | Wait for the completion of DMA transmission execution |
| Reserved | H'8C | This instruction is not allowed to use<br>With this instruction, no illegal instruction interrupt occurs |
| Reserved | H'8D | This instruction is not allowed to use<br>With this instruction, no illegal instruction interrupt occurs |
| JUMP | H'90 | Jump to the specified address |
| GOSUB | H'91 | Jump to the specified address |
| RET | H'99 | Jump to the instruction next to the GOSUB |
| WUP | H'A0 | Release the SLP of other modules |
| SLP | H'A1 | Wait for another module to execute WUP |
| Reserved | H'B0 | This instruction is not allowed to use<br>With this instruction, no illegal instruction interrupt occurs |
| Reserved | H'B1 | This instruction is not allowed to use<br>With this instruction, no illegal instruction interrupt occurs |
| Reserved | H'C0 | This instruction is not allowed to use<br>With this instruction, no illegal instruction interrupt occurs |

### U30.4.1 NOP (No OPeration)

#### (1) Function

It is No Operation instruction.
The NOP instruction causes the IMP DMAC to do nothing for the number of clock cycles specified in the N field before execution of the next instruction.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_0000 (H'80) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | N (1≦N≦16) | | | | |

### U30.4.2 TRAP (TRAP)

#### (1) Function

This is the instruction to close the CL.
It closes the CL execution. When the TRAE bit of SER register is 1, the TRA bit of SR0 register is set to 1.
However, regardless of the value of the TRAE bit in SER register, the value specified in the CODE field is set to TRAPCODE of SR1 register.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_1000 (H'88) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | CODE | | | | | | | |

### U30.4.3 WPR (Write PaRameter)

#### (1) Function

This instruction performs writing to the register.
Register write is performed only by the number of fields N from the offset address of the register, which is specified in the ADD field. The value to be written to the register is the value of the DATA N-1 field in order from the DATA 0 field.
The ADD field specifies only the offset value in 1-byte unit. (Thread0 and Thread1 have the same address specification)
Do not set the N field to 0. The Index number field specifies the number of INDEX register so that value written to the selected Index register will be added to the S0SAR, S1SAR and D0SAR specified in DATA field of this command. The range of Index Number is (0 to 7). When using WPR for registers other than above, set Index Number=0.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0101_0101 (H'55) | | | | | | | | Index Number | | | ADD | | | | | | | | | | | | | N(>=1) | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

…

| DATA N-1 |
|---|

### U30.4.4 INT (INTerrupt)

#### (1) Function

This instruction generates the INT interrupt cause.
When the INTE bit of SER register is 1, the INT bit of SR0 register is set to 1.
Regardless of the value of the INTE bit in SER register, the value specified in the CODE field is set to INTCODE of SR1 register.
The next instruction is not started until the INT bit of SR0 register is cleared to 0.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_0001 (H'81) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | CODE | | | | | | | |

### U30.4.5 SYNCS (SYNChronize DMAC ProcesS)

#### (1) Function

This is the instruction to wait for the completion of DMA transmission execution.
This instruction waits for the completion of DMA transmission when there is a DMA transmission currently being executed. After waiting for the completion of DMA transmission, the next instruction is started.
If there is no DMA transmission being executed, it is the same as NOP (the next command is started).
Right after having set the DMAEN bit of SCTLR0 register to 1 by the WPR instruction, be sure to always wait for the completion of DMA transmission with the SYNCS instruction. If you do not wait for the DMA transmission to complete, the operation cannot be guaranteed.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_1011 (H'8B) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |

### U30.4.6 JUMP

#### (1) Function

This is the command to jump to the specified address.
It allows jumping to the address specified in JMPADD field. Setting 1 to RLT field uses relative address to JUMP command address and setting 0 in RLT field uses absolute address of the command list. Specify 4-byte unit command address in the JMPADD field.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1001_0000(H'90) | | | | | | | | RLT | Reserved（all 0） | | | | | | | | | | | | | | | | | | | | | | |
| JMPADD[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U30.4.7 GOSUB (Go Subroutine)

#### (1) Function

This is the command to save current CL address to the hardware buffer and jump to the specified address.
It allows jumping to the address specified in GSADD field and returning to the address next to the GOSUB command by RET command. Specify 4-byte unit command address in the GSADD field. Nesting from the GOSUB up to 4 level can be possible. Setting 1 to RLT field uses GSADD as relative address to GOSUB command address and setting 0 in RLT field uses absolute address of the command list. Setting 1 in CON field and 0 in CLCNDGSBR bit of CLCNDGSBR register change GOSUB command into NOP command. After the CLCNDGSBR register is set, a SYNCS command should be added before using the GOSUB command.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1001_0001 (H'91) | | | | | | | | RLT | CON | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | |
| GSADD[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U30.4.8 RET (RETurn)

#### (1) Function

This is the instruction to jump to the instruction next to the GOSUB.
RET allows jumping to the CL address next to the address at which the GOSUB instruction is executed.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1001_1001 (H'99) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |

### U30.4.9 WUP (WakeUP)

#### (1) Function

This is the instruction to cancel the SLP state of the module, which is specified in the SYNCC enable field, in the intermodule synchronization system. By setting bit 'm' of SYNCC enable field to 1, WUP signal is issued to the module set in SYNCCm(m=0~15) of SYNCCRn(n=0~3) register. Be sure to insert 8 or more instructions other than the WUP instruction between two WUP instructions in the CL. For details, refer to the operation explanation of the intermodule synchronization system.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1010_0000 (H'A0) | | | | | | | | Reserved (all 0) | | | | | | | | SYNCC enable | | | | | | | | | | | | | | | |

### U30.4.10 SLP (SLeep)

#### (1) Function

This is the instruction to change the own module into the SLP state in the intermodule synchronization system.
The SLP state is cancelled by the WUP instruction of the module specified in the SYNCC enable field.
By setting bit 'm' of SYNCC enable field to 1, wait for the WUP signal from the module set in SYNCCm(m=0~15) of SYNCCRn(n=0~3) register. For details, refer to the operation explanation of the intermodule synchronization system.

#### (2) Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1010_0001 (H'A1) | | | | | | | | Reserved (all 0) | | | | | | | | SYNCC enable | | | | | | | | | | | | | | | |

## U30.5 Usage Notes

### U30.5.1 Power off

Refer to chapter 58.7 Usage Notes in IMP-X7 HW Manual for the power off.

### U30.5.2 Module Standby Mode

Refer to chapter chapter 58.7 Usage Notes in IMP-X7 HW Manual for the Module Standby Mode.

### U30.5.3 CPG Software reset

Refer to chapter chapter 58.7 Usage Notes in IMP-X7 HWM Manual for the software reset.

## U30.6 Safety Mechanisms

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### U30.6.1 EDC protection.

EDC protection is implemented to SRAM and Bus interface in IMPDMAC/Slim-IMP DMAC. EDC check is always enabled. When using the EDC function of memory (after every power on reset or software reset by CPG or Runtime Test or POST), it is necessary to initialize the memories. Set the **U30.2.29 RAMTSTR** register for memory initialization in IMPDMAC/Slim-IMP DMAC. When using both the thread0 and thread1, set the memory initialization by only thread0.

After setting MEMINIT bit of RAMTSTR to "1", SYNCS command need to insert (waiting initialize complete) when writing from command list or CPU polling to this status bit is needed for data coherency.

Execute following steps from (1) to (3) to initialize IMP DMAC memory by CPU
< Procedure to initialize memory by CPU>

(1) Set MEMINIT bit to 1 in RAMTSTR register
(2) Read 8 times the RAMTSTR register
(3) Wait till RAMTSTR bit becomes 0 by polling further the RAMTSTR register

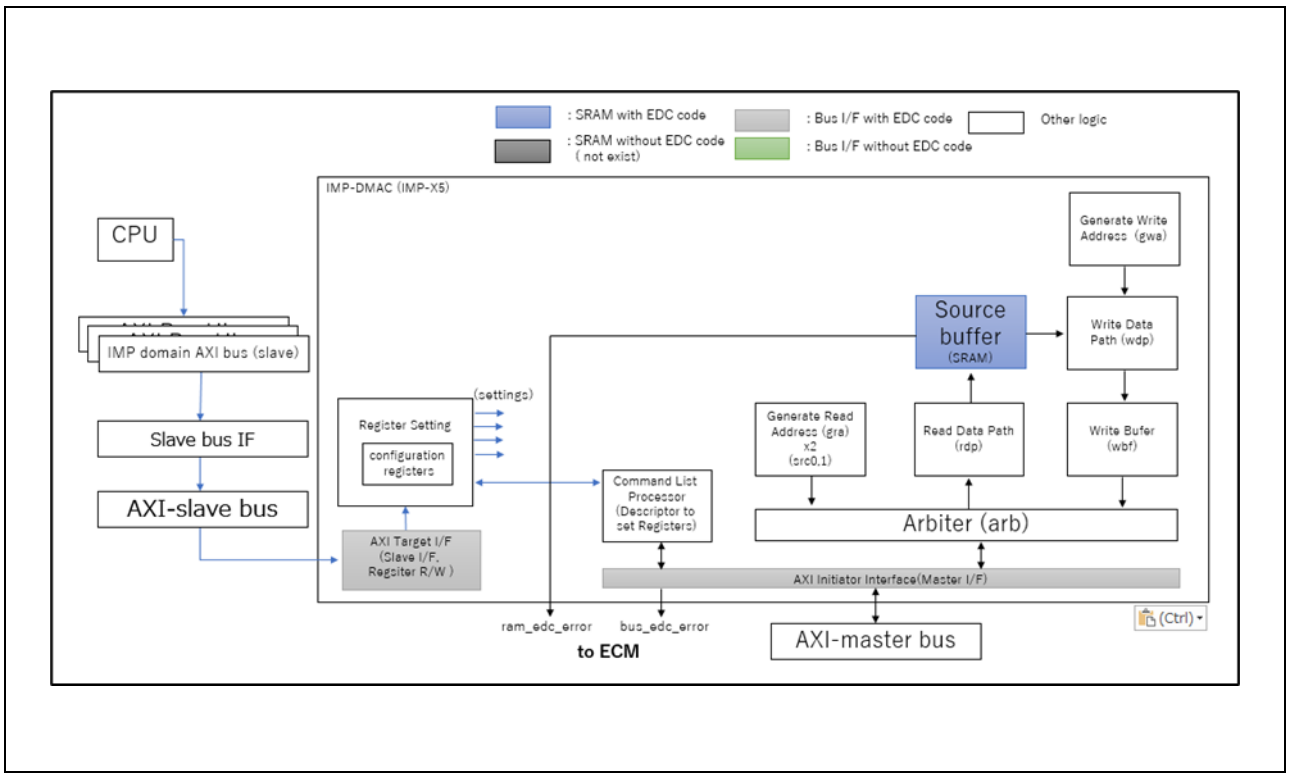



**Figure U30.6 Block Diagram of FUSA**

# U31. IMP PSC

## U31.1 Overview

The IMP PSC reads one image (hereinafter referred to as source image) from an external memory (hereinafter referred to as DDR-SDRAM) or an IMP dedicated internal memory (hereinafter referred to as "internal memory") and extracts up to four reduced images (hereinafter referred to as destination images) and writes these to DDR-SDRAM or Internal memory.

8 bpp (signed, unsigned), 10 bpp (unsigned), 12 bpp (unsigned), 16 bpp (signed) are supported as pixel format. The reduction ratio for each destination image (1 to 4), can be set between 0.5 to 1.0 for X and Y direction (step size 1/1048576). IMP PSC internal processing is done in little endian format.

The operation of IMP PSC is shown in the image below.

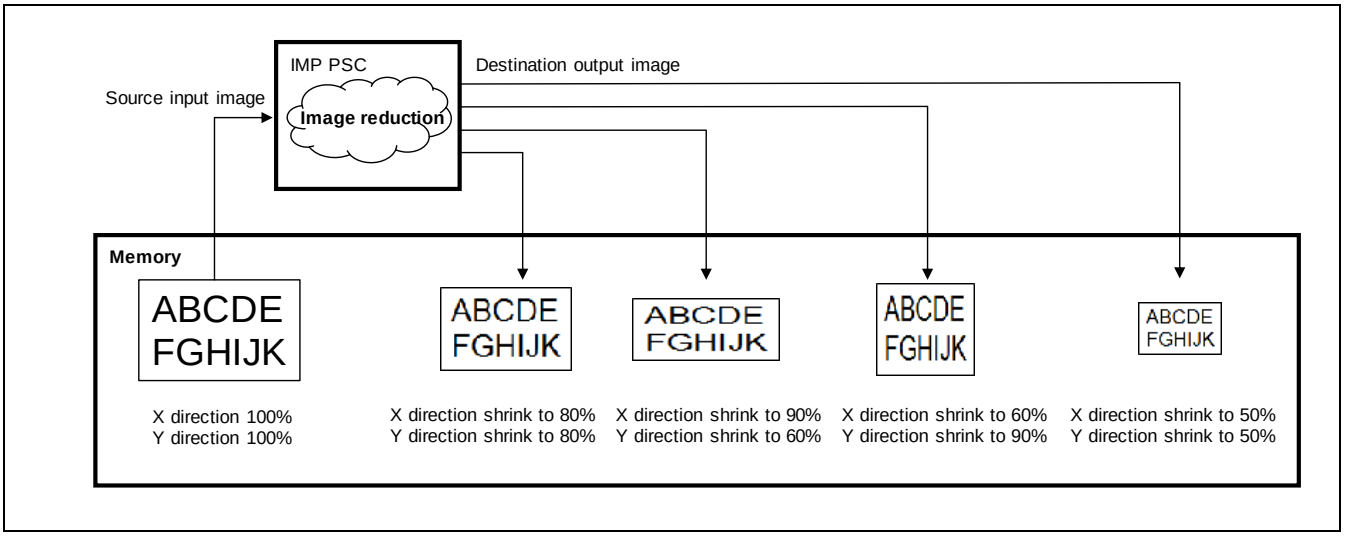



**Figure U31.1 Operation of the IMP PSC**

**Table U31B.1 Base Address of IMP PSC**

| Channel | Base Address |
|---|---|
| 0 | H'FFA8 4000 |

### U31.1.1 Features

- Command List (CL) system
- Register write instruction (WPR)
- Interrupt generation instruction (INT)
- Control instructions (NOP, TRAP, SYNCS, JUMP, GOSUB(stack based), RET)
- Conditional GOSUB, CL Break
- Relative and absolute addressing of JUMP and GOSUB instruction.
- Synchronization instructions (WUP, SLP) with other modules[*1]

- PSC
- Source image is input from DDR-SDRAM or from Internal memory by DMA transfer
- Reduced source image (Destination image) is output to DDR-SDRAM or Internal memory[*2] by DMA transfer
- Support of maximal 4096 pixels in the X direction and 4096 lines in the Y direction for source images and destination images[*3]
- Pixel format can be selected from unsigned 8, 10, 12 bpp or signed 8, 16 bpp[*4]
- The range setting in the X and Y directions of the reduction ratio is between 0.5 and 1.0[*5]
- Reduction operation is performed by bilinear interpolation
- IMP PSC activation with WPR instruction of CL[*6]

Notes: 1. For modules that can be synchronized with IMP PSC, see **U34.IMP-X7_system**.
2. 1 to 4 destination images can be output per source image.
3. When the pixel format is 10, 12, 16 bpp, the maximum in the X direction is 2048 pixels.
4. Please use the same pixel format for source and destination images.
5. It can be set with step size 1/1048576.
6. It is prohibited to write the EX bit of the EXCTLR register to 1 from the CPU to start the IMP PSC.

### U31.1.2 Block Diagram

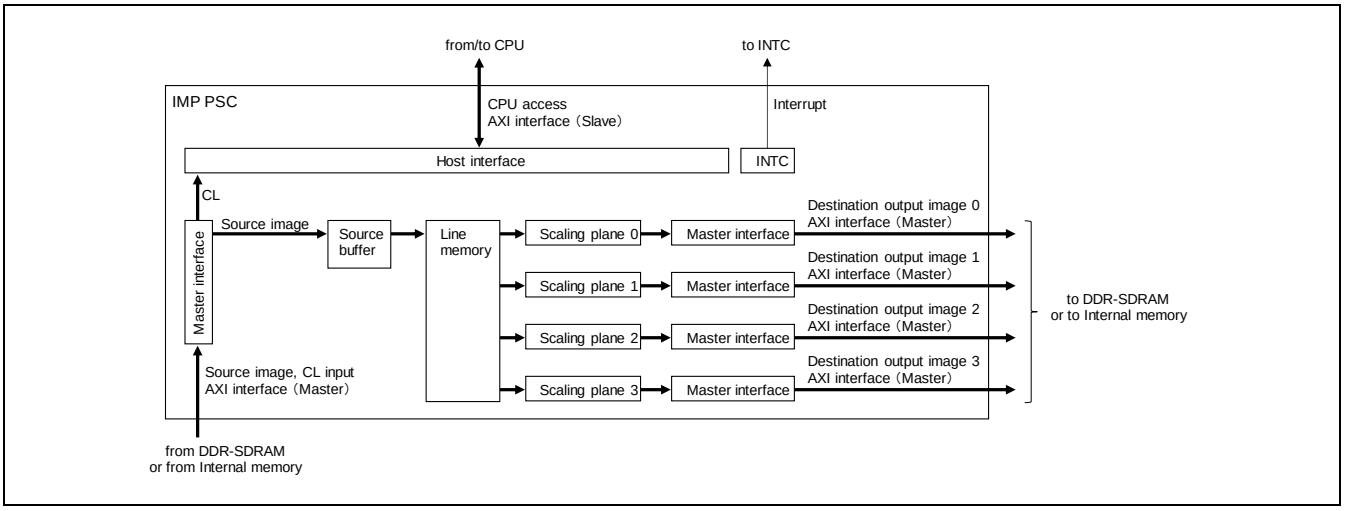



**Figure U31.2 Block Diagram**

— Host interface:

This is an interface module for AXI slave access. Handles AXI slave access (in 4-byte units only) set via registers.

CL is loaded from DDR-SDRAM or Internal memory by DMA transfer and executed.

— Interrupt control (INTC):

Controls output of various interrupt signals.

— Master interface:

This is an interface module for AXI master access.

— Source buffer:

Source image is input from DDR-SDRAM or Internal memory by DMA transfer and stored to Line memory.

— Line memory(LM):

This is the storage unit to realize the bilinear filtering.

— Scaling plane0-3:

According to register settings, the IMP PSC can generate 4 reduced-size destination images in parallel.

Destination image is output by DMA transfer to DDR-SDRAM or internal memory.

### U31.1.3 Register Configuration

Access to addresses not listed in this register list is prohibited. Operation is not guaranteed if access is made to an address not listed. Access size is 32-bit. Operation is not guaranteed if access is made except for 32-bit. Operation is not guaranteed when writing to registers that can not be accessed during IMP PSC execution.

**Table U31B.2 Register Configuration – Scaling Control Registers**

| Name | Symbol | R/W | Offset | CPU Write Access during IMP PSC execution[*1] | Initial value | Reset[*2] | WPR |
|---|---|---|---|---|---|---|---|
| Version Control Register | VCR | R | H'0000 | — | *3 | — | — |
| System Control Register 0 | SCTLR0 | R/W | H'0004 | — | H'0000 0000 | H | — |
| Status Register | SR | R | H'0008 | — | H'0000 0000 | S/H | — |
| Status Clear Register | SCR | -/W | H'000C | √ | H'0000 0000 | S/H | — |
| Status Enable Register | SER | R/W | H'0010 | √ | H'0000 0071 | S/H | — |
| Interrupt Mask Register | IMR | R/W | H'0014 | √ | H'0000 0471 | S/H | — |
| System Control Register 1 | SCTLR1 | R/W | H'0018 | — | H'0000 0000 | S/H | — |
| Command List Start Address Register | CLSAR | R/W | H'0038 | — | H'0000 0000 | S/H | — |
| Status Register1 | SR1 | R | H'003C | — | H'0000 0000 | S/H | — |
| Command List Fetch Address Register | CLFAR | R | H'0044 | — | Undefined | — | — |
| Command List Execute Instruction Register | CLEIR | R | H'0048 | — | H'0000 0000 | S/H | — |
| Synchronization Control Register 0 | SYNCCR0 | R/W | H'0058 | — | *4 | S/H | √ |
| Synchronization Control Register 1 | SYNCCR1 | R/W | H'0060 | — | *4 | S/H | √ |
| Synchronization Control Register 2 | SYNCCR2 | R/W | H'0064 | — | *4 | S/H | √ |
| Synchronization Control Register 3 | SYNCCR3 | R/W | H'0068 | — | *4 | S/H | √ |
| Performance Control Register | PCR | R/W | H'00E8 | — | H'00000000 | SH | — |
| Performance Busy Counter Register | PBSYCNTR | R/W | H'00EC | — | H'00000000 | SH | — |
| CL Break Address Register | CLBRKADDRR | R/W | H'03F4 | √ | H'00000000 | S/H | — |
| CL Conditinal GOSUB Register | CLCNDGSBR | R/W | H'03FC | √ | H'00000000 | S/H | — |

Notes: 1. IMP PSC execution is the state where the EX bit in the EXCTLR register = 1 or the CLE bit in the SCTLR1 register = 1.

2. H: Power-on reset, S: Software reset by the SWRST bit of the SCTLR register.

3. The value to be read varies depending on the product. Refer register description for the initial value.

4. Initial value is varies depending on the product. Refer register description for the initial value.

**Table U31B.3 Register Configuration - Memory Control Registers**

| Name | Symbol | R/W | Offset | CPU Write Access during IMP PSC execution[*1] | Initial value | Reset[*2] | WPR |
|---|---|---|---|---|---|---|---|
| Source Start Address Register | SSAR | R/W | H'0100 | — | Undefined | — | √ |
| Source Length Register | SLR | R/W | H'0104 | — | H'0000 0000 | S/H | √ |
| Source Stride Register | SSTR | R/W | H'0108 | — | H'0000 0000 | S/H | √ |
| Destination Start Address Register0 | DSAR0 | R/W | H'010C | — | Undefined | — | √ |
| Destination Start Address Register1 | DSAR1 | R/W | H'0110 | — | Undefined | — | √ |
| Destination Start Address Register2 | DSAR2 | R/W | H'0114 | — | Undefined | — | √ |
| Destination Start Address Register3 | DSAR3 | R/W | H'0118 | — | Undefined | — | √ |
| Destination Stride Register0 | DSTR0 | R/W | H'011C | — | H'0000 0000 | S/H | √ |
| Destination Stride Register1 | DSTR1 | R/W | H'0120 | — | H'0000 0000 | S/H | √ |
| Destination Stride Register2 | DSTR2 | R/W | H'0124 | — | H'0000 0000 | S/H | √ |
| Destination Stride Register3 | DSTR3 | R/W | H'0128 | — | H'0000 0000 | S/H | √ |
| Destination Length Register0 | DLNGR0 | R/W | H'0130 | — | H'0000 0000 | S/H | √ |
| Destination Length Register1 | DLNGR1 | R/W | H'0134 | — | H'0000 0000 | S/H | √ |
| Destination Length Register2 | DLNGR2 | R/W | H'0138 | — | H'0000 0000 | S/H | √ |
| Destination Length Register3 | DLNGR3 | R/W | H'013C | — | H'0000 0000 | S/H | √ |
| Data Control Register | DCTLR | R/W | H'0140 | — | H'0000 0000 | S/H | √ |
| Execute Control Register | EXCTLR | R/W | H'0144 | — | H'0000 0000 | S/H | √ |
| Execute Line Count Register | EXCNTR | R | H'0148 | — | H'0000 0000 | S/H | — |
| Destination Address Enable-Register 0 | DAER0 | R/W | H'0180 | — | H'FFFF FFFF | S/H | √ |
| Destination Address Enable-Register 1 | DAER1 | R/W | H'0184 | — | H'FFFF FFFF | S/H | √ |
| Destination Address Enable-Register 2 | DAER2 | R/W | H'0188 | — | H'FFFF FFFF | S/H | √ |
| Destination Address Enable-Register 3 | DAER3 | R/W | H'018C | — | H'FFFF FFFF | S/H | √ |

Notes: 1. IMP PSC execution is the state where the EX bit in the EXCTLR register = 1 or the CLE bit in the SCTLR1
register = 1.
2. H: Power-on reset
S: Software reset by the SWRST bit of the SCTLR register.

**Table U31B.4 Register Configuration - Scaling Control Registers**

| Name | Symbol | R/W | Offset | CPU Write Access during IMP PSC execution[*1] | Initial value | Reset[*2] | WPR |
|---|---|---|---|---|---|---|---|
| Scaling Parameter Register | SPR | R/W | H'0200 | — | H'0000 0000 | S/H | √ |
| Scaling Ratio Y for Destination Register 0 | SRYDR0 | R/W | H'0204 | — | H'0000 0000 | S/H | √ |
| Scaling Ratio X for Destination Register 0 | SRXDR0 | R/W | H'0208 | — | H'0000 0000 | S/H | √ |
| Scaling Offset Y for Destination Register 0 | SOYDR0 | R/W | H'020C | — | H'0000 0000 | S/H | √ |
| Scaling Offset X for Destination Register 0 | SOXDR0 | R/W | H'0210 | — | H'0000 0000 | S/H | √ |
| Scaling Ratio Y for Destination Register 1 | SRYDR1 | R/W | H'021C | — | H'0000 0000 | S/H | √ |
| Scaling Ratio X for Destination Register 1 | SRXDR1 | R/W | H'0220 | — | H'0000 0000 | S/H | √ |
| Scaling Offset Y for Destination Register 1 | SOYDR1 | R/W | H'0224 | — | H'0000 0000 | S/H | √ |
| Scaling Offset X for Destination Register 1 | SOXDR1 | R/W | H'0228 | — | H'0000 0000 | S/H | √ |
| Scaling Ratio Y for Destination Register 2 | SRYDR2 | R/W | H'0234 | — | H'0000 0000 | S/H | √ |
| Scaling Ratio X for Destination Register 2 | SRXDR2 | R/W | H'0238 | — | H'0000 0000 | S/H | √ |
| Scaling Offset Y for Destination Register 2 | SOYDR2 | R/W | H'023C | — | H'0000 0000 | S/H | √ |
| Scaling Offset X for Destination Register 2 | SOXDR2 | R/W | H'0240 | — | H'0000 0000 | S/H | √ |
| Scaling Ratio Y for Destination Register 3 | SRYDR3 | R/W | H'024C | — | H'0000 0000 | S/H | √ |
| Scaling Ratio X for Destination Register 3 | SRXDR3 | R/W | H'0250 | — | H'0000 0000 | S/H | √ |
| Scaling Offset Y for Destination Register 3 | SOYDR3 | R/W | H'0254 | — | H'0000 0000 | S/H | √ |
| Scaling Offset X for Destination Register 3 | SOXDR3 | R/W | H'0258 | — | H'0000 0000 | S/H | √ |
| Blend Mode Register0 | BMDR0 | R/W | H'0264 | — | H'0000 0000 | S/H | √ |
| Blend Mode Register1 | BMDR1 | R/W | H'0268 | — | H'0000 0000 | S/H | √ |
| Blend Mode Register2 | BMDR2 | R/W | H'026C | — | H'0000 0000 | S/H | √ |
| Blend Mode Register3 | BMDR3 | R/W | H'0270 | — | H'0000 0000 | S/H | √ |

Notes: 1. IMP PSC execution is the state where the EX bit in the EXCTLR register = 1 or the CLE bit in the SCTLR1
register = 1.
2. H: Power-on reset
S: Software reset by the SWRST bit of the SCTLR register.

## U31.2 Register Description

【Register description symbol】
Initial value: Register value after reset
—: Undefined value
R/W: Read and write enabled. The write value can be read.
R: Read-only. The write value should be the same as the read value. Only modify the target bit (Read Modify Write) because the initial value should not be changed.
-/W: Only write is available. The read value is undefined.

### U31.2.1 Version Control Register (VCR)

This register indicates the version of this module.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VER[31:16] | | | | | | | | | | | | | | | |
| Initial value: | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VER[15:0] | | | | | | | | | | | | | | | |
| Initial value: | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | VER | H’21111616 | R | Indicates the version of this module. |

### U31.2.2 System Control Register 0 (SCTLR0)

This register performs (software) reset control.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SWRST | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | SWRST | 0 | R/W | Software reset<br>Writing 1 to this bit applies a software reset and writing 0 cancels a software reset.<br>0: Cancels a software reset for this module.<br>1: Applies a software reset to this module.<br>When set to 1, the value of the register in which the notation of 'S' is written in the "Reset" column of Tables in section U31.1.3 Register Configuration.<br>Setting the EX bit in the EXCTLR register to 1 or the CLE bit in the SCTLR1 register to 1 is prohibited. In that case, operation is not guaranteed.<br>To cancel the reset, write 0 to this bit. After, read this bit to check that the value is 0. |
| 30 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.3 Status Register (SR)

This register displays each status generated by PSC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | WUPCOVF |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLBRK | — | — | — | INT | IER | TRA | — | — | — | END |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 16 | WUPCOVF | 0 | R | WUP counter overflow<br>This bit indicates the overflow of WUP instruction.<br>0: WUP overflow has not been occured<br>1: WUP overflow has been occured<br>Note: When WUPCOVFE bit in SER is 0, this bit is not set to 1 even if the WUP overflow occured.<br>When WUPCOVFE bit in SER is 1, this bit is set to 1 if the WUP overflow occur. |
| 15 to 11 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 10 | CLBRK | 0 | R | CL break status.<br>0: CL does not break<br>1: CL break<br>CL break can not be usable if CLBRKE of SER register is 0. Set CLBRKE of SER register to 1 to use CL break function.<br>Command list continue execution after clearing this bit using CLBRKC bit of SCR register. |
| 9 to 7 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | INT | 0 | R | Decoding of INT instruction/<br>This bit indicates decoding of the INT instruction.<br>Once this bit becomes 1, subsequent instructions are not started until this bit becomes 0.<br>0 : The INT instruction is not decoded<br>1 : The INT instruction is decoded<br>Note: When INTE bit in SER is 0, this bit is not set to 1 even if the INT instruction is decoded.<br>When INTE bit in SER is 1, this bit is set to 1 if the INT instruction is decoded. |
| 5 | IER | 0 | R | Decoding of illegal instruction or GOSUB overflow/underflow.<br>This bit indicates the decoding of illegal instructions.<br>Once this bit becomes 1, execution of CL is terminated.<br>An illegal instruction is an instruction not listed in Table U31B.8.<br>0 : Illegal instruction is not decoded<br>1 : Illegal instruction is decoded<br>Note: When IERE bit in SER is 0, this bit is not set to 1 even if illegal instructions are decoded.<br>When IERE bit in SER is 1, this bit is set to 1 if illegal instructions are decoded.<br>Reset IMP PSC by using SWRST bit of SCTLR0 when this bit is 1. |
| 4 | TRA | 0 | R | Decode TRAP instruction<br>This bit indicates decoding of the TRAP instruction.<br>When decoding the TRAP instruction, CL execution is terminated.<br>0: TRAP instruction is not decoded<br>1: TRAP instruction is decoded<br>Note: When TRAE bit in SER is 0, this bit is not set to 1 even if the TRAP instruction is decoded.<br>When TRAE bit in SER is 1, this bit is set to 1 if the TRAP instruction is decoded. |
| 3 to1 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 0 | END | 0 | R | Image processing end of IMP PSC<br>When the EX bit of the EXCTLR register = 1, the image processing of the IMP PSC is started and when the image processing of the IMP PSC ends, the EX bit of the EXCTLR register becomes 0 and this bit becomes 1.<br>0: IMP PSC is not running or image processing is in progress<br>1: IMP PSC image processing has ended<br>Note: When ENDE bit in SER is 0, this bit is not set to 1 even if the image processing has ended.<br>When ENDE bit in SER is 1, this bit is set to 1 if image processing has ended. |

### U31.2.4 Status Clear Register (SCR)

This register clears each status displayed in the SR register. Clearing is done by writing a 1, writing 0 is illegal.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | WUPCOVFC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | -/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLBRKC | — | — | — | INTC | IERC | TRAC | — | — | — | ENDC |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | -/W | R | -/W | R | R | R | -/W | -/W | -/W | R | R | R | -/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 16 | WUPCOVFC | 0 | -/W | Clear the WUPOVF bit.<br>By writing 1, the WUPOVF bit of the SR register is cleared to 0.<br>When this bit is read, 0 is always read. |
| 15 to 11 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 10 | CLBRKC | 0 | -/W | CL break status clear.<br>By writing 1, the 0 in the CLBRK bit of SR register will be cleared.<br>This bit is always read as 0.<br>Command list continue execution after clearing CLBRK bit of SR register. |
| 9 to 7 | — | 0 | R | Reserved<br>The read value is undefined.<br>Operation cannot be guaranteed if a value other than the read value is written. |
| 6 | INTC | 0 | -/W | Clear the INT bit<br>By writing 1, the INT bit of the SR register is cleared to 0.<br>When this bit is read, 0 is always read. |
| 5 | IERC | 0 | -/W | Clear the IER bit<br>By writing 1, the IER bit of the SR register is cleared to 0.<br>When this bit is read, 0 is always read. |
| 4 | TRAC | 0 | -/W | Clear TRA bit<br>By writing 1, the TRA bit in the SR register is cleared to 0.<br>When this bit is read, 0 is always read. |
| 3 to1 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | ENDC | 0 | -/W | Clear the END bit<br>By writing 1, the END bit of the SR register is cleared to 0.<br>When this bit is read, 0 is always read. |

### U31.2.5 Status Enable Register (SER)

The bits of this register are used to set whether to reflect the status of the corresponding SR register bit.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | WUPCOVFE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLBRKE | — | — | — | INTE | IERE | TRAE | — | — | — | ENDE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R/W | R | R | R | R/W | R/W | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 16 | WUPCOVFE | 0 | R/W | WUPOVF bit set enable<br>0: Even if the WUP overflow is detected, WUPOVF bit of SR register is not set to 1.<br>1: When WUP overflow is detected, the WUPOVF bit of SR register is set to 1.<br>Refer to the **U34.IMP-X7_system, U34.10.3 Inter Module Synchronization** when setting this bit. |
| 15 to 11 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 10 | CLBRKE | 0 | R/W | CL break status/CL break enable.<br>0: CL break disable<br>When CLBRK is not detected, 1 is not set to CLBRK bit of SR register.<br>1: CL break enable<br>When CLBRK is detected, 1 is set to CLBRK bit of SR register.<br>When this bit is 0, CL break operation is not be possible. |
| 9 to 7 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 6 | INTE | 1 | R/W | INT bit set enable<br>0: Even if the INT instruction is decoded, the INT bit of the SR register is not set to 1.<br>1: When the INT instruction is decoded, the INT bit of the SR register is set to 1. |
| 5 | IERE | 1 | R/W | Set IER bit Enable<br>0: Even if an illegal instruction is decoded, the IER bit of the SR register is not set to 1.<br>1: When an illegal instruction is decoded, the IER bit of the SR register<br>is set to 1. |
| 4 | TRAE | 1 | R/W | TRA bit set enable<br>0: Even if the TRAP instruction is decoded, the TRA bit in the SR register is not set to 1.<br>1: When the TRAP instruction is decoded, the TRA bit in the SR register is set to 1. |
| 3 to1 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 0 | ENDE | 1 | R/W | END bit set enable<br>0: Even after IMP PSC image processing is completed, the END bit of the SR register is not set to 1.<br>1: When image processing of IMP PSC is completed, END bit of the SR register is set to 1. |

### U31.2.6 Interrupt Mask Register (IMR)

It is a register that sets whether to generate an interrupt to CPU based on the internal interrupt cause or whether to mask the internal interrupt cause.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | WUPCOVFM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLBRKM | — | — | — | INTM | IERM | TRAM | — | — | — | ENDM |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R/W | R | R | R | R/W | R/W | R/W | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 16 | WUPCOVFM | 0 | R/W | Interrupt mask setting by WUPOVF status<br>0: Interrupt generation by WUPOVF is not masked.<br>1: Interrupt generation by WUPOVF is masked. |
| 15 to 11 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 10 | CLBRKM | 1 | R/W | Mask setting of interrupt by CLBRK status<br>0: The generation of CLBRK interrupt is not masked.<br>1: The generation of CLBRK interrupt is masked. |
| 9 to 7 | — | 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 6 | INTM | 1 | R/W | Interrupt mask setting by INT status<br>0: Interrupt generation by INT is not masked.<br>1: Interrupt generation by INT is masked. |
| 5 | IERM | 1 | R/W | Interrupt mask setting by IER status<br>0: Interrupt generation by IER is not masked.<br>1: Interrupt generation by IER is masked. |
| 4 | TRAM | 1 | R/W | Interrupt mask setting by TRA status<br>0: Interrupt generation by TRA is not masked.<br>1: Interrupt generation by TRA is masked. |
| 3 to1 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 0 | ENDM | 1 | R/W | Interrupt mask setting by END status<br>0: Interrupt generation by END is not masked.<br>1: Interrupt generation by END is masked. |

### U31.2.7 System Control Register 1 (SCTLR1)

It is a register that controls activation of CL.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 0 | CLE | 0 | R/W | IMP PSC CL (Command List) activation bit<br>[writing]<br>0:—<br>1: Launch CL<br>[reading]<br>0: CL is not running<br>1: CL is running<br>If 1 is written to this bit, IMP PSC fetches CL from the address set in CLSAR and executes the instruction.<br>This bit is read as 1 during CL execution and 0 is read after CL execution is completed. Writing 1 to this bit during CL execution is prohibited and operation is not guaranteed. |

### U31.2.8 Command List Start Address Register (CLSAR)

This register sets the start address for fetching CL.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLSAR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CLSAR | | | | | | | | | | | | | | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | CLSAR | H'0000 0000 | R/W | Start address of CL<br>Set the start address of CL execution in this register.<br>Please set it in units of 4 bytes. |
| 1 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.9 Status Register1 (SR1)

This register displays the code value when IMP PSC executes INT and TRAP instructions.

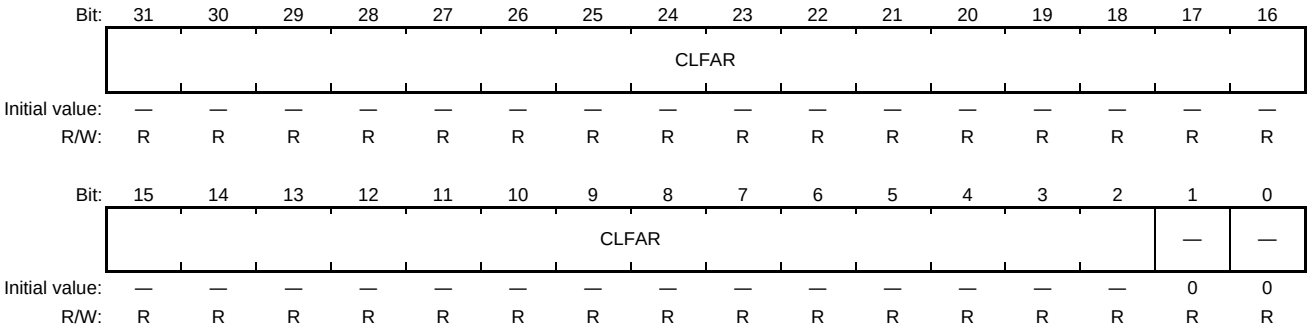
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCODE | | | | | | | | TRAPCODE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | INTCODE | H'00 | R | INT interrupt code<br>When the INT instruction is executed, the value specified in the CODE field (bit [7: 0]) of this instruction is reflected in these bits.<br>These bits are updated independently of the setting of the INTE bit in the SER register. |
| 23 to 16 | TRAPCODE | H'00 | R | TRAP interrupt code<br>When the TRAP instruction is executed, the value specified in the CODE field (bit [7: 0]) of this instruction is reflected in these bits.<br>These bits are updated independently of the setting of the TRAE bit in the SER register. |
| 15 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.10 Command List Fetch Address Register (CLFAR)

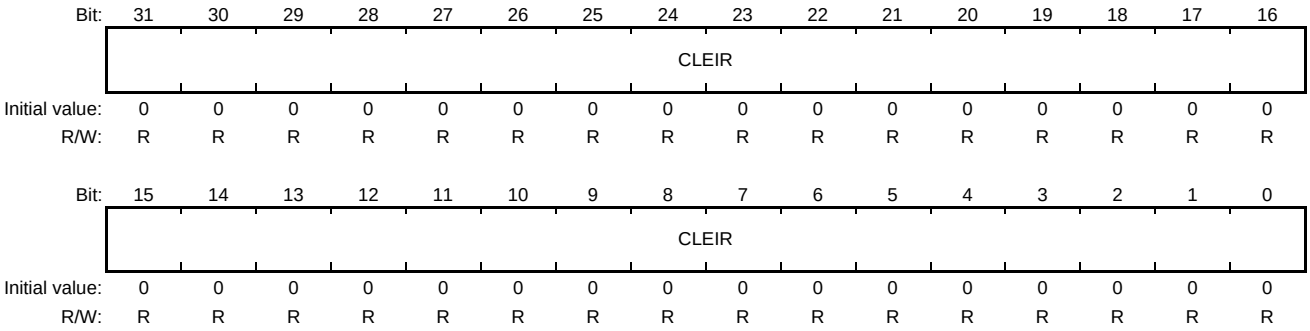
This register displays the address of the CL fetch destination (running).




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | CLFAR | Undefined | R | Displays the CL fetch destination address (running).<br>The initial value is undefined. |
| 1 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.11 Command List Execute Instruction Register (CLEIR)

This register displays the command being executed by CL.



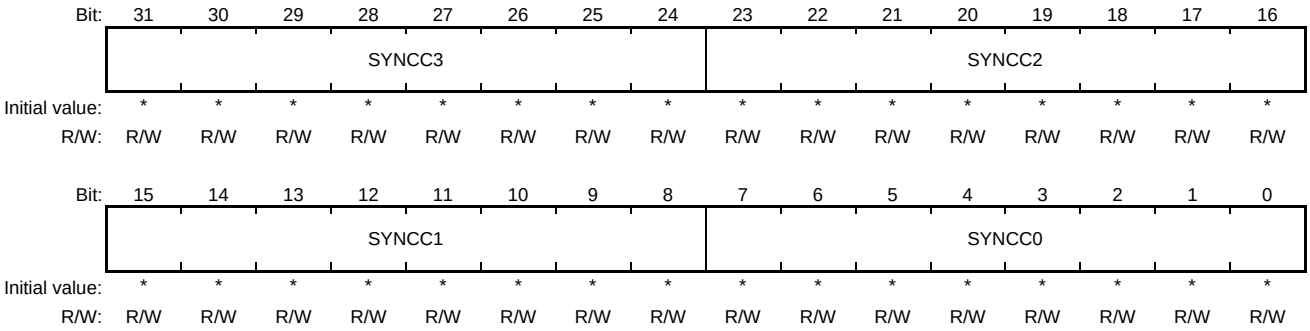

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CLEIR | H'0000 0000 | R | Display the command being executed by the CL. |

### U31.2.12 Synchronization Control Register 0 (SYNCCR0)

This register specifies the module corresponding to bits 0 to 3 of the SYNCC enable field of WUP or SLP instruction.

Set the module number of the module you want to specify in this register. For the module numbers, refer to U34.IMP-X7_system, U34.10.5 Inter Module Synchronization.

Set this register by the CPU before setting the CLE bit in the SCTLR1 register to 1. It is prohibited to rewrite this register while the CLE bit of the SCTLR1 register is 1.



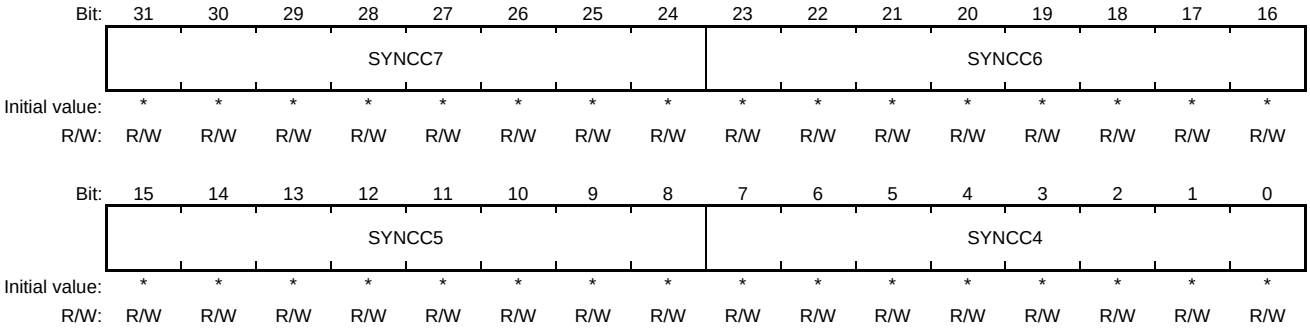

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC3 | H'FF | R/W | This register is used to specify the correlated module for Bit 3 of the SYNCC enable field of the WUP, SLP instruction. |
| 23 to 16 | SYNCC2 | H'FF | R/W | This register is used to specify the correlated module for Bit 2 of the SYNCC enable field of the WUP, SLP instruction. |
| 15 to 8 | SYNCC1 | H'FF | R/W | This register is used to specify the correlated module for Bit 1 of the SYNCC enable field of the WUP, SLP instruction. |
| 7 to 0 | SYNCC0 | H'FF | R/W | This register is used to specify the correlated module for Bit 0 of the SYNCC enable field of the WUP, SLP instruction. |

### U31.2.13 Synchronization Control Register 1 (SYNCCR1)

This register specifies the module corresponding to bits 4 to 7 of the SYNCC enable field of WUP or SLP instruction.

Set the module number of the module you want to specify in this register. For the module numbers, refer to U34.IMP-X7_system, U34.10.5 Inter Module Synchronization.

Set this register by the CPU before setting the CLE bit in the SCTLR1 register to 1. It is prohibited to rewrite this register while the CLE bit of the SCTLR1 register is 1.



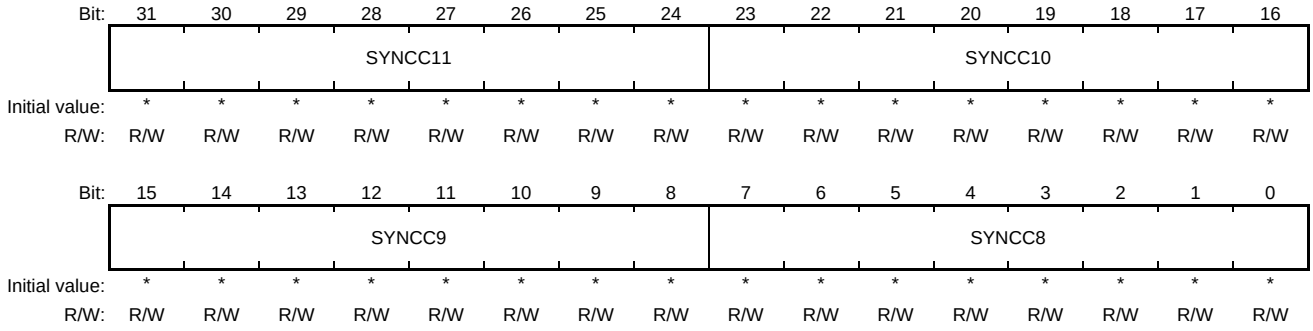

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC7 | H'FF | R/W | This register is used to specify the correlated module for Bit 7 of the SYNCC enable field of the WUP, SLP instruction. |
| 23 to 16 | SYNCC6 | H'FF | R/W | This register is used to specify the correlated module for Bit 6 of the SYNCC enable field of the WUP, SLP instruction. |
| 15 to 8 | SYNCC5 | H'FF | R/W | This register is used to specify the correlated module for Bit 5 of the SYNCC enable field of the WUP, SLP instruction. |
| 7 to 0 | SYNCC4 | H'FF | R/W | This register is used to specify the correlated module for Bit 4 of the SYNCC enable field of the WUP, SLP instruction. |

### U31.2.14 Synchronization Control Register 2 (SYNCCR2)

This register specifies the module corresponding to bits 8 to 11 of the SYNCC enable field of WUP or SLP instruction.

Set the module number of the module you want to specify in this register. For the module numbers, refer to U34.IMP-X7_system, U34.10.5 Inter Module Synchronization.

Set this register by the CPU before setting the CLE bit in the SCTLR1 register to 1. It is prohibited to rewrite this register while the CLE bit of the SCTLR1 register is 1.



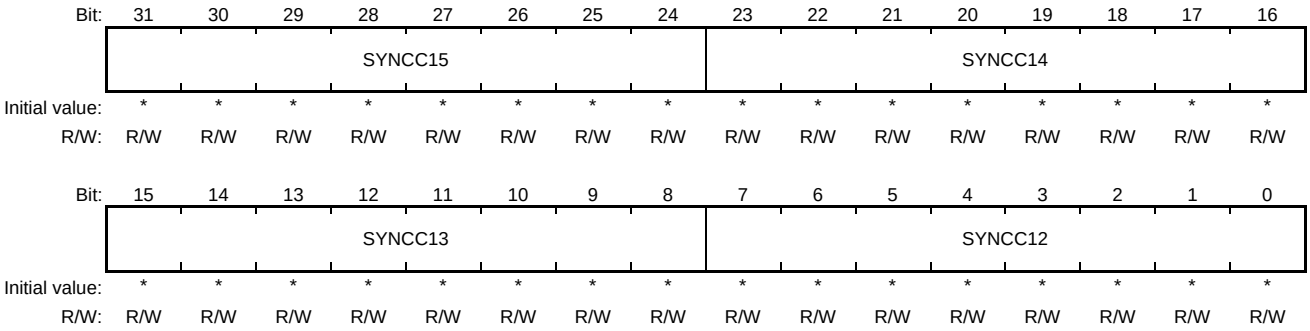

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC11 | H'FF | R/W | This register is used to specify the correlated module for Bit 11 of the SYNCC enable field of the WUP, SLP instruction. |
| 23 to 16 | SYNCC10 | H'FF | R/W | This register is used to specify the correlated module for Bit 10 of the SYNCC enable field of the WUP, SLP instruction. |
| 15 to 8 | SYNCC9 | H'FF | R/W | This register is used to specify the correlated module for Bit 9 of the SYNCC enable field of the WUP, SLP instruction. |
| 7 to 0 | SYNCC8 | H'FF | R/W | This register is used to specify the correlated module for Bit 8 of the SYNCC enable field of the WUP, SLP instruction. |

### U31.2.15 Synchronization Control Register 3 (SYNCCR3)

This register specifies the module corresponding to bits 12 to 15 of the SYNCC enable field of WUP or SLP instruction.

Set the module number of the module you want to specify in this register. For the module numbers, refer to U34.IMP-X7_system, U34.10.5 Inter Module Synchronization.

Set this register by the CPU before setting the CLE bit in the SCTLR1 register to 1. It is prohibited to rewrite this register while the CLE bit of the SCTLR1 register is 1.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC15 | H'FF | R/W | This register is used to specify the correlated module for Bit 15 of the SYNCC enable field of the WUP, SLP instruction. |
| 23 to 16 | SYNCC14 | H'FF | R/W | This register is used to specify the correlated module for Bit 14 of the SYNCC enable field of the WUP, SLP instruction. |
| 15 to 8 | SYNCC13 | H'FF | R/W | This register is used to specify the correlated module for Bit 13 of the SYNCC enable field of the WUP, SLP instruction. |
| 7 to 0 | SYNCC12 | H'FF | R/W | This register is used to specify the correlated module for Bit 12 of the SYNCC enable field of the WUP, SLP instruction. |

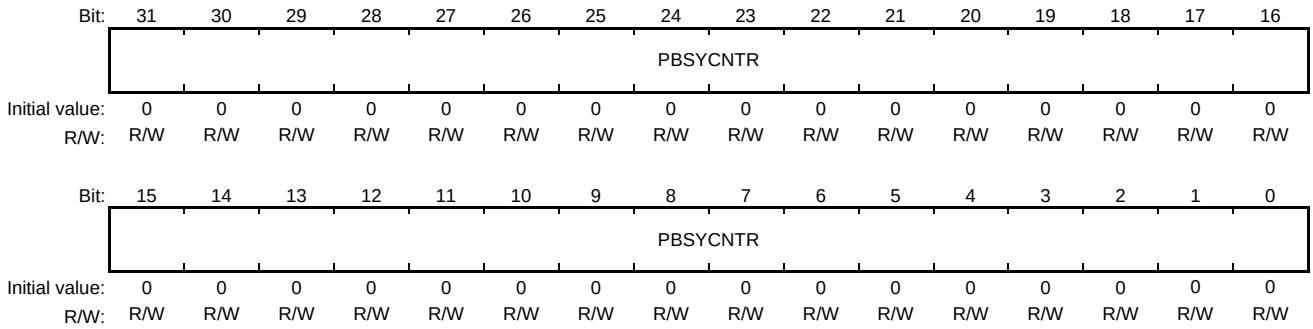
### U31.2.16 Performance Control Register (PCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | IFBSYCBTENB | IBSYCNTENB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

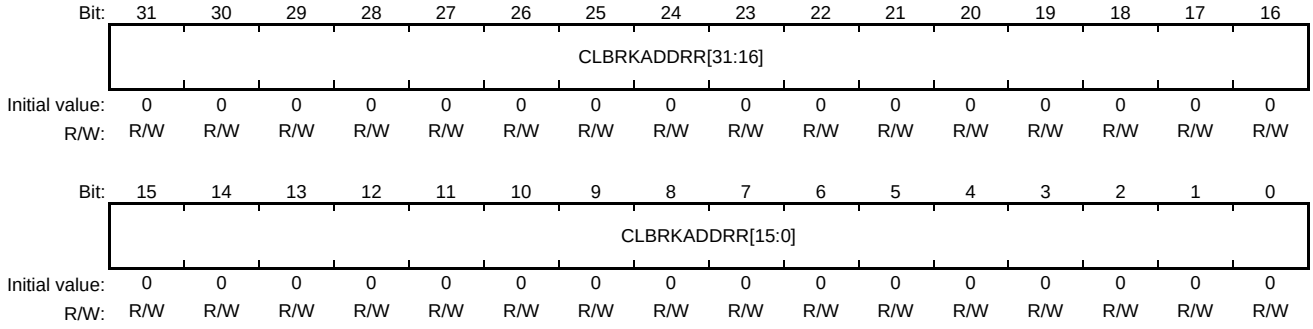
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 1 | IFBSYCNTENB | 0 | R/W | IFID operation count enable<br>When this bit is 1, CYCLE is counted while CL is running.<br>Write 1 in this bit during IBSYCNTENB= 1 is prohibited. |
| 0 | IBSYCNTENB | 0 | R/W | Image operation count enable<br>When this bit is 1, CYCLE is counted while EX bit in EXCNTR register = 1.<br>Write 1 in this bit during IFBSYCNTENB = 1 is prohibited. |

### U31.2.17 Performance Busy Counter Register (PBSYCNTR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PBSYCNTR | H'0000 0000 | R/W | This register count cycles during operation<br>This reads the value counted in the condition set in Performance Control Register (PCR). Count value of this register cannot be guarenteed.<br>When this register is written, the counter is set to the written value, and starts counts from set value onwards.<br>Writing this register during the counting is prohibited. |

### U31.2.18 CL Break Address Register(CLBRKADDRR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CLBRKADDRR | All 0 | R/W | CL break address. Specify the address(break point) of command list.<br>Specify a 32bit command list address.<br>Data address is prohibited. |

### U31.2.19 CL Conditinal GOSUB Register(CLCNDGSBR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | | — | — | — | — | — | — | — | CLCND GSBR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>The read value should always be 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 0 | CLCNDGSBR | All 0 | R/W | Condition for conditional GOSUB instruction:<br>.1: GOSUB is executed<br>0: GOSUB is not executed, NOP is instead executed<br>(Bit 22 of the GOSUB command should be 1 for NOP operation) |

### U31.2.20 Source Start Address Register (SSAR)

This register sets the start address of the source image to be read from DDR-SDRAM or Internal memory.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSAR | | | | | | | | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSAR | | | | | | | | | | | | | | | |
| Initial value: | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SSAR | Undefined | R/W | Set the start address of the source image in byte units.<br>When it is not 8 bpp, please set it by 2-byte unit.<br>(The lowest 1 bit is fixed to 0)<br>The initial value is undefined. |

### U31.2.21 Source Length Register (SLR)

This register sets the reading size of the source image from the DDR-SDRAM or Internal memory.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | YLNG | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | XLNG | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 28 to 16 | YLNG | H'0000 | R/W | Please specify the number of lines in the Y direction of the source image.<br>It can be specified in the range from 1 to 4096.<br>Operation is not guaranteed if a value outside the range is set. |
| 15 to 13 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 12 to 0 | XLNG | H'0000 | R/W | Please specify the number of pixels in the X direction of the source image.<br>8 bpp: It can be specified in the range of 1 to 4096.<br>10, 12, 16 bpp: It can be specified in the range of 1 to 2048.<br>Operation is not guaranteed if a value outside the range is set. |

### U31.2.22 Source Stride (SSTR)

This register sets the stride of the source image to be read from DDR-SDRAM or Internal memory.

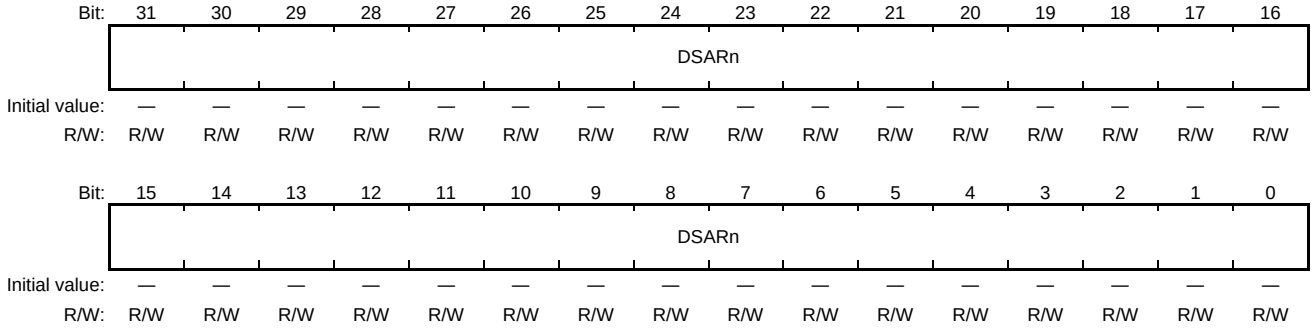
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SSTR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 15 to 0 | SSTR | H'0000 | R/W | Sets the stride of the source image in unsigned byte units.<br>If it is not 8 bpp, set it in 2-byte units (the lowest 1 bit is fixed to 0).<br>Set these bits to a value larger than or equal to the value obtained by converting the number of pixels in the X direction of the source image (set in the XLNG bit of the SLR register) to the number of bytes.<br>In case of 8bpp, convert one pixel as one byte, otherwise, convert one pixel as 2 bytes. |

### U31.2.23 Destination Start Address register n (DSARn) (n=0-3)

This register sets the start address of the destination image to be written to in DDR-SDRAM or Internal memory.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DSARn | Undefined | R/W | Please set the start address of the Destination image in byte units.<br>When it is not 8 bpp, please set it in 2-byte units.<br>(The lowest 1 bit is fixed to 0)<br><br>DSAR 0: Start address of destination image for output image 0<br>DSAR 1: Start address of destination image for output image 1<br>DSAR 2: Start address of destination image for output image 2<br>DSAR 3: Start address of destination image for output image 3<br><br>The initial value is undefined. |

### U31.2.24 Destination Stride Register n (DSTRn) (n=0-3)

This register sets the stride of the destination image to be written to in DDR-SDRAM or Internal memory.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DSTRn | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/w | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 15 to 0 | DSTRn | H'0000 | R/W | Sets the stride of the Destination image in unsigned byte units.<br>When it is not 8 bpp, please set it in 2-byte units.<br>(The lowest 1 bit is fixed to 0)<br>DSTR 0: stride of destination image of output image 0<br>DSTR 1: stride of destination image of output image 1<br>DSTR 2: stride of destination image of output image 2<br>DSTR 3: stride of destination image of output image 3<br>Set this bit to a value greater than or equal to the number of pixels in the X direction of the Destinatoion image (set in the DXLNGn bit of the DLNGRn register) converted to the number of bytes.<br>In case of 8 bpp, convert one pixel as one byte, otherwise, convert one pixel as 2 bytes. |

### U31.2.25 Destination Length Register n (DLNGRn) (n=0-3)

This register sets the size of the destination image to be written to in DDR-SDRAM or Internal memory.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DYLNGn | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | DXLNGn | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 28 to 16 | DYLNGn | H'0000 | R/W | Please specify the number of lines in the Y direction of the destination image.<br>It can be specified in the range from 1 to 4096.<br>DYLNG 0: Number of lines of destination image for output image 0<br>DYLNG 1: Number of lines of destination image for output image 1<br>DYLNG 2: Number of lines of destination image for output image 2<br>DYLNG 3: Number of lines of destination image for output image 3<br>Set these bits to a value equal or below the number of lines in the Y direction of the source image (set in the YLNG bit of the SLR register).<br>[important point]<br>When the value specified in these bits is less than or equal to the number of lines after reduction of the source image, the number of lines set by these bits are output.<br>If the value specified by these bits is larger than the number of lines after reduction of the source image, the number of lines set by these bits is output by repeating the last line of the source image. |
| 15 to 13 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 12 to 0 | DXLNGn | H'0000 | R/W | Please specify the number of pixels in the X direction of the Destination image.<br>8 bpp: It can be specified in the range of 1 to 4096.<br>10, 12, 16 bpp: It can be specified in the range of 1 to 2048.<br>DXLNG 0: Number of pixels of destination image for output image 0<br>DXLNG 1: Number of pixels of destination image for output image 1<br>DXLNG 2: Number of pixels of destination image for output image 2<br>DXLNG 3: Number of pixels of destination image for output image 3<br>Set these bits to a value equal or below the number of pixels in the X direction of the source image (set in the XLNG bit of the SLR register).<br>[important point]<br>When the value specified in these bits is less than or equal to the number of pixels after reduction of the source image, the number of pixels set by these bits is output.<br>If the value specified by these bits is larger than the number of pixels after reduction of the source image, the number of pixels set by these bits is output by repeating the last pixel of each row of the source image. |

### U31.2.26 Data Control Register (DCTLR)

This register specifies the pixel format of the source image and destination images.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFSRC | | | | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFDST | | | | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R/ | R/ | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 28 | PFSRC | H'0 | R/W | Specify the pixel format of the source image.<br>0: unsigned 8 bits<br>1: unsigned 10 bits<br>2: unsigned 12 bits<br>8: Signed 8bit<br>11: Signed 16bit<br>Other: Not defined (please do not set)<br>Set the same value as the PFDST bit. |
| 27 to 16 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 15 to 12 | PFDST | H'0 | R/W | Specifies the pixel format of the destination image.<br>0: unsigned 8 bits<br>1: unsigned 10 bits<br>2: unsigned 12 bits<br>8: Signed 8bit<br>11: Signed 16bit<br>Other: Not defined (please do not set)<br>Set the same value as the PFSRC bit. |
| 11 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.27 Execute Control Register (EXCTLR)

It is a register that controls startup of IMP PSC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EX | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | EX | 0 | R/W | IMP PSC activation bit<br>[writing]<br>0: NOP<br>1: Start IMP PSC<br>[reading]<br>0: IMP PSC is not activated<br>1: IMP PSC is running (during image processing)<br>When IMP PSC is activated, and image processing is completed, this bit automatically becomes 0. At this time, if the ENDE bit of the SER register is 1, the END bit of the SR register is set to 1.<br>Write 1 to this bit (activate IMP PSC) with CL's WPR instruction.<br>Setting this bit to 0 or 1 while this bit is 1 is prohibited. In that case, operation is not guaranteed. |
| 30 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.28 Execute Line Count Register (EXCNTR)

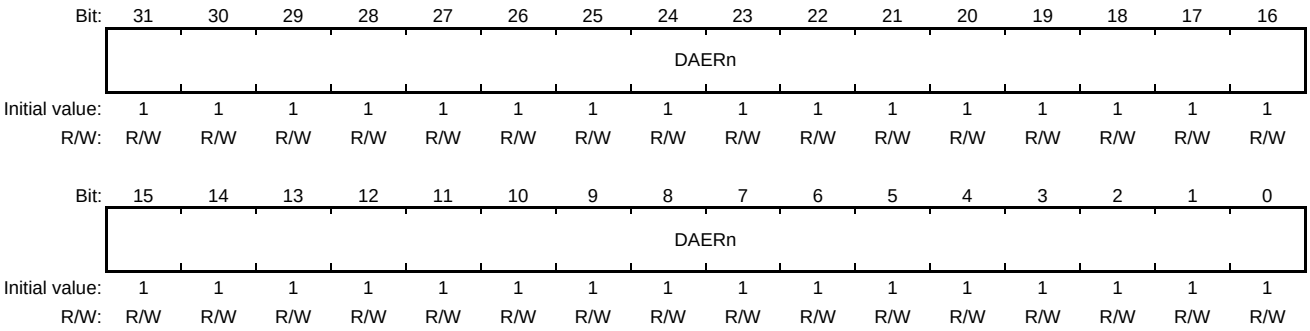
This register displays the number of output lines of the destination image。

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | EXCNT | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 13 to 0 | EXCNT | H'0000 | R | Displays the number of output lines of the destination image.<br>When the destination image is output in multiple directions by setting the SURFACE bit of SPR register, the minimum number of outputted lines in output image 0-3 is displayed. |

### U31.2.29 Destination Address Enable-Register n (DAERn) (n=0-3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DAERn | H'FFFF FFFF | R/W | DAER 0: Destination address of output image 0 bit Enable<br>DAER 1: Destination address of output image 1 bit Enable<br>DAER 2: Destination address of output image 2 bit Enable<br>DAER 3: Destination address of output image 3 bit Enable<br>When outputting destination image to DDR-SDRAM or Internal memory, set H'FFFF FFFF (initial value) to this register. |

### U31.2.30 Scaling Parameter Register (SPR)

Scaling operation setting register of IMP PSC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

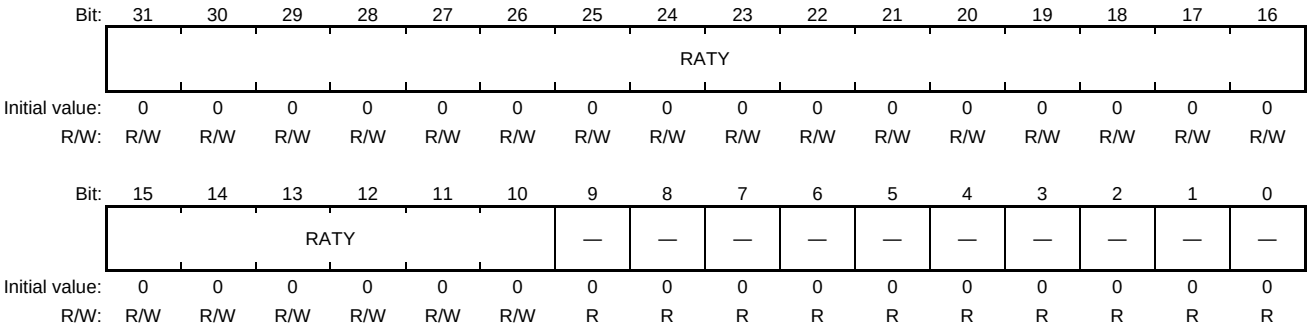
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | SURFACE[3:0] | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 3 to 0 | SURFACE[3:0] | H'0 | R/W | Depending on the number of destination images to be outputted, set the output image 0 to 3 to be used as follows.<br>0001: 1 sheet output (using only output image 0)<br>0011: 2 sheets output (using output image 0 to 1)<br>0111: 3 sheets output (using output image 0 to 2)<br>1111: 4 sheets output (using output image 0 to 3)<br>It is prohibited to start IMP PSC with settings other than the above. When setting other than the above, operation is not guaranteed.<br>Corresponding to the settings in of these bits, set the following registers accordingly.<br>DSARn, DSTRn, DLNGRn, DAERn, SRYDRn, SRXDRn, SOYDRn, SOXDRn, BMDRn<br>(N = 0-3)<br>For example, when SURFACE [3: 0] = 0111, please set each register n = 0 - 2 above. |

### U31.2.31 Scaling Ratio Y for Destination Register n (SRYDRn) (n=0-3)

This register sets the magnification in the Y direction.

Calculate the result of source Image size / destination Image size in Y direction and multiply it by $2^{20}$ (binary number with 20bits), then truncate the decimal part.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | RATY | H'00 0000 | R/W | Set the magnification in Y direction according to source image size / destination image size.<br>The register area is a fixed point value with 2 bits integer and 20 bits fraction part.<br>Since IMP PSC supports reduction ratio = 0.5 to 1.0, set the value between H '10 0000 ~ H'20 0000.<br>For example, when reducing 480 lines to 479 lines:<br>(480/479) x $2^{20}$ = H'10088D. |
| 9 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.32 Scaling Ratio X for Destination Register n (SRXDRn) (n=0-3)

This register sets the magnification in the X direction.

Calculate the result of source Image size / destination Image size in X direction and multiply it by $2^{20}$ (binary number with 20bits), then truncate the decimal part.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RATX | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RATX | | | | | | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 10 | RATX | H'00 0000 | R/W | Set the magnification in the X direction according to the source image size / destination image size.<br>The register area is a fixed point value with 2 bits integer and 20 bits fraction part.<br>Since IMP PSC supports reduction ratio = 0.5 to 1.0, set the value between H '10 0000 ~ H'20 0000.<br>For example, when reducing 640 pixels to 639 pixels:<br>(640/639) x$2^{20}$ = H'100668. |
| 9 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.33 Scaling Offset Y for Destination Register n (SOYDRn) (n=0-3)

This register specifies the start offset for reading the source image in the Y direction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | YOFS | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | YOFS | | | | | | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 30 to 10 | YOFS | H'00 0000 | R/W | Specify the offset value in the Y direction at the start of processing.<br>The register area is a fixed-point number of 1 integer digit and 20 decimal places, and it can set a value less than 2.<br>For example, when adding an offset of 0.5 at the start, specify H'080000. |
| 9 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.34 Scaling Offset X for Destination Register n (SOXDRn) (n=0-3)

This register specifies the start offset for reading the source image in the X direction.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | XOFS | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | XOFS | | | | | | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 30 to 10 | XOFS | H'00 0000 | R/W | Specify the offset value in the X direction at the start of processing.<br>The register area is a fixed-point number of 1 integer digit and 20 decimal places, and it can set a value less than 2.<br>For example, when adding an offset of 1.5 at the start, specify H'180000. |
| 9 to 0 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |

### U31.2.35 Blend Mode Register n (BMDRn) (n=0-3)

It is a register that controls blend of pixels (bilinear interpolation operation)。

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | BMP | BMA |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | — | All 0 | R | Reserve<br>The read value is always 0.<br>When values other than 0 are written, operation is not guaranteed. |
| 1 | RNDM | 0 | R/W | When shrinking in X direction, this bit is used for masking the rounding with bilinear interpolation calculation.<br>0: Perform rounding processing (do not mask).<br>1: Mask rounding processing.<br>In the case of 16 bpp, the value setting value of this bit is invalid, and rounding processing is not performed. |
| 0 | BLDM | 0 | R/W | When shrinking in X direction, this bit is used for masking the bilinear interpolation calculation.<br>0: Bilinear interpolation operation is executed (not masked).<br>1: Mask bilinear interpolation operation.<br>When bilinear interpolation operation is masked, the source image is reduced by thinning pixels instead of blending adjacent pixels. |

## U31.3 Operation

### U31.3.1 Image reduction

Reduction calculation of the input image is performed by bilinear interpolation. Figure U31.3 shows an outline of the operation of bilinear interpolation calculation.

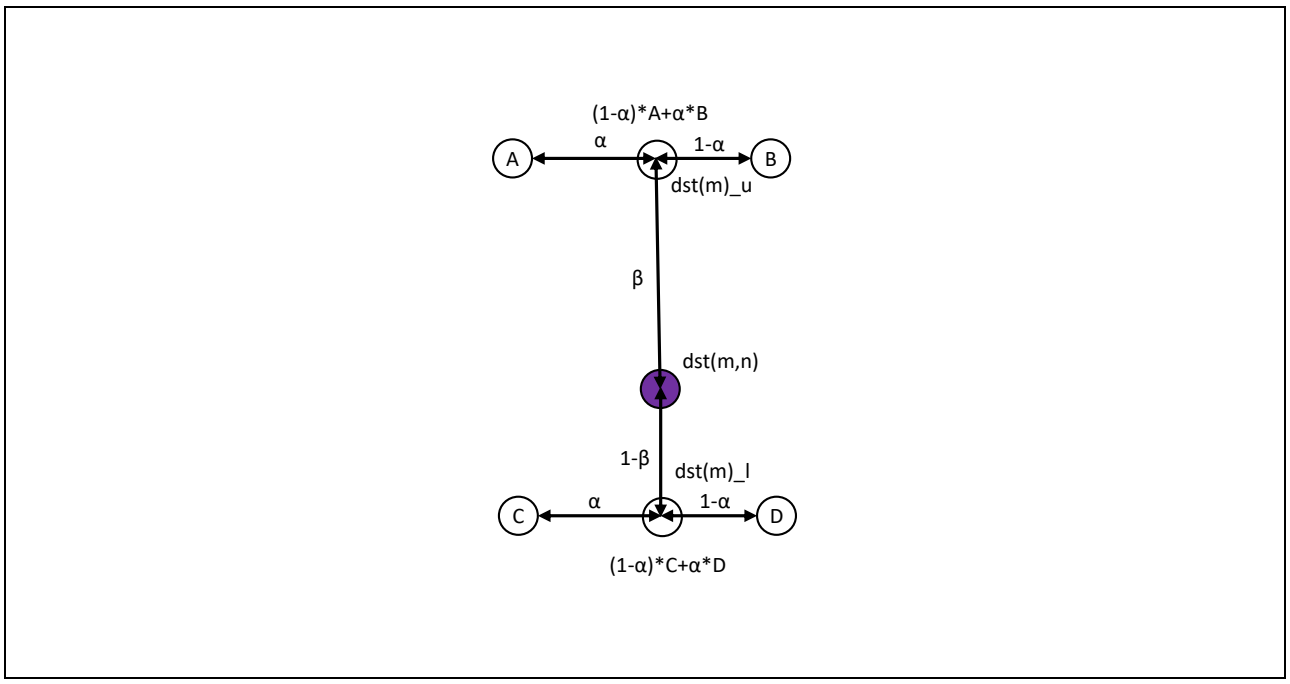



**Figure U31.3 Example of bilinear interpolation calculation operation**

A pixel dst (m, n) in a destination image can be found by bilinear interpolation calculation using 2 * 2 pixels (A, B, C, D) of the source image located around it. The calculation procedure at this time is as follows:

Step1：Bilinearly interpolate the pixels A and B of the source image to calculate the pixel dst(m)_u:

dst(m)_u = ( 1 - α ) * A + α * B

Step2：Bilinearly interpolate the pixels C and D of the source image to calculate the pixel dst(m)_l:

dst(m)_l = ( 1 - α ) * C + α * D

Step3：Bilinearly interpolate dst(m)_u and dst(m)_l obtained in Step1 and Step2 above to calculate the pixel dst (m, n):

dst(m,n) = ( 1 - β ) * dst(m)_u + β * dst(m)_l

### U31.3.2 Inter-Module Synchronization System

This functionality allows the synchronization of operations between the IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, IMP CNN, VDSP and IMP DTA. Executing the WUP and SLP instructions with target modules specified enables synchronization of the operations of the command list (CL) controllers in each of these modules.

For details of the WUP and SLP instructions in the IMP PSC, refer to Section U31.4.9 WUP, and Section U31.4.10 SLP. For selection of target modules, refer to the following section.

#### U31.3.2.1 Selection of Target Modules

Refer to the **U34.IMP-X7_system, U34.10.5 Inter Module Synchronization** chapter for the details.

### U31.3.3 CL break

To use the CL break function, following setting is required.

1. Specify the break point address of command list. Be sure that break point is command address of command list.

2. Set CLBRKENB of SER register to 1 and if interrupt is required, set CLBRKM bit in IMR bit to use CL break function.
3. Check the CLBRK status in SR register.

4. To continue execution after CL break, Set the CLBRKC bit of SCR register to 1.

## U31.4 Command List

It is prohibited to use instructions not listed on this section.

**Table U31B.8 Instruction list**

| Instruction | OP CODE | Description |
|---|---|---|
| NOP | H'80 | No Operation |
| TRAP | H'88 | Quit CL |
| WPR | H'55 | Write register |
| INT | H'81 | Generate an interrupt cause and wait until the interrupt is cleared. |
| SYNCS | H'8B | Wait for completion of DMA transfer execution |
| Reserved | H'8C | This OP CODE is prohibited. When this OP CODE is used, no illegal instruction interrupt occurs. |
| JUMP | H'90 | Jump to the specified address |
| GOSUB | H'91 | Jump to the specified address |
| RET | H'99 | Jump to the next instruction of GOSUB |
| WUP | H'A0 | Release SLP of other modules |
| SLP | H'A1 | Wait for other modules to execute WUP |

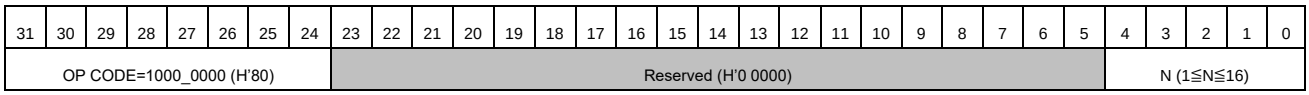
### U31.4.1 NOP (No OPeration)

#### U31.4.1.1 Functionality

This is an instruction to do No Operation.

The NOP instruction causes the IMP PSC to do nothing for the number of clock cycles specified in the N field before execution of the next instruction.

#### U31.4.1.2 Instruction Format



### U31.4.2 TRAP (TRAP)

#### U31.4.2.1 Functionality

This is an instruction to end CL execution.
TRAP terminates execution of the command list. Only when the TRAE bit of the SER register is 1, the TRA bit of the SR register is set to 1.
However, regardless of the value of the TRAE bit in the SER register, the value specified in the CODE field is set to the TRAPCODE bit of the SR1 register.

#### U31.4.2.2 Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_1000 (H'88) | | | | | | | | Reserved (H'0000) | | | | | | | | | | | | | | | | CODE | | | | | | | |

### U31.4.3 WPR (Write PaRameter)

#### U31.4.3.1 Functionality

This is an instruction to write data to registers.
Register write is performed for the number of fields N from the offset address specified by the ADD field. The value to be written to the register is the value of the DATA N - 1 field in order from the DATA 0 field.
Set the ADD field in 1-byte units. Setting 0 in the N field is prohibited.

#### U31.4.3.2 Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0101_0101 (H'55) | | | | | | | | ADD | | | | | | | | | | | | | | | | N(>=1) | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

…

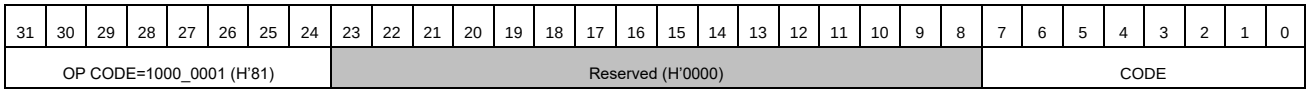
| DATA N-1 |
|---|

### U31.4.4 INT (INTerrupt)

#### U31.4.4.1 Functionality

This instruction generates an interrupt.
When the INTE bit of the SER register is 1, the INT bit of the SR register is set to 1. Regardless of the value of the INTE bit in the SER register, the value specified in the CODE field is set to the INTCODE bit of the SR1 register.
The next instruction is not started until the INT bit of the SR0 register is cleared to 0.

#### U31.4.4.2 Instruction Format




### U31.4.5 SYNCS (SYNChronize ProcesS)

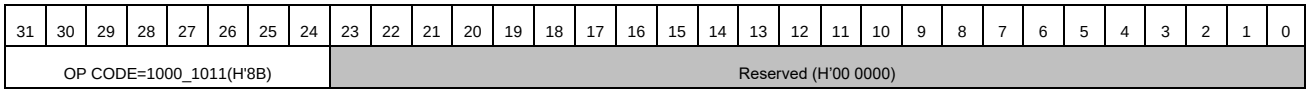
#### U31.4.5.1 Functionality

Wait for completion of DMA transfer execution.
If there is image processing currently being executed, wait for completion of the image processing. After waiting for the completion of the image processing, the next instruction is started.

If image processing is not being executed, it is the same as NOP (next instructions are started).

Always wait for image processing to be completed with the SYNCS instruction immediately after setting the EX bit in the EXCTLR register to 1 with the WPR instruction. If you do not wait for image processing to complete, the operation is not guaranteed.

#### U31.4.5.2 Instruction Format



### U31.4.6 JUMP

#### U30B.4.5.1 Functionality

This is the command to jump to the specified address.
It allows jumping to the address specified in JMPADD field. Setting 1 to RLT field uses relative address to JUMP command address and setting 0 in RLT field uses absolute address of the command list. Specify 4-byte unit command address in the JMPADD field.

#### U30B.4.5.2 Instruction format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1001_0000(H'90) | | | | | | | | RLT | Reserved（all 0） | | | | | | | | | | | | | | | | | | | | | | |
| JMPADD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U31.4.7 GOSUB (GOSUB)

#### U31.4.7.1 Functionality

This is the command to save current CL address to the hardware buffer and jump to the specified address.

It allows jumping to the address specified in GSADD field and returning to the address next to the GOSUB command by RET command. Specify 4-byte unit command address in the GSADD field. Nesting from the GOSUB up to 4 level can be possible. Setting 1 to RLT field uses GSADD as relative address to GOSUB command address and setting 0 in RLT field uses absolute address of the command list. Setting 1 in CON field and 0 in CLCNDGSBR bit of CLCNDGSBR register change GOSUB command into NOP command. After the CLCNDGSBR register is set, a SYNCS command should be added before using the GOSUB command.

#### U31.4.7.2 Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE = 1001_0001 (H'91) | | | | | | | | RLT | CON | Reserved (All 0) | | | | | | | | | | | | | | | | | | | | | |
| GSADD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U31.4.8 RET (RETurn)

#### U31.4.8.1 Functionality

This instruction jumps to the next instruction of GOSUB.
RET allows jumping to the CL address next to the address at which the GOSUB instruction was executed.

#### U31.4.8.2 Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1001_1001(H'99) | | | | | | | | Reserved (All 0) | | | | | | | | | | | | | | | | | | | | | | | |

### U31.4.9 WUP (WakeUP)

#### U31.4.9.1 Functionality

This instruction releases the modules specified in the SYNCC enable field from the sleep state in inter-module synchronization.

For details, see the explanation obout the operation on the intermodule synchronization system. Be sure to insert 8 or more instructions other than the WUP instruction between two WUP instructions in the CL.

#### U31.4.9.2 Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1010_0000 (H'A0) | | | | | | | | Reserved (H'00) | | | | | | | | SYNCC enable | | | | | | | | | | | | | | | |

### U31.4.10 SLP (SLeeP)

#### U31.4.10.1 Functionality

This instruction places the current module in the sleep state in inter-module synchronization.
When the modules specified in the SYNCC enable field executes the WUP instruction, the current module is released from the sleep state.
For details, see the explanation obout the operation on the intermodule synchronization system.

#### U31.4.10.2 Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1010_0001 (H'A1) | | | | | | | | Reserved (H'00) | | | | | | | | SYNCC enable | | | | | | | | | | | | | | | |

## U31.5 Usage Notes

### U31.5.1 Power shutdown

Refer to chapter 58.7 Usage Notes in IMP-X7 HW Manual for the power off.

### U31.5.2 Module Standby Mode

Refer to chapter chapter 58.7 Usage Notes in IMP-X7 HW Manual for the Module Standby Mode.

### U31.5.3 Software reset of CPG

Refer to chapter chapter 58.7 Usage Notes in IMP-X7 HWM Manual for the software reset.

## U31.6 Safety mechanism

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### U31.6.1 Overview

EDC protection is implemented to SRAM and Bus interface in IMPPSC. EDC check is always enabled.

### U31.6.2 Block diagram EDC protection

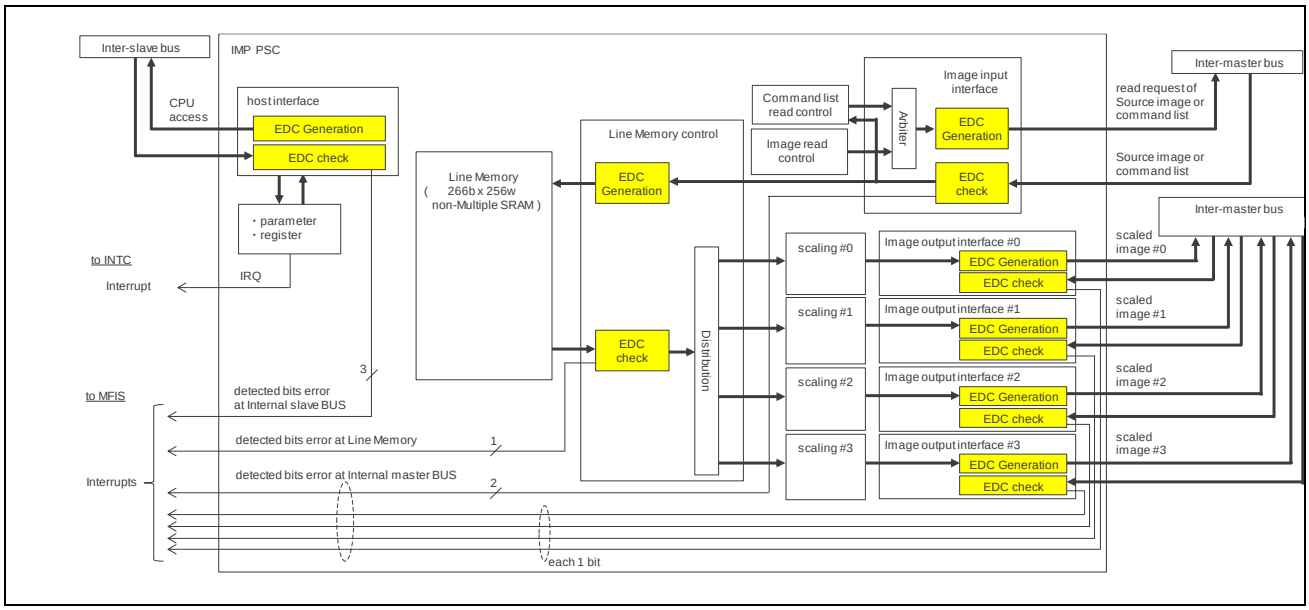



**Figure U31.9 Functional safety of IMP PSC.**

# U32. IMP-X7 CVengine

## U32.1 Overview

The IMP-X7 CVengine is a general purpose processor; its main target are algorithms of the computer vision area.

The IMP-X7 CVengine is configured of a hierarchy of three types of units referred to as the cluster, cores, and threads. Each of the units is described below.

A thread retains the thread state and is the element that executes a program based on the vertices it is assigned. As many programs as there are threads can be executed in parallel.

8 threads are collected in units called cores. The multiple threads of a core have local working memory (LWM) and Shared ArithMetic logic unit (SAM), both of which are shared among the multiple threads of the core.

Each thread in a core is configured by the command list as “master” or “slave”. The master and slave have own program area. Vertex Issue starts the master thread and slave threads are started by the master thread.

4 cores are collected in the unit called a cluster. A cluster serves as one processing unit of an IMP-X7 CVengine.

The multiple cores of a cluster have various caches (for data and instructions denoted by D$ and I$), a stream buffer output (SBO), global working memory (GWM) and a Shared Operation Unit (SOU). These are all shared among the multiple cores of the cluster.

The scratchpad memory and external memory are implemented outside of clusters.

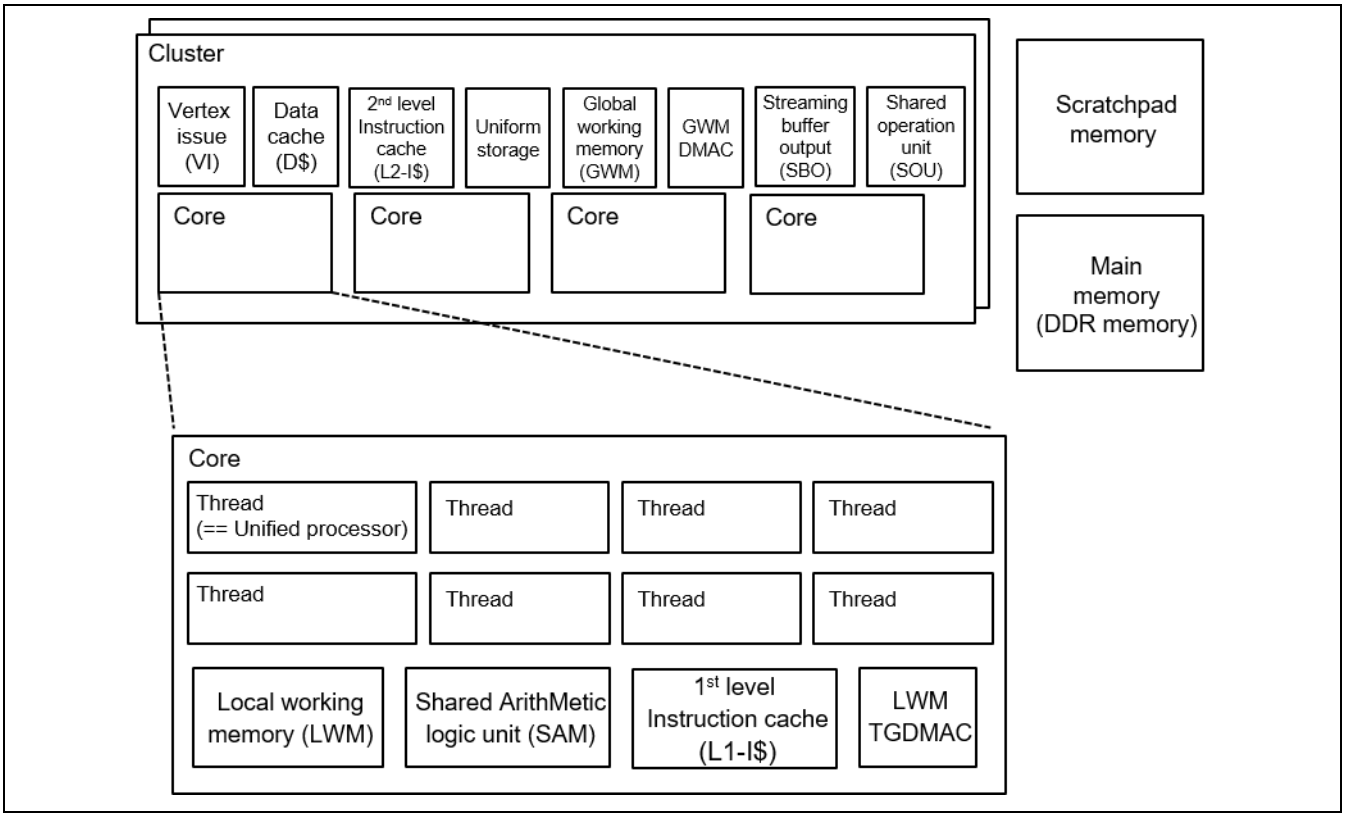



**Figure U32.1.1 Block Diagram of an CVengine Cluster**

The programming model focuses on the thread as the unit of operations by CVengine and its specifications are described in the remainder of this document.

The CVengine incorporates a single type of processor that is the CVengine thread. The CVengine thread is connected to the following ports or modules:

(1) Data input port (Master: Input from Vertex Issue / Slave: Input from Master thread)
(2) Data cache
(3) Instruction cache
(4) Local working memory
(5) Global working memory
(6) Uniform storage
(7) Main memory / Scratchpad memory
(8) Streaming buffer output
(9) Data output port (Master thread only)
(10) Data cache bypassing

Figure U32.1.2 shows an overview of the CVengine thread.

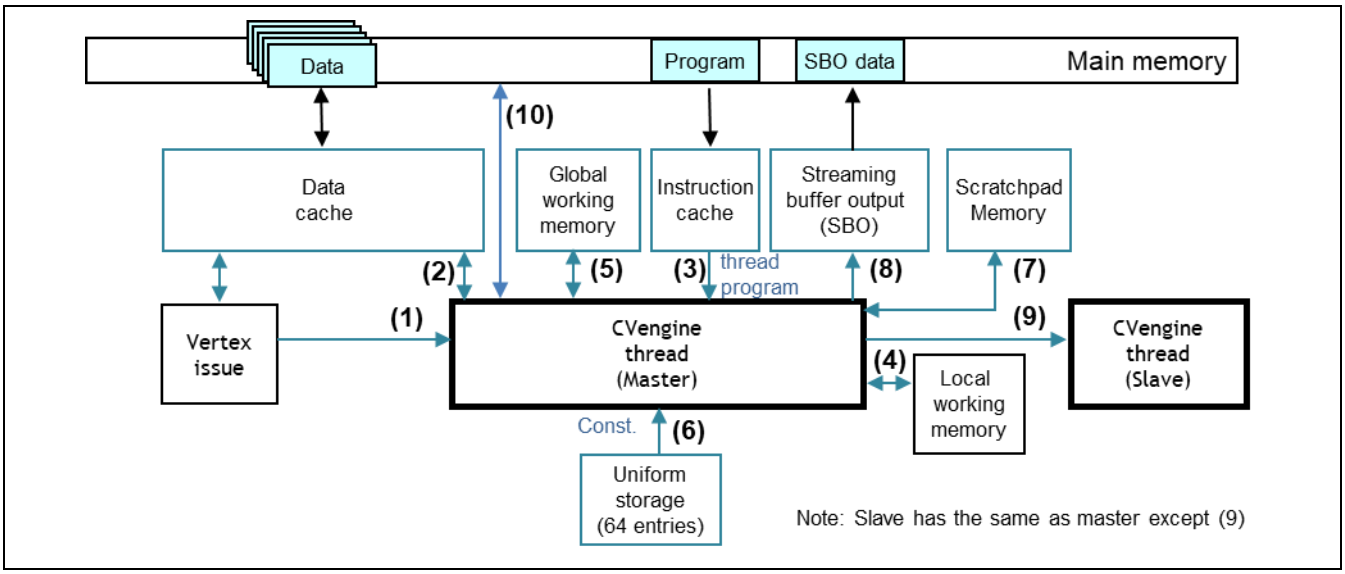



**Figure U32.1.2 Connection Diagram of CVengine Thread and Peripheral Blocks**

### U32.1.1 Input Data

The data input from the data input port is stored in registers from which the thread can read.

### U32.1.2 Cache Memory

There are data cache and instruction cache between main memory and CVengine.

**Table U32.1.1 Cache Memory List**

| | Data Cache | L1 Instruction Cache | L2 Instruction Cache |
|---|---|---|---|
| **Shared among** | All threads in a cluster | All threads in a core | All cores in a cluster |
| **Size** | 8kB x 1bank | 8kB | 16kB |
| **Line size** | 128byte | ← | ← |
| **Structure** | 4-way set associative | 2-way set associative | 4-way set associative |
| **Tag** | Upper 19bit of physical address | Upper 21bit of physical address | Upper 21bit of physical address |
| **Refill** | LRU(Least Recently Used) | ← | ← |
| **Purging** | Write-back | NA | NA |
| **Stage** | 7 stages (responded at stage 6) | ← | ← |
| **Remarks** | | | |

#### U32.1.2.1 Data Cache

A data cache memory consists of 1 bank of 8kbyte memory and is connected to CVengine for fast access to image data and variables in the external memory. The line size of the L1 data cache is128 bytes. Data is read in units of 128 bytes into L1 data cache. Output data can be written out in 32-, 64- or 128-byte unit. i.e., When in one line of 128 bytes, only 32 bytes are marked as dirty then only 32 bytes will be written out. Access to the data cache from the thread is specified by X-Y coordinates or linear address on Shader memory space. Refer to Figure U32.5.1 for this linear memory space. Addresses in the main memory are automatically calculated from the specified X-Y coordinates or linear address using the image, data format, base address, or stride. More than one base address can be handled, and they are specified by configuration registers. All threads of a cluster share the same L1 data cache. Use software to maintain data coherence between the threads if required. For bypassing this data cache, refer to section U32.8.9 .

The 0th level cache (named as “pbuf”) has a 32byte long write buffer with write-back method and a 32byte read buffer which are implemented at the data cache (D$) access port of each thread to improve data access performance at D$ conflicts. This cache is automatically flushed but not cleared at the thread end, TRAP instruction. Clearing this cache is user responsibility. A dedicated instruction SYNCP is also available to clear or flush the cache manually and it invalidates the content in pbuf. Be noticed that pbuf is only connected to D$ thus it does not work for accessing any other memory (e.g., LWM).

#### U32.1.2.2 Instruction Cache

The program which is executed by the thread is supplied from the instruction cache. The instruction cache consists of the level 1 cache which is 8kbyte of 2way set associative cache and shared by all 8 threads in a core. The level 2 cache is 16kbyte of 4way set associative cache which is implemented per cluster and is shared by all 32 threads.

In addition to the above, a 0th level cache is also implemented as 8 buffers with a size of 32byte. It is placed between the thread and the instruction cache allowing 64 instructions to be cached. For details on improving performance by effectively using the 0th level instruction cache, refer to section U32.14.4 .

### U32.1.3 Local Working Memory

The local working memory is readable and writable memory. In non-common access mode, each thread has its own dedicated local working memory bank, and it is only accessible by the thread. In common access mode, the full local working memory address space can be accessed by any thread inside a core. The size of local working memory is 3kB per thread in non-common mode and 24kB per core in common mode, refer to U32.5.8 for more information.
Each core of a cluster has local work memory.

The local working memory can be used as space for holding temporary variables which cannot be stored in general-purpose registers.

### U32.1.4 Global Working Memory

The global working memory is readable and writable and is shared by all threads of the CVengine. It can be used to hold common data which is shared by all threads. The size of global working memory is 8kB per cluster.

The global working memory has additionally two types of special storage. One storage is a semaphore register. This register is a 1-bit register that can be read from and written to by all threads. When this register is read, the register value is forcibly changed to 1 after the read operation. This allows the shared resources between the threads to be managed.

The other storage is an increment register. This register is a 32-bit register which can be read from and written to by all threads. When this register is read, the register value is incremented by 1 after the read operation. This allows the order between the threads to be controlled.

Since all threads share the global working memory, use software to maintain data coherence.

### U32.1.5 Uniform Storage

The uniform storage is used to hold constants that can be used by programs. Values that were stored as source operands from specific instructions can be used. Values need to be already set in the uniform storage by the Vertex Issue through the command list before programs are started. The uniform storage comprises 64 32bit entries.

### U32.1.6 Streaming Buffer Output

The streaming buffer output is used to consecutively write data to the external memory. Up to three types of data can be written. The type of data to be written can be set by the control register.

As the streaming buffer output is shared by multiple threads, the order in which data is written to the external memory cannot be guaranteed between the threads.

### U32.1.7 Data Output Port

The data output port of the thread is for use in starting slave threads by issuing the ACTST instruction. The data through the port is used as arguments, provided by the master thread for the slave threads.

## U32.2 Overview of Thread Hardware

The features and overview of a thread are shown in Figure U32.2.1.

Features

- Input/output interface to register file
- 32- and 16-bit signed-integer arithmetic units and floating-point arithmetic unit operating in the IEEE 754 single-precision format
- SIMD operation for multiple-accumulation with 64-bit width input.
- Data read from the constant area (uniform storage) can be used in operations.
- Conditional execution instructions are supported.
- SAM supports special arithmetic operations such as multiplying and rotating which are shared by multiple threads.
- SOU supports other special arithmetic operations (e.g., square root) which are shared by multiple cores.

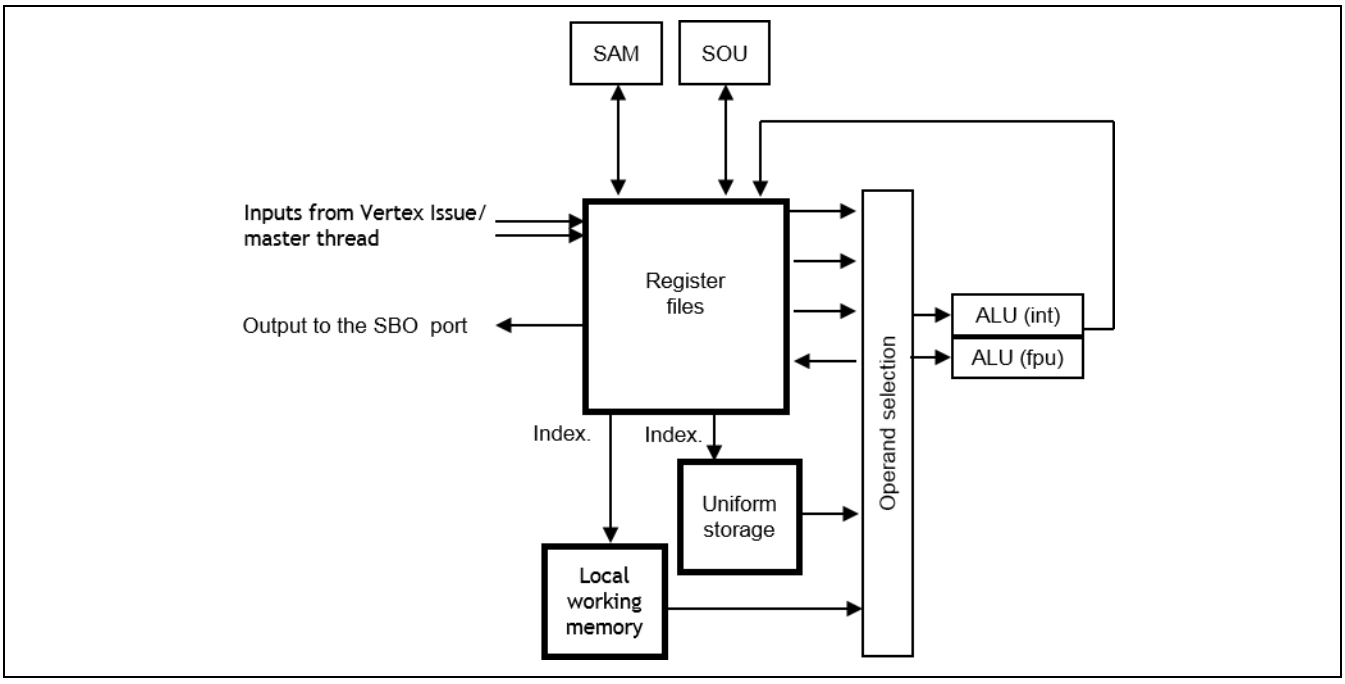



**Figure U32.2.1 Data Flow in CVengine Thread**

Input/output ports to interface with external modules are connected to the register files, and the thread can make access by reading from registers or writing to registers. Each register file consists of 48 32-bit registers.

## U32.3 Thread State

### U32.3.1 Program Start and Stop

The thread can be in one of two states: idle state or processing state. In the idle state, the PC (Program Counter) is stopped, and no operation is done. When the previous stage (vertex issue/master thread) has completed setting values to necessary input registers, an activation signal is generated. The thread loads the specified PC and transitions to processing state. To shift from the processing state back to the stopped state, execute the TRAP instruction, to bring the thread into idle state.

If a break occurs in the processing state due to a CVengine configuration register setting, the thread transits to the break state after execution of the decoded instruction has completed. Also, when an interrupt happens with the INT instruction, the thread goes to the break state. In break state, the PC is stopped, and the thread state can be read and changed by a CVengine configuration register. When the break state is released due to a CVengine configuration register setting, the thread returns to the processing state.

For information how a thread can be disabled, please refer to section U32.12.1.49 . The CVengine can also be reset by a configuration register but resetting while a thread is in running or breaking state is prohibited and behavior is not defined. For more information, please refer to section U32.12.1.2 .

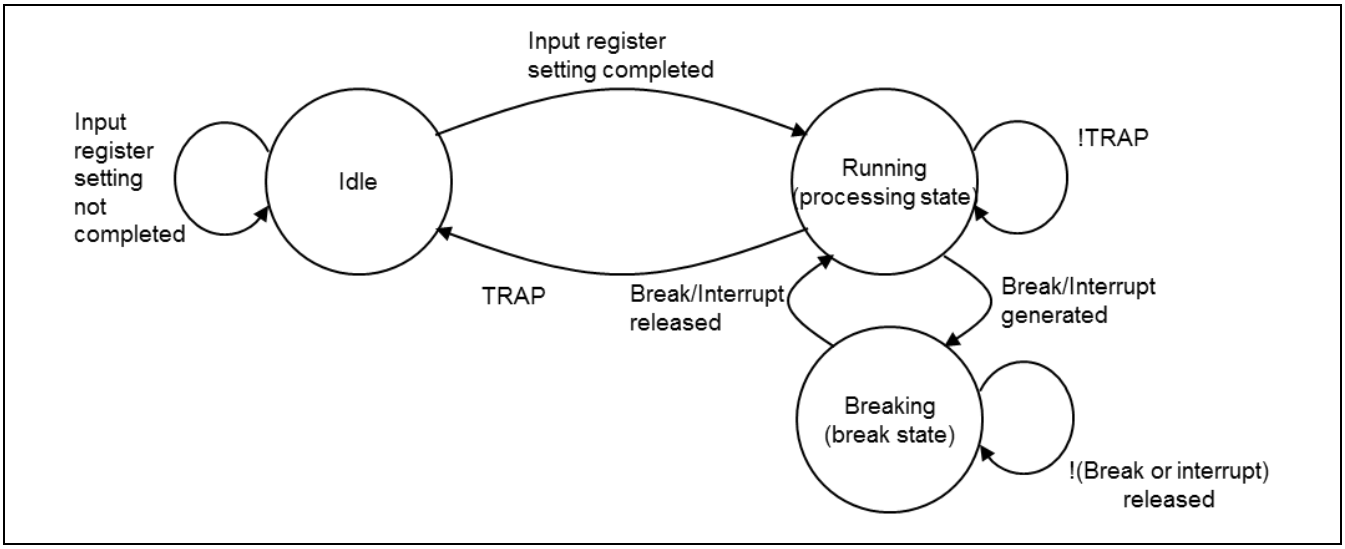



**Figure U32.3.1 State Transitions of CVengine Thread**

### U32.3.2 Master and Slave

Each thread in CVengine can take a role as master or slave. This role is configured for each thread by TGSEN1 configuration register and fulfilled for the commands POINT, POINTS and RECT. Differences between master and slave are as follows.

- The Vertex Issue starts master threads only.
- The slave threads are only called by master threads by ACTST, which slave thread is called is assigned by the queue arbiter.
- Only the master threads can issue SYNCG and wait for completion of the slave threads.
- R2 is used as parameter transferred from master to slave (e.g., master ID or LWM base address).
- The slave thread starts from a dedicated instruction area which is specified by PPCSAR.

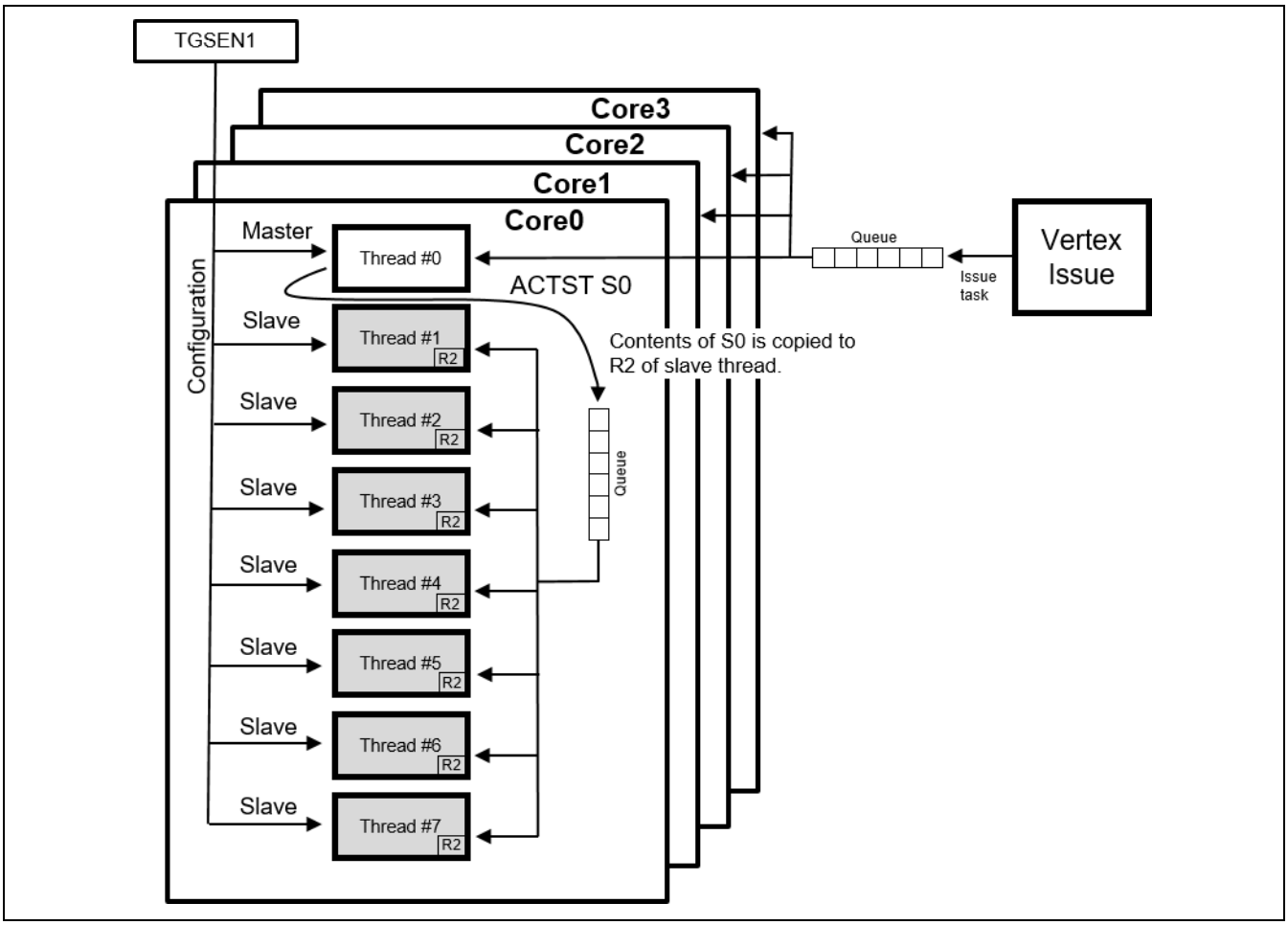



**Figure U32.3.2 Master/Slave Architecture (Case of 1 Master and 7 Slaves)**

Issuing SYNCG or ACTST command by a slave thread causes undefined result. Hardware is not initialized automatically when a slave thread is started. Initialize manually with SINI if necessary.

The following figure shows a typical and basic master-slave sequence in case of 1 master and 7 slaves' configuration.

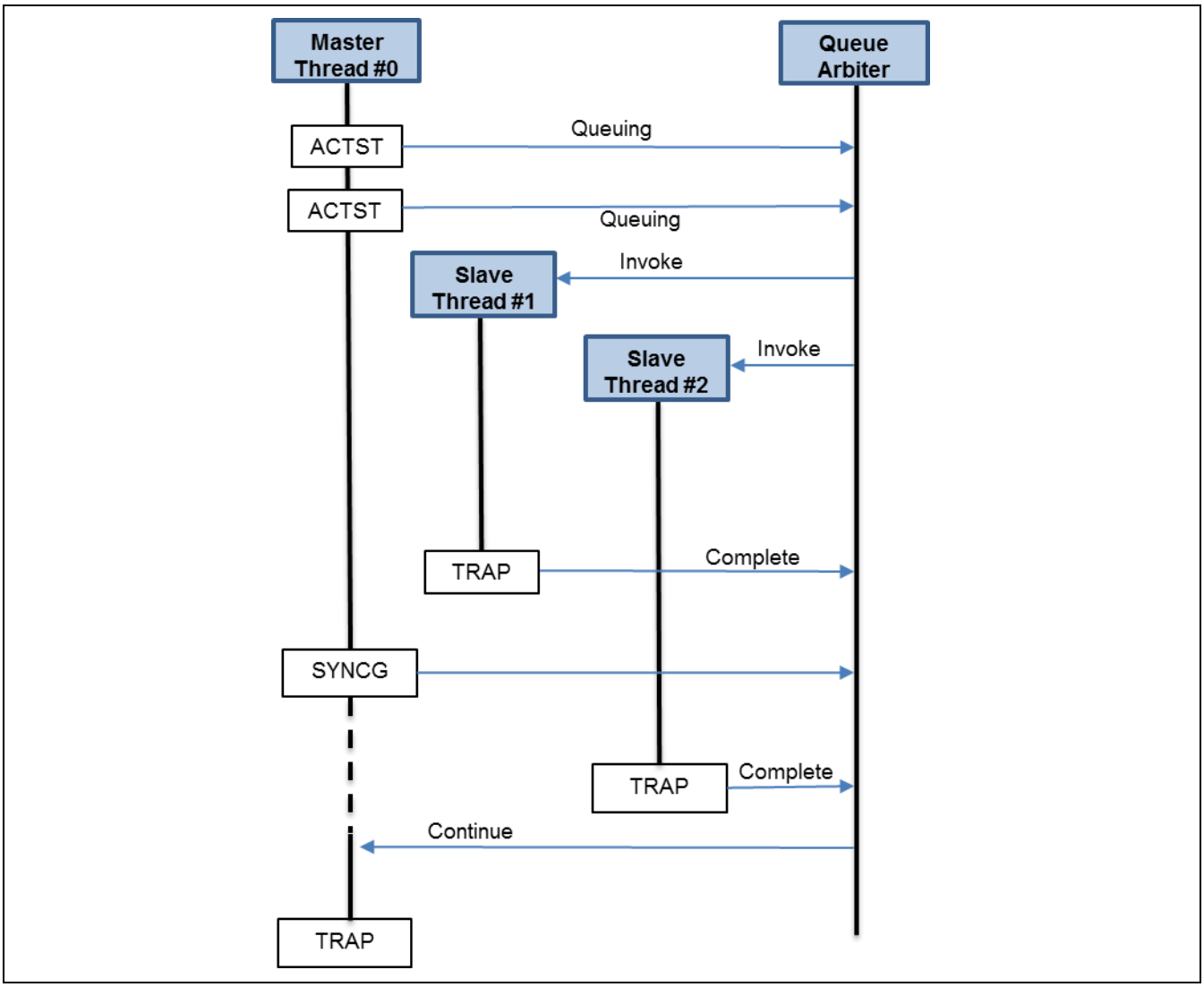



**Figure U32.3.3 Typical Master/Slave Sequence**

In the above figure, the master thread deploys two slave threads by issuing two ACTST via the queue arbiter, then two slave threads proceed their task. The queue arbiter detects completions (TRAP instruction) from all slaves and resumes the master thread which waits by SYNCG.

As derived cases from above:

- The master must issue SYNCG before TRAP.
- The master can issue SYNCG with no ACTST issuing. In this case, SYNCG is treated as NOP.

The master can issue ACTST at any time. If there is no vacant slave in the same core, the master stalls. SYNCG makes the master to wait for all the slave threads to finish that are invoked with ACTST by the master thread.

Only master threads can initiate an LWM-DMA transfer and only 1 DMA transfer can be active at a time.

All threads can be set to either master or slave with TGSEN1. This TGSEN1 setting is applied to all 4 cores. For more details, please refer to U32.12.1.49 . Any combination of master and slave is possible.

## U32.4 Memory Space for Program and Data Planes

The CVengine has two types of memory space. One is for program and data planes. The threads access the program area as instruction fetch and the data planes indexed by coordinates of dedicated instructions like LDRXY, STRXY, etc. These memory areas are the scope of this chapter.

The other type of memory space is used for variables and IO which is described in the next chapter.

### U32.4.1 Program Area

The program of the CVengine thread is located in an area allocated in the external memory.

The start address of the area allocated in the main memory needs to be set in the CVengine configuration register VIBAR before program execution starts.

The program is accessed via the instruction cache (I$). There are two types of instruction caches (I$): a 4kbyte 2way set-associative first-level (L1) cache which is shared by all threads in a core and an 8kbyte 4way set-associative second-level (L2) cache which is shared by all cores in a cluster.

The thread refers to the instruction located at the address specified by the VIBAR register as an instruction whose program counter (PC) is 0, and execution starts from the start PC set in the CVengine configuration register. When the SINI command of the vertex issue is used, execution starts from the PC set in the VSINISAR register. When other start commands are used, execution starts from the PC set in the VPCSAR register.

The main memory space that can be accessed as the program area by the thread is 64Kbytes from the start address specified by the VIBAR register.

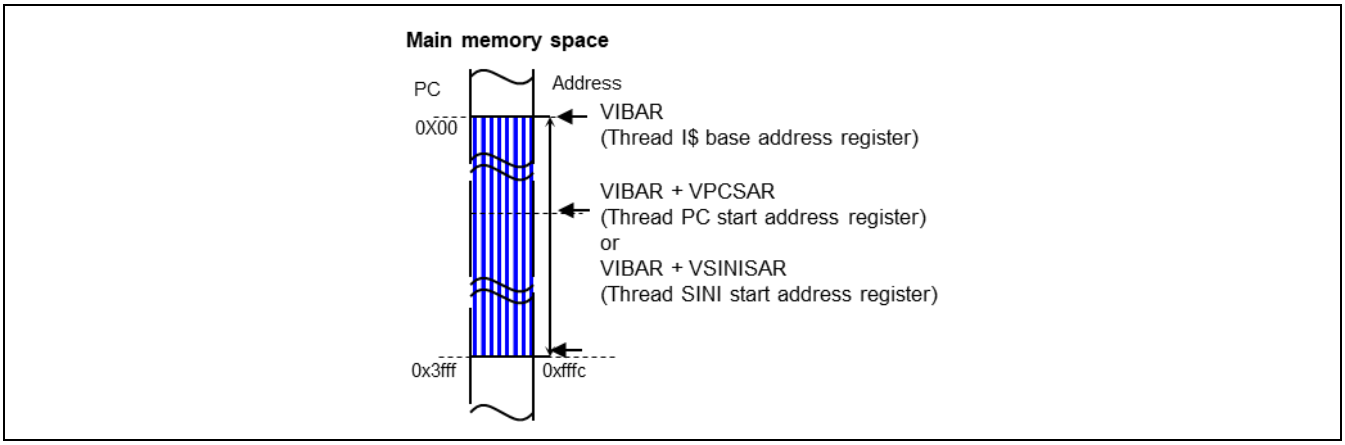



**Figure U32.4.1 Memory Space for Program Area**

### U32.4.2 Data Planes

This external memory space holds image data or variables. Data in external memory can be in variable format and is determined by the fields PFORMAT and PFORMAT2 of the configuration register IMGC0Rn. The area that is accessed by the thread via the data cache (D$) is a two-dimensional space that is always represented by the X and Y coordinates. At the data cache, the X and Y coordinates requested by the thread are converted into a memory address used for accessing the external memory.

With instructions for loading and storing of integers using X and Y coordinates e.g., LDRXY/STRXY, if the data format of the external memory is not 32-bit signed integer, in case of loading, data from the external memory are converted into 32-bit signed integers and then provided to the thread. In case of storing, the 32-bit signed integer from the thread is converted into the data format of the external memory and written afterwards.

With instructions for loading and storing of floating-point numbers using X and Y coordinates e.g., FLDRXY/FSTRXY, if the data format of the external memory is not 32-bit floating-point number, in case of loading, data from the external memory are converted into 32-bit floating-point numbers and then provided to the thread. In case of storing, the 32-bit floating-point number from the thread is converted into the data format of the external memory and written afterwards.

With instructions for loading and storing data using linear addressing e.g., LD/ST, data format conversion is not taken place and data is loaded and stored as-is.

The D$ is composed of one bank of 8-Kbyte 4-way set associative cache.

The CVengine configuration registers (IMGBAR0 to IMGBAR11, IMGSTR0 to IMGSTR11, and IMGC0R0 to IMGC0R11) need to be set before executing the program. The CVengine can handle up to 12 planes of memory space. The n-th plane registers are described as IMGBARn, IMGSTRn, and IMGC0Rn in this document. The planes are shared by all threads. The values of Control Registers CR8 and CR9 are used to select which plane to be read or written by a thread. Refer to section U32.6.2 .

The start address of the data area of the plane should be set in the IMGBARn register. The memory width of the plane should be set in the IMGSTRn register. Both of the settings should be made based on the unit of the data format specified by the IMGC0Rn register. The external memory data format, rounding mode at conversion between integer and floating-point, and saturation specification at conversion of the plane are set in the IMGC0Rn register.

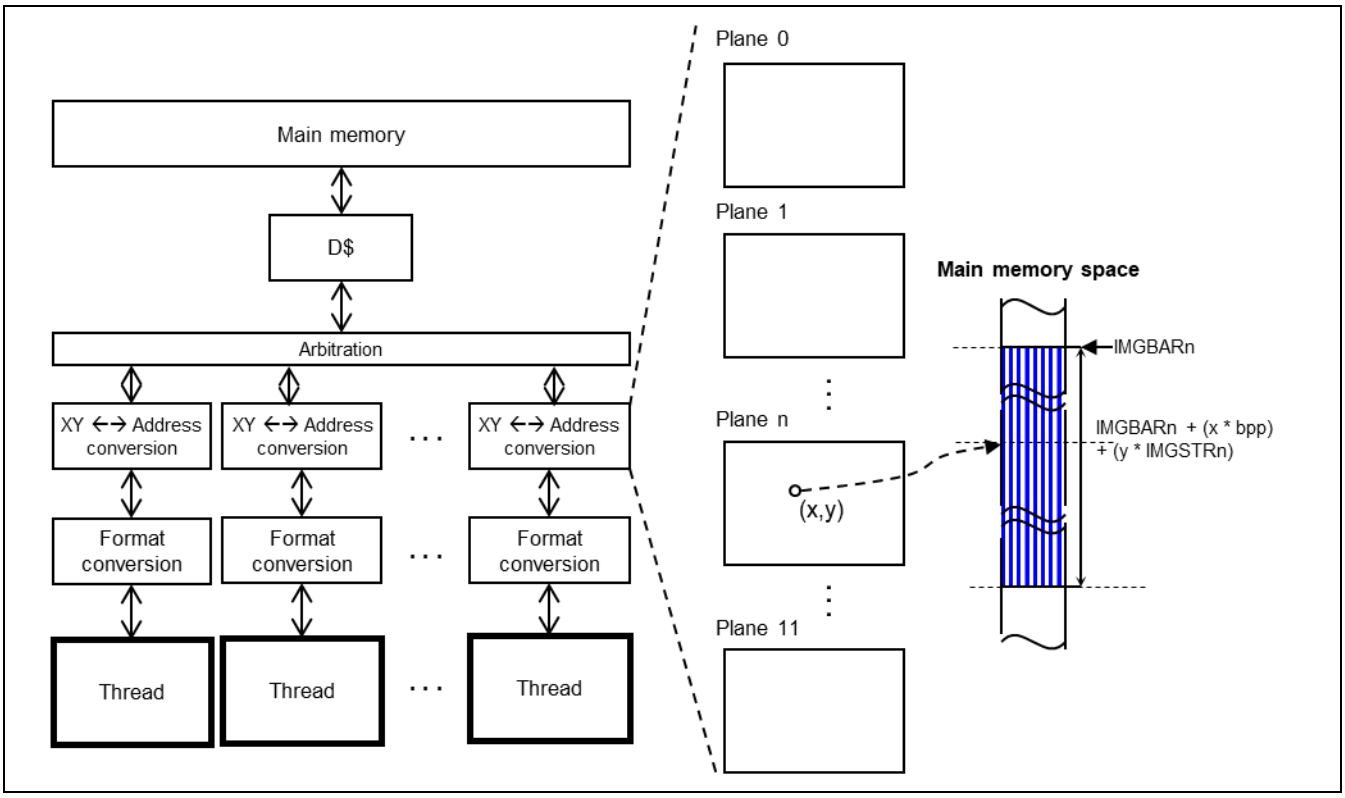



**Figure U32.4.2 Memory Space for Data Cache Area**

**Note: Maximum size of data plane accessed by the thread**

The physical address accessed by a thread is calculated with the following formula:

Physical Address = Base + (X × Format) + (Y × Stride)

Base: unsigned 32-bit, 0 ~ 4294967295 (IMGBARn)
Format: 1/2/4 (IMGC0Rn.PFORMAT/PFORMAT2)
Stride: signed 16-bit, -32768 ~ 32767 (IMGSTRn)
X: signed 32-bit, -2147483648 ~ 2147483647 (R0)
Y: signed 16-bit, -32768 ~ 32767 (R1)
(X × Format) + (Y × Stride) >= 0
Physical Address <= 0xFFFFFFFF

Since the final offset to Base cannot take a negative value, the maximum area size that can be accessed by a thread is 2147483648 × 1 byte for a one-dimensional plane. For a two-dimensional plane, maximum width is 32767 bytes since Stride is signed 16-bit, and the maximum height is 32768 lines since effective range of Y is $0 \leq Y \leq 32767$. The maximum size in pixels varies depending on the Format. (On the other hand, Vertex Issue can handle maximum 65536 × 65536 pixels regardless of the Format. (Refer to section U32.13.1 , U32.13.2 and 0.)

## U32.5 Memory Space for Variables and IO

### U32.5.1 Memory Map and Attributes

The CVengine has a 32-bit address space and any memory space for variables and I/O are mapped on it. It can access host memory space with the same address, which is useful to read/write directly from/to external memory. The Figure U32.5.1 shows this address space with some attributions for each memory area.

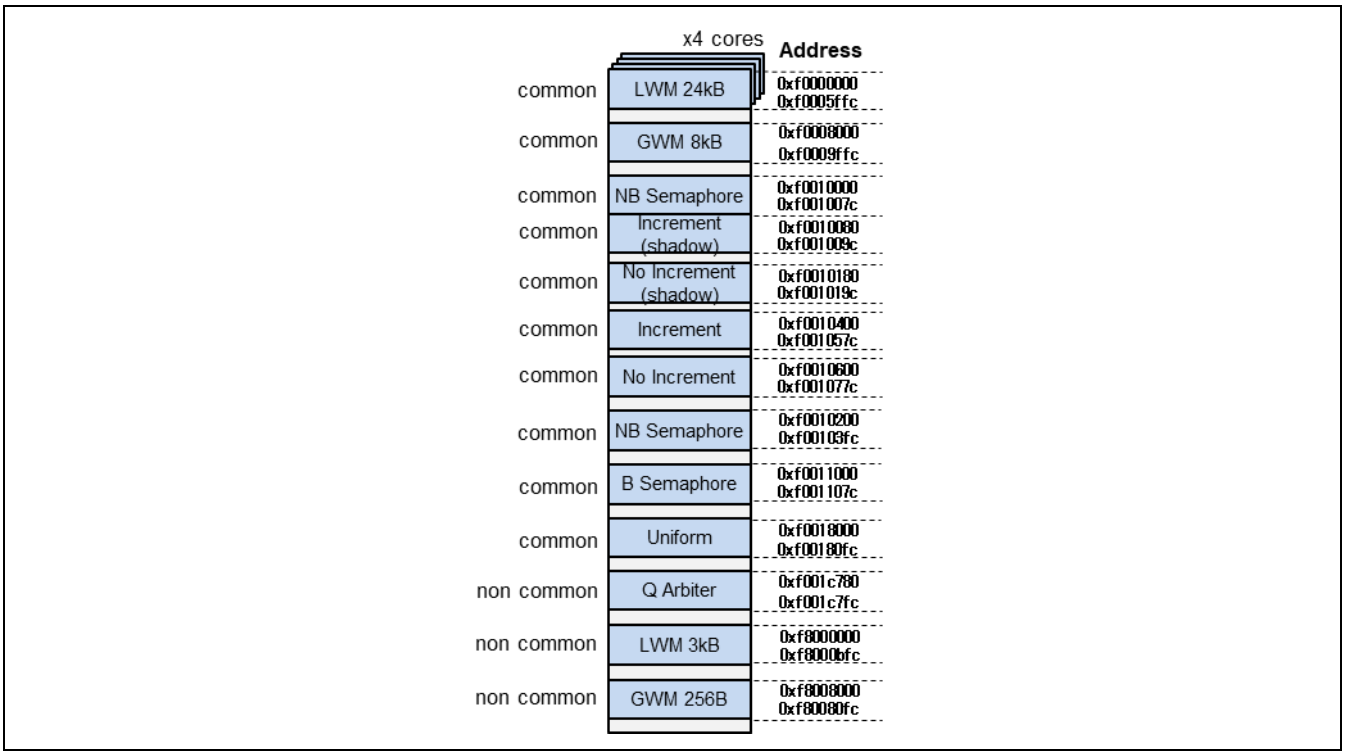



**Figure U32.5.1 Memory Map for Variables and IO**

For this address space, dedicated load/store instructions are used to access actual data. And these load/store instructions have variants to access memory with byte and half word size. For example, store instructions have ST (word size), STH (half word size) and STB (byte size). Any access to an area where nothing is mapped to has no effect, i.e., is dealt as NOP instruction. Regarding common area and non-common area, refer to the section U32.5.8 . Table U32.5.1 shows some attributes of each memory from each master.

**Table U32.5.1 Memory Attributes**

| Transferred Memory / Bus Master | LWM | GWM | Scratchpad | Host Memory Space (External Memory) |
|---|---|---|---|---|
| Host (Arm, System DMA) | Cannot access | Cannot access | Can access | Can access |

| | | | | |
|---|---|---|---|---|
| | | | - Mapped on virtual memory space of Arm<br>- Non-cacheable *1 | - Mapped on virtual memory space of Arm<br>- Cacheable |
| **CVengine Thread** | Can access<br>- Mapped on CVengine memory space<br>- Non-cacheable, but fast.<br>- Thread dedicated. Cannot be shared with other thread.(non-common access)<br>- Can be shared with other thread in the same core.(common access)<br>- Data only. CVengine instruction cannot be loaded<br>- General CVengine load / store instructions are used.<br>- 8-bit, 16-bit, 32-bit, 64-bit access are available | Can access<br>- Mapped on CVengine memory space<br>- Non-cacheable, but fast.<br>- Thread dedicated. Cannot be shared with other thread.(non-common access)<br>- Can be shared with other thread in the same cluster.(common access)<br>- Data only. CVengine instruction cannot be loaded<br>- General CVengine load / store instructions are used.<br>- 8-bit, 16-bit, 32-bit access are available. | Can access<br>- Mapped on CVengine memory space<br>- Cacheable<br>- Both data and CVengine instruction are loaded<br>- General CVengine load / store instructions are used.<br>- 8-bit, 16-bit, 32-bit access are available.<br>- Dedicated instructions e.g., LDRXY/STRXY are also available to access the data.<br>- Can be shared among threads and clusters | Can access<br>- Mapped on CVengine memory space.<br>- Cacheable<br>- Both data and CVengine instruction are loaded<br>- General CVengine load / store instructions are used.<br>- 8-bit, 16-bit, 32-bit access are available.<br>- Dedicated instructions e.g., LDRXY/STRXY are also available to access the data. |
| **CVengine DMA** | Can access<br>- Master thread / program can use this DMA.<br>- Arm cannot use this DMA.<br>- Not through cache. Physical memory to memory. | Can access<br>- CVengine thread / program cannot use this DMA.<br>- Arm can use this DMA.<br>- Not through cache. Physical memory to memory. | Can access<br>- Not through cache. Physical memory to memory. | Can access<br>- Not through cache. Physical memory to memory. |
| **SBO** | Cannot access | Cannot access | Can access<br>- Write only | Can access<br>- Write only |

*1 Arm can access the scratchpad memory with 4byte width only. Accessing the scratchpad memory via data cache of Arm is not supported.

### U32.5.2 Constant Memory Area

This memory area is for storing uniform storage data (constants that can be read from the CVengine threads). CVengine has 64 entries, and each entry has 32-bit long and is mapped in the memory space.

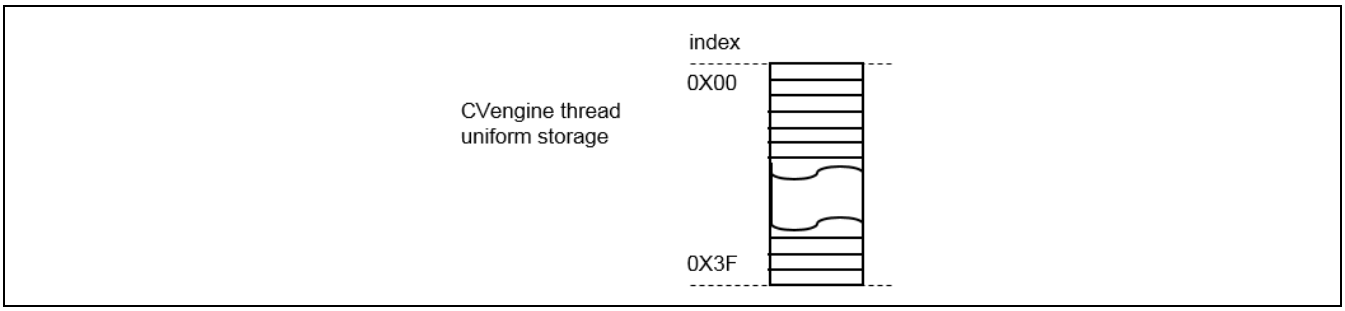



**Figure U32.5.2 Address Space for Constant Memory Area**

The uniform storage area can be accessed by the dedicated instructions like MOVU, ORU, ADDU using a dedicated index. The uniform storage can always be accessed by these instructions without penalty cycles.

In addition, general memory load instruction LD is available to read the uniform storage area. This general load instruction can access the constant memory area described in the section U32.5.1 by word, half word and byte unit.

To mention is, the dedicated instructions like e.g., MOVU specify an address on this area by “index”, namely address is in a word unit. However, the general load/store instructions specify an address by “byte address”, same as other memory areas.

### U32.5.3 Local Working Memory Area

The local working memory size is in total 24 Kbytes for all 8 threads in a core and it has a 32-bit width. This memory is not cached. There are common and non-common accessing modes to this memory, for details refer to section U32.5.8 . In non-common mode this memory cannot be accessed from other threads, so it can be used as temporal data storage by each thread of CVengine. Each thread can access this local working memory via same memory window, but actual

physical memory is dedicated to each thread. See Figure U32.5.3 for this mapping. In common-access mode, all the threads in a core share one memory space. Accessing LWM in another core is impossible.

The local working memory can be accessed by general load/store instructions like LD/ST.

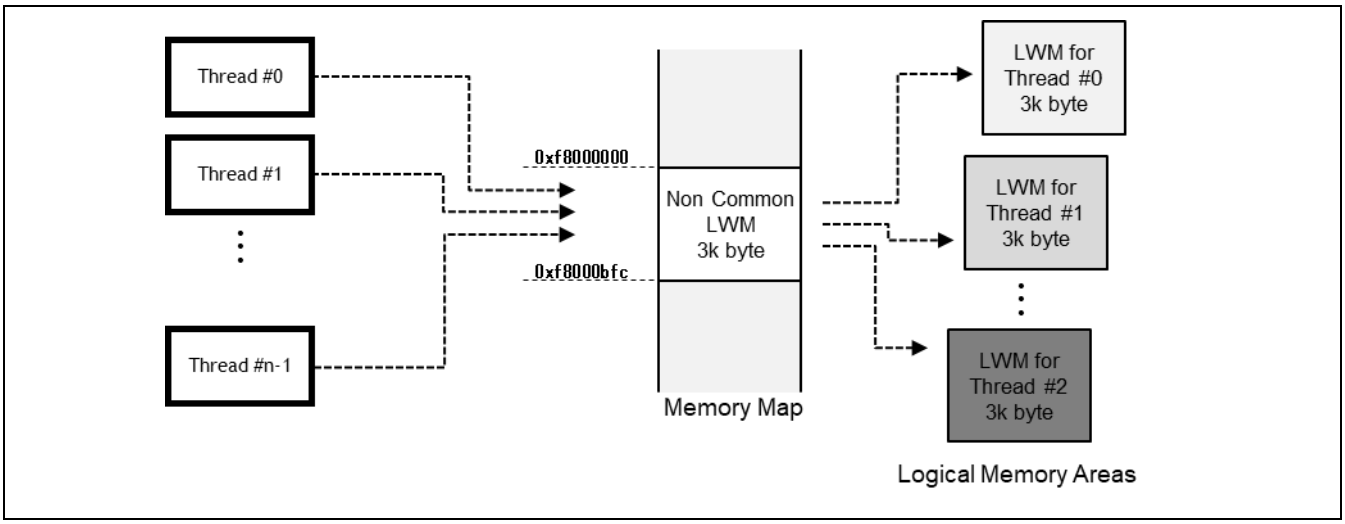



**Figure U32.5.3 Local Working Memory Space**

### U32.5.4 Global Working Memory Area

The global working memory is an area that can be accessed by all threads of the CVengine. This memory is not cached. There are common and non-common accessing modes to this memory, for details, refer to section U32.5.8 . The global working memory of the CVengine is composed of an 8 Kbyte data area and two types of special registers (semaphore register and increment register).

The data area is a 32-bit width memory area to which values can be read from or written to freely. When data coherency needs to be guaranteed, synchronization by the executed program is necessary.

The global working memory can be accessed by general load/store instructions like LD/ST.

### U32.5.5 Semaphore and Increment Register Area

This common area is divided into 2 parts: semaphore and increment part. These register areas must be accessed with a word (32-bit) access instruction. Accessing these areas with instructions in 16-bit or 8-bit length is prohibited.

#### U32.5.5.1 Semaphore Register Area

The semaphore register area is from 0xF0010200 to 0xF00103FC. There are 128 32-bit long semaphore registers. There is also another semaphore register area from 0xF0010000 to 0xF001007C, which is the mirror of the first 32 of the 128 semaphores.

The semaphore register holds only 0 or 1 and it can be read from and written to by all threads. When the semaphore register is read, the register value is forcibly changed to 1 after the read operation.

This feature can be used to control accesses to shared resources. Each register is assigned the state of a resource. 0 indicates that the resource is available, whereas 1 indicates that the resource is not available. During program execution, as long as the relevant semaphore register value is 1, a wait routine to wait for the semaphore register value to become 0 shall be executed. When the value has changed to 0 and is read, perform the processing using that resource and write 0 to the relevant semaphore register at the end. This allows another thread that was waiting to use that resource to get exclusive access.

Operation is not guaranteed when a value other than 0 or 0.0f is written to the semaphore register.

The non-blocking (read without blocking) semaphore register area is from 0xF0010200 to 0xF00103FC, the blocking (read with blocking) semaphore area is available from 0xF0011000 till 0xF001107C.

Same as the non-blocking semaphore register, the blocking semaphore register also holds only 0 or 1. Each blocking semaphore register is assigned the state of a resource. 0 indicates that the resource is available, whereas 1 indicates that the resource is not available. When the blocking semaphore register is read, the register value is forcibly changed to 1 after the read operation. Only 1 thread is able to read the blocking semaphore register at one time. While the register value is 1, any thread trying to read the blocking semaphore register stalls so that the read value of the blocking semaphore is always 0. When the register value is changed from 1 to 0, the stalling thread restarts and read the register value again. Only 0 can be written to the blocking semaphore register regardless of the value stored in the register. Unlike the non-blocking semaphore registers explained above, waiting loop is not needed to wait for the register value becoming 0 to control resources. The round robin arbitration scheme is used to control the threads accessing to the blocking semaphore register. Writing access has higher priority than reading access.

After a load instruction reading a blocking register, an instruction reading the loaded value must be inserted. For example, instruction sequence below is valid.

```
LD    R26, R13, R3; R13 points to the blocking semaphore register
MOV R3, R26; an instruction reading the loaded value
```

Note that there is no need to insert the read instruction just after the load instruction. Instructions that do not access resources protected by the semaphore can be scheduled in between. Generally, compiler for CVengine does not support this insertion automatically, insertion is user responsibility.

**Note:** For non-blocking semaphore, value is returned regardless of successful acquisition of a semaphore. Because of the hardware implementation, a semaphore may not be acquired forever depending on access timings from threads. This is an example case of deadlock. When the value of a shared variable is changed after acquisition of a semaphore, and if both reading and writing from/to the shared variable use the same semaphore, reading from the shared variable by a thread can disturb writing to the shared variable from another thread so that deadlock can happen. To avoid this kind of deadlock, logic must be designed carefully. For example, only the thread writing to the shared variable uses the semaphore.

#### U32.5.5.2 Increment Register Area

The increment register area is from 0xF0010400 to 0xF001057C. There are 96 32-bit long increment registers.

The increment register is a 32-bit register in the integer format, and it can be read from and written to by all threads.

When the increment register is read, the register value is incremented by 1 after the read operation. This feature can be used by each thread to acquire unique serial numbers that are consecutive.

The default value of the increment register is 0. User can write 0 to reset the increment register. Any other value can also be written to the increment register.

A second access area for the increment registers exists for read access from 0xF0010600 to 0xF001077C. When using this access area for a read access, the register value is read without incrementing the value by 1. A write to this access area is not allowed.

For the first 8 increment register, a shadow area exists from 0xF0010080 to 0xF001009C for an access with incrementation and from 0xF0010180 to 0xF001019C for an access without incrementation.

### U32.5.6 Scratchpad Memory Area

CVengine can access to a 16MB scratchpad memory area. Scratchpad memory is formerly called “IMPC”. This memory is accessible by all threads of all clusters. This is useful to exchange data with any external core like IMP Core or Arm without using DDR memory. There are common and non-common accessing modes to this memory.

The CVengine memory map prepares 16MB area for scratchpad, but the mapping to the physical memory is decided by the bus system which is outside of the scope of CVengine. For the used address map, refer to User's Manual: Hardware, section U34.1.3, Address map.

### U32.5.7 Queue Arbiter Register Area

This area has some configuration registers which are read/written by each thread to control its behavior. TGDMCR0, TGDMEXTMEMOFSADDR, TGDMLWMBADDR, TGDMLWMSTRR, TGLNGR, TGSBOCNTR0 are implemented to configure TGDMAC “Thread Group DMAC” and start/stop DMA transfer between external memory and local working memories.

The start address of this area is 0xf001c780 and the end address is 0xf001c7fc.

These registers are so called “non-common”, namely, each thread has his own register set which cannot be directly accessed by other threads. And they must be accessed by word (32-bit) unit. Half word (16-bit) and byte access are prohibited.

### U32.5.8 Common and Non Common Area

In the following, the relation between common and non-common address will be shown. For LWM and GWM, common and non-common base addresses can be taken from Figure U32.5.1. For scratchpad memory, the common base address can be taken from the User's Manual: Hardware, section U34.1.3, Address map, the non-common base address can be calculated as follows:

non-common base address = 0xee000000 + (common base address - 0xed000000) / 32

After, each address can be converted between common and non-common style. In the case of common address calculation from non-common address, do as follows:

1.) Calculate non-common offset from non-common address by subtracting non-common base address from non-common address as follows:

2.) Calculate common offset from non-common offset as follows:

common offset = non-common offset * (32 + thread #x)

3.) Calculate common address from common offset by adding common base address to common offset.

In the case of non-common address calculation from common address, do as follows:

1.) Calculate common offset from common address by subtracting common base address from common address.

2.) Calculate non-common offset from common offset as follows:

non-common offset = (common offset / 32) for thread ((common offset / 4) % 32)

3.) Calculate non-common address from non-common offset by adding non-common base address to non-common offset.

#### U32.5.8.1 Non-Common Access to LWM

Some memory areas have two memory regions, one is for non-common area and the other is for common area. The non-common area provides one virtual area on the memory map. Through this area, each thread can access a physical memory “bank” dedicated to each thread. This non-common access is typically used in the case of local working memory.

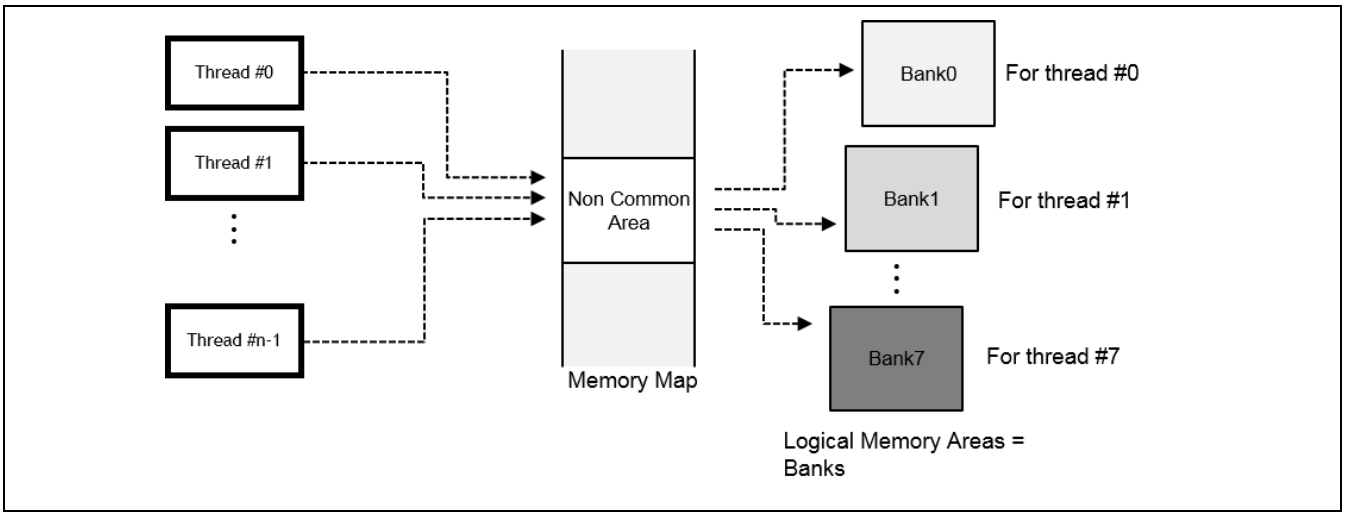



**Figure U32.5.4 Non-common Access to LWM**

### U32.5.8.2 Common Access to LWM

The common access provides capability to access all banks from any thread. For example, through common area, a thread can access the LWM owned by other threads. However, each bank is not mapped on this common area continuously. A bank appears on the common area every 64 byte (8byte × 8threads), because of the 64-bit access instructions to LWM. This 8 byte access is only to LWM. Access to other memories is in 4 bytes. Figure U32.5.5 shows this mapping for LWM case. A thread can access any bank of other threads, but software should have responsible to manage memory consistency among all threads.

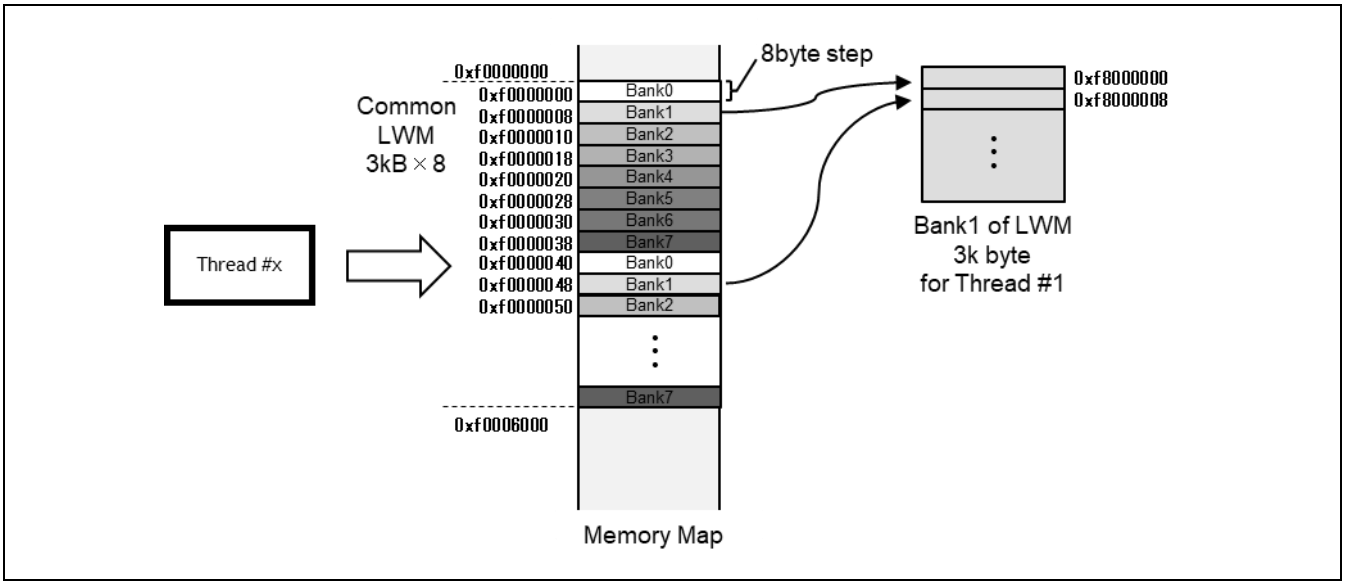



**Figure U32.5.5 Common Access to LWM**

### U32.5.8.3 Access Collision to LWM Common Area

When multiple threads access to the same LWM bank at the same time in common mode, lower prioritized thread stalls until access is ready. The priority is: TGDMAC > thread 0 > thread 1 > … > thread 7. This stall happens for both, reading and writing accesses. For the LWM bank structure in detail, refer to section U32.5.8.2 .

For 8bit, 16bit, 32bit and 64bit LD read accesses to LWM in common mode, a data sharing is implemented so that if multiple threads are reading the same memory address in the same clock cycle, the read data is sent to all these requesting threads. For reading access no stall happens. For writing access stall still happens.

### U32.5.8.4 Common and Non-Common Access to GWM

GWM can be also accessed in either common or non-common mode. GWM is designed to be used in common mode access in general, so that threads can share one global memory area. But, if necessary, GWM can be accessed in non-common mode, so that threads can have a slightly larger dedicated memory than only using LWM.

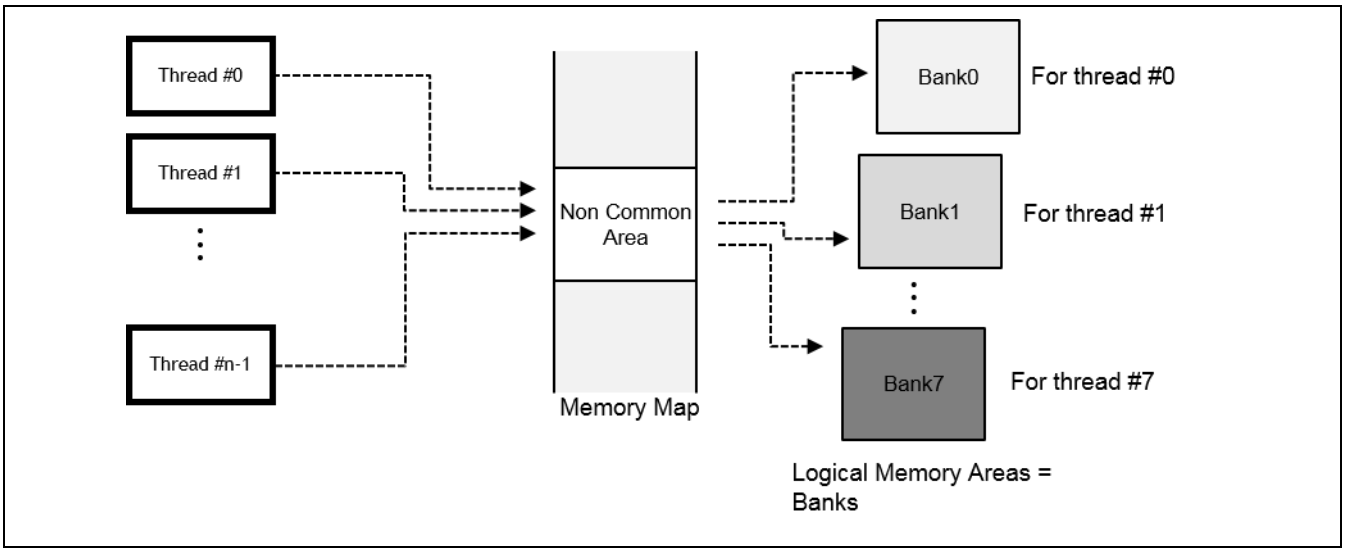



**Figure U32.5.6 Non-common Access to GWM**

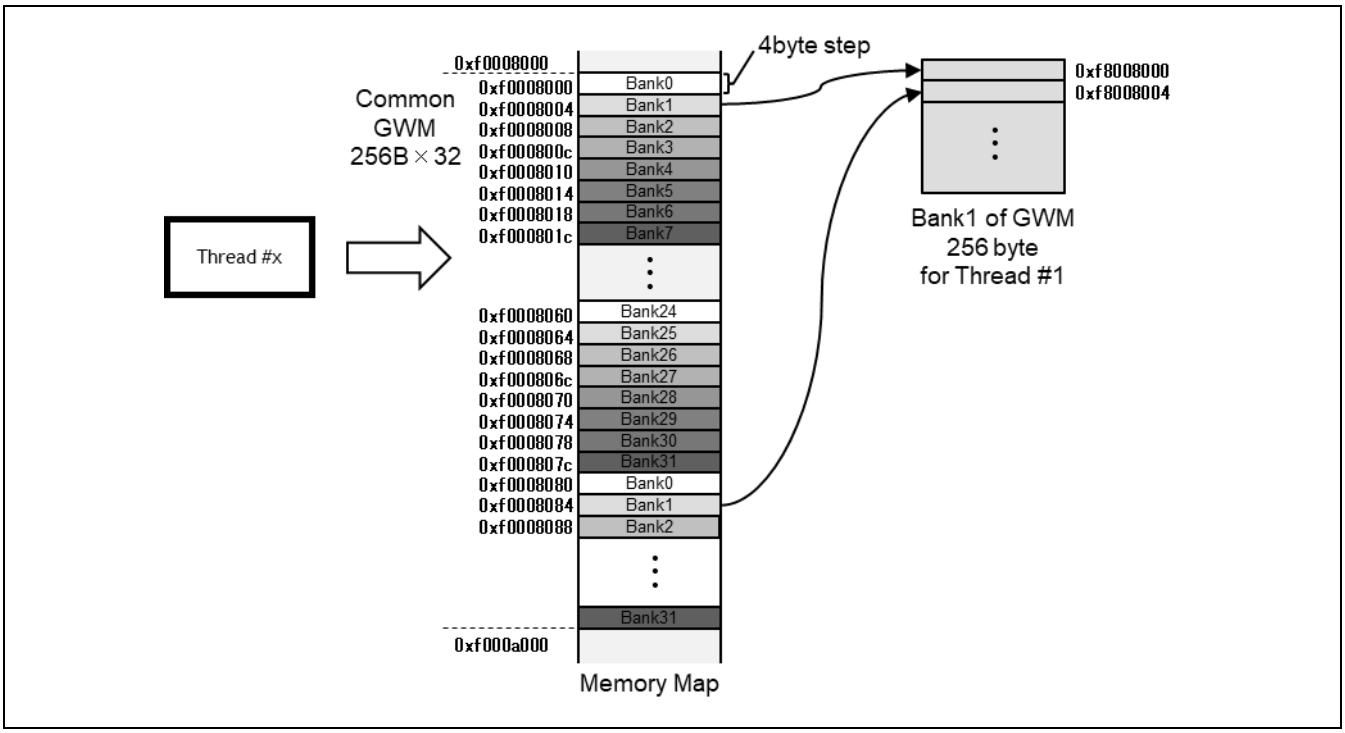



**Figure U32.5.7 Common Access to GWM**

### U32.5.8.5 Common and Non-Common Access to Scratchpad

Scratchpad can be also accessed either in common or non-common mode. Note that Scratchpad memory is shared among clusters even in non-common access mode. Maintain data coherence is user responsibility.

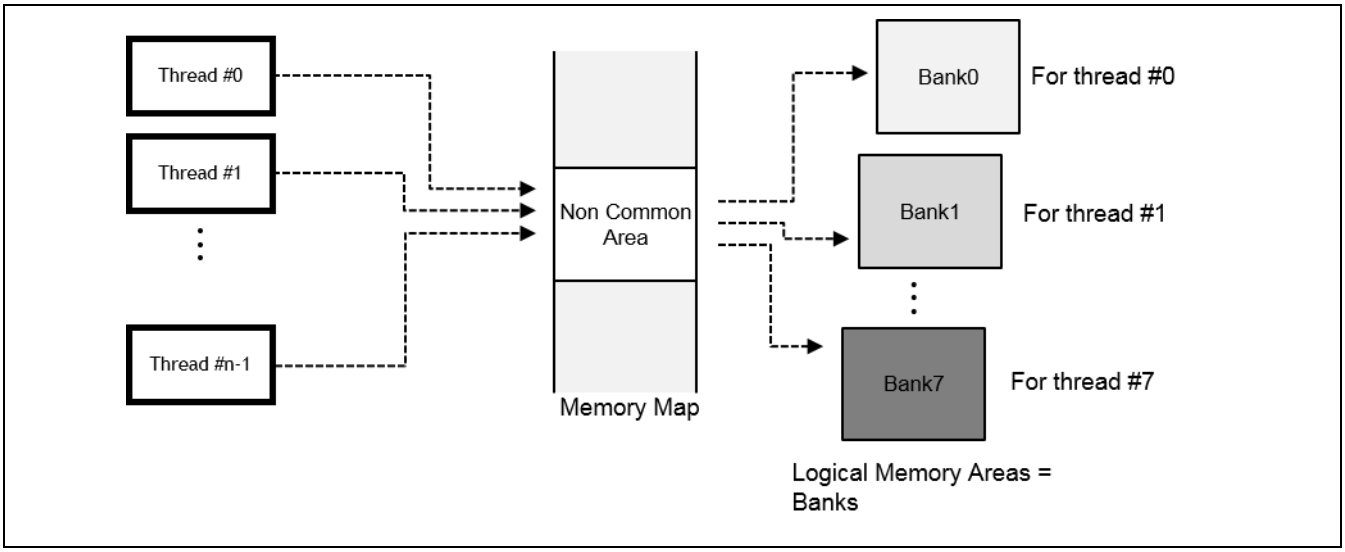



**Figure U32.5.8 Non-common Access to Scratchpad**

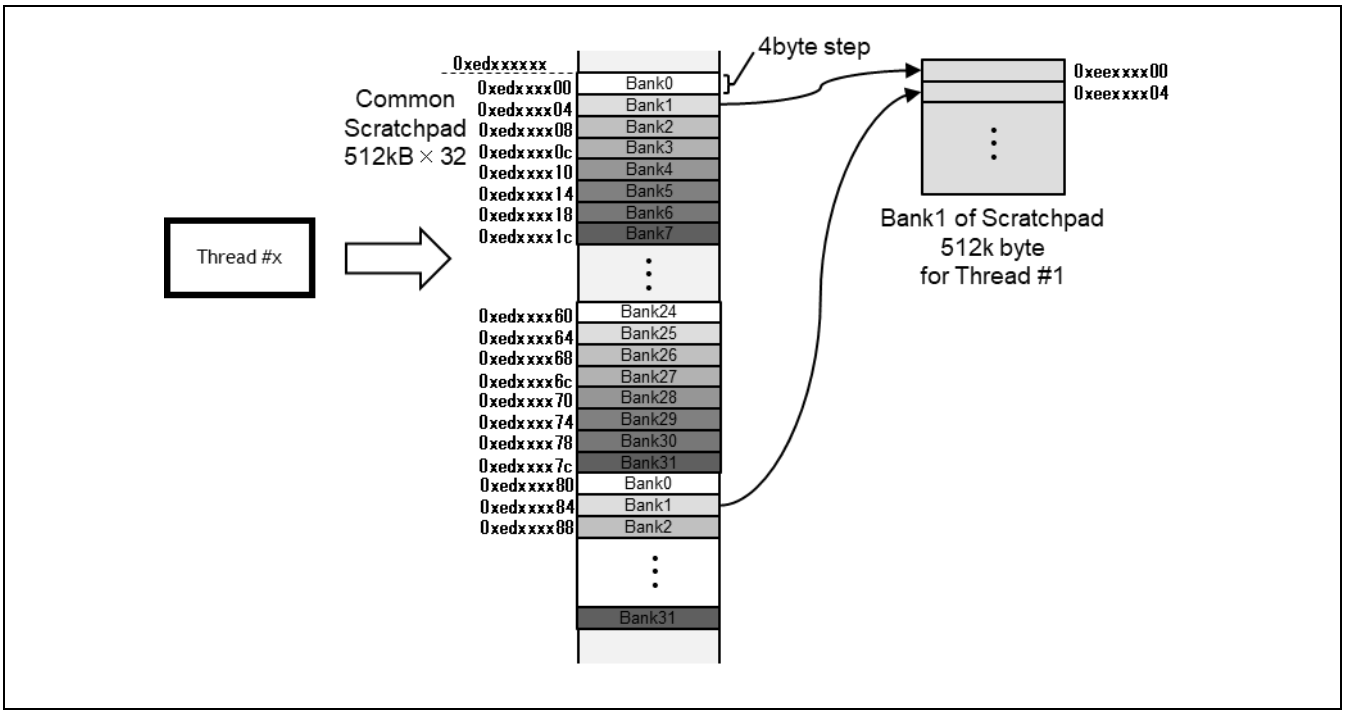



**Figure U32.5.9 Common Access to Scratchpad**

## U32.6 Register Description: CVengine Thread

### U32.6.1 List of CVengine Thread Registers

The CVengine thread has 48 registers with 32-bit width. All these registers are read/write able from thread and at the beginning uninitialized, they are not used as control registers but as data registers to hold e.g., temporary processed data from the thread instructions of U32.9 .

Registers are classified into input registers, output registers, address registers, and constant registers, each of which has special features. Also, there is one status register (SR) which has flags that are set to 1 or 0 depending on the results of compare instructions.

| | |
|---|---|
| R0 and R1 | Input registers in which X and Y coordinate information is set |
| R2 | Inter-Thread-Register to pass a value by ACTST from master thread to slave thread |
| R3 | ZERO register, reading this register returns always zero |
| R4/R5 | General purpose registers. |
| R6 | The register is used either to specify an index address for use in access to the uniform storage, or as a general purpose register. For detail, refer to Note 4 in U32.9 . |
| R7 | General purpose register |
| R8 | Register to calculate the sine of the written value / General purpose register |
| R9 | Register to calculate the cosine of the written value / General purpose register |
| R10 | Register to calculate the inverse of square root of the written value / General purpose register |
| R11 | Register to calculate the exponent of the written value with 2 as the base / General purpose register |
| R12 | Register to calculate the logarithm of the written value with 2 as the base / General purpose register |
| R13 | Register to calculate the square root of the written value / General purpose register |
| R14 and R15 | Registers to calculate the reciprocal of the written value. Independent values can be written in these two registers, and the reciprocal operations are executed in pipeline. / General purpose registers |
| R16 to R47 | General purpose registers |
| SR | Status register, which stores the results of compare instructions |

About R8 to R15 registers, refer to Section U32.6.2 for SOU/GP switching. Also refer to section U32.7.2 for exception handlings.

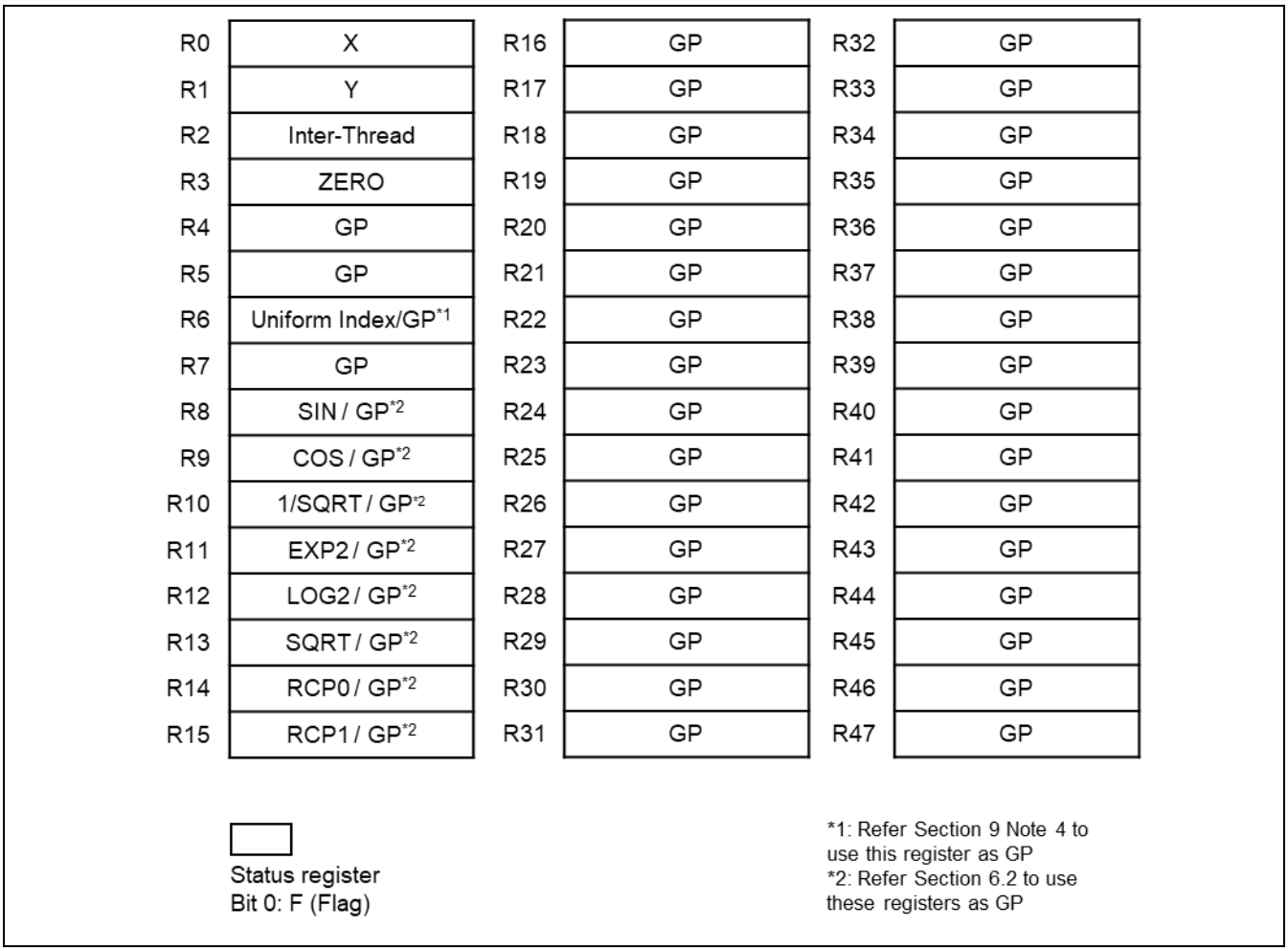



**Figure U32.6.1 Register Files of CVengine Thread**

The following subsections show details of the CVengine thread registers

#### U32.6.1.1 Coordinate Input Registers (R0 and R1)

These are 32bit registers to which the coordinates are input. The X coordinate is input to R0, and the Y coordinate is input to R1. These coordinate values are used when the coordinates are output to the SBO port. The coordinate information can be read as well as changed by the thread.

The coordinates to be input are the values of the X and Y coordinates issued by the vertex issue.

#### U32.6.1.2 Inter-Thread Register (R2)

This register is for a slave thread to receive a 32-bit small information from a master thread. The master thread specifies the register having the value in an operand of ACTST. The slave thread can get the value in its register R2.

Software can use this register as one of general purpose register, if conflict by ACTST writing is managed by software accordingly.

#### U32.6.1.3 Zero Register (R3)

When CVengine program reads this register, 0 “zero” is always returned. Writing any value to this register is ignored.

#### U32.6.1.4 Uniform Index Register (R6)

This register is used to specify an index (address offset in word unit) for use in access to uniform storage.

An instruction uses the value obtained by adding the index specified in this register to the address specified in the instruction as the address for access.

The value of this register is automatically cleared to 0 each time a thread is started.

This register can be used as a general purpose register. Refer to Section U32.9 Note 4 for detail.

#### U32.6.1.5 Sine Operation Register (R8)

This is a register to calculate the sine.

When a value (floating point) is written to this register, input value is multiplied by pi automatically in the hardware, then sine is calculated from the value, namely compute sin(pi*x). Upon completion of operation, the result is written back to this register.

If this register is read from an instruction as a source operand before operation has completed, the thread stalls.

If this register is read before being written, the result is undefined.

This register can be used as a general purpose register. Refer to Section U32.6.2 for detail.

#### U32.6.1.6 Cosine Operation Register (R9)

This is a register to calculate the cosine.

When a value (floating point) is written to this register, input value is multiplied by pi automatically in the hardware, then cosine is calculated from the value, namely compute cos(pi*x). Upon completion of operation, the result is written back to this register.

If this register is read from an instruction as a source operand before operation has completed, the thread stalls.

If this register is read before being written, the result is undefined.

This register can be used as a general purpose register. Refer to Section U32.6.2 for detail.

#### U32.6.1.7 Inverse of Square Root Operation Register (R10)

This is a register to calculate the inverse of square root.

When a value (floating point) is written to this register, square root and inverse operation are executed with the value used as an operand. Upon completion of operation, the result is written back to this register. If a negative value is specified as an input, NaN (exponent=0xff, fraction!=0) is returned.

If this register is read from an instruction as a source operand before operation has completed, the thread stalls.

If this register is read before being written, the result is undefined.

This register can be used as a general purpose register. Refer to Section U32.6.2 for detail.

#### U32.6.1.8 Exponent Operation Register (R11)

This is a register to calculate the exponent with 2 as the base.

When a value (floating point) is written to this register, exponent operation with base 2 is executed with the value used as an operand. Upon completion of operation, the result is written back to this register.

If this register is read from an instruction as a source operand before operation has completed, the thread stalls.

If this register is read before being written, the result is undefined.

This register can be used as a general purpose register. Refer to Section U32.6.2 for detail.

#### U32.6.1.9 Logarithm Operation Register (R12)

This is a register to calculate the logarithm with 2 as the base.

When a value (floating point) is written to this register, logarithm operation with base 2 is executed with the value used as an operand. Upon completion of operation, the result is written back to this register.

If this register is read from an instruction as a source operand before operation has completed, the thread stalls.

If this register is read before being written, the result is undefined.

This register can be used as a general purpose register. Refer to Section U32.6.2 for detail.

#### U32.6.1.10Square Root Register (R13)

This is a register to calculate the square root.

When a value (floating point) is written to this register, square root operation is executed with the value used as an operand. Upon completion of operation, the result is written back to this register. If a negative value is specified as an input, NaN (exponent=0xff, fraction≠0) is returned.

If this register is read from an instruction as a source operand before operation has completed, the thread stalls.

If this register is read before being written, the result is undefined.

This register can be used as a general purpose register. Refer to Section U32.6.2 for detail.

#### U32.6.1.11Reciprocal Operation Registers (R14 and R15)

These are registers to calculate the reciprocal.

When a value (floating point) is written to either of these registers, reciprocal operation is executed with the value used as an operand. Upon completion of operation, the result is written back to that register.

If the register is read from an instruction as a source operand before operation has completed, the thread stalls.

If these registers are read before being written, the result is undefined.

RCP0 (R14) and RCP1 (R15) are both registers to calculate reciprocal in the same manner. Since parallel processing of these registers is possible, the required number of cycles can be hidden by operating them in parallel when two reciprocals are needed.

Rounding mode of the registers is fixed to “Round to the nearest, ties to even”.

These registers can be used as a general purpose register. Refer to Section U32.6.2 for detail.

Note: RCP0 and RCP1 share the same reciprocal arithmetic unit. Therefore, if it is used frequently by multiple threads, contention will occur frequently, and it may not be possible to successfully hide the number of cycles.

#### U32.6.1.12R4, R5, R7, R16 to R31/R47 and 64-bit Pair Register

These are general-purpose and temporary registers (readable and writable) with 32-bit width. Initial value of these registers is undefined.

In addition, some instructions, which are handling 64-bit width data, utilize two consecutive general purpose registers in little endian, with the lower register being at an even index. For example, "MULLS R16, R30, R31" means lower 32-bit of 64-bit result is stored in R16, and upper 32-bit is stored in R17. And when specify an odd register as 64-bit register, result of execution is not guaranteed. Only R16(R17) to R46(R47) can be used for the 64-bit pair registers.

Further, the new small register block load/store instructions to local work memory LDSRB, STSRB, LDSRBI, STSRBI, LDSRBC and STSRBC have the possibility to access the registers R16 to R23 by specifying R16, R24 to R31 by specifying R24, R32 to R39 by specifying R32 and R40 to R47 by specifying R40, while the new large register block load/store instructions to local work memory LDLRB, STLRB, LDLRBI, STLRBI, LDLRBC and STLRBC have the possibility to access the registers R16 to R31 by specifying R16 and the registers R32 to R47 by specifying R32.

#### U32.6.1.13Status Register

This register has bits that are set to 0 or 1 by the results of compare instructions. This register is used to determine whether to branch or conditionally execute instructions, for details refer to section U32.9.1 . This register cannot be read as an operand by instructions.

### U32.6.2 Control Registers of CVengine Thread

The CVengine thread has control registers (CR) to control operation.

The control registers can be changed by the STC instruction or STCI instruction and read by the LDC instruction.

An integer value has to be set to the control registers.

These registers are initialized only upon a software reset. Refer to section U32.12.1.2 .

| | |
|---|---|
| CR0 | When read, the core number (ID) is read. This register cannot be rewritten. |
| CR1 | When read, the thread number (ID) is read. This register cannot be rewritten. |
| CR2 to CR7 | Reserved. Writing these registers is prohibited. |
| CR8 | This register specifies the data plane number to be read via the D$.<br>The initial value is undefined. |
| CR9 | This register specifies the data plane number to be written via the D$.<br>The initial value is undefined. |
| CR10 | This register specifies the output mask for each element of SBO.<br>Bit 2, bit 1 and bit 0 correspond to R0 (X coordinate), R1 (Y coordinate) and D0 (data) respectively. When each bit is set to 1, the corresponding element is not output.<br>For example, when bit 2 = 0, bit 1 = 1, and bit 0 = 1, only R0 (X coordinate) is output from the SBO port. |
| CR11 | Reserved. Writing this register is prohibited. |
| CR12 | This register holds offset number from thread's own TGDMAC configuration register.<br>When this register holds value m, thread #n accesses TGDMAC configuration register for thread #((n+m)%8), for details refer to section U32.8.3 . User can change the value of this register freely but avoiding resource conflict is user responsibility. |
| CR13 to CR15 | Control registers that have not been implemented. Rewriting these registers is prohibited. |
| CR16 | This register specifies the rounding mode of floating-point operations. The initial value is 0.<br>0: Round to the nearest, ties to even<br>1: Round toward zero<br>This setting specifies the behavior of addition or multiplication performed by an instruction and also the behavior of conversion performed by the I2F instruction.<br>When changing value of this register, 4 or more cycles must have passed from the last floating-point operation. Alternatively, insert 4 NOPs to clear pipeline before changing its value. |
| CR17 | This register specifies the rounding mode of conversion between floating-point and integer. The initial value is 1.<br>0: Round to the nearest, ties to even<br>1: Round toward zero<br>This setting specifies the behavior of conversion performed by the F2I instruction.<br>When changing value of this register, 4 or more cycles must have passed from the last F2I instruction. Or insert 4 NOPs to clear pipeline before changing its value. |
| CR18 | This register specifies the rounding mode to be used in casting by the FFRC and FFLR instructions. The initial value is 1.<br>Changing the value from the initial value leads to the instruction not operating in the manner of the operating model. Accordingly, do not change the value unless this is essential.<br>0: Round to the nearest, ties to even<br>1: Round toward zero<br>When changing value of this register, 4 or more cycles must have passed from the last FFRC or FFLR instruction. Or insert 4 NOPs to clear pipeline before changing its value. |

| | |
|---|---|
| CR19 | Writing 1 to bit-n of this register turns Rn to the general purpose register (n=8-15).<br>Reading this register shows current mode of each of R8 to R15. The default mode is SOU for all the registers.<br>0: SOU<br>1: General purpose register<br>When changing each bit of this register, 4 or more cycles must have passed from the last corresponding SOU operation to the bit. Or the result of SOU operation must be used as an operand of an instruction before changing the corresponding bit. (Because pipeline stalls until result of SOU operation becomes ready.) Or insert 4 NOPs to clear pipeline before changing a bit. |
| CR20 | Writing 1 to bit-n of this register turns Rn to the SOU register (n=8-15).<br>This register is write-only. Behavior of reading this register is undefined.<br>When value of this register is changed, corresponding SOU operation must be executed 4 or more cycles after the changing instruction. Or insert 4 NOPs to clear pipeline after changing its value. |

## U32.7 Operation: Floating-Point

### U32.7.1 Floating-Point Data Format

The CVengine thread handles 32-bit floating-point values. The floating-point data format is shown below.

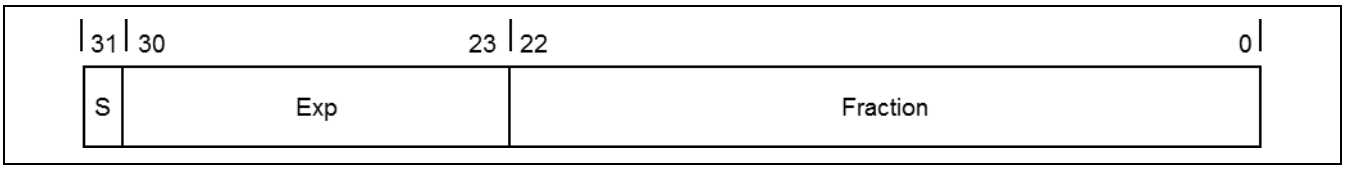



**Figure U32.7.1 32-bit Floating-Point Data Format**

— S (sign): Indicates the sign.
0: positive value or 0/1: negative value or 0

— Exp (exponent): Indicates the N-th power of 2 as a bias.
H'7F is the 0th power of 2, H'80 is the 1st power of 2, H'7E is the –1st power of 2,
H'FF is infinite and H'00 is 0.

Fraction: Indicates the fraction (a total of 24 bits with an implicit one bit added is regarded as the fraction part).

### U32.7.2 IEEE754 Conformance

The CVengine is partially conformant to IEEE754-1985 in floating-point number calculation accuracy.

**Table U32.7.1 IEEE754 Conformance**

| Operation | Bit Accuracy Compliancy to IEEE754, i.e., < 0.5ulp | IEEE754 Compliancy except accuracy, e.g., handling NaN, denormal, exception, etc. |
|---|---|---|
| **Float to Integer (F2I)** | Comply | Comply |
| **Integer to Float (I2F)** | Comply | Comply |
| **Addition, Subtraction (FADD, etc.)** | Comply | Comply |
| **Multiplication (FMUL, etc.)** | Comply | Comply |

| | | |
|---|---|---|
| **Sine (R8)** | maximum 2^(-23) | Comply |
| **Cosine (R9)** | maximum 2^(-23) | Comply |
| **Inverse Square Root (R10)** | 1ulp | Comply |
| **Exponent (R11)** | 1ulp | Comply |
| **Logarithm (R12)** | maximum 2^(-23) | Comply |
| **Square Root (R13)** | 1ulp | Comply |
| **Reciprocal (R14, R15)** | 1ulp | Comply |
| **Comparison (FCMP, etc.)** | N/A | Comply |

## U32.8 Peripheral Functions

### U32.8.1 Streaming Buffer Output (SBO)

Executing SBOCRD instructions leads to the writing of an X coordinate, Y coordinate, and value of a specified register to consecutive areas of the main memory via the streaming buffer output (SBO) port. Masking can be specified individually for the output of the X coordinate, Y coordinate, and specified register value. When masking is specified, only unmasked data is written to consecutive areas. The X and Y coordinates to be output are the X and Y coordinates specified in R0 and R1 with the #IMM1 value added to X and the #IMM2 value added to Y and the data format converted into the integer format. The register value is output as the 32-bit bit string without any change.

The CVengine configuration register SBOBAR0 is used to specify the start address of each SBO. The SBOMMSR0 register is used to specify the maximum size of the sum of data output from each SBO port. When the sum of the output data exceeds this setting, data output is stopped, and an interrupt is generated.

Control register CR10 is used to specify whether to mask the output of the X coordinate, Y coordinate, and specified register value. Setting examples and the output results after the SBOCRD instruction is executed for n times are shown in the figure below. The CVengine has multiple threads, and thus data output via the SBO port are in the same order as that in which the SBOCRD instructions are issued. Note that the output of data through the SBO is not in the order of activation of the threads.

Note:

SBO buffer size is 128 bytes. So SBO transmits data in 128-byte long each time. User must maintain the data size to be transmitted via SBO in multiple of 128-byte.

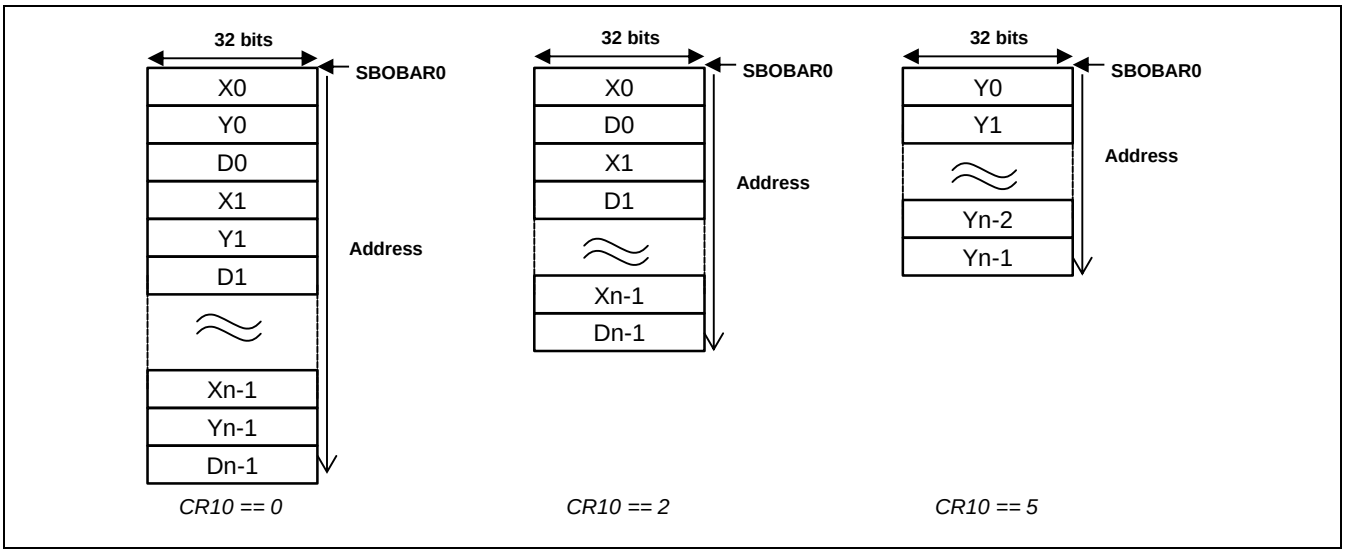



**Figure U32.8.1 Examples of Data Format Output from the SBO Port (XYPCK=0)**

The X coordinate and Y coordinate output can be controlled by the CVengine configuration registers SBOCR0. When the XYPCK bit is set to 1, the lower 16 bits of the X coordinate and Y coordinate which are used when the XYPCK bit is 0 are packed to form 32 bits, then output as the X and Y coordinates. The order of the X coordinate and Y coordinate can be swapped by changing the PCKMODE bit to 0 or 1. When the XYPCK bit is 1 and no mask is specified, the 32 bits output as the X coordinate and the 32 bits output as the Y coordinate are the same value. Setting examples and the output results with those settings are shown in the figure below.

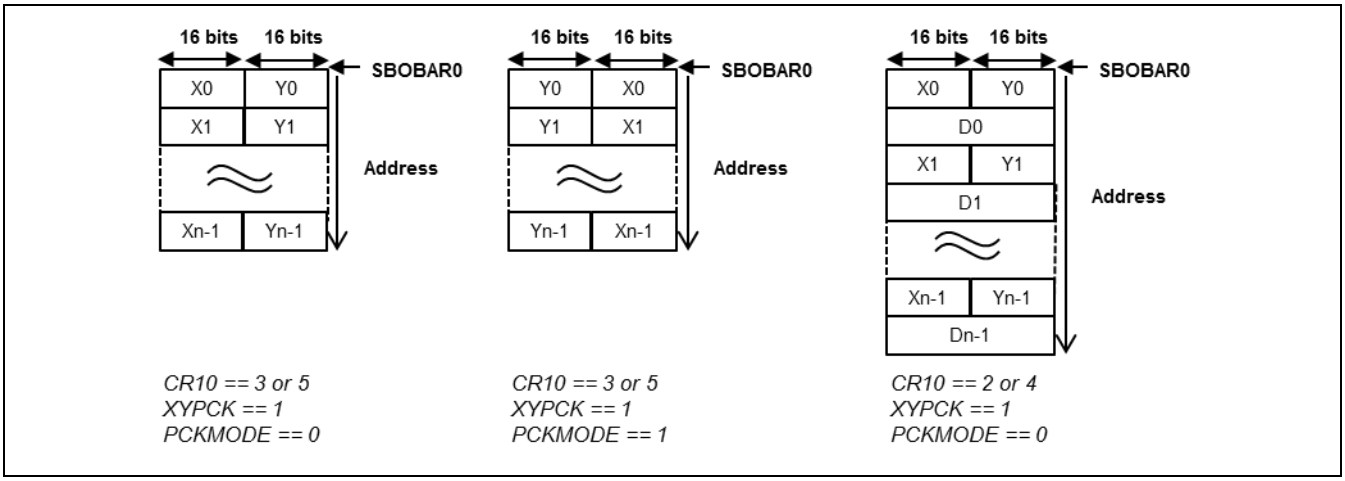



**Figure U32.8.2 Examples of X and Y Coordinate Packs Output from the SBO Port**

### U32.8.2 SOU and SAM

CVengine has some shared logic among threads or cores to consider trade-off between chip size and calculation efficiency. The Shared ArithMetic logic unit (SAM) is implemented for each core, and it is shared by threads belonging to a core. This SAM is further divided in SAM2, SAM4 and SAM8. The SAM8 unit is shared among all 8 threads of a core. The SAM4 unit consists of SAM4E unit and SAM4O unit. The SAM4E is shared among even threads of a core and the SAM4O is shared odd threads. The SAM2 unit consists of SAM2A to SAM2D units. Each SAM2 unit is shared by two threads. The following figure shows how these units are shared by the threads.

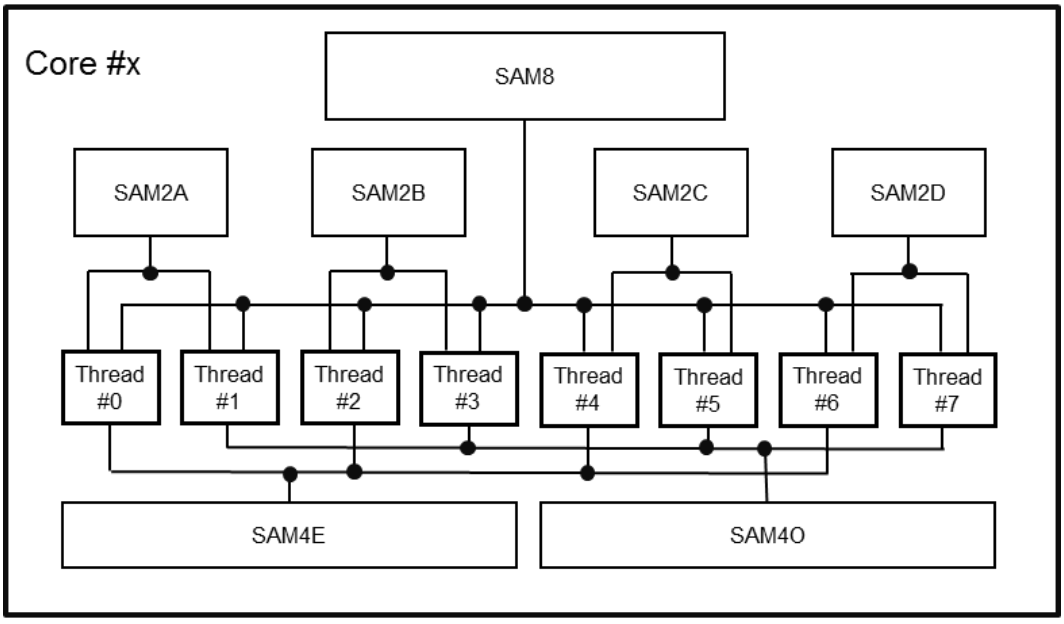



**Figure U32.8.3 SAM Sharing among Threads**

The instructions using the SAM unit are clarified in the Table U32.10.2.

On the other hand, the shared operation unit (SOU) is implemented for each cluster, and it is shared by all threads in the cluster. The SOU is used through special registers, namely EXP2(R11), LOG2(R12), SQRT(R13), RCP(R14 and R15), 1/SQRT(R10), SIN(R8), COS(R9).

For both SAM and SOU case, the sharing is done by hardware automatically. A thread waits for availability of a SAM/SOU operation if the SAM/SOU operation is already used by another thread.

### U32.8.3 DMA

CVengine supports DMA transfer from/to the common memory regions of LWM/GWM to/from the data plane on the external memory by utilizing TGDMAC “Thread Group DMAC” which can be initiated in CVengine program of any master thread. DMA transfer between LWM and GWM is not supported.

The TGDMAC for transfer to/from LWM is named LWM-TGDMAC and it is implemented one in a core. The LWM DMA supports line size units of 32, 64 and 128 bytes to SPMC and 128-byte units to DDR, while on internal side to LWM, 32-byte units are used. The GWM-DMAC is for transfer from/to GWM which is implemented one time in a cluster. It transfers data in 128-byte units to DDR/SPMC and 32-byte units to GWM.

There is one LWM DMA channel per core so there are 4 LWM DMA channels per cluster. Only one DMA channel can transfer data at a time slot.

Each DMA transfer from/to LWM is configured by TGDMAC configuration registers. This bunch of registers is implemented for each thread and mapped on the queue arbiter register area, so a thread can transfer any data with certain configuration. TGDMCR0 specifies transfer direction, data plane and has control bits to start/stop DMA transfer. Offset or X/Y coordinates of transferred data on the data plane are decided by TGDMEXTMEMOFSADDR. The LWM region is specified by TGDMLWMBADDR as a base address and TGDMLWMSTRR as an image stride. TGLNGR is used to indicate lengths for X and Y direction. See chapter U32.12.1 for detail description of each register.

Only master thread can read/write LWM-TGDMAC configuration registers and manage DMA transfer. This means only master thread initiate DMA transfer and issue WAITDMA instruction to wait for completion of transfer. If those DMA operations are done by slave threads, behavior is undefined. When multiple master threads issue DMA transfers, requests are prioritized by an arbiter which has a fair arbitration policy.

Due to the fact that only master threads can initiate a DMA transfer, CVengine has the function that enables master thread to control slave thread’s LWM-TGDMAC. The master thread can also access to the slave threads’ LWM-TGDMAC configuration registers to be able to control up to 8 different DMA transfers. To achieve this, CR12 register and DMAWAITS instruction were added to specify DMA transfer for which slave thread to control. For details, refer to section U32.6.2 and section U32.9 .

To transfer any data from/to GWM, the GWM-DMAC must be used. GWM-DMAC has the same bunch of registers as LWM-TGDMAC, but these registers are implemented in CVengine configuration register area which can be accessed while in processing state by a command list only, while Arm or thread cannot access these registers.

The following is a typical DMA procedure for LWM-TGDMAC and the Figure U32.8.4 is for better understanding of this transfer.

1. Master thread configures DMA through his LWM-TGDMAC configuration registers.
2. Master thread starts DMA transfer by writing 1 to DMACEX bit of TGDMCR0.
3. After starting DMA, the master thread issues WAITDMA instruction, then waits for completion of DMA.

4. After DMA completion, the master thread resumes from WAITDMA, then proceeds next steps like invoking slave threads.

Considering use cases, DMA transfer and thread accessing to LWM should be executed via LWM common area to simplify splitting image area to each thread.

A typical procedure for GWM-DMAC is as follows.

1. Configure GWM-DMAC registers by a command list.
2. A command list starts DMA transfer by writing 1 to DMACEX bit of DMCR0.
3. After starting DMA, the command list issues SYNCS command, then waits for completion of DMA.
4. After DMA completion, the command list resumes from SYNCS, then proceeds next steps like invoking RECT command.

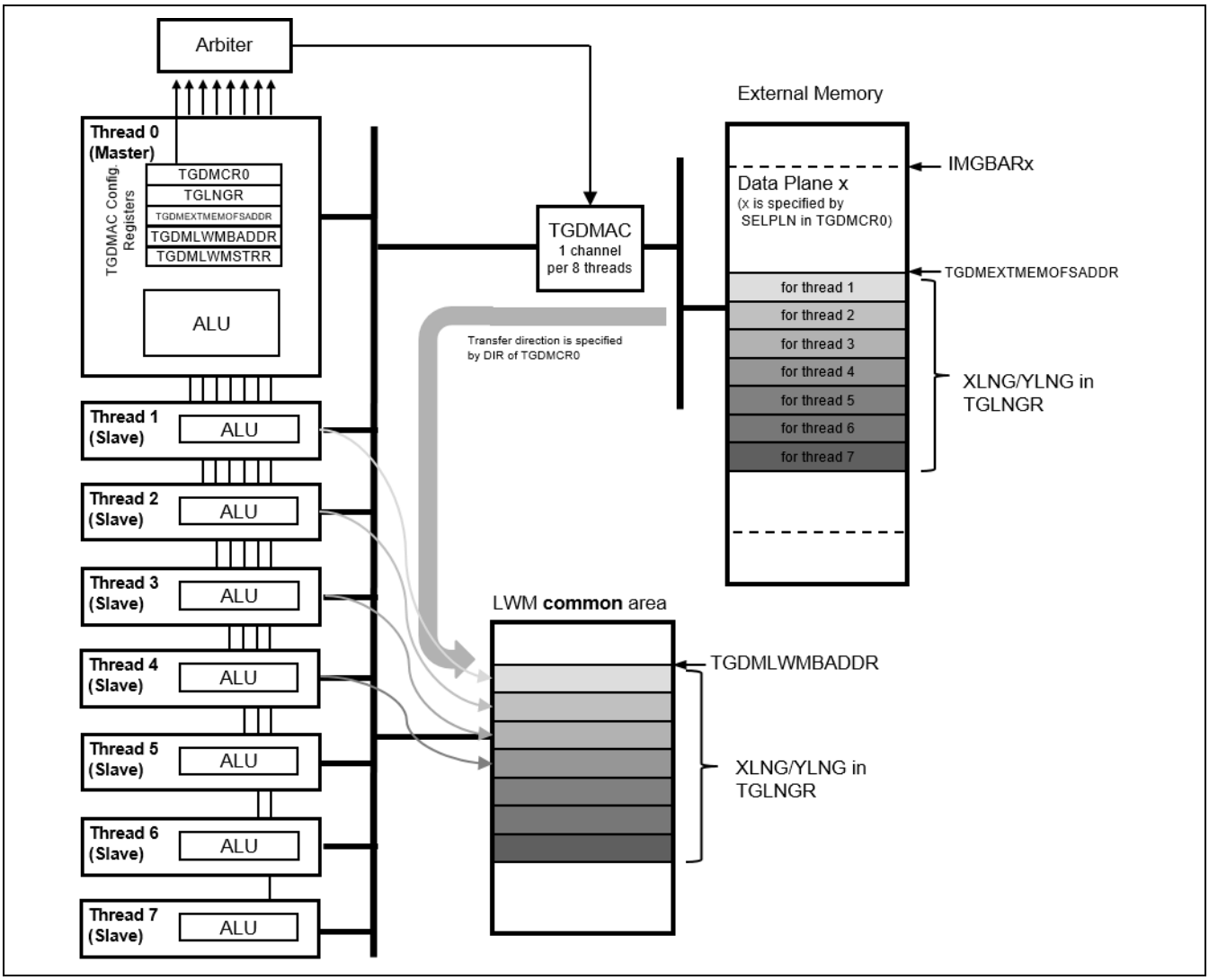



**Figure U32.8.4 Data Transfer by TGDMAC between LWM and External DDR Memory**

### U32.8.4 Debug Support

On the CVengine, a debug resource is available with limitations described below. This debug resource is only available to a dedicated debugger.

1. All clusters have the same debug resources.
2. Only core 0 of each cluster has the debug resource. Other cores do not.
3. Only thread 0 and thread 1 have the debug resource. Other threads do not.
4. The two threads of point 3 are intended for debugging master/slave behavior. Thread 0 can be debugged only as master and thread 1 only as slave.

### U32.8.5 Interrupt

Each thread is able to issue an interrupt to the host processor by the INT instruction. After issuing interrupt, the corresponding bit of status register SR2 configuration register is set to 1. The thread completes execution of the current instruction and updates the thread internal program counters with the address of the next instruction to be executed. If the corresponding bit of interrupt control register ICR2 is unmasked, the bit13 of SR1 is set to 1. Finally, the host processor is interrupted if it is not masked by interrupt mask register IMR1.After the host processor received an interrupt, the host can check which thread is interrupted via two status registers SR1 and SR2. The host can resume the interrupted thread by writing 0 to two status clear register SCR1 and SCR2.

See the Figure U32.8.5 for better understanding of this interrupt routing. And some remarks should be described as follows:

**Remarks:**

- A corresponding bit of SR2 is set to 1 after previous instruction including load/store and SOU calculation is completed.
- If INT instruction and status clearing take place at the same time, the clearing is prioritized.
- Clearing SR1 does not affect SR2, i.e., the interrupt is retained.
- DMA operation is not affected by the INT instruction, as it works independently. Software must take care data synchronization transferred by DMA.
- The thread internal program counters are for debugging purpose only. Refer to section U32.8.4 .

**This interrupt and its handshake with the host processor have detrimental effects on performance and seamless thread processing. Therefore, this INT should be used for debugging purpose only.**

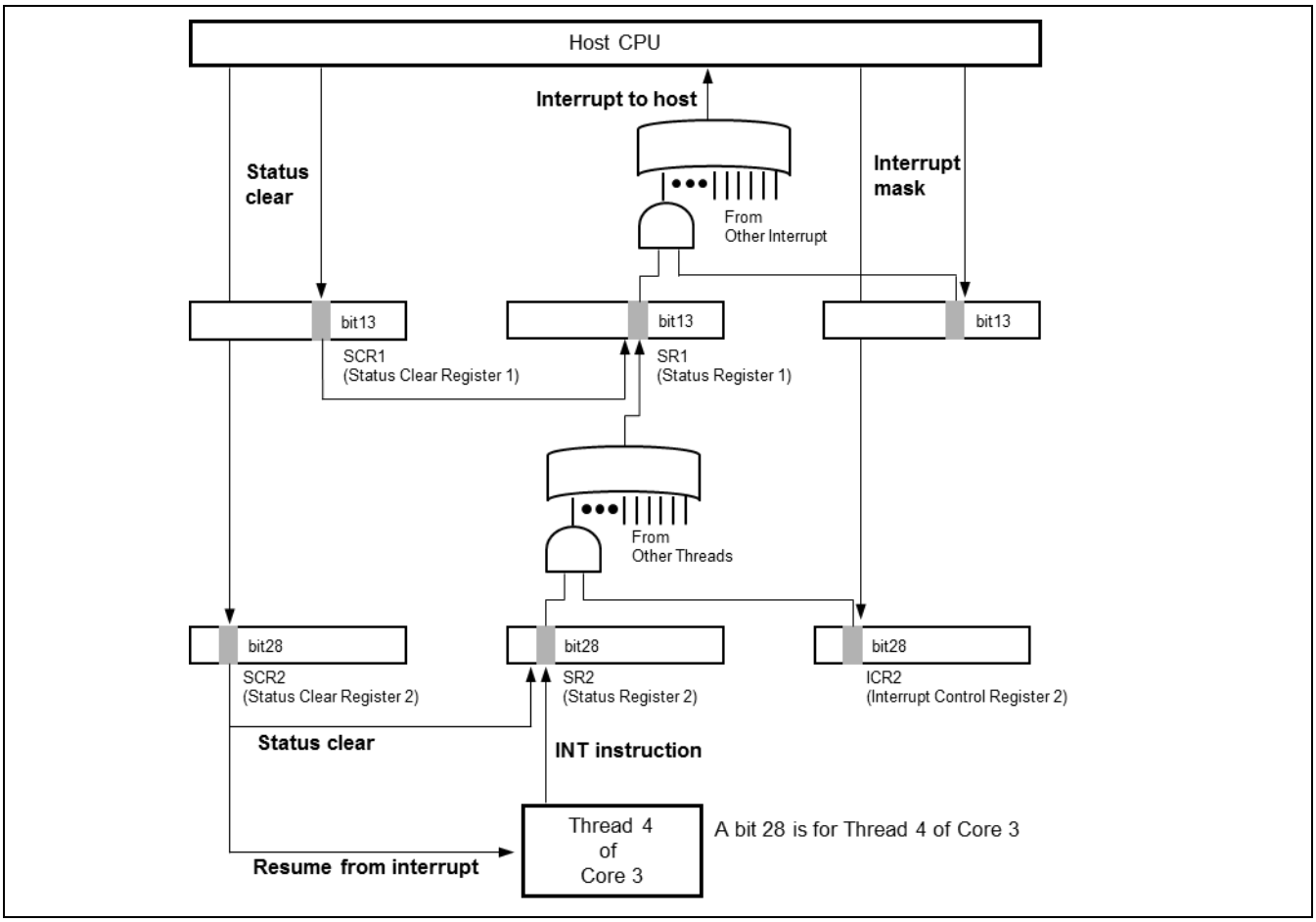



**Figure U32.8.5 Interrupt from Thread to Host**

### U32.8.6 Performance Counter

There are two kinds of hardware counters ready to measure performance.

One kind is a hardware counter is available in a cluster to measure performance of the cluster. It counts the clock cycles while CVengine is active (while CL is running). This counter can be enabled or disabled by the settings of configuration registers, SPCR and SPBSYCNTR. This counter does not take care for overflow. Refer to sections U32.12.1.19 and U32.12.1.20 for the details of the configuration registers for this counter.

The other kind is to measure performance of each thread. This counter can count stalled cycles of a thread. To use this counter, corresponding configuration registers must be set before invocation of CVengine. The configuration registers TGCR0, USPFCCTLR and PFCTOTALR0 are available for this counter. Refer to sections U32.12.1.14 , U32.12.1.50 and U32.12.1.51 . By checking the overflow bit, this counter can detect whether overflow happened or not. This counter can count only one thread in a cluster per execution.

**Note:**

Counter values are just for your information. These counters could be useful to improve software performance in some cases, but any relationship between counter values and actual processing time is not guaranteed. Also, any detailed condition of counting will not be opened.

### U32.8.7 Image Clipping

CVengine supports the image clipping feature. This feature allows user to clip the entire image into some necessary parts of the image to process. Refer to Figure U32.8.6.

To clip the image, CLPMINR and CLPMAXR configuration registers of the CVengine are used. Refer to section U32.12.1.17 and section U32.12.1.18 .

To enable this image clipping feature, set 1 to the clip bit in POINT/POINTS/RECT command. Refer to section U32.13.1 , section U32.13.2 and section U32.13.3 .

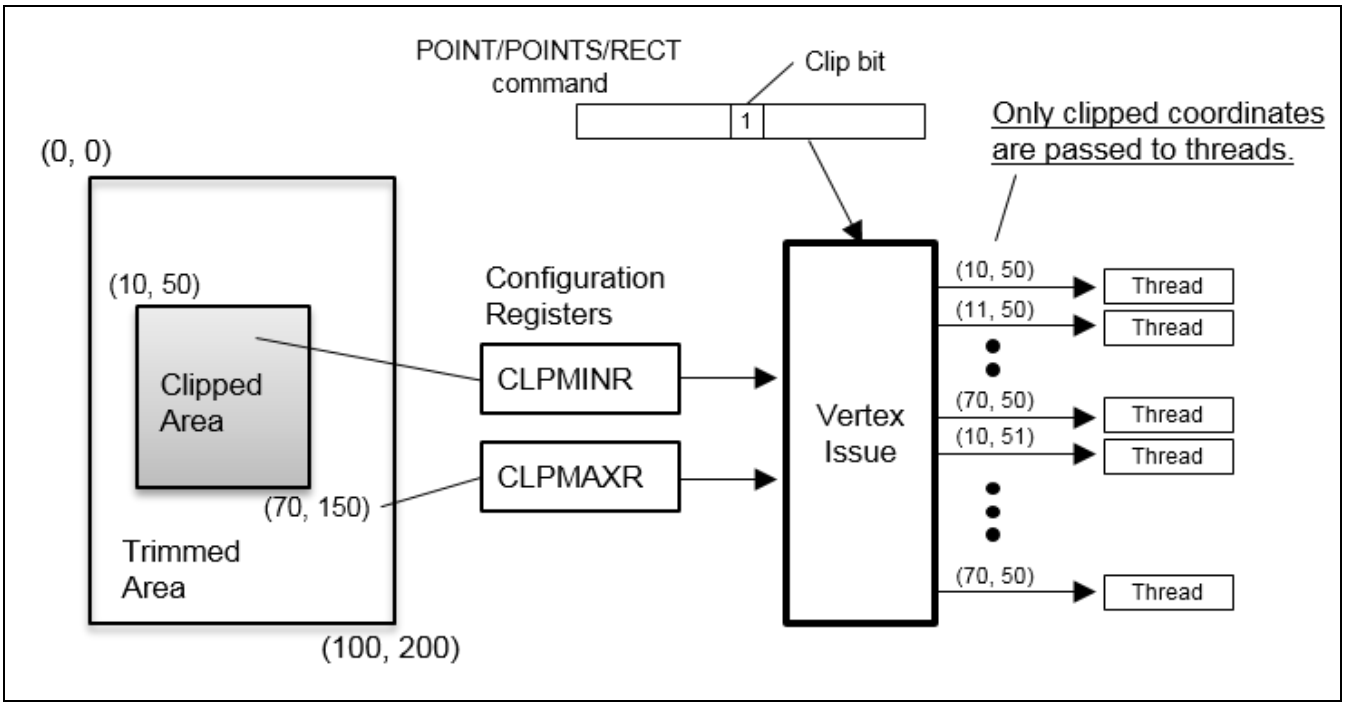



**Figure U32.8.6 Image Clipping Feature**

### U32.8.8 Guard Timer

The guard timer function is to check if CVengine processing time exceeds expected time. The guard timer is an additional function of the performance counter. Refer to section U32.8.6 .

To show the processing time has exceeded the threshold, this function can set two types of flags, SR1.PBCOVF and SR1.INT. SR1.PBCOVF flag is automatically set at the moment when SPBSYCNTR exceeds SPBCTHRR while threads are running. To utilize SR1.INT flag, an INT command with INT_PBCOVF bit enabled is needed after a CVengine operation command (e.g., RECT) in the Command List. When the INT command in Command List is executed, and if INT.INT_PBCOVF and SR1.PBCOVF are both 1, SR1.INT flag is set so that the execution of command list is halted, and the host CPU can recognize an overflow interrupt has happened. Thus, the host CPU can take some action for the processing time overflow like rewriting commands in the Command List. When the INT command in Command List is executed, and if INT.INT_PBCOVF is set to 1 but SR1.PBCOVF is 0, so no overflow happened, then instead of the INT a NOP is executed.

When SCR1.PBCOVFCLR is set to 1, SR1.PBCOVF and SPBSYCNTR are both cleared. Figure U32.8.7shows flowchart of this function. Also refer to section U32.12.1.5 , U32.12.1.6 , U32.12.1.21 and U32.13.7 .

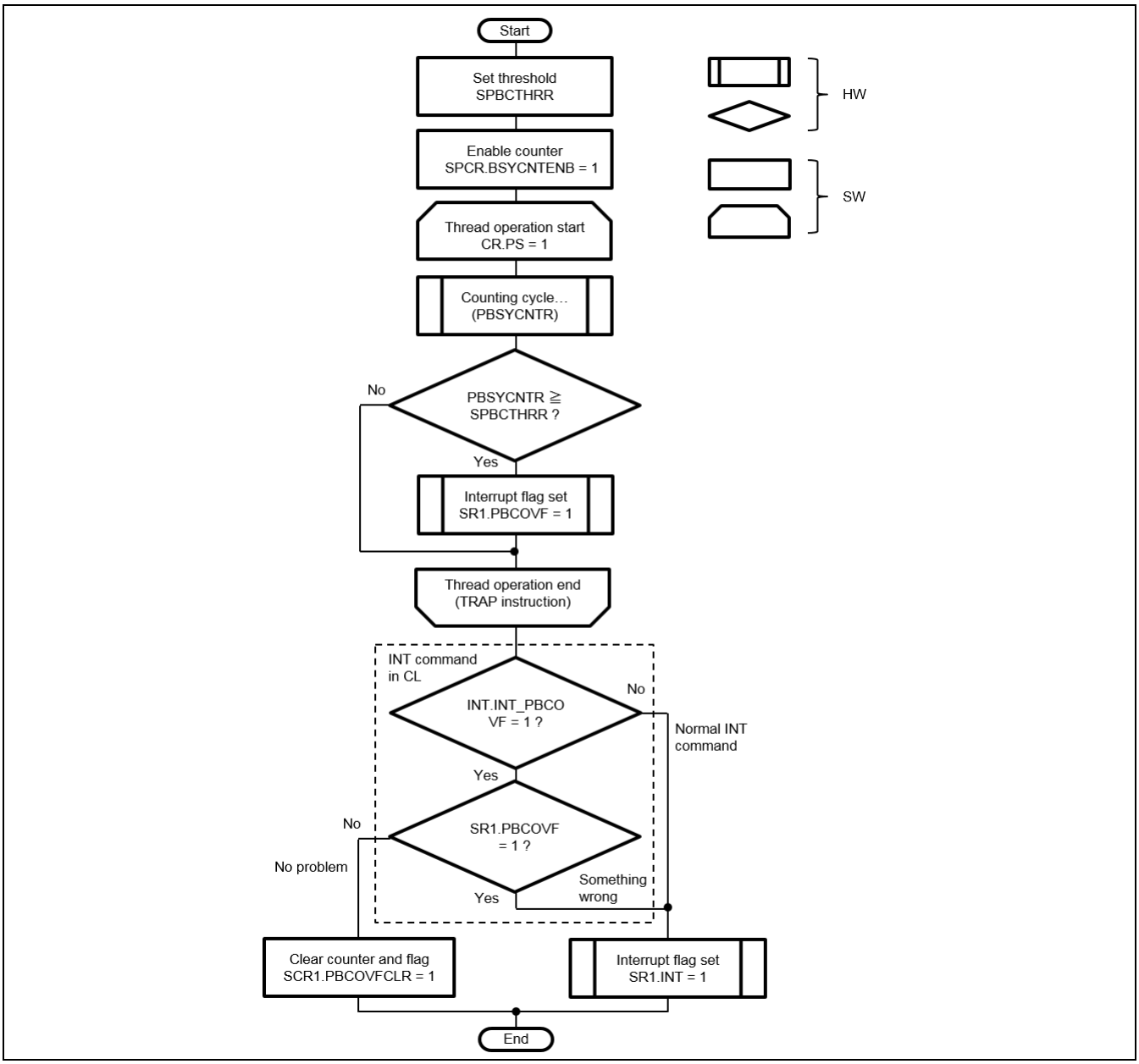



**Figure U32.8.7 Guard Timer Flow Chart**

Figure U32.8.8 shows a use case of the Guard Timer. In this case IMR1.PBCOVFMSK is set to 1 so the interrupt during RECT command, SR1.PBCOVF is masked. When counter value exceeds threshold value at the end of the 1st RECT command, the INT command in CL signals the CPU that a processing time overflow happened. So, the host CPU alters the threshold value set by WPR command in CL to the new value.

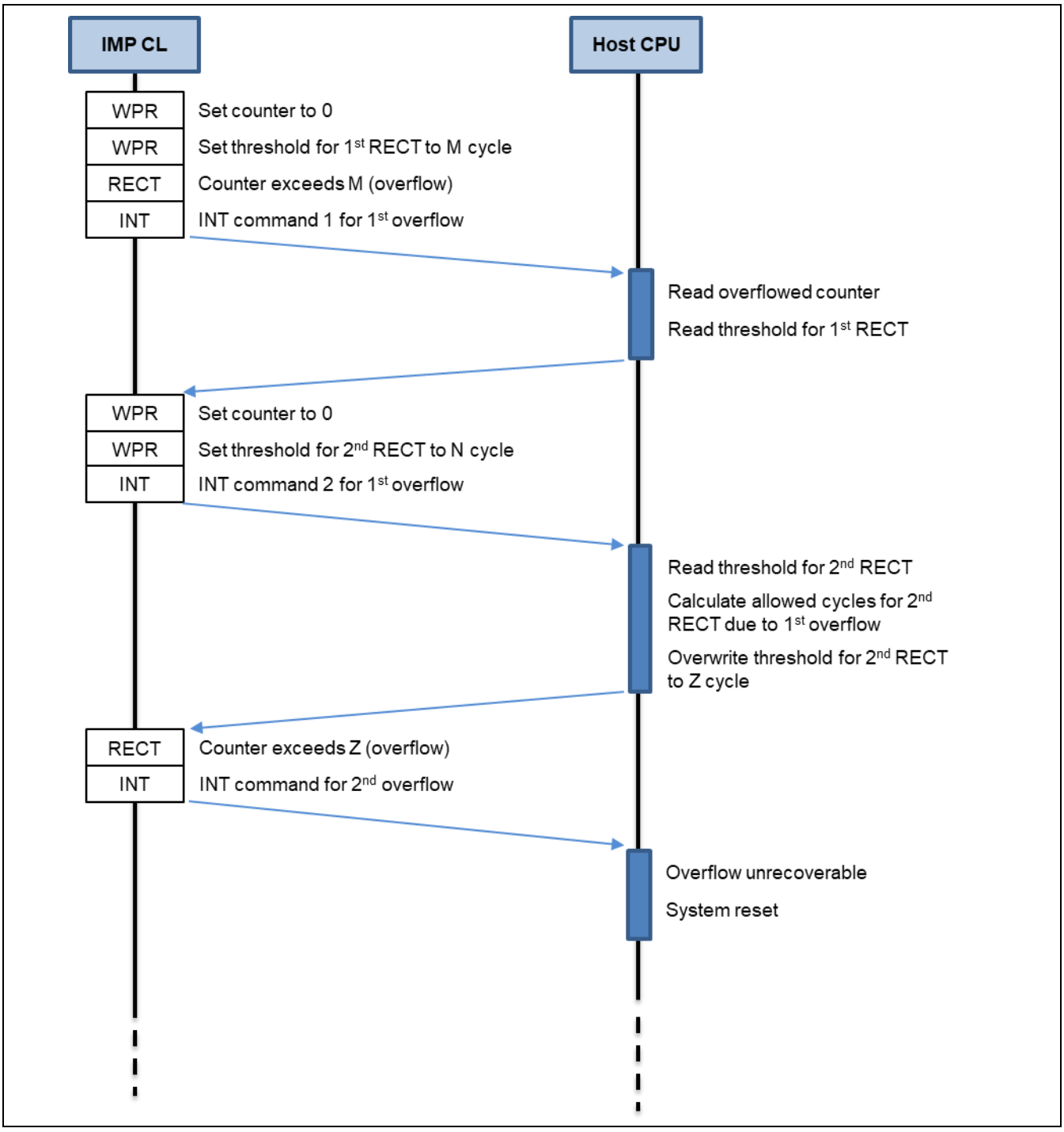



**Figure U32.8.8 Guard Timer Use Case**

### U32.8.9 Data Cache Bypassing

Data cache bypassing area can be set with configuration registers DCBYPA0MIN and DCBYPA0MAX. The default values of the registers are 0xFFFFFFFF and 0x00000000 respectively, so no bypassing area is set. Only 4-byte access is supported for D$ bypassing. This feature can be enabled and disabled for each thread with DCBYPEN register. Refer to sections U32.12.1.22 , U32.12.1.23 , and U32.12.1.24 .

Additionally, store data replacement feature is implemented. When both DCBYPEN and DCBYPCTRL are enabled for a thread, upper 16 bits of writing data with bypassing D$ by the thread are replaced with Core# (bits 24 and 25) and Thread# (bits 16 till 18), which can be utilized for debugging purpose.

The D$ bypassing function is available for reading and writing data of all threads in a cluster, but it is recommended to use only thread 0 of each core, total 4 threads in a cluster to prevent large performance degradation. Only Arm and Command List can change the values of the configuration registers for this function. When changing the register values from CL, insert SYNCM just before it.

Through the D$ bypassing, registers for IMP Core, IMP DMAC, PSC, CNN and CoreSight can be accessed from a thread. Also registers for CVengine clusters can be accessed, except for the cluster to which the thread belongs. Scratchpad cannot be accessed via D$ bypassing.

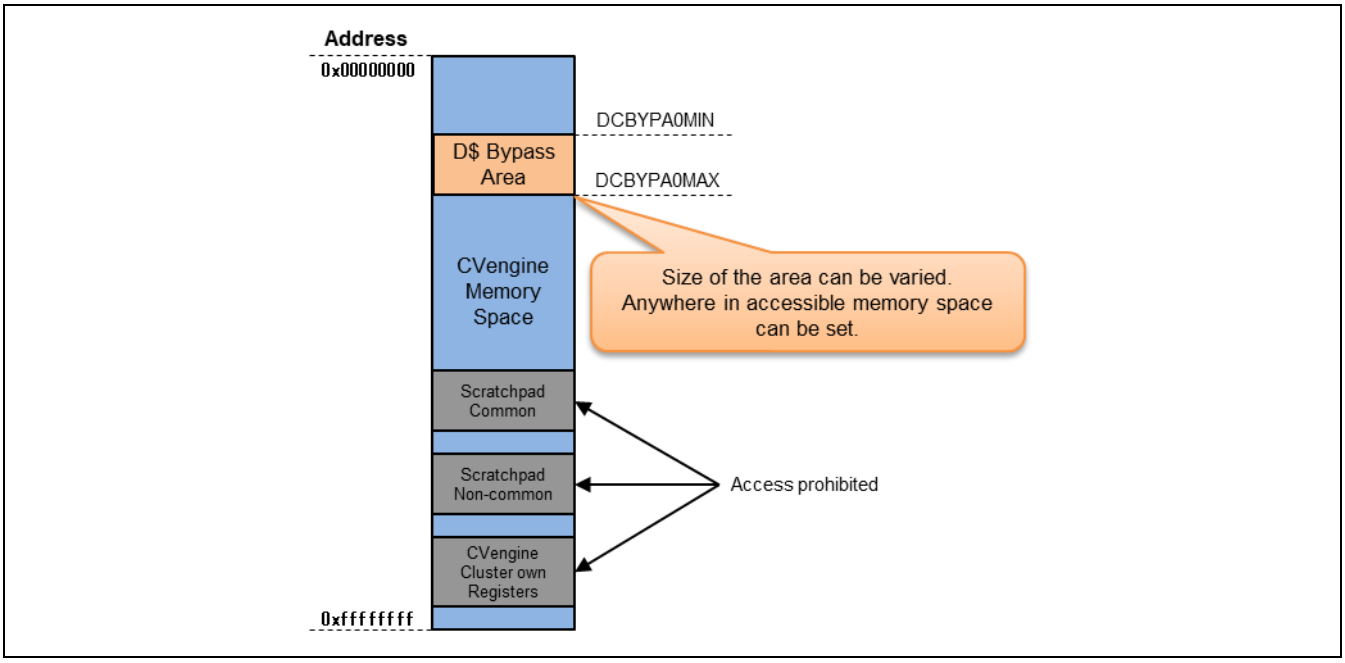



**Figure U32.8.9 Data Cache Bypassing Area**

## U32.9 Instruction Set

The CVengine thread instructions are classified into the following three types of instructions.

(1) Flow control instructions
- Branch instructions
- Compare instructions to generate the branch conditions
- Instructions to control the stop of program execution

(2) Data transfer instructions

(3) Operation instructions

All of the instructions can be executed conditionally with the condition of the status register value being 0 or 1.

**Note 1:**

Division instructions are not implemented in the CVengine threads of the CVengine.

**Note 2:**
#ASMIMM is the immediate description for the CVengine Assembler. This #ASMIMM is interpreted by the assembler and is translated to #IMM which is the immediate used in an actual instruction code defined in Section U32.10.1 .
The translation from #ASMIMM to #IMM is defined instruction-by-instruction in the following description.

**Note 3:**

For 8-bit and 16-bit group instructions such as GADDxx, GMADxx, GSADxx or GSUBxx, s0 and s1 are treated as 64-bit pair registers. Refer to Section U32.6.1.12 . A pair register contains 4 16-bit data or 8 8-bit data as a data array. For 16-bit group instructions, S0[0] means S0(n)[15:0] and S0[3] means S0(n+1)[31:16]. For 8-bit group instructions, S0[0] means S0(n)[7:0] and S0[7] means S0(n+1)[31:24].

**Note 4:**

R6 register can be used either as offset base for Uniform storage immediate value indexing, or as a general purpose register. Refer to section U32.10.1 about the R6 bit of instruction format. To use R6 as the offset base with the CVengine Assembler, add '$' mark to the 3rd operand for the RU type instructions. See the examples below.

ADDU R16, R17, 1 → R16 = R17 + Uniform[1];
ADDU R16, R17, $1 → R16 = R17 + Uniform[R6+1];

#R6IMM is used at behavior model in the description of the RU type instructions. If R6 is used as the offset base, the value of #R6IMM is R6 + #IMM. If R6 is used as a general purpose register, the value of #R6IMM is equal to #IMM.

#IMM: 7-bit signed integer
R6: 8-bit signed integer

If the value of R6 + #IMM exceeds the area of uniform storage, result is undefined.

### U32.9.1 Conditional instruction execution

For all instructions, it is possible to control the execution/non-execution of the instruction by referencing the status register SR. By writing "SR" at the end of an operand, the instruction is executed if the status register is true, and the instruction is regarded as the NOP instruction if the status register is false. By writing "!SR" at the end of an operand, the instruction is executed if the status register is false, and the instruction is regarded as the NOP instruction if the status register is true.

**Examples:**

```
CMP.EQ   R3,   R16,   R17
ADD      R20,  R21,   R22,   SR      # When R16 == R17, execute ADD.
SUB      R20,  R21,   R22,   !SR     # When R16 != R17, execute SUB.
```

| **ABS** | **ABSolute** |
|---|---|
| Usage | ABS D0, S0 |
| Function | Stores the absolute value of S0 in D0.<br>The value of S0 is handled as an integer in the operation. |
| Behavior model | D0 = \|(int)S0\|; |

| **ABS8** | **ABSolute 8-bit values** |
|---|---|
| Usage | ABS8 D0, S0 |
| Function | This is an instruction to get the absolute value from 8 signed 8-bit input values from source S0. |
| Behavior model | D0[n] = ABS(S0[n]), n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer<br>S0: 8 times 8-bit signed integer |

| **ABS16** | **ABSolute 16-bit values** |
|---|---|
| Usage | ABS16 D0, S0 |
| Function | This is an instruction to get the absolute value from 4 signed 16-bit input values from source S0. |
| Behavior model | D0[n] = ABS(S0[n]), n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 16-bit signed integer |

| **ACTST** | **ACTivate Shader Thread** |
|---|---|
| Usage | ACTST S0 |
| Function | Master thread activates slave thread. Which slave is going to be activated is determined by the queue arbiter. The values of R0 and R1 are sent as the X and Y coordinates. In addition, the register value specified by S0 is sent to R2 of the slave thread.<br>This instruction must not be executed by the slave thread and must not be executed by threads started by the SINI CL command<br>. |
| Behavior model | - |
| Operand type | S0: -(32-bit array) |

| **ADD** | **ADD** |
|---|---|
| Usage | ADD D0, S0, S1 |
| Function | Adds S0 and S1, then stores the result in D0.<br>S0 and S1 are handled as integers in the operation. |
| Behavior model | D0 = (int)S0 + (int)S1; |

| **ADDS** | **ADD with Saturation** |
|---|---|
| Usage | ADDS D0, S0, S1 |
| Function | This is an instruction to add 2 signed 32-bit values from source S0 and source S1 with saturation. Mapped on Thread. |
| Behavior model | D0 = SAT(S0 + S1, 32); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer<br>S1: 32-bit signed integer |

| **ADDSS8** | **ADD Signed with Saturation 8-bit values** |
|---|---|
| Usage | ADDSS8 D0, S0, S1 |
| Function | This is an instruction to add 8 times 2 signed 8-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] + S1[n], 8), n = 0...7; |
| Operand type | D0: 8 times 8-bit signed integer<br>S0: 8 times 8-bit signed integer<br>S1: 8 times 8-bit signed integer |

| **ADDSS16** | **ADD Signed with Saturation 16-bit values** |
|---|---|
| Usage | ADDSS16 D0, S0, S1 |
| Function | This is an instruction to add 4 times 2 signed 16-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] + S1[n], 16), n = 0...3; |
| Operand type | D0: 4 times 16-bit signed integer<br>S0: 4 times 16-bit signed integer<br>S1: 4 times 16-bit signed integer |

| **ADDSU** | **ADD with Saturation signed Uniform value** |
|---|---|
| Usage | ADDSU D0, S0, #IMM |
| Function | This is an instruction to add 2 signed 32-bit values from source S0 and Uniform value indexed by #IMM with saturation. Mapped on Thread. |
| Behavior model | D0 = SAT(S0 + Uniform[#R6IMM], 32); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer |

| **ADDSI** | **ADD with Saturation signed Immediate value** |
|---|---|
| Usage | ADDSI D0, S0, #IMM |
| Function | This is an instruction to add a signed 32-bit values from source S0 and the immediate 8-bit value #IMM with saturation. Mapped on Thread. |
| Behavior model | D0 = SAT(S0 + #IMM, 32); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer<br>#IMM: 8-bit signed integer |

| **ADDDBL** | **ADD DouBL** |
|---|---|
| Usage | ADDDBL D0, S0, S1 |
| Function | This instruction treats S0 as two 32-bit integers, upper and lower, and S1 as two 15bit integers, upper and lower. And adds those integers respectively. Then stores the result to D0.<br>Typical use case of this instruction is to increment two pointers in a loop with one instruction.<br>D0 and S0 are 64-bit pair registers, always the lower even register is specified, which holds the lower 32-bit. S1 is a 32-bit register. |
| Behavior model | D0[63:32] = {S0[63:48], (16-bit unsign extended(S0[46:32])+16-bit unsign extended(S1[30:16]))}<br>D0[31:0] = {S0[31:16], (16-bit unsign extended(S0[14:0])+16-bit unsign extended(S1[14:0]))} |
| Operand type | D0: Concatenated two 32-bit unsigned integers<br>S0: Concatenated two 32-bit unsigned integers.<br>S1: Concatenated two 15-bit unsigned integers (16-bit aligned) |

| **ADDDBLU** | **ADD DouBL with Uniform** |
|---|---|
| Usage | ADDDBLU D0, S0, #IMM |
| Function | This instruction treats S0 as two 32-bit integers, upper and lower, and the uniform value indexed by #IMM as two 15bit integers, upper and lower. And adds those integers respectively. Then stores the result to D0.<br>Typical use case of this instruction is to increment two pointers in a loop with one instruction.<br>D0 and S0 are 64-bit pair registers, always the lower even register is specified, which holds the lower 32-bit. S1 is a 32-bit register. |
| Behavior model | D0[63:32] = {S0[63:48], (16-bit unsign extended(S0[46:32]) + 16-bit unsign extended(Uniform[#R6IMM][30:16]))};<br>D0[31:0] = {S0[31:16], (16-bit unsign extended(S0[14:0]) + 16-bit unsign extended(Uniform[#R6IMM][14:0]))}; |
| Operand type | D0: Concatenated two 32-bit unsigned integers<br>S0: Concatenated two 32-bit unsigned integers<br>Uniform: Concatenated two 15-bit unsigned integers (16-bit aligned) |

| **ADDI** | **ADD to Immediate value** |
|---|---|
| Usage | ADDI D0, S0, #IMM |
| Function | Adds S0 and #IMM, then stores the result in D0.<br>S0 and #IMM are handled as integers in the operation. |
| Behavior model | D0 = (int)S0 + (char)#IMM; |

| **ADDU** | **ADD to Uniform** |
|---|---|
| Usage | ADDU D0, S0, #IMM |
| Function | Adds S0 and the uniform value indexed by #IMM, then stores the result in D0.<br>S0 is handled as an integer in the operation. |
| Behavior model | D0 = S0 + (int)Uniform[#R6IMM]; |

| **ADDUS8** | **ADD Unsigned with Saturation 8-bit values** |
|---|---|
| Usage | ADDUS8 D0, S0, S1 |
| Function | This is an instruction to add 8 times 2 unsigned 8-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] + S1[n], 8), n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer<br>S0: 8 times 8-bit unsigned integer<br>S1: 8 times 8-bit unsigned integer |

| **ADDUS16** | **ADD Unsigned with Saturation 16-bit values** |
|---|---|
| Usage | ADDUS16 D0, S0, S1 |
| Function | This is an instruction to add 4 times 2 unsigned 16-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] + S1[n], 16), n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 16-bit unsigned integer<br>S1: 4 times 16-bit unsigned integer |

| **AND** | **AND** |
|---|---|
| Usage | AND D0, S0, S1 |
| Function | Performs the logical AND operation between S0 and S1, then stores the result in D0. |
| Behavior model | D0 = S0 & S1; |

| **AND64** | **AND 64-bit values** |
|---|---|
| Usage | AND64 D0, S0, S1 |
| Function | This is an instruction to AND 2 64-bit (8 times 8-bit or 4 times 16-bit) values. |
| Behavior model | D0[n] = S0[n] AND S1[n], n = 0...63; |
| Operand type | D0: 64-bit integer<br>S0: 64-bit integer<br>S1: 64-bit integer |

| **ANDU** | **AND with Uniform** |
|---|---|
| Usage | ANDU D0, S0, #IMM |
| Function | Performs the logical AND operation between S0 and the uniform value indexed by #IMM, then stores the result in D0. |
| Behavior model | D0 = S0 & Uniform[#R6IMM]; |

| **ASM** | **Add and Sub, then each Multiplied** |
|---|---|
| Usage | ASM D0, S0, S1 |
| Function | This instruction is assumed to be used in Haar Wavelet or FFT butterfly. D0, S0, S1 has 64-bit width which is concatenated by two 32-bit signed integers. Calculate summation and subtraction between higher 32-bit and lower 32-bit of S0. Then each summation and subtraction are multiplied by higher 32-bit and lower 32-bit of S1 respectively. Finally, these two values are concatenated into 64-bit width D0.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0[63:32] = (S0[63:32] + S0[31:0]) * S1[63:32]; // Each 32-bit is “signed”.<br>D0[31:0] = (S0[63:32] - S0[31:0]) * S1[31:0]; // Each 32-bit is “signed”. |
| Operand type | D0: Concatenated two 32-bit signed integers<br>S0: Concatenated two 32-bit signed integers<br>S1: Concatenated two 32-bit signed integers |

| **ASMU** | **Add and Sub, then each Multiplied by Uniform** |
|---|---|
| Usage | ASMU D0, S0, #IMM |
| Function | This instruction is assumed to be used in Haar Wavelet or FFT butterfly. D0, S0 has 64-bit width which is concatenated by two 32-bit signed integers. Calculate summation and subtraction between higher 32-bit and lower 32-bit of S0. Then each summation and subtraction are multiplied by a constant in uniform value indexed by #IMM. Finally, there two values are concatenated into 64-bit width D0.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0[63:32] = (S0[63:32] + S0[31:0]) * Uniform[#R6IMM];<br>D0[31:0] = (S0[63:32] - S0[31:0]) * Uniform[#R6IMM]; |
| Operand type | D0: Concatenated two 32-bit signed integers<br>S0: Concatenated two 32-bit signed integers<br>Uniform: 32-bit signed integer |

| **BRA** | **BRAnch** |
|---|---|
| Usage | BRA #LABEL |
| Function | Branches to #LABEL. |
| Behavior model | PC=#LABEL; |
| Operand type | #LABEL: 16-bit unsigned integer |

| **BRAR** | **BRAnch (Register)** |
|---|---|
| Usage | BRAR S0 |
| Function | Branches to the register value specified by S0.<br>The 16 lower-order bits of S0 are handled as an unsigned integer. |
| Behavior model | PC= (ushort)(S0&0xffff); |
| Operand type | S0: 16-bit unsigned integer |

| **BRCID** | **BRanch based on Core ID** |
|---|---|
| Usage | BRCID #IMM |
| Function | Jump to new PC which is defined as the sum of by 1 incremented old PC address and the product of the own core ID times the specified 8-bit signed immediate value. The address is a word address. |
| Behavior model | PC=PC+(core ID * #IMM)+1; |

| **BRTID** | **BRanch based on Thread ID** |
|---|---|
| Usage | BRTID #IMM |
| Function | Jump to new PC which is defined as the sum of by 1 incremented old PC address and the product of the own thread ID times the specified 8-bit signed immediate value. The address is a word address. |
| Behavior model | PC=PC+(thread ID * #IMM)+1; |

| **BTST** | **Bit TeST** |
|---|---|
| Usage | BTST D0, S0, S1 |
| Function | This instruction copies the S1th bit from S0 to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | D0 = (S0 >> S1) & 1 |
| Operand type | D0: 1bit integer<br>S0: 32-bit integer<br>S1: 5bit unsigned integer |

| **BTSTU** | **Bit TeST with S0th bit from UNI[#IMM]** |
|---|---|
| Usage | BTSTU D0, S0, #IMM |
| Function | This instruction copies the S0th bit from UNI[#IMM] to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | D0 = (UNI[#IMM] >> S0) & 1 |
| Operand type | D0: 1bit integer<br>S0: 5bit unsigned integer<br>#IMM: 6bit unsigned integer |

| **BTSTI** | **Bit TeST with Immediate bit selection** |
|---|---|
| Usage | BTSTI D0, S0, #IMM |
| Function | This instruction copies the #IMMth bit from S0 to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | D0 = (S0 >> #IMM) & 1 |
| Operand type | D0: 1bit integer<br>S0: 32-bit integer<br>#IMM: 5bit unsigned integer |

| **CALL** | **sub routine CALL** |
|---|---|
| Usage | CALL D0, #LABEL |
| Function | PC+1 is saved to D0, and branch to #LABEL. |
| Behavior model | D0 = (PC+1); PC=#LABEL; |

| **CALLA** | **sub routine CALL Alternative** |
|---|---|
| Usage | CALLA D0, #LABEL |
| Function | Saves PC+1 to the temporary-storage PC register specified by D0 and branches to #LABEL. Although this temporary-storage PC register is not described in the list of registers, but it is implicitly implemented.<br>The register to be specified by D0 is not the general-purpose register number, but the CALLA/RETA dedicated temporary-storage PC register number. There are eight registers for this purpose and 0 to 7 can be specified.<br>The 3 lower-order bits of D0 are handled as an unsigned integer.<br>For example, to save the return address to the 3rd PC register, let D0 to have the value 3 before CALLA instruction. |
| Behavior model | PCR[(D0&0x7)]=PC+1; PC=#LABEL; |
| Operand type | D0: 3-bit unsigned integer<br>#LABEL: 16-bit unsigned integer |

| **CALLR** | **sub routine CALL(Register)** |
|---|---|
| Usage | CALLR D0, S0 |
| Function | PC+1 is saved to D0, and branch to the PC specified by S0. |
| Behavior model | D0 = (PC+1); PC=S0; |

| **CMP.EQ** | **EQual** |
|---|---|
| Usage | CMP.EQ D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is equal to S1, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is equal to S1, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands S0 and S1 are handled as 32-bit signed integers in the comparison. |
| Behavior model | SR.F = (S0 == S1 ); (D0 == R3)<br>D0 = (S0 == S1 ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>S1: 32-bit signed integer |

| **CMP.EQ8** | **CoMPare EQual 8-bit values** |
|---|---|
| Usage | CMP.EQ8 D0, S0, S1 |
| Function | This is an instruction to compare (equal) 8 8-bit values with 8 8-bit value. The result of the nth compare is stored in the nth byte of D0. |
| Behavior model | D0[n] = S0[n] == S1[n], n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer, only lowest bit in each byte used<br>S0: 8 times 8-bit integer<br>S1: 8 times 8-bit integer |

| **CMP.EQ16** | **CoMPare EQual 16-bit values** |
|---|---|
| Usage | CMP.EQ16 D0, S0, S1 |
| Function | This is an instruction to compare (equal) 4 16-bit values with 4 16-bit value. The result of the nth compare is stored in the nth 16-bit word of D0. |
| Behavior model | D0[n] = S0[n] == S1[n], n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer, only lowest bit in each 16-bit word used<br>S0: 4 times 16-bit integer<br>S1: 4 times 16-bit integer |

| **CMP.NEQ** | **Not EQual** |
|---|---|
| Usage | CMP.NEQ D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is not equal to S1, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is not equal to S1, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands S0 and S1 are handled as 32-bit signed integers in the comparison. |
| Behavior model | SR.F = (S0 != S1 ); (D0 == R3)<br>D0 = (S0 != S1 ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>S1: 32-bit signed integer |

| **CMP.NEQ8** | **CoMPare Not EQual 8-bit values** |
|---|---|
| Usage | CMP.NEQ8 D0, S0, S1 |
| Function | This is an instruction to compare (not equal) 8 8-bit values with 8 8-bit value. The result of the nth compare is stored in the nth byte of D0. |
| Behavior model | D0[n] = S0[n] != S1[n] , n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer, only lowest bit in each byte used<br>S0: 8 times 8-bit integer<br>S1: 8 times 8-bit integer |

| **CMP.NEQ16** | **CoMPare Not EQual 16-bit values** |
|---|---|
| Usage | CMP.NEQ16 D0, S0, S1 |
| Function | This is an instruction to compare (not equal) 4 16-bit values with 4 16-bit value. The result of the nth compare is stored in the nth 16-bit word of D0. |
| Behavior model | D0[n] = S0[n] != S1[n], n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer, only lowest bit in each 16-bit word used<br>S0: 4 times 16-bit integer<br>S1: 4 times 16-bit integer |

| **CMP.SGE** | **Greater than or Equal** |
|---|---|
| Usage | CMP.SGE D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is greater than or equal to S1, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is greater than or equal to S1, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands S0 and S1 are handled as 32-bit signed integers in the comparison. |
| Behavior model | SR.F = (S0 >= S1 ); (D0 == R3)<br>D0 = (S0 >= S1 ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>S1: 32-bit signed integer |

| **CMP.SGT** | **Greater Than** |
|---|---|
| Usage | CMP.SGT D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is greater than S1, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is greater than S1, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands S0 and S1 are handled as 32-bit signed integers in the comparison. |
| Behavior model | SR.F = (S0 > S1 ); (D0 == R3)<br>D0 = (S0 > S1 ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>S1: 32-bit signed integer |

| **CMP.SLE** | **Less than or Equal** |
|---|---|
| Usage | CMP.SLE D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is smaller than or equal to S1, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is smaller than or equal to S1, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands are handled as 32-bit signed integers in the comparison. |
| Behavior model | SR.F = (S0 <= S1 ); (D0 == R3)<br>D0 = (S0 <= S1 ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>S1: 32-bit signed integer |

| **CMP.SLE8** | **CoMPare Signed Less than or Equal 8-bit values** |
|---|---|
| Usage | CMP.SLE8 D0, S0, S1 |
| Function | This is an instruction to compare (lower or equal than) 8 signed 8-bit values with 8 signed 8-bit value. The result of the nth compare is stored in the nth byte of D0. Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] <= S1[n] , n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer, only lowest bit in each byte used<br>S0: 8 times 8-bit signed integer<br>S1: 8 times 8-bit signed integer |

| **CMP.SLE16** | **CoMPare Signed Less than or Equal 16-bit values** |
|---|---|
| Usage | CMP.SLE16 D0, S0, S1 |
| Function | This is an instruction to compare (lower or equal than) 4 signed 16-bit values with 4 signed 16-bit value. The result of the nth compare is stored in the nth 16-bit word of D0. |
| Behavior model | D0[n] = S0[n] <= S1[n] , n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer, only lowest bit in each 16-bit word used<br>S0: 4 times 16-bit signed integer<br>S1: 4 times 16-bit signed integer |

| **CMP.SLT** | **Less than** |
|---|---|
| Usage | CMP.SLT D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is smaller than S1, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is smaller than S1, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands are handled as 32-bit signed integers in the comparison. |
| Behavior model | SR.F = (S0 < S1 ); (D0 == R3)<br>D0 = (S0 < S1 ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>S1: 32-bit signed integer |

| **CMP.SLT8** | **CoMPare Signed Less than 8-bit values** |
|---|---|
| Usage | CMP.SLT8 D0, S0, S1 |
| Function | This is an instruction to compare (lower than) 8 signed 8-bit values with 8 signed 8-bit value. The result of the nth compare is stored in the nth byte of D0. |
| Behavior model | D0[n] = S0[n] < S1[n] , n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer, only lowest bit in each byte used<br>S0: 8 times 8-bit signed integer<br>S1: 8 times 8-bit signed integer |

| **CMP.SLT16** | **CoMPare Signed Less than 16-bit values** |
|---|---|
| Usage | CMP.SLT16 D0, S0, S1 |
| Function | This is an instruction to compare (lower than) 4 signed 16-bit values with 4 signed 16-bit value. The result of the nth compare is stored in the nth 16-bit word of D0. |
| Behavior model | D0[n] = S0[n] < S1[n] , n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer, only lowest bit in each 16-bit word used<br>S0: 4 times 16-bit signed integer<br>S1: 4 times 16-bit signed integer |

| **CMP.ULE8** | **CoMPare Unsigned Less than or Equal 8-bit values** |
|---|---|
| Usage | CMP.ULE8 D0, S0, S1 |
| Function | This is an instruction to compare (lower or equal than) 8 unsigned 8-bit values with 8 unsigned 8-bit value. The result of the nth compare is stored in the nth byte of D0. |
| Behavior model | D0[n] = S0[n] <= S1[n] , n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer, only lowest bit in each byte used<br>S0: 8 times 8-bit unsigned integer<br>S1: 8 times 8-bit unsigned integer |

| **CMP.ULE16** | **CoMPare Unsigned Less than or Equal 16-bit values** |
|---|---|
| Usage | CMP.ULE16 D0, S0, S1 |
| Function | This is an instruction to compare (lower or equal than) 4 unsigned 16-bit values with 4 unsigned 16-bit value. The result of the nth compare is stored in the nth 16-bit word of D0. |
| Behavior model | D0[n] = S0[n] <= S1[n] , n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer, only lowest bit in each 16-bit word used<br>S0: 4 times 16-bit unsigned integer<br>S1: 4 times 16-bit unsigned integer |

| **CMP.ULT8** | **CoMPare Unsigned Less than 8-bit values** |
|---|---|
| Usage | CMP.ULT8 D0, S0, S1 |
| Function | This is an instruction to compare (lower than) 8 unsigned 8-bit values with 8 unsigned 8-bit value. The result of the nth compare is stored in the nth byte of D0. |
| Behavior model | D0[n] = S0[n] < S1[n] , n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer, only lowest bit in each byte used<br>S0: 8 times 8-bit unsigned integer<br>S1: 8 times 8-bit unsigned integer |

| **CMP.ULT16** | **CoMPare Unsigned Less than 16-bit values** |
|---|---|
| Usage | CMP.ULT16 D0, S0, S1 |
| Function | This is an instruction to compare (lower than) 4 unsigned 16-bit values with 4 unsigned 16-bit value. The result of the nth compare is stored in the nth 16-bit word of D0. |
| Behavior model | D0[n] = S0[n] < S1[n] , n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer, only lowest bit in each 16-bit word used<br>S0: 4 times 16-bit unsigned integer<br>S1: 4 times 16-bit unsigned integer |

| **CMPU.EQ** | **EQual to Uniform** |
|---|---|
| Usage | CMPU.EQ D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is equal to the uniform value indexed by #IMM, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands are handled as signed integers in the comparison. |
| Behavior model | SR.F = (S0 == Uniform[#R6IMM] ); (D0 == R3)<br>D0 = (S0 == Uniform[#R6IMM] ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>R6: 8-bit unsigned integer<br>Uniform: 32-bit signed integer |

| **CMPU.NEQ** | **Not EQual to Uniform** |
|---|---|
| Usage | CMPU.NEQ D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is not equal to the uniform value indexed by #IMM, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is not equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands are handled as signed integers in the comparison. |
| Behavior model | SR.F = (S0 != Uniform [#R6IMM] ); (D0 == R3)<br>D0 = (S0 != Uniform [#R6IMM]); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>R6: 8-bit unsigned integer<br>Uniform: 32-bit signed integer |

| **CMPU.SGE** | **Greater than or Equal to Uniform** |
|---|---|
| Usage | CMPU.SGE D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is greater than or equal to the uniform value indexed by #IMM, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is greater than or equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands are handled as signed integers in the comparison. |
| Behavior model | SR.F = (S0 >= Uniform[#R6IMM] ); (D0 == R3)<br>D0 = (S0 >= Uniform[#R6IMM] ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>R6: 8-bit unsigned integer<br>Uniform: 32-bit signed integer |

| **CMPU.SGT** | **Greater Than to Uniform** |
|---|---|
| Usage | CMPU.SGT D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is greater than the uniform value indexed by #IMM, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is greater than the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands are handled as signed integers in the comparison. |
| Behavior model | SR.F = (S0 > Uniform[#R6IMM] ); (D0 == R3)<br>D0 = (S0 > Uniform[#R6IMM] ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>R6: 8-bit unsigned integer<br>Uniform: 32-bit signed integer |

| **CMPU.SLE** | **Less than or Equal to Uniform** |
|---|---|
| Usage | CMPU.SLE D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is smaller than or equal to the uniform value indexed by #IMM, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is smaller than or equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands are handled as signed integers in the comparison. |
| Behavior model | SR.F = (S0 <= Uniform[#R6IMM] ); (D0 == R3)<br>D0 = (S0 <= Uniform[#R6IMM] ); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>R6: 8-bit unsigned integer<br>Uniform: 32-bit signed integer |

| **CMPU.SLT** | **Less Than Uniform** |
|---|---|
| Usage | CMPU.SLT D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is smaller than the uniform value indexed by #IMM, sets the SR.F bit to 1, otherwise sets it to 0.<br>D0 != R3 (zero register): If S0 is smaller than the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. In this case, SR.F is not updated.<br>Both operands are handled as signed integers in the comparison. |
| Behavior model | SR.F = (S0 < Uniform[#R6IMM]); (D0 == R3)<br>D0 = (S0 < Uniform[#R6IMM); (D0 != R3) |
| Operand type | S0: 32-bit signed integer<br>R6: 8-bit unsigned integer<br>Uniform: 32-bit signed integer |

| **CMV8** | **Conditional MoVe of 8-bit values** |
|---|---|
| Usage | CMV8 D0, S0, S1 |
| Function | This is an instruction to copy dependent on 2 condition bits stored in S0[n] and S0[32+n] the value from S1[7:0] to D0[(n+1)*8-1:n*8]. The values of other bits are retained. |
| Behavior model | if (S0[n] && (S0[32+n]) then D0[(n+1)*8-1:n*8] = S1[7:0], n=0...7 |
| Operand type | D0: 8 times 8-bit value (=64-bit)<br>S0: 64-bit value (only bits 0...7 and 32...39 used)<br>S1: 8-bit value in lower 8-bit |

| **CMV16** | **Conditional MoVe of 16-bit values** |
|---|---|
| Usage | CMV16 D0, S0, S1 |
| Function | This is an instruction to copy dependent on 2 condition bits stored in S0[n] and S0[32+n] the value from S1[15:0] to D0[(n+1)*16-1:n*16]. The values of other bits are retained. |
| Behavior model | if (S0[n] && (S0[32+n]) then D0[(n+1)*16-1:n*16] = S1[15:0], n=0...3 |
| Operand type | D0: 4 times 16-bit value (=64-bit)<br>S0: 64-bit value (only bits 0...3 and 32...35 used)<br>S1: 16-bit value in lower 16-bit |

| **CNT0** | **Count 0** |
|---|---|
| Usage | CNT0 D0, S0 |
| Function | Treats S0 as a 32-bit array, counts the number of zero-valued bits in this array, and stores the result in D0. |
| Behavior model | D0 = {number of zero-valued bits in S0} |

| **CNT1** | **Count 1** |
|---|---|
| Usage | CNT1 D0, S0 |
| Function | Treats S0 as a 32-bit array, counts the number of one-valued bits in this array, and stores the result in D0. |
| Behavior model | D0 = {number of one-valued bits in S0} |

| **CSUSS** | **Signed census** |
|---|---|
| Usage | CSUSS D0, S0, S1 |
| Function | D0 = (S0[23:16] < D0[15:8]) << 31 \|<br>(D0[23:16] < D0[15:8]) << 30 \|<br>(S1[23:16] < D0[15:8]) << 29 \|<br>(S0[15:8] < D0[15:8]) << 28 \|<br>(S1[15:8] < D0[15:8]) << 27 \|<br>(S0[7:0] < D0[15:8]) << 26 \|<br>(D0[7:0] < D0[15:8]) << 25 \|<br>(S1[7:0] < D0[15:8]) << 24 \|<br>D0[23:0];<br>S0, S1 and D0 are 3 24bit vectors with each holding 3 8-bit signed elements. The instruction is calculating the census operation by comparing the center byte stored in D0[15:8] with each of the other 8 bytes. Each comparison delivers a 0 or 1 depending on the result of the comparison. These 8 comparison results are stored into D0[31:24] before D0 is stored back to the register file. |
| Behavior model | D0 = (S0[23:16] < D0[15:8]) << 31 \|<br>(D0[23:16] < D0[15:8]) << 30 \|<br>(S1[23:16] < D0[15:8]) << 29 \|<br>(S0[15:8] < D0[15:8]) << 28 \|<br>(S1[15:8] < D0[15:8]) << 27 \|<br>(S0[7:0] < D0[15:8]) << 26 \|<br>(D0[7:0] < D0[15:8]) << 25 \|<br>(S1[7:0] < D0[15:8]) << 24 \|<br>D0[23:0]; |

| **CSUSU** | **Unsigned census** |
|---|---|
| Usage | CSUSU D0, S0, S1 |
| Function | D0 = (S0[23:16] < D0[15:8]) << 31 |<br>(D0[23:16] < D0[15:8]) << 30 |<br>(S1[23:16] < D0[15:8]) << 29 |<br>(S0[15:8] < D0[15:8]) << 28 |<br>(S1[15:8] < D0[15:8]) << 27 |<br>(S0[7:0] < D0[15:8]) << 26 |<br>(D0[7:0] < D0[15:8]) << 25 |<br>(S1[7:0] < D0[15:8]) << 24 |<br>D0[23:0];<br>S0, S1 and D0 are 3 24bit vectors with each holding 3 8-bit unsigned elements. The instruction is calculating the census operation by comparing the center byte stored in D0[15:8] with each of the other 8 bytes. Each comparison delivers a 0 or 1 depending on the result of the comparison. These 8 comparison results are stored into D0[31:24] before D0 is stored back to the register file. |
| Behavior model | D0 = (S0[23:16] < D0[15:8]) << 31 |<br>(D0[23:16] < D0[15:8]) << 30 |<br>(S1[23:16] < D0[15:8]) << 29 |<br>(S0[15:8] < D0[15:8]) << 28 |<br>(S1[15:8] < D0[15:8]) << 27 |<br>(S0[7:0] < D0[15:8]) << 26 |<br>(D0[7:0] < D0[15:8]) << 25 |<br>(S1[7:0] < D0[15:8]) << 24 |<br>D0[23:0]; |

| **DMAWAITS** | **Wait for the specified DMA** |
|---|---|
| Usage | DMAWAITS #IMM |
| Function | This is a command to wait for the finish of DMA transfers. With #IMM you are specifying the thread of the core. For thread #n, each k-th bit of #IMM corresponds to TGDMAC for thread #((n+k)%8), which the master thread should wait for. Refer to section U32.8.3 . |
| Behavior model | #IMM[k] == 1 ? Wait for finishing DMA for thread #(n+k)%8. |
| Operand type | #IMM: 8-bit unsigned integer |

| **EOR** | **Exclusive OR** |
|---|---|
| Usage | EOR D0, S0, S1 |
| Function | Treats S0 and S1 as 32-bit arrays, takes the exclusive OR (XOR) of S0 and S1, and stores the result in D0. |
| Behavior model | D0 =S0 ^ S1; |

| **EOR64** | **Exclusive OR 64-bit values** |
|---|---|
| Usage | EOR64 D0, S0, S1 |
| Function | This is an instruction to XOR 2 64-bit (8 times 8-bit or 4 times 16-bit) values. |
| Behavior model | D0[n] = S0[n] XOR S1[n], n = 0...63; |
| Operand type | D0: 64-bit integer<br>S0: 64-bit integer<br>S1: 64-bit integer |

| **EORU** | **Exclusive OR with Uniform** |
|---|---|
| Usage | EORU D0, S0, #IMM |
| Function | Treats S0 and the uniform value indexed by #IMM as 32-bit arrays, takes the exclusive OR (XOR) of S0 and the uniform value, and stores the result in D0. |
| Behavior model | D0 =S0 ^ Uniform[#R6IMM]; |

| **F2I** | **Float to Integer** |
|---|---|
| Usage | F2I D0, S0 |
| Function | Assumes the S0 value is an IEEE754 single-precision floating-point number, converts it into a 32-bit signed integer, then stores the result in D0.<br>The rounding method can be changed by the control register. |
| Behavior model | D0 =int((float)S0 ); |

| **FABS** | **Float ABSolute** |
|---|---|
| Usage | FABS D0, S0 |
| Function | Stores the absolute value of S0 in D0.<br>S0 is handled as a floating-point number in the operation. |
| Behavior model | D0 = \|(float)S0\|; |

| **FADD** | **Float ADD** |
|---|---|
| Usage | FADD D0, S0, S1 |
| Function | Adds S0 and S1, then stores the result in D0.<br>S0 and S1 are handled as floating-point numbers in the operation. |
| Behavior model | D0 = (float)S0 + (float)S1; |

| **FADDU** | **Float ADD to Uniform** |
|---|---|
| Usage | FADDU D0, S0, #IMM |
| Function | Adds S0 and the uniform value indexed by #IMM, then stores the result in D0.<br>S0 and the uniform value are handled as floating-point numbers in the operation. |
| Behavior model | D0 = (float)S0 + (float)Uniform[#R6IMM]; |

| **FCMP.O** | **Ordered Floating point** |
|---|---|
| Usage | FCMP.O D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 and S1 are not NaN, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and S1 are not NaN, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = S0 != NaN and S1 != NaN; (D0 == R3)<br>D0 = S0 != NaN and S1 != NaN; (D0 != R3) |

| **FCMP.OEQ** | **Ordered Floating point equal** |
|---|---|
| Usage | FCMP.OEQ D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 and S1 are not NaN and S0 is equal to S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and S1 are not NaN and S0 is equal to S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 == S1 ) and S0 != NaN and S1 != NaN; (D0 == R3)<br>D0 = (S0 == S1 ) and S0 != NaN and S1 != NaN; (D0 != R3) |

| **FCMP.OGE** | **Ordered Floating point greater than or equal** |
|---|---|
| Usage | FCMP.OGE D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 and S1 are not NaN and S0 is greater than or equal S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and S1 are not NaN and S0 is greater than or equal S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 >= S1 ) and S0 != NaN and S1 != NaN; (D0 == R3)<br>D0 = (S0 >= S1 ) and S0 != NaN and S1 != NaN; (D0 != R3) |

| **FCMP.OGT** | **Ordered Floating point greater than** |
|---|---|
| Usage | FCMP.OGT D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 and S1 are not NaN and S0 is greater than S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and S1 are not NaN and S0 is greater than S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 > S1 ) and S0 != NaN and S1 != NaN; (D0 == R3)<br>D0 = (S0 > S1 ) and S0 != NaN and S1 != NaN; (D0 != R3) |

| **FCMP.OLE** | **Ordered Floating point less than or equal** |
|---|---|
| Usage | FCMP.OLE D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 and S1 are not NaN and S0 is less than or equal S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and S1 are not NaN and S0 is less than or equal S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 <= S1 ) and S0 != NaN and S1 != NaN; (D0 == R3)<br>D0 = (S0 <= S1 ) and S0 != NaN and S1 != NaN; (D0 != R3) |

| **FCMP.OLT** | **Ordered Floating point less than** |
|---|---|
| Usage | FCMP.OLT D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 and S1 are not NaN and S0 is less than S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and S1 are not NaN and S0 is less than S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 < S1 ) and S0 != NaN and S1 != NaN; (D0 == R3)<br>D0 = (S0 < S1 ) and S0 != NaN and S1 != NaN; (D0 != R3) |

| **FCMP.ONE** | **Ordered Floating point not equal** |
|---|---|
| Usage | FCMP.ONE D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 and S1 are not NaN and S0 is not equal S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and S1 are not NaN and S0 is not equal S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 != S1 ) and S0 != NaN and S1 != NaN; (D0 == R3)<br>D0 = (S0 != S1 ) and S0 != NaN and S1 != NaN; (D0 != R3) |

| **FCMP.UEQ** | **Unordered Floating point equal** |
|---|---|
| Usage | FCMP.UEQ D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is NaN or S1 is NaN or S0 is equal to S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or S1 is NaN or S0 is equal to S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 == S1 ) or S0 == NaN or S1 == NaN; (D0 == R3)<br>D0 = (S0 == S1 ) or S0 == NaN or S1 == NaN; (D0 != R3) |

| **FCMP.UGE** | **Unordered Floating point greater than or equal** |
|---|---|
| Usage | FCMP.UGE D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is NaN or S1 is NaN or S0 is greater than or equal S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or S1 is NaN or S0 is greater than or equal S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 >= S1 ) or S0 == NaN or S1 == NaN; (D0 == R3)<br>D0 = (S0 >= S1 ) or S0 == NaN or S1 == NaN; (D0 != R3) |

| **FCMP.UGT** | **Unordered Floating point greater than** |
|---|---|
| Usage | FCMP.UGT D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is NaN or S1 is NaN or S0 is greater than S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or S1 is NaN or S0 is greater than S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 > S1 ) or S0 == NaN or S1 == NaN; (D0 == R3)<br>D0 = (S0 > S1 ) or S0 == NaN or S1 == NaN; (D0 != R3) |

| **FCMP.ULE** | **Unordered Floating point less than or equal** |
|---|---|
| Usage | FCMP.ULE D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is NaN or S1 is NaN or S0 is less than or equal S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or S1 is NaN or S0 is less than or equal S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 <= S1 ) or S0 == NaN or S1 == NaN; (D0 == R3)<br>D0 = (S0 <= S1 ) or S0 == NaN or S1 == NaN; (D0 != R3) |

| **FCMP.ULT** | **Unordered Floating point less than** |
|---|---|
| Usage | FCMP.ULT D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is NaN or S1 is NaN or S0 is less than S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or S1 is NaN or S0 is less than S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 < S1 ) or S0 == NaN or S1 == NaN; (D0 == R3)<br>D0 = (S0 < S1 ) or S0 == NaN or S1 == NaN; (D0 != R3) |

| **FCMP.UNE** | **Unordered Floating not equal** |
|---|---|
| Usage | FCMP.UNE D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is NaN or S1 is NaN or S0 is not equal S1, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or S1 is NaN or S0 is not equal S1, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 != S1 ) or S0 == NaN or S1 == NaN; (D0 == R3)<br>D0 = (S0 != S1 ) or S0 == NaN or S1 == NaN; (D0 != R3) |

| **FCMP.UO** | **Unordered Floating point** |
|---|---|
| Usage | FCMP.UO D0, S0, S1 |
| Function | D0 == R3 (zero register): If S0 is NaN or S1 is NaN, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or S1 is NaN, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = S0 == NaN or S1 == NaN; (D0 == R3)<br>D0 = S0 == NaN or S1 == NaN; (D0 != R3) |

| **FCMPU.O** | **Ordered Floating point with Uniform** |
|---|---|
| Usage | FCMPU.O D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and the uniform value indexed by #IMM, are not NaN, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = S0 != NaN and Uniform[#R6IMM] != NaN; (D0 == R3)<br>D0 = S0 != NaN and Uniform[#R6IMM] != NaN; (D0 != R3) |

| **FCMPU.OEQ** | **Ordered Floating point equal to Uniform** |
|---|---|
| Usage | FCMPU.OEQ D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is equal to the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and Uniform[R6+#IMM] are not NaN and S0 is equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 == Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 == R3)<br>D0 = (S0 == Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 != R3) |

| **FCMPU.OGE** | **Ordered Floating point greater than or equal to Uniform** |
|---|---|
| Usage | FCMPU.OGE D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is greater than or equal to the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is greater than or equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 >= Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 == R3)<br>D0 = (S0 >= Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 != R3) |

| **FCMPU.OGT** | **Ordered Floating point greater than Uniform** |
|---|---|
| Usage | FCMPU.OGT D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is greater than the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is greater than the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 > Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 == R3)<br>D0 = (S0 > Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 != R3) |

| **FCMPU.OLE** | **Ordered Floating point less than or equal to Uniform** |
|---|---|
| Usage | FCMPU.OLE D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is less than or equal to the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is less than or equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 <= Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 == R3)<br>D0 = (S0 <= Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 != R3) |

| **FCMPU.OLT** | **Ordered Floating point less than Uniform** |
|---|---|
| Usage | FCMPU.OLT D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is less than the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is less than the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 < Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 == R3)<br>D0 = (S0 < Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 != R3) |

| **FCMPU.ONE** | **Ordered Floating point not equal to Uniform** |
|---|---|
| Usage | FCMPU.ONE D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is not equal the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 and the uniform value indexed by #IMM are not NaN and S0 is not equal the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 != Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 == R3)<br>D0 = (S0 != Uniform[#R6IMM]) and S0 != NaN and Uniform[#R6IMM] != NaN; (D0 != R3) |

| **FCMPU.UEQ** | **Unordered Floating point equal to Uniform** |
|---|---|
| Usage | FCMPU.UEQ D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is equal to the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 == Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 == R3)<br>D0 = (S0 == Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 != R3) |

| **FCMPU.UGE** | **Unordered Floating point greater than or equal to Uniform** |
|---|---|
| Usage | FCMPU.UGE D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is greater than or equal the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is greater than or equal the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 >= Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 == R3)<br>D0 = (S0 >= Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 != R3) |

| **FCMPU.UGT** | **Unordered Floating point greater than Uniform** |
|---|---|
| Usage | FCMPU.UGT D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is greater than the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is greater than the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 > Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 == R3)<br>D0 = (S0 > Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 != R3) |

| **FCMPU.ULE** | **Unordered Floating point less than or equal to Uniform** |
|---|---|
| Usage | FCMPU.ULE D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is less than or equal to the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is less than or equal to the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 <= Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 == R3)<br>D0 = (S0 <= Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 != R3) |

| **FCMPU.ULT** | **Unordered Floating point less than Uniform** |
|---|---|
| Usage | FCMPU.ULT D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is less than the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is less than the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 < Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 == R3)<br>D0 = (S0 < Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 != R3) |

| **FCMPU.UNE** | **Unordered Floating not equal to Uniform** |
|---|---|
| Usage | FCMPU.UNE Do, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is not equal the uniform value indexed by #IMM, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN or S0 is not equal the uniform value indexed by #IMM, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = (S0 != Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 == R3)<br>D0 = (S0 != Uniform[#R6IMM]) or S0 == NaN or Uniform[#R6IMM] == NaN; (D0 != R3) |

| **FCMPU.UO** | **Unordered Floating point with Uniform** |
|---|---|
| Usage | FCMPU.UO D0, S0, #IMM |
| Function | D0 == R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN, status register flag (SR.F) is set to 1, else it is set to 0.<br>D0 != R3 (zero register): If S0 is NaN or the uniform value indexed by #IMM is NaN, D0 is set to 1, else it is set to 0. |
| Behavior model | SR.F = S0 == NaN or Uniform[#R6IMM] == NaN; (D0 == R3)<br>D0 = S0 == NaN or Uniform[#R6IMM] == NaN; (D0 != R3) |

| **FFLR** | **Float FLooR** |
|---|---|
| Usage | FFLR D0, S0 |
| Function | Stores the integer part of S0 in D0. This integer part is calculated according to a configuration of CR18. The default value of CR 18 is 1 and the logic is integer part of floating value (round towards zero). When CR18 is 0 floating value is rounded to "nearest" integer. The result is stored to D0 and S0 is handled as a floating-point number in the operation. |
| Behavior model | D0=floor ((float)S0); |

| **FFRC** | **Float FRaCtion** |
|---|---|
| Usage | FFRC D0, S0 |
| Function | Subtracts the integer part of S0 from S0. This integer part is calculated according to a configuration of CR18. value of CR 18 is 1 and the logic is integer part of floating value (round towards zero). When CR18 is 0 floating value is rounded to "nearest" integer. The result is stored to D0 and S0 is handled as a floating-point number in the operation. |
| Behavior model | D0=S0 - floor ((float)S0); |

| **FLDCI** | **Float Load from Control Register** |
|---|---|
| Usage | FLDCI D0, #IMM |
| Function | This is an instruction for loading data from a control register.<br>Writes the data read from the control register at the address specified by #IMM to D0.<br>The value written to D0 is floating-point.<br>Specifying #IMM as the number of a control register that does not exist is prohibited. |
| Behavior model | D0 = (float)ControlRegister[#IMM]; |

| **FLDRCI** | **Float Load from Relative Coordinate XY with Immediate offset** |
|---|---|
| Usage | FLDRCI D0, #IMM1, #IMM2 |
| Function | This is an instruction for data transfer from memory to a register.<br>X and Y coordinates are specified in R0 and R1 respectively. Reads the value from memory at the coordinates obtained by adding the #IMM1 value to X and the #IMM2 value to Y and stores the value in D0.<br>The full 32 bits of R0 and the 16 lower-order bits of R1 are handled as signed integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>The value stored in D0 is floating-point. |
| Behavior model | D0 = (float) *(R0+(char)#IMM1, (short)((short)(R1&0xffff)+(char)#IMM2)); |
| Operand type | D0: 32-bit signed floating-point<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **FLDRCIU** | **Float Load from Relative Coordinate XY with Immediate offset, then source Updated** |
|---|---|
| Usage | FLDRCIU D0, #IMM1, #IMM2 |
| Function | This is an instruction for data transfer from memory to a register.<br>X and Y coordinates are specified in R0 and R1 respectively. Reads the value from memory at the coordinates obtained by adding the #IMM1 value to X and the #IMM2 value to Y and stores the value in D0.<br>The full 32 bits of R0 and the 16 lower-order bits of R1 are handled as signed integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers. After operation, X(R0) and Y(R1) are updated by adding #IMM1 and #IMM2 respectively.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>The value stored in D0 is floating-point. |
| Behavior model | D0 = (float) *(R0+(char)#IMM1, (short)((short)(R1&0xffff)+(char)#IMM2));<br>R0 = R0+(char)#IMM1; R1 = R1+ (char)#IMM2; |
| Operand type | D0: 32-bit signed floating-point<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **FLDRXY** | **Float Load from Relative Coordinate XY with register offset** |
|---|---|
| Usage | FLDRXY D0, S0, S1 |
| Function | This is an instruction for data transfer from memory to a register.<br>X and Y coordinates are specified in R0 and R1 respectively. Reads the value from memory at the coordinates obtained by adding the S0 value to X and the S1 value to Y and stores the value in D0.<br>The full 32 bits of R0 and S0 and the 16 lower-order bits of R1 and S1 are handled as signed integers.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>The value stored in D0 is floating-point. |
| Behavior model | D0 = (float) *(R0+S0, (short)((short)(R1&0xffff)+(short)(S1&0xffff)));<br> |
| Operand type | D0: 32-bit signed floating-point<br>S0: 32-bit signed integer<br>S1: 16-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **FLDRXYU** | **Float Load from Relative Coordinate XY with register offset, then source Updated** |
|---|---|
| Usage | FLDRXYU D0, S0, S1 |
| Function | This is an instruction for data transfer from memory to a register.<br>X and Y coordinates are specified in R0 and R1 respectively. Reads the value from memory at the coordinates obtained by adding the S0 value to X and the S1 value to Y and stores the value in D0.<br>The full 32 bits of R0 and S0 and the 16 lower-order bits of R1 and S1 are handled as signed integers. After operation, X(R0) and Y(R1) are updated by adding S0 and S1 respectively.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>The value stored in D0 is floating-point. |
| Behavior model | D0 = (float) *(R0+S0, (short)((short)(R1&0xffff)+(short)(S1&0xffff)));<br>R0 = R0+S0; R1 = (short)((short)(R1&0xffff)+(short)(S1&0xffff)); |
| Operand type | D0: 32-bit signed floating-point<br>S0: 32-bit signed integer<br>S1: 16-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **FMAD** | **Float Multiply and ADd** |
|---|---|
| Usage | FMAD D0, S0, S1 |
| Function | Adds D0 and the product of S0 and S1, then stores the result in D0.<br>D0, S0 and S1 are handled as floating-point numbers in the operation.<br>The result of this instruction matches that of the similar operation obtained by combining the FMUL and FADD instructions. |
| Behavior model | D0 = S0*S1 + D0; |

| **FMADU** | **Float Multiply and Add Uniform** |
|---|---|
| Usage | FMADU D0, S0, #IMM |
| Function | Adds D0 and the product of S0 and the uniform value indexed by #IMM, then stores the result in D0.<br>D0, S0 and S1 are handled as floating-point numbers in the operation.<br>The result of this instruction matches that of the similar operation obtained by combining the FMULU and FADD instructions. |
| Behavior model | D0 = S0*Uniform[#R6IMM] + D0; |

| **FMAX** | **Float MAXimum** |
|---|---|
| Usage | FMAX D0, S0, S1 |
| Function | Selects the larger value between S0 and S1, then stores the result in D0.<br>S0 and S1 are handled as floating-point numbers in the operation. |
| Behavior model | D0 = (S0 > S1) ? S0 : S1; |

| **FMAXU** | **Float MAXimum compared to Uniform** |
|---|---|
| Usage | FMAXU D0, S0, #IMM |
| Function | Selects the larger value between S0 and the uniform value indexed by #IMM, then stores the result in D0.<br>S0 and the uniform value are handled as floating-point numbers in the operation. |
| Behavior model | D0 = (S0 > Uniform[#R6IMM]) ? S0 : Uniform[#R6IMM]); |

| **FMIN** | **Float MINimum** |
|---|---|
| Usage | FMIN D0, S0, S1 |
| Function | Selects the smaller value between S0 and S1, then stores the result in D0.<br>S0 and S1 are handled as floating-point numbers in the operation. |
| Behavior model | D0 = (S0 > S1) ? S1 : S0; |

| **FMINU** | **Float MINimum compared to Uniform** |
|---|---|
| Usage | FMINU D0, S0, #IMM |
| Function | Selects the smaller value between S0 and the uniform value indexed by #IMM, then stores the result in D0.<br>S0 and the uniform value are handled as floating-point numbers in the operation. |
| Behavior model | D0 = (S0 > Uniform[#R6IMM]) ? Uniform[#R6IMM]) : S0; |

| **FMP** | **Float Middle Point** |
|---|---|
| Usage | FMP D0, S0, S1 |
| Function | Divides the sum of S0 and S1 by 2.0, then stores the result in D0.<br>S0 and S1 are handled as floating-point numbers in the operation. |
| Behavior model | D0 = (S0 + S1) / 2.0; |

| **FMSU** | **Float Multiply and Sub** |
|---|---|
| Usage | FMSU D0, S0, S1 |
| Function | Subtracts the product of S0 and S1 from D0, then stores the result in D0.<br>D0, S0 and S1 are handled as floating-point numbers in the operation.<br>The result of this instruction matches that of the similar operation obtained by combining the FMUL and FSUB instructions. |
| Behavior model | D0 = -S0*S1 + D0; |

| **FMSUU** | **Float Multiply and Sub Uniform** |
|---|---|
| Usage | FMSUU D0, S0, #IMM |
| Function | Subtracts the product of S0 and the uniform value indexed by #IMM from D0, then stores the result in D0.<br>D0, S0 and S1 are handled as floating-point numbers in the operation.<br>The result of this instruction matches that of the similar operation obtained by combining the FMULU and FSUB instructions. |
| Behavior model | D0 = -S0*Uniform[#R6IMM] + D0; |

| **FMUL** | **Float MULtiply** |
|---|---|
| Usage | FMUL D0, S0, S1 |
| Function | Multiplies S0 and S1, then stores the result in D0.<br>S0 and S1 are handled as floating-point numbers in the operation. |
| Behavior model | D0 = S0 * S1; |

| **FMULU** | **Float MULtiply by Uniform** |
|---|---|
| Usage | FMULU D0, S0, #IMM |
| Function | Multiplies S0 and the uniform value indexed by #IMM, then stores the result in D0.<br>S0 and the uniform value are handled as floating-point numbers in the operation. |
| Behavior model | D0 = S0 * Uniform[#R6IMM]; |

| **FSTC** | **Float STore to Control Register** |
|---|---|
| Usage | FSTC D0, S0 |
| Function | This is an instruction for storing data in a control register.<br>The S0 value is converted to integer and written to the control register D0. Specifying D0 as a control register that does not exist is prohibited. |
| Behavior model | ControlRegisterD0 = (int)S0; |

| **FSTRCI** | **Float STore to Relative Coordinate with Immediate offset** |
|---|---|
| Usage | FSTRCI D0, #IMM1, #IMM2 |
| Function | This is an instruction for data transfer from a register to memory.<br>Stores the D0 value in memory at X and Y coordinates which are specified in R0 and R1 respectively, with the #IMM1 value added to X and the #IMM2 value added to Y.<br>The full 32 bits of R0 and the 16 lower-order bits of R1 are handled as signed integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>D0 is handled as a floating-point number. |
| Behavior model | @(R0+(char)#IMM1, (short)((short)(R1&0xffff)+(char)#IMM2))= D0 |
| Operand type | D0: 32-bit signed floating-point<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **FSTRCIU** | **Float STore to Relative Coordinate with Immediate offset, then source Updated** |
|---|---|
| Usage | FSTRCIU D0, #IMM1, #IMM2 |
| Function | This is an instruction for data transfer from a register to memory.<br>Stores the D0 value in memory at X and Y coordinates which are specified in R0 and R1 respectively, with the #IMM1 value added to X and the #IMM2 value added to Y.<br>The full 32 bits of R0 and the 16 lower-order bits of R1 are handled as signed integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers. After operation, X(R0) and Y(R1) are updated by adding #IMM1 and #IMM2 respectively.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>D0 is handled as a floating-point number. |
| Behavior model | @(R0+(char)#IMM1, (short)((short)(R1&0xffff)+(char)#IMM2)) = D0;<br>R0 = R0+(char)#IMM1; R1 = R1+ (char)#IMM2; |
| Operand type | D0: 32-bit signed floating-point<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **FSTRXY** | **Float STore to Relative Coordinate XY with register offset** |
|---|---|
| Usage | FSTRXY D0, S0, S1 |
| Function | This is an instruction for data transfer from a register to memory.<br>Stores the D0 value in memory at X and Y coordinates which are specified in R0 and R1 respectively, with the S0 value added to X and the S1 value added to Y.<br>The full 32 bits of R0 and S0 and the 16 lower-order bits of R1 and S1 are handled as signed integers.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>D0 is handled as a floating-point number. |
| Behavior model | @(R0+S0, (short)((short)(R1&0xffff)+(short)(S1&0xffff))) = D0; |
| Operand type | D0: 32-bit signed floating-point<br>S0: 32-bit signed integer<br>S1: 16-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **FSTRXYU** | **Float STore to Relative Coordinate XY with register offset, then source Updated** |
|---|---|
| Usage | FSTRXYU D0, S0, S1 |
| Function | This is an instruction for data transfer from a register to memory.<br>Stores the D0 value in memory at X and Y coordinates which are specified in R0 and R1 respectively, with the S0 value added to X and the S1 value added to Y.<br>The full 32 bits of R0 and S0 and the 16 lower-order bits of R1 and S1 are handled as signed integers. After operation, X(R0) and Y(R1) are updated by adding S0 and S1 respectively.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>D0 is handled as a floating-point number. |
| Behavior model | @(R0+S0, (short)((short)(R1&0xffff)+(short)(S1&0xffff))) = D0;<br>R0 = R0+S0; R1 = (short)((short)(R1&0xffff)+(short)(S1&0xffff)); |
| Operand type | D0: 32-bit signed floating-point<br>S0: 32-bit signed integer<br>S1: 16-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **FSUB** | **Float SUB** |
|---|---|
| Usage | FSUB D0, S0, S1 |
| Function | Subtracts S1 from S0, then stores the result in D0.<br>S0 and S1 are handled as floating-point numbers in the operation. |
| Behavior model | D0 = S0 - S1; |

| **FSUBU** | **Float SUB Uniform from** |
|---|---|
| Usage | FSUBU D0, S0, #IMM |
| Function | Subtracts the uniform value indexed by #IMM from S0, then stores the result in D0.<br>S0 and the uniform value are handled as floating-point numbers in the operation. |
| Behavior model | D0 = S0 - Uniform[#R6IMM]; |

| **GADDBS** | **Group add and sum up for 8-bit signed input data** |
|---|---|
| Usage | GADDBS D0, S0, S1 |
| Function | D0 = D0+S0[0]+S1[0]+S0[1]+S1[1]+S0[2]+S1[2]+S0[3]+S1[3]+S0[4]+S1[4]+S0[5]+S1[5]+S0[6]+S1[6] +S0[7]+S1[7]<br>S0 and S1 are signed 8-bit vectors with each 8 elements. The instruction is performing an addition of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]+S1[0]+S0[1]+S1[1]+S0[2]+S1[2]+S0[3]+S1[3]+S0[4]+S1[4]+S0[5]+S1[5]+S0[6] +S1[6]+S0[7]+S1[7] |
| Operand type | D0: 32-bit signed integer<br>S0: 8 * 8-bit signed integer<br>S1: 8 * 8-bit signed integer |

| **GADDBU** | **Group add and sum up for 8-bit unsigned input data** |
|---|---|
| Usage | GADDBU D0, S0, S1 |
| Function | D0 = D0+S0[0]+S1[0]+S0[1]+S1[1]+S0[2]+S1[2]+S0[3]+S1[3]+S0[4]+S1[4]+S0[5]+S1[5]+S0[6]+S1[6] +S0[7]+S1[7]<br>S0 and S1 are unsigned 8-bit vectors with each 8 elements. The instruction is performing an addition of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]+S1[0]+S0[1]+S1[1]+S0[2]+S1[2]+S0[3]+S1[3]+S0[4]+S1[4]+S0[5]+S1[5]+S0[6] +S1[6]+S0[7]+S1[7] |
| Operand type | D0: 32-bit signed integer<br>S0: 8 * 8-bit unsigned integer<br>S1: 8 * 8-bit unsigned integer |

| **GADDHS** | **Group add and sum up for 16-bit signed input data** |
|---|---|
| Usage | GADDHS D0, S0, S1 |
| Function | D0 = D0+S0[0]+S1[0]+S0[1]+S1[1]+S0[2]+S1[2]+S0[3]+S1[3]<br>S0 and S1 are signed 16-bit vectors with each 4 elements. The instruction is performing an addition of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0.D0 is a 32-bit signed value.S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]+S1[0]+S0[1]+S1[1]+S0[2]+S1[2]+S0[3]+S1[3] |
| Operand type | D0: 32-bit signed integer<br>S0: 4 * 16-bit signed integer<br>S1: 4 * 16-bit signed integer |

| **GADDHU** | **Group add and sum up for 16-bit unsigned input data** |
|---|---|
| Usage | GADDHU D0, S0, S1 |
| Function | D0 = D0+S0[0]+S1[0]+S0[1]+S1[1]+S0[2]+S1[2]+S0[3]+S1[3]<br>S0 and S1 are unsigned 16-bit vectors with each 4 elements. The instruction is performing an addition of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0.D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]+S1[0]+S0[1]+S1[1]+S0[2]+S1[2]+S0[3]+S1[3] |
| Operand type | D0: 32-bit signed integer<br>S0: 4 * 16-bit unsigned integer<br>S1: 4 * 16-bit unsigned integer |

| **GMADBS** | **Group multiply and add for 8-bit signed input data** |
|---|---|
| Usage | GMADBS D0, S0, S1 |
| Function | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]+S0[4]*S1[4]+S0[5]*S1[5]+S0[6]*S1[6]+S0[7]*S1[7]<br>S0 and S1 are signed 8-bit vectors with each 8 elements. The instruction is performing a multiplication of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]+S0[4]*S1[4]+S0[5]*S1[5]+S0[6]*S1[6]+S0[7]*S1[7] |
| Operand type | D0: 32-bit signed integer<br>S0: 8 * 8-bit signed integer<br>S1: 8 * 8-bit signed integer |

| **GMADBSL** | **Group multiply and add for 8-bit signed input data long** |
|---|---|
| Usage | GMADBSL D0, S0, S1 |
| Function | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]+S0[4]*S1[4]+S0[5]*S1[5]+S0[6]*S1[6]+S0[7]*S1[7]<br>S0 and S1 are signed 8-bit vectors with each 8 elements. The instruction is performing a multiplication of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 64-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]+S0[4]*S1[4]+S0[5]*S1[5]+S0[6]*S1[6]+S0[7]*S1[7] |
| Operand type | D0: 64-bit signed integer<br>S0: 8 * 8-bit signed integer<br>S1: 8 * 8-bit signed integer |

| **GMADBU** | **Group multiply and add for 8-bit unsigned input data** |
|---|---|
| Usage | GMADBU D0, S0, S1 |
| Function | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]+S0[4]*S1[4]+S0[5]*S1[5]+S0[6]*S1[6]+S0[7]*S1[7]<br>S0 and S1 are unsigned 8-bit vectors with each 8 elements. The instruction is performing a multiplication of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]+S0[4]*S1[4]+S0[5]*S1[5]+S0[6]*S1[6]+S0[7]*S1[7] |
| Operand type | D0: 32-bit signed integer<br>S0: 8 * 8-bit unsigned integer<br>S1: 8 * 8-bit unsigned integer |

| **GMADBUL** | **Group multiply and add for 8-bit unsigned input data long** |
|---|---|
| Usage | GMADBUL D0, S0, S1 |
| Function | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]+S0[4]*S1[4]+S0[5]*S1[5]+S0[6]*S1[6]+S0[7]*S1[7]<br>S0 and S1 are unsigned 8-bit vectors with each 8 elements. The instruction is performing a multiplication of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 64-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]+S0[4]*S1[4]+S0[5]*S1[5]+S0[6]*S1[6]+S0[7]*S1[7] |
| Operand type | D0: 64-bit unsigned integer<br>S0: 8 * 8-bit unsigned integer<br>S1: 8 * 8-bit unsigned integer |

| **GMADHS** | **Group multiply and add for 16-bit signed input data** |
|---|---|
| Usage | GMADHS D0, S0, S1 |
| Function | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3]<br>S0 and S1 are signed 16-bit vectors with each 4 elements. The instruction is performing a multiplication of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers.<br>Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3] |
| Operand type | D0: 32-bit signed integer<br>S0: 4 * 16-bit signed integer<br>S1: 4 * 16-bit signed integer |

| **GMADHSL** | **Group multiply and add for 16-bit signed input data long** |
|---|---|
| Usage | GMADHSL D0, S0, S1 |
| Function | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3] |
| | S0 and S1 are signed 16-bit vectors with each 4 elements. The instruction is performing a multiplication of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 64-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| | Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3] |
| Operand type | D0: 64-bit signed integer |
| | S0: 4 * 16-bit signed integer |
| | S1: 4 * 16-bit signed integer |

| **GMADHU** | **Group multiply and add for 16-bit unsigned input data** |
|---|---|
| Usage | GMADHU D0, S0, S1 |
| Function | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3] |
| | S0 and S1 are unsigned 16-bit vectors with each 4 elements. The instruction is performing a multiplication of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit unsigned value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| | Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3] |
| Operand type | D0: 32-bit unsigned integer |
| | S0: 4 * 16-bit unsigned integer |
| | S1: 4 * 16-bit unsigned integer |

| **GMADHUL** | **Group multiply and add for 16-bit unsigned input data long** |
|---|---|
| Usage | GMADHUL D0, S0, S1 |
| Function | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3] |
| | S0 and S1 are unsigned 16-bit vectors with each 4 elements. The instruction is performing a multiplication of each element between the vectors S0 and S1 and adding all results to the input D0 before storing back the result to D0. D0 is a 64-bit unsigned value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| | Hardware does not take care any overflow handlings, so software must take care of it. |
| Behavior model | D0 = D0+S0[0]*S1[0]+S0[1]*S1[1]+S0[2]*S1[2]+S0[3]*S1[3] |
| Operand type | D0: 64-bit unsigned integer |
| | S0: 4 * 16-bit unsigned integer |
| | S1: 4 * 16-bit unsigned integer |

| **GSADBS** | **Group SAD and sum up for 8-bit signed input data** |
|---|---|
| Usage | GSADBS D0, S0, S1 |
| Function | D0 = D0+\|S0[0]-S1[0]\|+\|S0[1]-S1[1]\|+\|S0[2]-S1[2]\|+\|S0[3]-S1[3]\|+\|S0[4]-S1[4]\|+\|S0[5]-S1[5]\|+\|S0[6]-S1[6]\|+\|S0[7]-S1[7]\|<br>S0 and S1 are signed 8-bit vectors with each 8 elements. The instruction is performing difference calculation of each element of vector S1 from vector S0 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = D0+\|S0[0]-S1[0]\|+\|S0[1]-S1[1]\|+\|S0[2]-S1[2]\|+\|S0[3]-S1[3]\|+\|S0[4]-S1[4]\|+\|S0[5]-S1[5]\|+\|S0[6]-S1[6]\|+\|S0[7]-S1[7]\| |
| Operand type | D0: 32-bit signed integer<br>S0: 8 * 8-bit signed integer<br>S1: 8 * 8-bit signed integer |

| **GSADBU** | **Group SAD and sum up for 8-bit unsigned input data** |
|---|---|
| Usage | GSADBU D0, S0, S1 |
| Function | D0 = D0+\|S0[0]-S1[0]\|+\|S0[1]-S1[1]\|+\|S0[2]-S1[2]\|+\|S0[3]-S1[3]\|+\|S0[4]-S1[4]\|+\|S0[5]-S1[5]\|+\|S0[6]-S1[6]\|+\|S0[7]-S1[7]\|<br>S0 and S1 are unsigned 8-bit vectors with each 8 elements. The instruction is performing a difference calculation of each element of vector S1 from vector S0 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit unsigned value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = D0+\|S0[0]-S1[0]\|+\|S0[1]-S1[1]\|+\|S0[2]-S1[2]\|+\|S0[3]-S1[3]\|+\|S0[4]-S1[4]\|+\|S0[5]-S1[5]\|+\|S0[6]-S1[6]\|+\|S0[7]-S1[7]\| |
| Operand type | D0: 32-bit unsigned integer<br>S0: 8 * 8-bit unsigned integer<br>S1: 8 * 8-bit unsigned integer |

| **GSADHS** | **Group SAD and sum up for 16-bit signed input data** |
|---|---|
| Usage | GSADHS D0, S0, S1 |
| Function | D0 = D0+\|S0[0]-S1[0]\|+\|S0[1]-S1[1]\|+\|S0[2]-S1[2]\|+\|S0[3]-S1[3]\|<br>S0 and S1 are signed 16-bit vectors with each 4 elements. The instruction is performing a difference calculation of each element of vector S1 from vector S0 and adding all results to the input D0 before storing back the result to D0.D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = D0+\|S0[0]-S1[0]\|+\|S0[1]-S1[1]\|+\|S0[2]-S1[2]\|+\|S0[3]-S1[3]\| |
| Operand type | D0: 32-bit signed integer<br>S0: 4 * 16-bit signed integer<br>S1: 4 * 16-bit signed integer |

| **GSADHU** | **Group SAD and sum up for 16-bit unsigned input data** |
|---|---|
| Usage | GSADHU D0, S0, S1 |
| Function | D0 = D0+\|S0[0]-S1[0]\|+\|S0[1]-S1[1]\|+\|S0[2]-S1[2]\|+\|S0[3]-S1[3]\| |
| | S0 and S1 are unsigned 16-bit vectors with each 4 elements. The instruction is performing difference calculation of each element of vector S1 from vector S0 and adding all results to the input D0 before storing back the result to D0.D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = D0+\|S0[0]-S1[0]\|+\|S0[1]-S1[1]\|+\|S0[2]-S1[2]\|+\|S0[3]-S1[3]\| |
| Operand type | D0: 32-bit signed integer |
| | S0: 4 * 16-bit unsigned integer |
| | S1: 4 * 16-bit unsigned integer |

| **GSUBBS** | **Group sub and sum up for 8-bit signed input data** |
|---|---|
| Usage | GSUBBS D0, S0, S1 |
| Function | D0 = D0+(S0[0]-S1[0])+(S0[1]-S1[1])+(S0[2]-S1[2])+(S0[3]-S1[3])+(S0[4]-S1[4])+(S0[5]-S1[5])+(S0[6]-S1[6])+(S0[7]-S1[7]) |
| | S0 and S1 are signed 8-bit vectors with each 8 elements. The instruction is performing a subtraction of each element of vector S1 from vector S0 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = D0+(S0[0]-S1[0])+(S0[1]-S1[1])+(S0[2]-S1[2])+(S0[3]-S1[3])+(S0[4]-S1[4])+(S0[5]-S1[5])+(S0[6]-S1[6])+(S0[7]-S1[7]) |
| Operand type | D0: 32-bit signed integer |
| | S0: 8 * 8-bit signed integer |
| | S1: 8 * 8-bit signed integer |

| **GSUBBU** | **Group sub and sum up for 8-bit unsigned input data** |
|---|---|
| Usage | GSUBBU D0, S0, S1 |
| Function | D0 = D0+(S0[0]-S1[0])+(S0[1]-S1[1])+(S0[2]-S1[2])+(S0[3]-S1[3])+(S0[4]-S1[4])+(S0[5]-S1[5])+(S0[6]-S1[6])+(S0[7]-S1[7]) |
| | S0 and S1 are unsigned 8-bit vectors with each 8 elements. The instruction is performing a subtraction of each element of vector S1 from vector S0 and adding all results to the input D0 before storing back the result to D0. D0 is a 32-bit signed value. S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = D0+(S0[0]-S1[0])+(S0[1]-S1[1])+(S0[2]-S1[2])+(S0[3]-S1[3])+(S0[4]-S1[4])+(S0[5]-S1[5])+(S0[6]-S1[6])+(S0[7]-S1[7]) |
| Operand type | D0: 32-bit signed integer |
| | S0: 8 * 8-bit unsigned integer |
| | S1: 8 * 8-bit unsigned integer |

| **GSUBHS** | **Group sub and sum up for 16-bit signed input data** |
|---|---|
| Usage | GSUBHS D0, S0, S1 |
| Function | D0 = D0+(S0[0]-S1[0])+(S0[1]-S1[1])+(S0[2]-S1[2])+(S0[3]-S1[3]) |
| | S0 and S1 are signed 16-bit vectors with each 4 elements. The instruction is performing a subtraction of each element of vector S1 from vector S0 and adding all results to the input D0 before storing back the result to D0.D0 is a 32-bit signed value.S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = D0+(S0[0]-S1[0])+(S0[1]-S1[1])+(S0[2]-S1[2])+(S0[3]-S1[3]) |
| Operand type | D0: 32-bit signed integer |
| | S0: 4 * 16-bit signed integer |
| | S1: 4 * 16-bit signed integer |

| **GSUBHU** | **Group sub and sum up for 16-bit unsigned input data** |
|---|---|
| Usage | GSUBHU D0, S0, S1 |
| Function | D0 = D0+(S0[0]-S1[0])+(S0[1]-S1[1])+(S0[2]-S1[2])+(S0[3]-S1[3]) |
| | S0 and S1 are unsigned 16-bit vectors with each 4 elements. The instruction is performing a subtraction of each element of vector S1 from vector S0 and adding all results to the input D0 before storing back the result to D0.D0 is a 32-bit signed value.S0 and S1 are 64-bit vectors, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = D0+(S0[0]-S1[0])+(S0[1]-S1[1])+(S0[2]-S1[2])+(S0[3]-S1[3]) |
| Operand type | D0: 32-bit signed integer |
| | S0: 4 * 16-bit unsigned integer |
| | S1: 4 * 16-bit unsigned integer |

| **I2F** | **Integer to Float** |
|---|---|
| Usage | I2F D0, S0 |
| Function | Assumes the S0 value is an integer, converts it into an IEEE754 single-precision floating-point number, then stores the result in D0. The rounding method can be changed by the control register. |
| Behavior model | D0 = float((int)S0); |

| **INT** | **Interrupt** |
|---|---|
| Usage | INT |
| Function | Interrupts current processing in a thread and waits for resuming by host. |
| Behavior model | - |

| **INV** | **INVerse** |
|---|---|
| Usage | INV D0, S0 |
| Function | Treats S0 as a 32-bit array, inverts all bits of this array, and stores the result in D0. |
| Behavior model | D0 = ~S0; |

| **INV64** | **INVerse 64-bit values** |
|---|---|
| Usage | INV64 D0, S0 |
| Function | This is an instruction to invert a 64-bit (8 times 8-bit or 4 times 16-bit) value. |
| Behavior model | D0[n] = !S0[n], n = 0...63; |
| Operand type | D0: 64-bit integer |
| | S0: 64-bit integer |

| **L2F** | **Long to Float** |
|---|---|
| Usage | L2F D0, S0 |
| Function | Convert 64-bit value stored in register pair specified by even register S0 to IEEE754 single floating point format, then store the result to register D0. The rounding method is specified by the control register. |
| Behavior model | D0 = float(S0); |

| **LD** | **Load word** |
|---|---|
| Usage | LD D0, S0, S1 |
| Function | It is a data transfer instruction from memory to the register.<br>The 32-bit data is loaded to register D0 from the 32-bit aligned address specified by the sum of the values S0 and S1*4. S1 is specifying a 32-bit word offset.<br>The value of S0 must be a multiple of 4 (word aligned) otherwise behavior is undefined. |
| Behavior model | D0 = *(S0+S1*4); |

| **LDBS** | **Load byte with signed extension** |
|---|---|
| Usage | LDBS D0, S0, S1 |
| Function | It is a data transfer instruction from memory to the register.<br>The 8-bit data is loaded and signed 32-bit extended to register D0 from the 8-bit aligned address specified by the sum of the values S0 and S1. S1 is specifying a signed byte offset. |
| Behavior model | D0 = (signed int)*(S0+S1); |

| **LDBSI** | **Load byte with signed extension and immediate offset** |
|---|---|
| Usage | LDBSI D0, S0, #IMM |
| Function | It is a data transfer instruction from memory to the register.<br>The 8-bit data is loaded and signed 32-bit extended to register D0 from the 8-bit aligned address specified by the sum of the values S0 and #IMM. The 8-bit signed immediate value is specifying a byte offset. |
| Behavior model | D0 = (signed int)*(S0+#IMM); |

| **LDBU** | **Load byte with unsigned extension** |
|---|---|
| Usage | LDBU D0, S0, S1 |
| Function | It is a data transfer instruction from memory to the register.<br>The 8-bit data is loaded, and unsigned 32-bit extended to register D0 from the 8-bit aligned address specified by the sum of the values S0 and S1. S1 is specifying a signed byte offset. |
| Behavior model | D0 = (unsigned int)*(S0+S1); |

| **LDBUI** | **Load byte with unsigned extension and immediate offset** |
|---|---|
| Usage | LDBUI D0, S0, #IMM |
| Function | It is a data transfer instruction from memory to the register.<br>The 8-bit data is loaded, and unsigned 32-bit extended to register D0 from the 8-bit aligned address specified by the sum of the values S0 and #IMM. The 8-bit signed immediate value is specifying a byte offset. |
| Behavior model | D0 = (unsigned int)*(S0+#IMM); |

| **LDCI** | **Load from Control Register** |
|---|---|
| Usage | LDCI D0, #IMM |
| Function | This is an instruction for loading data from a control register.<br>Writes the data read from the control register specified by #IMM to the general purpose register D0. The value written to D0 is an integer.<br>Specifying #IMM as the number of a control register that does not exist is prohibited. |
| Behavior model | D0 = ControlRegister[#IMM]; |
| Operand type | D0: General purpose register<br>#IMM: 8-bit unsigned integer |

| **LDHS** | **Load half word with signed extension** |
|---|---|
| Usage | LDHS D0, S0, S1 |
| Function | It is a data transfer instruction from memory to the register.<br>The 16-bit data is loaded and signed 32-bit extended to register D0 from the 16-bit aligned address specified by the sum of the values S0 and S1*2. S1 is specifying a 16-bit short offset.<br>The value of S0 must be a multiple of 2 (half word aligned) otherwise behavior is undefined. |
| Behavior model | D0 = (signed int)*(S0+S1*2); |

| **LDHSI** | **Load half word with signed extension and immediate offset** |
|---|---|
| Usage | LDHSI D0, S0, #ASMIMM |
| Function | It is a data transfer instruction from memory to the register.<br>The 16-bit data is loaded and signed 32-bit extended to register D0 from the 16-bit aligned address specified by the sum of the values S0 and #ASMIMM. The 8-bit signed immediate value is specifying a byte offset.<br>The value of S0 must be a multiple of 2 (half word aligned) otherwise behavior is undefined. |
| Behavior model | #IMM = #ASMIMM>>1<br>D0 = (signed int)*(S0+#IMM*2); |

| **LDHU** | **Load half word with unsigned extension** |
|---|---|
| Usage | LDHU D0, S0, S1 |
| Function | It is a data transfer instruction from memory to the register.<br>The 16-bit data is loaded, and unsigned 32-bit extended to register D0 from the 16-bit aligned address specified by the sum of the values S0 and S1*2. S1 is specifying a 16-bit short offset.<br>The value of S0 must be a multiple of 2 (half word aligned) otherwise behavior is undefined. |
| Behavior model | D0 = (unsigned int)*(S0+S1*2); |

| **LDHUI** | **Load half word with unsigned extension and immediate offset** |
|---|---|
| Usage | LDHUI D0, S0, #ASMIMM |
| Function | It is a data transfer instruction from memory to the register.<br>The 16-bit data is loaded, and unsigned 32-bit extended to register D0 from the 16-bit aligned address specified by the sum of the values S0 and #ASMIMM. The 8-bit signed immediate value is specifying a byte offset.<br>The value of S0 must be a multiple of 2 (half word aligned) otherwise behavior is undefined. |
| Behavior model | #IMM = #ASMIMM>>1<br>D0 = (unsigned int)*(S0+#IMM*2); |

| **LDI** | **Load word with immediate offset** |
|---|---|
| Usage | LDI D0, S0, #ASMIMM |
| Function | It is a data transfer instruction from memory to the register.<br>The 32-bit data is loaded to register D0 from the 32-bit aligned address specified by the sum of the values S0 and #ASMIMM. The 8-bit signed immediate value is specifying a byte offset.<br>The value of S0 must be a multiple of 4 (word aligned) otherwise behavior is undefined. |
| Behavior model | #IMM = #ASMIMM>>2<br>D0 = *(S0+#IMM*4); |

| **LDLL** | **Load Long from Local working memory** |
|---|---|
| Usage | LDLL D0, S0, S1 |
| Function | This is an instruction to transfer 64-bit data from local working memory to a 64-bit pair register. This instruction supports the common and non-common address space.<br>Writes the data read from the address specified by adding the S1×8 value to address S0 of local working memory to D0.<br>When this instruction is used for other memory accesses, this instruction transfer 32-bit from the specified memory address into the even register and set the odd register to 0.<br>The 64-bit access to LWM must be aligned, when unaligned, function is prohibited.<br>D0 is a 64-bit pair register, always the lower even register is specified, which holds the lower 32-bit. S0 and S1 are 32-bit registers.<br>The value of S0 must be a multiple of 8 (long word aligned) otherwise behavior is undefined. |
| Behavior model | D0 = *(S0 + S1*8); |
| Operand type | D0: 64-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>S1: 32-bit signed integer |

| **LDLLI** | **Load Long from Local working memory with Immediate offset** |
|---|---|
| Usage | LDLLI D0, S0, #ASMIMM |
| Function | This is an instruction to transfer 64-bit data from local working memory to a 64-bit pair register. This instruction supports the common and non-common address space.<br>Writes the data read from the address specified by adding the #IMM×8 value to address S0 of local working memory to D0.<br>When this instruction is used for other memory accesses, this instruction transfer 32-bit from the specified memory address into the even register and set the odd register to 0.<br>The 64-bit access to LWM must be aligned, when unaligned, function is prohibited.<br>D0 is a 64-bit pair register, always the lower even register is specified, which holds the lower 32-bit. S0 is a 32-bit register.<br>The value of S0 must be a multiple of 8 (long word aligned) otherwise behavior is undefined. |
| Behavior model | #IMM = #ASMIMM>>3<br>D0 = *(S0 + (char)#IMM*8); |
| Operand type | D0: 64-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>#IMM: 8-bit signed integer |

| **LDLRB** | **LoaD Large Register Block** |
|---|---|
| Usage | LDLRB D0, S0, S1 |
| Function | This is an instruction to transfer 512-bit data from local working memory to a 512-bit large register block. This instruction supports only the common address space.<br>Writes the data read from the address specified by adding S1*64 to address S0 of local working memory to D0.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R31) and 32 (for register block R32-R47) |
| Behavior model | D0 = *(S0 + S1*64); |
| Operand type | D0: 512-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>S1: 16-bit signed integer |

| **LDLRBI** | **LoaD Large Register Block with Immediate offset** |
|---|---|
| Usage | LDLRBI D0, S0, #IMM |
| Function | This is an instruction to transfer 512-bit data from local working memory to a 512-bit large register block. This instruction supports only the common address space.<br>Writes the data read from the address specified by adding #IMM*64 to address S0 of local working memory to D0.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R31) and 32 (for register block R32-R47) |
| Behavior model | D0 = *(S0 + #IMM*64); |
| Operand type | D0: 512-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>#IMM: 8-bit signed integer |

| **LDLRBC** | **LoaD Large Register Block with LPC offset** |
|---|---|
| Usage | LDLRBC D0, S0 |
| Function | This is an instruction to transfer 512-bit data from local working memory to a 512-bit large register block. This instruction supports only the common address space.<br>Writes the data read from the address specified by adding LPC (loop counter)*64 to address S0 of local working memory to D0.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R31) and 32 (for register block R32-R47) |
| Behavior model | D0 = *(S0 + LPC*64); |
| Operand type | D0: 512-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>LPC: 9-bit signed integer |

| **LDRCI** | **Load from Relative Coordinate XY with Immediate offset** |
|---|---|
| Usage | LDRCI D0, #IMM1, #IMM2 |
| Function | This is an instruction for data transfer from memory to a register.<br>X and Y coordinates are specified in R0 and R1 respectively. Reads the value from memory at the coordinates obtained by adding the #IMM1 value to X and the #IMM2 value to Y and stores the value in D0.<br>The full 32 bits of R0 and the 16 lower-order bits of R1 are handled as signed integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>The value stored in D0 is an integer. |
| Behavior model | D0 = (int)* (R0+(char)#IMM1, (short)((short)(R1&0xffff)+(char)#IMM2)); |
| Operand type | D0: 32-bit signed integer<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **LDRCIU** | **Load from Relative Coordinate XY with Immediate offset, then source Updated** |
|---|---|
| Usage | LDRCIU D0, #IMM1, #IMM2 |
| Function | This is an instruction for data transfer from memory to a register.<br>X and Y coordinates are specified in R0 and R1 respectively. Reads the value from memory at the coordinates obtained by adding the #IMM1 value to X and the #IMM2 value to Y and stores the value in D0.<br>The full 32 bits of R0 and the 16 lower-order bits of R1 are handled as signed integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers. After operation, X(R0) and Y(R1) are updated by adding #IMM1 and #IMM2 respectively.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>The value stored in D0 is an integer. |
| Behavior model | D0 = (int)* (R0+(char)#IMM1, (short)((short)(R1&0xffff)+(char)#IMM2));<br>R0 = R0+(char)#IMM1; R1 = (short)(R1+(char)#IMM2); |
| Operand type | D0: 32-bit signed integer<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **LDRXY** | **Load from Relative Coordinate XY with register offset** |
|---|---|
| Usage | LDRXY D0, S0, S1 |
| Function | This is an instruction for data transfer from memory to a register.<br>X and Y coordinates are specified in R0 and R1 respectively. Reads the value from memory at the coordinates obtained by adding the S0 value to X and the S1 value to Y and stores the value in D0.<br>The full 32 bits of R0 and S0 and the 16 lower-order bits of R1 and S1 are handled as signed integers.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>The value stored in D0 is an integer. |
| Behavior model | D0 = (int) *(R0+S0, (short)((short)(R1&0xffff)+(short)(S1&0xffff)));<br> |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer<br>S1: 16-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **LDRXYU** | **Load from Relative Coordinate XY with register offset, then source Updated** |
|---|---|
| Usage | LDRXYU D0, S0, S1 |
| Function | This is an instruction for data transfer from memory to a register.<br>X and Y coordinates are specified in R0 and R1 respectively. Reads the value from memory at the coordinates obtained by adding the S0 value to X and the S1 value to Y and stores the value in D0.<br>The full 32 bits of R0 and S0 and the 16 lower-order bits of R1 and S1 are handled as signed integers. After operation, X(R0) and Y(R1) are updated by adding S0 and S1 respectively.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>The value stored in D0 is an integer. |
| Behavior model | D0 = (int) *(R0+S0, (short)((short)(R1&0xffff)+(short)(S1&0xffff)));<br>R0 = R0+S0; R1 = (short)((short)(R1&0xffff)+(short)(S1&0xffff)); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer<br>S1: 16-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **LDSRB** | **LoaD Small Register Block** |
|---|---|
| Usage | LDSRB D0, S0, S1 |
| Function | This is an instruction to transfer 256-bit data from local working memory to a 256-bit large register block. This instruction supports only the common address space.<br>Writes the data read from the address specified by adding S1*32 to address S0 of local working memory to D0.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R23), 24 (for register block R24-R31), 32 (for register block R32-R39) and 40 (for register block R40-R47) |
| Behavior model | D0 = *(S0 + S1*32); |
| Operand type | D0: 256-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>S1: 16-bit signed integer |

| **LDSRBI** | **LoaD Small Register Block with Immediate offset** |
|---|---|
| Usage | LDSRBI D0, S0, #IMM |
| Function | This is an instruction to transfer 256-bit data from local working memory to a 256-bit large register block. This instruction supports only the common address space.<br>Writes the data read from the address specified by adding #IMM*32 to address S0 of local working memory to D0.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R23), 24 (for register block R24-R31), 32 (for register block R32-R39) and 40 (for register block R40-R47) |
| Behavior model | D0 = *(S0 + #IMM*32); |
| Operand type | D0: 256-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>#IMM: 8-bit signed integer |

| **LDSRBC** | **LoaD Small Register Block with LPC offset** |
|---|---|
| Usage | LDSRBC D0, S0 |
| Function | This is an instruction to transfer 256-bit data from local working memory to a 256-bit large register block. This instruction supports only the common address space.<br>Writes the data read from the address specified by adding LPC*32 to address S0 of local working memory to D0.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R23), 24 (for register block R24-R31), 32 (for register block R32-R39) and 40 (for register block R40-R47) |
| Behavior model | D0 = *(S0 + LPC*32); |
| Operand type | D0: 256-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>LPC: 9-bit signed integer |

| **LLROT** | **Long left rotate** |
|---|---|
| Usage | LLROT D0, S0, S1 |
| Function | D0 = (S0 << S1) \| (S0 >> (64-(S1&0x3f))) |
| | S0 is a 64-bit vector and S1 is a 32-bit vector. The instruction is left rotating the 64-bit source S0 by an unsigned value specified in the lower 6bits of source S1. D0 and S0 are 64-bit values stored in 2 consecutive 32-bit registers, which are specified by the even of 2 consecutive 32-bit registers. |
| Behavior model | D0 = (S0 << S1) \| (S0 >> (64-S1)) |

| **LMBD** | **Left Most Bit Detection** |
|---|---|
| Usage | LMBD D0, S0 |
| Function | This is an instruction which searches the first "1" from left side inside S0 and gives back the position in D0. Position counting from left side. Output == 0x80000000 if "1" is not available. Implemented in SAM4. |
| Behavior model | S0[31] == "1"?D0=0: (S0[30] == "1"?D0=1: (S0[29] == "1"?D0=2:( .... S0[1] == "1"?D0=30: (S0[0] == "1"?D0=31:D0=0x80000000))) |
| Operand type | D0: 32bit unsigned integer (6bit used) |
| | S0: 32bit unsigned integer |

| **LPE** | **LooP End** |
|---|---|
| Usage | LPE |
| Function | LPC = LPC - LPC_DEC |
| | (LPC >=0)? PC = LPC_ADD : PC = PC+1 |
| | This is an instruction to subtract the content from the internal register LPC_DEC from LPC and depending on the result, jump back to the address stored in the internal register LPC_ADD (in case LPC is positive) without 1 cycle delay or continue with the instruction flow (1 cycle delay will appear). |
| Behavior model | LPC = LPC - LPC_DEC |
| | (LPC >=0)? PC = LPC_ADD : PC = PC+1 |

| **LPS** | **LooP Start** |
|---|---|
| Usage | LPS S0, #IMM |
| Function | This is an instruction to fast set up a most inner loop with loop counter and loop decrement value. The value specified in S0 is written to the internal loop counter register LPC that can have a 9-bit signed integer. The loop counter decrement LPC_DEC is initialized with the immediate value #IMM. The word address following the LPS instruction is saved into an internal register LPC_ADD. |
| Behavior model | LPC = S0; |
| | LPC_DEC = #IMM; |
| | LPC_ADD = PC+1; |
| Operand type | S0: 8-bit unsigned integer |
| | #IMM: 5-bit unsigned integer |

| **MACS** | **Multiply and ACcumulate** |
|---|---|
| Usage | MACS D0, S0, S1 |
| Function | D0 = D0 + S0 * S1 |
| | S0 and S1 are 2 32-bit signed values and D0 is a 64-bit signed element. The instruction is multiplying the 2 32-bit elements and the result is added to the 64-bit signed element D0. After, the 64-bit are stored back to the register file. D0 is a register pair specified by its even register number. |
| Behavior model | D0 = D0 + S0 * S1 |

| **MACSS** | **Multiply and Accumulate Short** |
|---|---|
| Usage | MACSS D0, S0, S1 |
| Function | D0 = D0 + S0 * S1 |
| | D0, S0 and S1 are 3 32-bit signed values. The instruction is multiplying the 2 32-bit source elements S0 and S1 and the result is added to the 32-bit signed element D0. After, the lower 32-bit are stored back to D0 in the register file. The upper 32bit is getting discarded, no overflow handling is implemented. |
| Behavior model | D0 = D0 + S0 * S1 |

| **MACSSRE** | **Multiply and Accumulate Short with Register Exchange** |
|---|---|
| Usage | MACSS D0, S0, S1 |
| Function | D0 = S0 + D0 * S1 |
| | D0, S0 and S1 are 3 32-bit signed values. The instruction is multiplying D0 times S1 and the result is added to S0. After, the lower 32-bit are stored back to D0 in the register file. The upper 32bit is getting discarded, no overflow handling is implemented. |
| Behavior model | D0 = S0 + D0 * S1 |

| **MAXS** | **MAXimum** |
|---|---|
| Usage | MAXS D0, S0,S1 |
| Function | Selects the larger value between S0 and S1, then stores the result in D0. |
| | S0 and S1 are handled as integers in the operation. |
| Behavior model | D0 = ((int)S0 > (int)S1) ? S0 : S1; |

| **MAXS8** | **MAXimum Signed 8-bit value** |
|---|---|
| Usage | MAXS8 D0, S0, S1 |
| Function | This is an instruction to get the maximum value from 8 times 2 signed 8-bit input values from source S0 and S1. Implemented in SAM4. |
| Behavior model | D0[n] = MAX(S0[n], S1[n]), n = 0...7; |
| Operand type | D0: 8 times 8-bit signed integer |
| | S0: 8 times 8-bit signed integer |
| | S1: 8 times 8-bit signed integer |

| **MAXS16** | **MAXimum Signed 16-bit value** |
|---|---|
| Usage | MAXS16 D0, S0, S1 |
| Function | This is an instruction to get the maximum value from 4 times 2 signed 16-bit input values from source S0 and S1. Implemented in SAM4. |
| Behavior model | D0[n] = MAX(S0[n], S1[n]), n = 0...3; |
| Operand type | D0: 4 times 16-bit signed integer<br>S0: 4 times 16-bit signed integer<br>S1: 4 times 16-bit signed integer |

| **MAXSU** | **MAXimum compared to Uniform** |
|---|---|
| Usage | MAXSU D0, S0,#IMM |
| Function | Selects the larger value between S0 and the uniform value indexed by #IMM, then stores the result in D0.<br>S0 and the uniform value are handled as integers in the operation. |
| Behavior model | D0 = ((int)S0 > (int)Uniform[#R6IMM] ? S0 : (int)Uniform[#R6IMM]; |

| **MAXU8** | **MAXimum Unsigned 8-bit value** |
|---|---|
| Usage | MAXU8 D0, S0, S1 |
| Function | This is an instruction to get the maximum value from 8 times 2 unsigned 8-bit input values from source S0 and S1. Implemented in SAM4. |
| Behavior model | D0[n] = MAX(S0[n], S1[n]), n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer<br>S0: 8 times 8-bit unsigned integer<br>S1: 8 times 8-bit unsigned integer |

| **MAXU16** | **MAXimum Unsigned 16-bit value** |
|---|---|
| Usage | MAXU16 D0, S0, S1 |
| Function | This is an instruction to get the maximum value from 4 times 2 unsigned 16-bit input values from source S0 and S1. Implemented in SAM4. |
| Behavior model | D0[n] = MAX(S0[n], S1[n]), n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 16-bit unsigned integer<br>S1: 4 times 16-bit unsigned integer |

| **MINS** | **MINimum** |
|---|---|
| Usage | MINS D0, S0,S1 |
| Function | Selects the smaller value between S0 and S1, then stores the result in D0.<br>S0 and S1 are handled as integers in the operation. |
| Behavior model | D0 = ((int)S0 > (int)S1) ? S1 : S0; |

| **MINS8** | **MINimum Signed 8-bit value** |
|---|---|
| Usage | MINS8 D0, S0, S1 |
| Function | This is an instruction to get the minimum value from 8 times 2 signed 8-bit input values from source S0 and S1. Implemented in SAM4. |
| Behavior model | D0[n] = MIN(S0[n], S1[n]), n = 0...7; |
| Operand type | D0: 8 times 8-bit signed integer<br>S0: 8 times 8-bit signed integer<br>S1: 8 times 8-bit signed integer |

| **MINS16** | **MINimum Signed 16-bit value** |
|---|---|
| Usage | MINS16 D0, S0, S1 |
| Function | This is an instruction to get the minimum value from 4 times 2 signed 16-bit input values from source S0 and S1. Implemented in SAM4. |
| Behavior model | D0[n] = MIN(S0[n], S1[n]), n = 0...3; |
| Operand type | D0: 4 times 16-bit signed integer<br>S0: 4 times 16-bit signed integer<br>S1: 4 times 16-bit signed integer |

| **MINSU** | **MINimum compared to Uniform** |
|---|---|
| Usage | MINSU D0, S0,#IMM |
| Function | Selects the smaller value between S0 and the uniform value indexed by #IMM, then stores the result in D0.<br>S0 and the uniform value are handled as integers in the operation. |
| Behavior model | D0 = ((int)S0 > (int)Uniform[#R6IMM] ? (int)Uniform[#R6IMM] : S0; |

| **MINU8** | **MINimum Unsigned 8-bit value** |
|---|---|
| Usage | MINU8 D0, S0, S1 |
| Function | This is an instruction to get the minimum value from 8 times 2 unsigned 8-bit input values from source S0 and S1. Implemented in SAM4. |
| Behavior model | D0[n] = MIN(S0[n], S1[n]), n = 0...7; |
| Operand type | D0: 8 times 8-bit unsigned integer<br>S0: 8 times 8-bit unsigned integer<br>S1: 8 times 8-bit unsigned integer |

| **MINU16** | **MINimum Unsigned 16-bit value** |
|---|---|
| Usage | MINU16 D0, S0, S1 |
| Function | This is an instruction to get the minimum value from 4 times 2 unsigned 16-bit input values from source S0 and S1. Implemented in SAM4. |
| Behavior model | D0[n] = MIN(S0[n], S1[n]), n = 0...3; |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 16-bit unsigned integer<br>S1: 4 times 16-bit unsigned integer |

| **MOV** | **MOVe** |
|---|---|
| Usage | MOV D0, S0 |
| Function | This is an instruction for data transfer between registers. Stores the S0 value in D0. |
| Behavior model | D0 = S0; |
| Operand type | D0: -(32-bit array)<br>S0: -(32-bit array) |

| **MOVI** | **MOVe Immediate value** |
|---|---|
| Usage | MOVI D0, #IMM |
| Function | The #IMM value is handled as a 16-bit signed integer, then sign-extended into a 32-bit signed integer, and the result is stored in D0. |
| Behavior model | D0 = (int)(#IMM & 0x0000ffff); |
| Operand type | D0: Signed integer<br>#IMM: 16-bit Signed integer |

| **MOVU** | **MOVe Uniform** |
|---|---|
| Usage | MOVU D0, #IMM |
| Function | Stores the index value in Uniform specified by #IMM in D0. |
| Behavior model | D0 = Uniform[#R6IMM]; |
| Operand type | D0: -(32-bit array)<br>Uniform value: -(32-bit array) |

| **MUL** | **MULtiply** |
|---|---|
| Usage | MUL D0, S0, S1 |
| Function | Multiply the 32-bit signed input values S0 and S1 and store the lower 32-bit of the result to D0. |
| Behavior model | D0 = S0 * S1; |

| **MULS** | **MULtiply with Saturation** |
|---|---|
| Usage | MULS D0, S0, S1 |
| Function | This is an instruction to multiply 2 signed 32-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0 = SAT(S0 * S1, 32); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer<br>S1: 32-bit signed integer |

| **MULSS8** | **MULtiply with Saturation Signed 8-bit values** |
|---|---|
| Usage | MULSS8 D0, S0, S1 |
| Function | This is an instruction to multiply 8 times 2 signed 8-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] * S1[n], 8), n = 0...7; |
| Operand type | D0: 8 times 8-bit signed integer<br>S0: 8 times 8-bit signed integer<br>S1: 8 times 8-bit signed integer |

| **MULSS16** | **MULtiply with Saturation Signed 16-bit values** |
|---|---|
| Usage | MULSS16 D0, S0, S1 |
| Function | This is an instruction to multiply 4 times 2 signed 16-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] * S1[n], 16), n = 0...3; |
| Operand type | D0: 4 times 16-bit signed integer<br>S0: 4 times 16-bit signed integer<br>S1: 4 times 16-bit signed integer |

| **MULSU** | **MULtiply Saturation with Uniform** |
|---|---|
| Usage | MULSU D0, S0, #IMM |
| Function | This is an instruction to multiply 2 signed 32-bit values from source S0 and Uniform storage indexed by #IMM with saturation. |
| Behavior model | D0 = SAT(S0 * Uniform[#R6IMM], 32); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer |

| **MULLS** | **MULtiply long signed** |
|---|---|
| Usage | MULLS D0, S0, S1 |
| Function | Multiply the 32-bit signed input values S0 and S1 and store the 64-bit result to D0. |
| Behavior model | D0 = S0 * S1; |

| **MULLS8** | **MULtiply Long Signed 8-bit values** |
|---|---|
| Usage | MULLS8 D0, S0, S1 |
| Function | This is an instruction to multiply 4 times 2 signed 8-bit values from source S0 and source S1 with storage of 16-bit result. |
| Behavior model | D0[n] = S0[n] * S1[n], n = 0...3; |
| Operand type | D0: 4 times 16-bit signed integer<br>S0: 4 times 8-bit signed integer<br>S1: 4 times 8-bit signed integer |

| **MULLS16** | **MULtiply Long Signed 16-bit values** |
|---|---|
| Usage | MULLS16 D0, S0, S1 |
| Function | This is an instruction to multiply 2 times 2 signed 16-bit values from source S0 and source S1 with storage of 32-bit result. |
| Behavior model | D0[n] = S0[n] * S1[n], n = 0...1; |
| Operand type | D0: 2 times 32-bit signed integer<br>S0: 2 times 16-bit signed integer<br>S1: 2 times 16-bit signed integer |

| **MULLSU** | **MULtiply long signed by Uniform** |
|---|---|
| Usage | MULLSU D0, S0, #IMM |
| Function | Multiply the 32-bit signed input values S0 with the 32-bit signed input value read from uniform storage indexed by #IMM and store the 64-bit result to D0. |
| Behavior model | D0 = S0 * Uniform[#R6IMM]; |

| **MULLU8** | **MULtiply Long Unsigned 8-bit values** |
|---|---|
| Usage | MULLU8 D0, S0, S1 |
| Function | This is an instruction to multiply 4 times 2 unsigned 8-bit values from source S0 and source S1 with storage of 16-bit result. |
| Behavior model | D0[n] = S0[n] * S1[n], n = 0..3; |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 8-bit unsigned integer<br>S1: 4 times 8-bit unsigned integer |

| **MULLU16** | **MULtiply Long Unsigned 16-bit values** |
|---|---|
| Usage | MULLU16 D0, S0, S1 |
| Function | This is an instruction to multiply 2 times 2 unsigned 16-bit values from source S0 and source S1 with storage of 32-bit result. |
| Behavior model | D0[n] = S0[n] * S1[n], n = 0..1; |
| Operand type | D0: 2 times 32-bit unsigned integer<br>S0: 2 times 16-bit unsigned integer<br>S1: 2 times 16-bit unsigned integer |

| **MULU** | **MULtiply by Uniform** |
|---|---|
| Usage | MULU D0, S0, #IMM |
| Function | Multiply the 32-bit signed input value S0 with the 32-bit signed input value read from uniform storage indexed by #IMM and store the lower 32-bit of the result to D0. |
| Behavior model | D0 = S0 * Uniform[#R6IMM]; |

| **MULUS8** | **MULtiply Unsigned with Saturation 8-bit values** |
|---|---|
| Usage | MULUS8 D0, S0, S1 |
| Function | This is an instruction to multiply 8 times 2 unsigned 8-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] * S1[n], 8), n = 0..7; |
| Operand type | D0: 8 times 8-bit unsigned integer<br>S0: 8 times 8-bit unsigned integer<br>S1: 8 times 8-bit unsigned integer |

| **MULUS16** | **MULtiply Unsigned with Saturation 16-bit values** |
|---|---|
| Usage | MULUS16 D0, S0, S1 |
| Function | This is an instruction to multiply 4 times 2 unsigned 16-bit values from source S0 and source S1 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] * S1[n], 16), n = 0..3; |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 16-bit unsigned integer<br>S1: 4 times 16-bit unsigned integer |

| **NCSCS** | **Non-symmetric Census Signature Calculation Step** |
|---|---|
| Usage | NCSCS D0, S0, S1 |
| Function | S0 is compared against S1 and -S1 as signed values. Each comparison is adding a 1 or 0 depending on whether the result is true (==1) or false (==0) to the earlier by 2 bits to the left shifted D0. The result is again stored to D0. |
| Behavior model | D0 = (D0 << 2) + (S0 > S1) + (S0 > -S1) |

| **NOP** | **No OPeration** |
|---|---|
| Usage | NOP |
| Function | No operation is executed. One cycle of bubble is inserted in the thread operation pipeline. |
| Behavior model | ; // empty statement |

| **OR** | **OR** |
|---|---|
| Usage | OR D0, S0, S1 |
| Function | Treats S0 and S1 as 32-bit arrays, takes the logical OR of S0 and S1, and stores the result in D0. |
| Behavior model | D0 = S0 \| S1; |

| **OR64** | **OR 64-bit values** |
|---|---|
| Usage | OR64 D0, S0, S1 |
| Function | This is an instruction to OR 2 64-bit (8 times 8-bit or 4 times 16-bit) values. |
| Behavior model | D0[n] = S0[n] OR S1[n], n = 0...63; |
| Operand type | D0: 64-bit integer<br>S0: 64-bit integer<br>S1: 64-bit integer |

| **ORU** | **OR with Uniform** |
|---|---|
| Usage | ORU D0, S0, #IMM |
| Function | Treats S0 and the uniform value indexed by #IMM as 32-bit arrays, takes the logical OR of S0 and the uniform value, and stores the result in D0. |
| Behavior model | D0 = S0 \| Uniform[#R6IMM]; |

| **PACKB0** | **PACK byte 0** |
|---|---|
| Usage | PACKB0 D0, S0, S1 |
| Function | D0 = (S0[39:32] << 24) \| (S0[7:0] << 16) \| (S1[39:32] << 8) \| (S1[7:0]) |
| | S0 and S1 are 64-bit vectors with each 2 32-bit elements, which are specified by the even of 2 consecutive 32-bit registers. The instruction is taking from each of the four 32-bit elements the lowest 8-bit part and packing them together in D0. D0 is handled as a 32-bit signed value. |
| Behavior model | D0 = (S0[39:32] << 24) \| (S0[7:0] << 16) \| (S1[39:32] << 8) \| (S1[7:0]) |

| **PACKB1** | **PACK byte 1** |
|---|---|
| Usage | PACKB1 D0, S0, S1 |
| Function | D0 = (S0[47:40] << 24) \| (S0[15:8] << 16) \| (S1[47:40] << 8) \| (S1[15:8]) |
| | S0 and S1 are 64-bit vectors with each 2 32-bit elements, which are specified by the even of 2 consecutive 32-bit registers. The instruction is taking from each of the four 32-bit elements the second lowest 8-bit part and packing them together in D0. D0 is handled as a 32-bit signed value. |
| Behavior model | D0 = (S0[47:40] << 24) \| (S0[15:8] << 16) \| (S1[47:40] << 8) \| (S1[15:8]) |

| **PACKB2** | **PACK byte 2** |
|---|---|
| Usage | PACKB2 D0, S0, S1 |
| Function | D0 = (S0[55:48] << 24) \| (S0[23:16] << 16) \| (S1[55:48] << 8) \| (S1[23:16]) |
| | S0 and S1 are 64-bit vectors with each 2 32-bit elements, which are specified by the even of 2 consecutive 32-bit registers. The instruction is taking from each of the four 32-bit elements the second highest 8-bit part and packing them together in D0. D0 is handled as a 32-bit signed value. |
| Behavior model | D0 = (S0[55:48] << 24) \| (S0[23:16] << 16) \| (S1[55:48] << 8) \| (S1[23:16]) |

| **PACKB3** | **PACK byte 3** |
|---|---|
| Usage | PACKB3 D0, S0, S1 |
| Function | D0 = (S0[63:56] << 24) \| (S0[31:24] << 16) \| (S1[63:56] << 8) \| (S1[31:24]) |
| | S0 and S1 are 64-bit vectors with each 2 32-bit elements, which are specified by the even of 2 consecutive 32-bit registers. The instruction is taking from each of the four 32-bit elements the highest 8-bit part and packing them together in D0. D0 is handled as a 32-bit signed value. |
| Behavior model | D0 = (S0[63:56] << 24) \| (S0[31:24] << 16) \| (S1[63:56] << 8) \| (S1[31:24]) |

| **PACKH0** | **PACK half word 0** |
|---|---|
| Usage | PACKH0 D0, S0, S1 |
| Function | D0 = (S0[15:0] << 16) \| (S1[15:0]) |
| | S0 and S1 are 32-bit vectors with each 1 32-bit element. The instruction is taking from each 32-bit element the lowest 16-bit part and packing them together in D0. D0 is handled as a 32-bit signed value. |
| Behavior model | D0 = (S0[15:0] << 16) \| (S1[15:0]) |

| **PACKH1** | **PACK half word 1** |
|---|---|
| Usage | PACKH1 D0, S0, S1 |
| Function | D0 = (S0[31:16] << 16) \| (S1[31:16])<br>S0 and S1 are 32-bit vectors with each 1 32-bit element. The instruction is taking from each 32-bit element the highest 16-bit part and packing them together in D0. D0 is handled as a 32-bit signed value. |
| Behavior model | D0 = (S0[31:16] << 16) \| (S1[31:16]) |

| **PCMPAGT** | **Parallel CoMPare Anded Greater Than** |
|---|---|
| Usage | PCMPAGT D0, S0, S1 |
| Function | This instruction compares (greater) in parallel the upper and lower halves of S0 and S1 and writes the ANDed comparison result to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | SR.F = (S0[31:0] > S1[31:0]) && (S0[63:32] > S1[63:32]); (D0 == R3)<br>D0 = (S0[31:0] > S1[31:0]) && (S0[63:32] > S1[63:32]); (D0 != R3) |
| Operand type | D0: 1bit integer<br>S0: 2 32-bit signed integer<br>S1: 2 32-bit signed integer |

| **PCMPALT** | **Parallel CoMPare Anded Less Than** |
|---|---|
| Usage | PCMPALT D0, S0, S1 |
| Function | This instruction compares (smaller) in parallel the upper and lower halves of S0 and S1 and writes the ANDed comparison result to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | SR.F = (S0[31:0] < S1[31:0]) && (S0[63:32] < S1[63:32]) ; (D0 == R3)<br>D0 = (S0[31:0] < S1[31:0]) && (S0[63:32] < S1[63:32]) ; (D0 != R3) |
| Operand type | D0: 1bit integer<br>S0: 2 32-bit signed integer<br>S1: 2 32-bit signed integer |

| **PCMPAEQ** | **Parallel CoMPare Anded EQual** |
|---|---|
| Usage | PCMPAEQ D0, S0, S1 |
| Function | This instruction compares (equal) in parallel the upper and lower halves of S0 and S1 and writes the ANDed comparison result to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | SR.F = (S0[31:0] == S1[31:0]) && (S0[63:32] == S1[63:32]) ; (D0 == R3)<br>D0 = (S0[31:0] == S1[31:0]) && (S0[63:32] == S1[63:32]) ; (D0 != R3) |
| Operand type | D0: 1bit integer<br>S0: 2 32-bit signed integer<br>S1: 2 32-bit signed integer |

| **PCMPOGT** | **Parallel CoMPare Ored Greater Than** |
|---|---|
| Usage | PCMPOGT D0, S0, S1 |
| Function | This instruction compares (greater) in parallel the upper and lower halves of S0 and S1 and writes the ORed comparison result to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | SR.F = (S0[31:0] > S1[31:0]) \|\| (S0[63:32] > S1[63:32]) ; (D0 == R3)<br>D0 = (S0[31:0] > S1[31:0]) \|\| (S0[63:32] > S1[63:32]) ; (D0 != R3) |
| Operand type | D0: 1bit integer<br>S0: 2 32-bit signed integer<br>S1: 2 32-bit signed integer |

| **PCMPOLT** | **Parallel CoMPare Ored Less Than** |
|---|---|
| Usage | PCMPOLT D0, S0, S1 |
| Function | This instruction compares (smaller) in parallel the upper and lower halves of S0 and S1 and writes the ORed comparison result to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | SR.F = (S0[31:0] < S1[31:0]) \|\| (S0[63:32] < S1[63:32]) ; (D0 == R3)<br>D0 = (S0[31:0] < S1[31:0]) \|\| (S0[63:32] < S1[63:32]) ; (D0 != R3) |
| Operand type | D0: 1bit integer<br>S0: 2 32-bit signed integer<br>S1: 2 32-bit signed integer |

| **PCMPOEQ** | **Parallel CoMPare Ored EQual** |
|---|---|
| Usage | PCMPOEQ D0, S0, S1 |
| Function | This instruction compares (equal) in parallel the upper and lower halves of S0 and S1 and writes the ORed comparison result to D0, in case D0 is R3, SR is set with this bit. Mapped on each Thread. |
| Behavior model | SR.F = (S0[31:0] == S1[31:0]) \|\| (S0[63:32] == S1[63:32]); (D0 == R3)<br>D0 = (S0[31:0] == S1[31:0]) \|\| (S0[63:32] == S1[63:32]); (D0 != R3) |
| Operand type | D0: 1bit integer<br>S0: 2 32-bit signed integer<br>S1: 2 32-bit signed integer |

| **PLSB8** | **Pack the LSB's of 4 8-bit values into the 4 lowest bits of dst** |
|---|---|
| Usage | PLSB8 D0, S0, S1 |
| Function | This is an instruction to take the LSB's of 4 8-bit values in SRC0 and concatenate them to the 4-bit left shifted value of SRC1 and store the result to D0. Implemented in each thread. |
| Behavior model | D0[31:4] = S1[27:0]; D0[0] = S0[0]; D0[1] = S0[8]; D0[2] = S0[16]; D0[3] = S0[24]; |
| Operand type | D0: 32-bit unsigned integer<br>S0: 32-bit unsigned integer<br>S1: 32-bit unsigned integer |

| **PLSB16** | **Pack the LSB's of 2 16-bit values into the 2 lowest bits of dst** |
|---|---|
| Usage | PLSB16 D0, S0, S1 |
| Function | This is an instruction to take the LSB's of 2 16-bit values in SRC0 and concatenate them to the 2-bit left shifted value of SRC1 and store the result to D0. Implemented in each thread. |
| Behavior model | D0[31:2] = S1[29:0]; D0[0] = S0[0]; D0[1] = S0[16]; |
| Operand type | D0: 32-bit unsigned integer<br>S0: 32-bit unsigned integer<br>S1: 32-bit unsigned integer |

| **RETA** | **RETurn from subroutine Alternative** |
|---|---|
| Usage | RETA D0 |
| Function | Returns to the PC stored in the temporary-storage PC register specified by D0.<br>The register to be specified by D0 is not the general-purpose register number, but the CALLA/RETA dedicated temporary-storage PC register number.<br>There are eight registers for this purpose and 0 to 7 can be specified.<br>When a number not written by the CALLA instruction is specified, operation is undefined.<br>The 3 lower-order bits of D0 are handled as an unsigned integer. |
| Behavior model | PC=PCR[(D0&0x7)] |
| Operand type | D0: 3-bit unsigned integer |

| **RMBD** | **Right Most Bit Detection** |
|---|---|
| Usage | RMBD D0, S0 |
| Function | This is an instruction which searches the first "1" from right side inside S0 and gives back the position in D0. Position counting from right side. Output == 0x80000000 if "1" is not available. Implemented in SAM4. |
| Behavior model | S0[0] == "1"?D0=0: (S0[1] == "1"?D0=1: (S0[2] == "1"?D0=2: (.... S0[30] == "1"?D0=30: (S0[31] == "1"?D0=31:D0=0x80000000))))) |
| Operand type | D0: 32bit unsigned integer (6bit used)<br>S0: 32bit unsigned integer |

| **RXCH** | **Register eXCHange** |
|---|---|
| Usage | RXCH D0, S0 |
| Function | This instruction exchanges the content of 2 registers of a register pair. Mapped on each Thread. |
| Behavior model | D0[31:0] = S0[63:32]; D0[63:32] = S0[31:0]; |
| Operand type | D0: 2 32-bit signed integer<br>S0: 2 32-bit signed integer |

| **SBOCRD** | **SBO of CooRDinate** |
|---|---|
| Usage | SBOCRD D0, #IMM1, #IMM2 |
| Function | This is an instruction for outputting the X coordinate, Y coordinate, and D0 (data) to memory via the SBO port.<br>The X and Y coordinates to be output are X and Y coordinates which are specified in R0 and R1 respectively, with the #IMM1 value added to X and the #IMM2 value added to Y.<br>The 16 lower-order bits of R0 and R1 are handled as integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers.<br>The output mask can be specified by control register CR10. Refer to section U32.6.2 .<br>The X and Y data to be output can be packed or expanded by configuration register SBOCR0. Refer to section U32.12.1.34 . |
| Behavior model | SBO((ushort((ushort)(R0&0xffff)+(char)#IMM1), (ushort)((ushort)(R1&0xffff)+#IMM2), D0); |
| Operand type | D0: -(32-bit array)<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 16-bit signed integer<br>R1: 16-bit signed integer |

| **SCSCS** | **Symmetric Census Signature Calculation Step** |
|---|---|
| Usage | SCSCS D0, S0, S1 |
| Function | S0 is compared against S1 and INV(S1) as signed values. Each comparison is adding a 1 or 0 depending on whether the result is true (==1) or false (==0) to the earlier by 2 bits to the left shifted D0. The result is again stored to D0. |
| Behavior model | D0 = (D0 << 2) + (S0 > S1) + (S0 > INV(S1)) |

| **SETHI** | **Set HIgher bits** |
|---|---|
| Usage | SETHI D0, #ASMIMM |
| Function | Higher 16 bits of #ASMIMM replaces the 16 higher-order bits of D0, and the result is retained by D0. |
| Behavior model | #IMM = (#ASMIMM & 0xffff0000)>>16<br>D0 = (D0 & 0x0000ffff) \| #IMM<<16; |
| Operand type | D0: -(32-bit array)<br>#ASMIMM: -(32-bit array)<br>#IMM: -(16-bit array) |

| **SHA8** | **SHift Arithmetic left of signed 8-bit values** |
|---|---|
| Usage | SHA8 D0, S0, S1 |
| Function | This is an instruction to arithmetic shift 8 signed 8-bit values by 8 signed 4-bit values stored in S1 as 8 signed 8-bit values. Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] <<< S1[n], n = 0...7; (S1[n]>0)<br>D0[n] = S0[n] >>> S1[n], n = 0...7; (S1[n]<0)<br>D0[n] = S0[n], n = 0...7; (S1[n]==0) |
| operand type | D0: 8 times 8-bit signed integer<br>S0: 8 times 8-bit signed integer<br>S1: 8 times 8-bit signed integer, only lower 4 bits are used |

| **SHA16** | **SHift Arithmetic of signed 16-bit values** |
|---|---|
| Usage | SHA16 D0, S0, S1 |
| Function | This is an instruction to arithmetic shift 4 signed 16-bit values by 4 signed 5-bit values stored in S1 as 4 signed 16-bit values. Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] <<< S1[n], n = 0...3; (S1[n]>0)<br>D0[n] = S0[n] >>> S1[n], n = 0...3; (S1[n]<0)<br>D0[n] = S0[n], n = 0...3; (S1[n]==0) |
| Operand type | D0: 4 times 16-bit signed integer<br>S0: 4 times 16-bit signed integer<br>S1: 4 times 16-bit signed integer, only lower 5 bits are used |

| **SHAI8** | **SHift Arithmetic by signed Immediate value signed 8-bit values** |
|---|---|
| Usage | SHAI8 D0, S0, #IMM |
| Function | This is an instruction to shift 8 signed 8-bit values by 1 signed 4-bit value stored as 1 signed 8-bit immediate value. Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] <<< #IMM, n = 0...7; (#IMM>0)<br>D0[n] = S0[n] >>> #IMM, n = 0...7; (#IMM<0)<br>D0[n] = S0[n], n = 0...7; (#IMM==0) |
| Operand type | D0: 8 times 8-bit signed integer<br>S0: 8 times 8-bit signed integer<br>#IMM: 1 8-bit signed integer, only lower 4 bits are used |

| **SHAI16** | **SHift Arithmetic by signed Immediate value signed 16-bit values** |
|---|---|
| Usage | SHAI16 D0, S0, #IMM |
| Function | This is an instruction to shift 4 signed 16-bit values by 1 signed 5-bit value stored as 1 signed 8-bit immediate value. Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] <<< #IMM, n = 0...3; (#IMM>0)<br>D0[n] = S0[n] >>> #IMM, n = 0...3; (#IMM<0)<br>D0[n] = S0[n] , n = 0...3; (#IMM==0) |
| Operand type | D0: 4 times 16-bit signed integer<br>S0: 4 times 16-bit signed integer<br>#IMM: 1 8-bit signed integer, only lower 5 bits are used |

| **SHL** | **SHift Left** |
|---|---|
| Usage | SHL D0, S0, S1 |
| Function | Shifts S0 to the left by the number of bits given by the value of S1 and stores the result in D0. The lower 5bits of S1 are used to decide shift size. |
| Behavior model | D0 = S0 << (S1 & 0x1f); |

| **SHL8** | **SHift Logical of unsigned 8-bit values** |
|---|---|
| Usage | SHL8 D0, S0, S1 |
| Function | This is an instruction to shift 8 unsigned 8-bit values by 8 signed 4-bit values stored in S1 as 8 signed 8-bit values. Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] << S1[n] , n = 0...7; (S1[n]>0)<br>D0[n] = S0[n] >> S1[n] , n = 0...7; (S1[n]<0)<br>D0[n] = S0[n] , n = 0...7; (S1[n]==0) |
| Operand type | D0: 8 times 8-bit unsigned integer<br>S0: 8 times 8-bit unsigned integer<br>S1: 8 times 8-bit signed integer, only lower 4 bits are used |

| **SHL16** | **SHift Logical of unsigned 16-bit values** |
|---|---|
| Usage | SHL16 D0, S0, S1 |
| Function | This is an instruction to shift 4 unsigned 16-bit values by 4 signed 5-bit values stored in S1 as 4 signed 16-bit values. Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] << S1[n] , n = 0...3; (S1[n]>0)<br>D0[n] = S0[n] >> S1[n] , n = 0...3; (S1[n]<0)<br>D0[n] = S0[n] , n = 0...3; (S1[n]==0) |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 16-bit unsigned integer<br>S1: 4 times 16-bit signed integer, only lower 5 bits are used |

| **SHLI** | **SHift Left with Immediate value** |
|---|---|
| Usage | SHLI D0, S0, #IMM |
| Function | Shifts S0 to the left by the number of bits given by the value of #IMM and stores the result in D0. The lower 5bits of #IMM are used to decide shift size. |
| Behavior model | D0 = S0 << (#IMM & 0x1f); |

| **SHLI8** | **SHift Logical by signed Immediate value unsigned 8-bit values** |
|---|---|
| Usage | SHLI8 D0, S0, #IMM |
| Function | This is an instruction to shift 8 unsigned 8-bit values by 1 signed 4-bit value stored as 1 signed 8-bit immediate value (sign bit expanded). Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] << #IMM, n = 0...7; (#IMM>0)<br>D0[n] = S0[n] >> #IMM, n = 0...7; (#IMM<0)<br>D0[n] = S0[n], n = 0...7; (#IMM==0) |
| Operand type | D0: 8 times 8-bit unsigned integer<br>S0: 8 times 8-bit unsigned integer<br>#IMM: 1 8-bit signed integer, only lower 4 bits are used |

| **SHLI16** | **SHift Logical by signed Immediate value unsigned 16-bit values** |
|---|---|
| Usage | SHLI16 D0, S0, #IMM |
| Function | This is an instruction to shift 4 unsigned 16-bit values by 1 signed 5-bit value stored as 1 signed 8-bit immediate value (sign bit expanded). Implemented in SAM4. |
| Behavior model | D0[n] = S0[n] << #IMM, n = 0...3; (#IMM>0)<br>D0[n] = S0[n] >> #IMM, n = 0...3; (#IMM<0)<br>D0[n] = S0[n] , n = 0...3; (#IMM==0) |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 16-bit unsigned integer<br>#IMM: 1 8-bit signed integer, only lower 5 bits are used |

| **SHRA** | **SHift Right Arithmetically** |
|---|---|
| Usage | SHRA D0, S0, S1 |
| Function | Treats S0 as a 32-bit signed integer, arithmetically shifts S0 to the right by the number of bits given by the value of S1 and stores the result in D0. The lower 5bits of S1 are used to decide sift size. |
| Behavior model | D0 = (signed int)S0 >> (S1 & 0x1f); |

| **SHRAI** | **SHift Right Arithmetically with Immediate value** |
|---|---|
| Usage | SHRAI D0, S0, #IMM |
| Function | Treats S0 as a 32-bit signed integer, arithmetically shifts S0 to the right by the number of bits given by the value of #IMM and stores the result in D0. The lower 5bits of #IMM are used to decide shift size. |
| Behavior model | D0 = (signed int)S0 >> (#IMM & 0x1f); |

| **SHRL** | **SHift Right Logically** |
|---|---|
| Usage | SHRL D0, S0, S1 |
| Function | Treats S0 as a 32-bit unsigned integer, logically shifts S0 to the right by the number of bits given by the value of S1 and stores the result in D0. The lower 5bits of S1 are used to decide sift size. |
| Behavior model | D0 = (unsigned int)S0 >> (S1 & 0x1f); |

| **SHRLI** | **SHift Right Logically with Immediate value** |
|---|---|
| Usage | SHRLI D0, S0, #IMM |
| Function | Treats S0 as a 32-bit unsigned integer, logically shifts S0 to the right by the number of bits given by the value of #IMM and stores the result in D0. The lower 5bits of #IMM are used to decide shift size. |
| Behavior model | D0 = (unsigned int)S0 >> (#IMM & 0x1f); |

| **SHUF** | **SHUFfle** |
|---|---|
| Usage | SHUF D0, S0, S1 |
| Function | This shuffle instruction is shuffling the data in S0 based on the given parameters in S1 and overwrites dependent on the parameter setting in S1 the whole or parts of the destination register. The instruction can be used to set specific bytes of an 8-byte sequence to zero, to shuffle bytes inside an 8-byte sequence, to reverse bits inside the bytes of an 8-byte sequence or to combine the operations mentioned before. By combining the operations, the user can achieve e.g., bit reversal operations in 32-bit values. |
| Behavior model | if (S1[n*8+3] == 1) then D0[(n+1)*8-1:n*8] = D0[(n+1)*8-1:n*8]<br>else if (S1[n*8+4] == 1) then D0[(n+1)*8-1:n*8] = 8'b0<br>else if (S1[n*8+5] == 1) then D0[(n+1)*8-1:n*8] = S0[S1[n*8+2:n*8]*8 : (S1[n*8+2:n*8]+1)*8-1]<br>else D0[(n+1)*8-1:n*8] = S0[ (S1[n*8+2:n*8]+1)*8-1 : S1[n*8+2:n*8]*8] |
| Operand type | D0: 64bit unsigned integer<br>S0: 64bit unsigned integer<br>S1: 64bit unsigned integer |

| **ST** | **Store word** |
|---|---|
| Usage | ST D0, S0, S1 |
| Function | It is a data transfer instruction from register to memory.<br>The 32-bit data from register D0 is stored to the 32-bit aligned address specified by the sum of the values S0 and S1*4. S1 is specifying a 32-bit word offset.<br>The value of S0 must be a multiple of 4 (word aligned) otherwise behavior is undefined. |
| Behavior model | *(S0+S1*4) = D0; |

| **STB** | **Store byte** |
|---|---|
| Usage | STB D0, S0, S1 |
| Function | It is a data transfer instruction from register to memory.<br>The lower 8-bit data from register D0 is stored to the 8-bit aligned address specified by the sum of the values S0 and S1. S1 is specifying a byte offset. |
| Behavior model | *(S0+S1) = D0; |

| **STBI** | **Store byte with immediate offset** |
|---|---|
| Usage | STBI D0, S0, #IMM |
| Function | It is a data transfer instruction from register to memory.<br>The lower 8-bit data from register D0 is stored to the 8-bit aligned address specified by the sum of the values S0 and #IMM. The 8-bit signed immediate value is specifying a byte offset. |
| Behavior model | *(S0+#IMM) = D0; |

| **STC** | **STore to Control Register** |
|---|---|
| Usage | STC D0, S0 |
| Function | This is an instruction for storing data in a control register.<br>Writes the lower 16 bits of the data in the general purpose register S0 value to the control register D0. Specifying D0 as a control register that does not exist is prohibited. |
| Behavior model | ControlRegisterD0 = S0&0xffff; |
| Operand type | D0: Control Register.<br>S0: 16-bit array |

| **STCI** | **STore Immediate to Control Register** |
|---|---|
| Usage | STCI D0 , #IMM |
| Function | This is an instruction for storing data in a control register.<br>Writes the #IMM value to the control register D0. Specifying D0 as a control register that does not exist is prohibited. |
| Behavior model | ControlRegisterD0 = #IMM; |
| Operand type | D0: Control Register<br>#IMM: 16-bit array |

| **STH** | **Store half word** |
|---|---|
| Usage | STH D0, S0, S1 |
| Function | It is a data transfer instruction from register to memory.<br>The lower 16-bit data from register D0 is stored to the 16-bit aligned address specified by the sum of the values S0 and S1*2. S1 is specifying a 16-bit short offset.<br>The value of S0 must be a multiple of 2 (half word aligned) otherwise behavior is undefined. |
| Behavior model | *(S0+S1*2) = D0; |

| **STHI** | **Store half word with immediate offset** |
|---|---|
| Usage | STHI D0, S0, #ASMIMM |
| Function | It is a data transfer instruction from register to memory.<br>The lower 16-bit data from register D0 is stored to the 16-bit aligned address specified by the sum of the values S0 and #ASMIMM. The 8-bit signed immediate value is specifying a byte offset.<br>The value of S0 must be a multiple of 2 (half word aligned) otherwise behavior is undefined. |
| Behavior model | #IMM = #ASMIMM>>1<br>*(S0+#IMM*2) = D0; |

| **STI** | **Store word with immediate offset** |
|---|---|
| Usage | STI D0, S0, #ASMIMM |
| Function | It is a data transfer instruction from register to memory.<br>The 32-bit data from register D0 is stored to the 32-bit aligned address specified by the sum of the values S0 and #ASMIMM. The 8-bit signed immediate value is specifying a byte offset.<br>The value of S0 must be a multiple of 4 (word aligned) otherwise behavior is undefined. |
| Behavior model | #IMM = #ASMIMM>>2<br>*(S0+#IMM*4) = D0; |

| **STLL** | **STore Long to Local working memory** |
|---|---|
| Usage | STLL D0, S0, S1 |
| Function | This is an instruction to transfer 64-bit data from a 64-bit pair register to local working memory. This instruction supports the common and non-common address space.<br>Writes the D0 data to the address specified by adding the S1×8 value to address S0 of local working memory.<br>When this instruction is used for other memory accesses, this instruction transfer 32-bit from the specified even register to the specified memory address.<br>The 64-bit access to LWM must be aligned, when unaligned, function is prohibited.<br>D0 is a 64-bit pair register, always the lower even register is specified, which holds the lower 32-bit. S0 and S1 are 32-bit registers.<br>The value of S0 must be a multiple of 8 (long word aligned) otherwise behavior is undefined. |
| Behavior model | *(S0 + S1*8) = D0; |
| Operand type | D0: 64-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>S1: 32-bit signed integer |

| **STLLI** | **STore Long to Local working memory with Immediate offset** |
|---|---|
| Usage | STLLI D0, S0, #ASMIMM |
| Function | This is an instruction to transfer 64-bit data from a 64-bit pair register to local working memory. This instruction supports the common and non-common address space.<br>Writes the D0 data to the address specified by adding the #IMM×8 value to address S0 of local working memory.<br>When this instruction is used for other memory accesses, this instruction transfer 32-bit from the specified even register to the specified memory address.<br>The 64-bit access to LWM must be aligned, when unaligned, function is prohibited.<br>D0 is a 64-bit pair register, always the lower even register is specified, which holds the lower 32-bit. S0 is a 32-bit register.<br>The value of S0 must be a multiple of 8 (long word aligned) otherwise behavior is undefined. |
| Behavior model | #IMM = #ASMIMM>>3<br>*(S0 + (char)#IMM*8) = D0; |
| Operand type | D0: 64-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>#IMM: 8-bit signed integer |

| **STLRB** | **STore Large Register Block** |
|---|---|
| Usage | STLRB D0, S0, S1 |
| Function | This is an instruction to transfer 512-bit data from 512-bit large register block to local working memory. This instruction supports only the common address space.<br>Writes the data read from D0 to the address specified by adding S1*64 to address S0 of local working memory.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R31) and 32 (for register block R32-R47) |
| Behavior model | *(S0 + S1*64) = D0; |
| Operand type | D0: 512-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>S1: 16-bit signed integer |

**STore Large Register Block with Immediate offset**

**STLRBI**

| | |
|---|---|
| Usage | STLRBI D0, S0, #IMM |
| Function | This is an instruction to transfer 512-bit data from 512-bit large register block to local working memory. This instruction supports only the common address space.<br>Writes the data read from D0 to the address specified by adding #IMM*64 to address S0 of local working memory.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R31) and 32 (for register block R32-R47) |
| Behavior model | *(S0 +#IMM*64) = D0; |
| Operand type | D0: 512-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>#IMM: 8-bit signed integer |

**STLRBC** **STore Large Register Block with LPC offset**

| | |
|---|---|
| Usage | STLRBC D0, S0 |
| Function | This is an instruction to transfer 512-bit data from 512-bit large register block to local working memory. This instruction supports only the common address space.<br>Writes the data read from D0 to the address specified by adding LPC*64 to address S0 of local working memory.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R31) and 32 (for register block R32-R47) |
| Behavior model | *(S0 + LPC*64) = D0; |
| Operand type | D0: 512-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>LPC: 9-bit signed integer |

**STRCI** **STore to Relative Coordinate with Immediate offset**

| | |
|---|---|
| Usage | STRCI D0, #IMM1, #IMM2 |
| Function | This is an instruction for data transfer from a register to memory.<br>Stores the D0 value in memory at X and Y coordinates which are specified in R0 and R1 respectively, with the #IMM1 value added to X and the #IMM2 value added to Y.<br>The full 32 bits of R0 and the 16 lower-order bits of R1 are handled as signed integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>D0 is handled as an integer. |
| Behavior model | @(R0+(char)#IMM1, (short)((short)(R1&0xffff)+(char)#IMM2)) = D0; |
| Operand type | D0: 32-bit signed integer<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **STRCIU** | **STore to Relative Coordinate with Immediate offset, then source Updated** |
|---|---|
| Usage | STRCIU D0, #IMM1, #IMM2 |
| Function | This is an instruction for data transfer from a register to memory.<br>Stores the D0 value in memory at X and Y coordinates which are specified in R0 and R1 respectively, with the #IMM1 value added to X and the #IMM2 value added to Y.<br>The full 32 bits of R0 and the 16 lower-order bits of R1 are handled as signed integers. #IMM1 and #IMM2 need to be specified with 8-bit signed integers. After operation, X(R0) and Y(R1) are updated by adding #IMM1 and #IMM2 respectively.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine.<br>D0 is handled as an integer. |
| Behavior model | @(R0+(char)#IMM1, (short)((short)(R1&0xffff)+(char)#IMM2)) = D0;<br>R0 = R0+(char)#IMM1; R1 = (short)(R1+(char)#IMM2); |
| Operand type | D0: 32-bit signed integer<br>#IMM1: 8-bit signed integer<br>#IMM2: 8-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **STRXY** | **STore to Relative Coordinate XY with register offset** |
|---|---|
| Usage | STRXY D0, S0, S1 |
| Function | This is an instruction for data transfer from a register to memory.<br>Stores the D0 value in memory at X and Y coordinates which are specified in R0 and R1 respectively, with the S0 value added to X and the S1 value added to Y.<br>The full 32 bits of R0 and S0 and the 16 lower-order bits of R1 and S1 are handled as signed integers.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine. |
| Behavior model | @(R0+S0, (short)((short)(R1&0xffff)+(short)(S1&0xffff))) = D0; |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer<br>S1: 16-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **STRXYU** | **STore to Relative Coordinate XY with register offset, then source Updated** |
|---|---|
| Usage | STRXYU D0, S0, S1 |
| Function | This is an instruction for data transfer from a register to memory.<br>Stores the D0 value in memory at X and Y coordinates which are specified in R0 and R1 respectively, with the S0 value added to X and the S1 value added to Y.<br>The full 32 bits of R0 and S0 and the 16 lower-order bits of R1 and S1 are handled as signed integers. After operation, X(R0) and Y(R1) are updated by adding S0 and S1 respectively.<br>Operation cannot be guaranteed if the calculated offset is negative to the base address or obtained address is outside from the range handled by the CVengine. |
| Behavior model | @(R0+S0, (short)((short)(R1&0xffff)+(short)(S1&0xffff))) = D0;<br>R0 = R0+S0; R1 = (short)((short)(R1&0xffff)+(short)(S1&0xffff)); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer<br>S1: 16-bit signed integer<br>R0: 32-bit signed integer<br>R1: 16-bit signed integer |

| **STSRB** | **STore Small Register Block** |
|---|---|
| Usage | STSRB D0, S0, S1 |
| Function | This is an instruction to transfer 256-bit data from 256-bit large register block to local working memory. This instruction supports only the common address space.<br>Writes the data read from D0 to the address specified by adding S1*32 to address S0 of local working memory.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R23), 24 (for register block R24-R31), 32 (for register block R32-R39) and 40 (for register block R40-R47) |
| Behavior model | *(S0 + S1*32) = D0; |
| Operand type | D0: 256-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>S1: 16-bit signed integer |

| **STSRBI** | **STore Small Register Block with Immediate offset** |
|---|---|
| Usage | STSRBI D0, S0, #IMM |
| Function | This is an instruction to transfer 256-bit data from 256-bit large register block to local working memory. This instruction supports only the common address space.<br>Writes the data read from D0 to the address specified by adding #IMM*32 to address S0 of local working memory.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R23), 24 (for register block R24-R31), 32 (for register block R32-R39) and 40 (for register block R40-R47) |
| Behavior model | *(S0 +#IMM*32) = D0; |
| Operand type | D0: 256-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>#IMM: 8-bit signed integer |

**STore Small Register Block with LPC offset**

**STSRBC**

| | |
|---|---|
| Usage | STSRBC D0, S0 |
| Function | This is an instruction to transfer 256-bit data from 256-bit large register block to local working memory. This instruction supports only the common address space.<br>Writes the data read from D0 to the address specified by adding LPC*32 to address S0 of local working memory.<br>The access to LWM must be 64-bit aligned, when unaligned, function is prohibited.<br>D0 is specifying the lowest register of the register block. Allowed values are 16 (for register block R16-R23), 24 (for register block R24-R31), 32 (for register block R32-R39) and 40 (for register block R40-R47) |
| Behavior model | *(S0 + LPC*32) = D0; |
| Operand type | D0: 256-bit unsigned integer<br>S0: 32-bit unsigned integer (Multiples of 8)<br>LPC: 9-bit signed integer |

**SUB** **SUB**

| | |
|---|---|
| Usage | SUB D0, S0, S1 |
| Function | Subtracts S1 from S0, then stores the result in D0.<br>S0 and S1 are handled as integers in the operation. |
| Behavior model | D0 = S0 - S1; |

**SUBS** **SUBtract with Saturation**

| | |
|---|---|
| Usage | SUBS D0, S0, S1 |
| Function | This is an instruction to subtract a signed 32-bit source value S1 from a signed 32-bit source value S0 with saturation. Mapped on Thread. |
| Behavior model | D0 = SAT(S0 - S1, 32); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer<br>S1: 32-bit signed integer |

**SUBSS8** **SUBtract Signed with Saturation 8-bit values**

| | |
|---|---|
| Usage | SUBSS8 D0, S0, S1 |
| Function | This is an instruction to subtract 8 times signed 8-bit value source S1 from signed 8-bit value source S0 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] - S1[n], 8), n = 0...7; |
| Operand type | D0: 8 times 8-bit signed integer<br>S0: 8 times 8-bit signed integer<br>S1: 8 times 8-bit signed integer |

| **SUBSS16** | **SUBtract Signed with Saturation 16-bit values** |
|---|---|
| Usage | SUBSS16 D0, S0, S1 |
| Function | This is an instruction to subtract 4 times signed 16-bit value source S1 from signed 16-bit value source S0 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] - S1[n], 16), n = 0...3; |
| Operand type | D0: 4 times 16-bit signed integer<br>S0: 4 times 16-bit signed integer<br>S1: 4 times 16-bit signed integer |

| **SUBRU** | **SUB Uniform from Register** |
|---|---|
| Usage | SUBRU D0, S0, #IMM |
| Function | Subtracts the uniform value indexed by #IMM from S0, then stores the result in D0.<br>S0 and the uniform value are handled as integers in the operation. |
| Behavior model | D0 = S0 - Uniform[#R6IMM]; |

| **SUBSRU** | **SUBtract with Saturation from Register value a Unified memory value** |
|---|---|
| Usage | SUBSRU D0, S0, #IMM |
| Function | This is an instruction to subtract a signed 32-bit value specified by a Uniform index #IMM from a signed 32-bit source value S0 with saturation. Mapped on Thread. |
| Behavior model | D0 = SAT(S0 - Uniform[#R6IMM], 32); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer |

| **SUBSUR** | **SUBtract with Saturation from Unified memory value a Register value** |
|---|---|
| Usage | SUBSUR D0, S0, #IMM |
| Function | This is an instruction to subtract a signed 32-bit source value S0 from a signed 32-bit value specified by a Uniform index #IMM with saturation. Mapped on Thread. |
| Behavior model | D0 = SAT(Uniform[#R6IMM] - S0, 32); |
| Operand type | D0: 32-bit signed integer<br>S0: 32-bit signed integer |

| **SUBUR** | **SUB Register from Uniform** |
|---|---|
| Usage | SUBUR D0, S0, #IMM |
| Function | Subtracts S0 from the uniform value indexed by #IMM, then stores the result in D0.<br>S0 and the uniform value are handled as integers in the operation. |
| Behavior model | D0 = Uniform[#R6IMM] - S0; |

| **SUBUS8** | **SUBtract Unsigned with Saturation 8-bit values** |
|---|---|
| Usage | SUBUS8 D0, S0, S1 |
| Function | This is an instruction to subtract 8 times unsigned 8-bit value source S1 from unsigned 8-bit value source S0 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] - S1[n], 8), n = 0..7; |
| Operand type | D0: 8 times 8-bit unsigned integer<br>S0: 8 times 8-bit unsigned integer<br>S1: 8 times 8-bit unsigned integer |

| **SUBUS16** | **SUBtract Unsigned with Saturation 16-bit values** |
|---|---|
| Usage | SUBUS16 D0, S0, S1 |
| Function | This is an instruction to subtract 4 times unsigned 16-bit value source S1 from unsigned 16-bit value source S0 with saturation. |
| Behavior model | D0[n] = SAT(S0[n] - S1[n], 16), n = 0..3; |
| Operand type | D0: 4 times 16-bit unsigned integer<br>S0: 4 times 16-bit unsigned integer<br>S1: 4 times 16-bit unsigned integer |

| **SYNCG** | **SYNC Group** |
|---|---|
| Usage | SYNCG |
| Function | Synchronize all threads inside a master-slave group. A SYNCG inside a thread causes the queue arbiter to set the bit for this thread inside the SYNCG register and the thread is waiting till a continue is received from the queue arbiter, After all bits from a master-slave group are set in the SYNCG register, the queue arbiter is sending the continue signal and the bits are cleared inside the SYNCG register. A master thread can issue this instruction. |
| Behavior model | - |

| **SYNCP** | **SYNC PBUF** |
|---|---|
| Usage | SYNCP #IMM |
| Function | Clear or flush pbuf. The behavior of the instruction is configured as follows.<br>1: Clear read buffer<br>2: Flush write buffer<br>3: Clear read buffer and flush write buffer.<br>Other immediate value is prohibited.<br>The “pbuf” is an alias of $0^{th}$ data cache associated with each thread. |
| Behavior model | - |

| **TRAP** | **TRAP program** |
|---|---|
| Usage | TRAP (no operand) |
| Function | Stops the program and transits the thread to the idle state. |
| Behavior model | - |

| **UPLSB8** | **Unpack the 4 lowest bits of S0 into the LSB's of 4 8-bit values stored in D0** |
|---|---|
| Usage | UPLSB8 D0, S0 |
| Function | This is an instruction to unpack the 4 lowest bits of S0 into the LSB's of 4 8-bit values stored in D0. The values of other bits are retained. Implemented in each thread. |
| Behavior model | D0[0] = S0[0]; D0[8] = S0[1]; D0[16] = S0[2]; D0[24] = S0[3]; |
| Operand type | D0: 32bit unsigned integer<br>S0: 32bit unsigned integer (4bit used) |

| **UPLSB16** | **Unpack the 2 lowest bits of S0 into the LSB's of 2 16-bit values stored in D0** |
|---|---|
| Usage | UPLSB16 D0, S0 |
| Function | This is an instruction to unpack the 2 lowest bits of S0 into the LSB's of 2 16-bit values stored in D0. The values of other bits are retained. Implemented in each thread. |
| Behavior model | D0[0] = S0[0]; D0[16] = S0[1]; |
| Operand type | D0: 32bit unsigned integer<br>S0: 32bit unsigned integer (2bit used) |

| **VPEBS** | **Variable Position Extract Byte Signed extension** |
|---|---|
| Usage | VPEBS D0, S0, #IMM |
| Function | The byte value stored at a position defined by #IMM in S0 gets sign extended to 32-bit and stored into D0. #IMM can take values 0 till 24. |
| Behavior model | D0 = ((S0 >> #IMM) << 24) >> 24 |

| **VPEBU** | **Variable Position Extract Byte Unsigned extension** |
|---|---|
| Usage | VPEBU D0, S0, #IMM |
| Function | The byte value stored at a position defined by #IMM in S0 gets unsigned extended to 32-bit and stored into D0. #IMM can take values 0 till 24. |
| Behavior model | D0 = ((S0 >> #IMM) << 24) >>> 24 |

| **VPEHS** | **Variable Position Extract Half word Signed extension** |
|---|---|
| Usage | VPEHS D0, S0, #IMM |
| Function | The half word value stored at a position defined by #IMM in S0 gets sign extended to 32-bit and stored into D0. #IMM can take values 0 till 16. |
| Behavior model | D0 = ((S0 >> #IMM) << 16) >> 16 |

| **VPEHU** | **Variable Position Extract Half word Unsigned extension** |
|---|---|
| Usage | VPEHU D0, S0, #IMM |
| Function | The half word value stored at a position defined by #IMM in S0 gets unsigned extended to 32-bit and stored into D0. #IMM can take values 0 till 16. |
| Behavior model | D0 = ((S0 >> #IMM) << 16) >>> 16 |

| **WAITDMA** | **Wait for DMA** |
|---|---|
| Usage | WAITDMA |
| Function | Wait till the queue arbiter sends continue signal which happens after the DMA transfer is finished. |
| Behavior model | - |

## U32.10 Instruction Format, Code and Pipeline Operation

Instructions have a fixed length of 32 bits and are classified into the following six instruction types: NI, TypeI16, TypeI8(xy), TypeI8(i), TypeRI/TypeRU and TypeR. The bit format of each instruction type is described in section U32.10.1 .

All instructions have the F1 bit and F0 bit to conditionally execute the instruction depending on the status register value. If the F0 and F1 bits are both 0, then the instruction is always executed regardless of the status register value. If the F1 bit is 1, then the instruction is executed when the status register is 1, whereas if the status register is 0, the instruction is regarded as NOP. If the F0 bit is 0, then the instruction is executed when the status register is 0, whereas if the status register is 1, the instruction is regarded as NOP. If the F1 and F0 bits are both 1, operation is not guaranteed.

The opcode of each instruction is listed in section U32.10.2 . The instruction group to which the instruction belongs to is also described in section U32.10.2 .

The pipeline operation of each instruction group is described in section U32.10.3 . Since the pipeline is stalled depending on the combination of the instructions, the conditions described in section U32.10.3 must be considered during programming.

### U32.10.1 Instruction Format

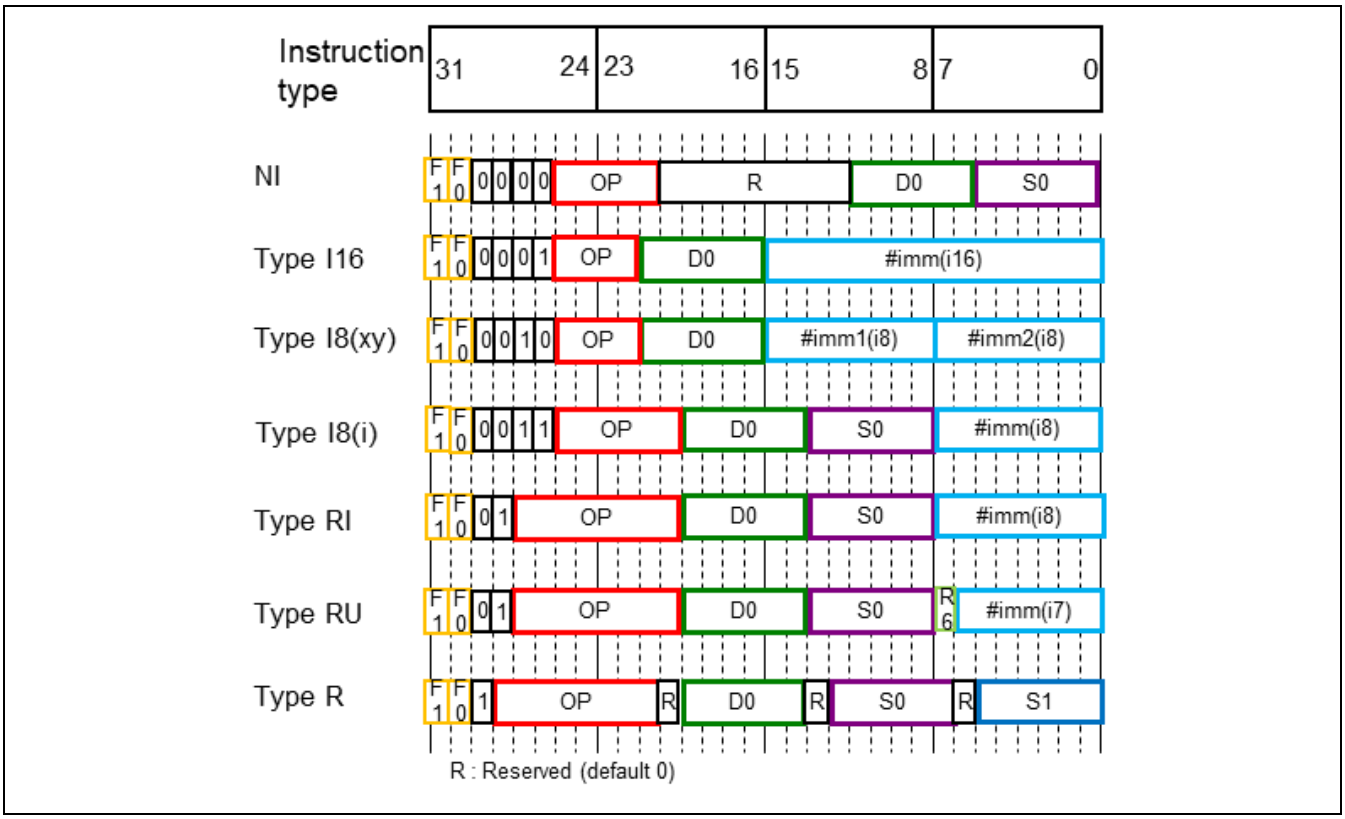



**Figure U32.10.1 List of Instruction Formats**

All instructions have a fixed length of 32 bits. Each instruction belongs to one of the six types of instructions: TypeNI, TypeI16, TypeI8(xy), TypeI8(i), TypeRI,/TypeRU, and TypeR.

| | |
|---|---|
| NI | No Immediate. Instruction format with two register operands. |
| I16 | Instruction format with one register operand and one immediate source value of 16-bit signed integer data. |
| I8(xy) | Instruction format with one register operand and two immediate source values of 8-bit signed integer data. |
| I8(i) | Instruction format with two register operands and one immediate source value of 8-bit integer data. |
| RI | Instruction format with two register operands and one immediate source value of 8-bit integer data. Whether an immediate value is signed or unsigned differs with the instruction. |
| RU | Instruction format with two register operands and one immediate source value of 7-bit integer data. The R6 bit is used to distinguish if R6 is referred to access the uniform storage. When the R6 bit is 1, R6 can be used as a general purpose register. To control this bit, refer to Section U32.9 Note 4. |
| R | Instruction format with three register operands. |

Execution or non-execution of the instruction can be selected depending on the value of the status register (SR) by setting either the F0 bit or F1 bit of the instruction to 1. This enables the instruction to be conditionally executed without using a branch instruction.

The instruction is always executed when the F0 and F1 bits are both set to 0. Setting both the F0 bit and F1 bit to 1 is prohibited.

**Table U32.10.1 Execution/Non-Execution Based on the Status Register and F0/F1 Flag State**

| | **{F1, F0} == {0, 0}** | **{F1, F0} == {0, 1}** | **{F1, F0} == {1, 0}** | **{F1, F0} == {1, 1}** |
|---|---|---|---|---|
| SR==0 | Always executed*1 | Executed*1 | Not executed*2 | Setting prohibited*2 |
| SR==1 | | Not executed*2 | Executed*1 | |

*1: Illegal instruction error can take place when the command is corrupted. *2: Operation is not guaranteed. No illegal instruction error takes place even if other bits are corrupted.

### U32.10.2 Instruction Code List

**Table U32.10.2 Instruction Code List**

| Instruction Type | Mnemonic | Instruction Group | Number of Execution Steps | Number of Issuance Cycles | Code | Code2 {INST[20], INST[13]} | SAM*1 |
|---|---|---|---|---|---|---|---|
| NI | NOP | CO | - (Control instruction) | 1 | 0x00 | - | - |
| | CALLR | BR | - (Branch instruction) | 1 | 0x02 | - | - |
| | ACTST | ET1 | 1 | 1 | 0x04 | - | - |
| | BRAR | BR | - (Branch instruction) | 1 | 0x05 | - | - |
| | TRAP | CO | - (Control instruction) | 1 | 0x06 | - | - |
| | RETA | BR | - (Branch instruction) | 1 | 0x07 | - | - |
| | INT | CO | - (Control instruction) | 1 | 0x08 | - | - |
| | LMBD | ES2 | 2 | 1 | 0x09 | - | SAM4 |
| | RMBD | ES2 | 2 | 1 | 0x0A | - | SAM4 |
| | SYNCG | CO | - (Control instruction) | 1 | 0x0B | - | - |
| | UPLSB8 | ET1 | 1 | 1 | 0x0C | - | - |
| | WAITDMA | CO | - (Control instruction) | 1 | 0x0D | - | - |
| | UPLSB16 | ET1 | 1 | 1 | 0x0E | - | - |
| | RXCH | ET1 | 1 | 1 | 0x0F | - | - |
| | INV64 | ET1 | 1 | 1 | 0x10 | - | - |
| | LPE | BR | 1 | 1 | 0x11 | - | - |
| | LDLRBC | LS | 6 | 1 | 0x12 | - | - |
| | STLRBC | LS | 1 | 1 | 0x13 | - | - |
| | LDSRBC | LS | 6 | 1 | 0x14 | - | - |
| | STSRBC | LS | 1 | 1 | 0x15 | - | - |
| | ABS8 | ES2 | 2 | 1 | 0x16 | - | SAM4 |
| | ABS16 | ES2 | 2 | 1 | 0x17 | - | SAM4 |
| I16 | MOVI | ET1 | 1 | 1 | 0x00 | - | - |
| | SETHI | ET1 | 1 | 1 | 0x01 | - | - |
| | CALL | BR | - (Branch instruction) | 1 | 0x02 | - | - |
| | CALLA | BR | - (Branch instruction) | 1 | 0x03 | - | - |
| | STCI | CO | - | 1 | 0x04 | - | - |
| | BRA | BR | - (Branch instruction) | 1 | 0x05 | - | - |
| | LDCI | CO | - (Control instruction) | 1 | 0x06 | - | - |
| | FLDCI | CO | - (Control instruction) | 1 | 0x07 | - | - |
| I8(XY) | LDRCI | LS | 5 | 1 | 0x00 | - | - |
| | FLDRCI | LS | 5 | 1 | 0x01 | - | - |
| | STRCI | LS | 1 | 1 | 0x02 | - | - |
| | FSTRCI | LS | 1 | 1 | 0x03 | - | - |
| | LDRCIU | LS | 5 | 1 | 0x04 | - | - |
| | FLDRCIU | LS | 5 | 1 | 0x05 | - | - |
| | STRCIU | LS | 1 | 1 | 0x06 | - | - |
| | FSTRCIU | LS | 1 | 1 | 0x07 | - | - |

| Instruction Type | Mnemonic | Instruction Group | Number of Execution Steps | Number of Issuance Cycles | Code | Code2 {INST[20], INST[13]} | SAM*1 |
|---|---|---|---|---|---|---|---|
| | SBOCRD | LS | 1 | 1 | 0x08 | - | - |
| I8(i) | LDBUI | LS | 5 | 1 | 0x00 | - | - |
| | LDBSI | LS | 5 | 1 | 0x01 | - | - |
| | STBI | LS | 1 | 1 | 0x02 | - | - |
| | LDHUI | LS | 5 | 1 | 0x04 | - | - |
| | LDHSI | LS | 5 | 1 | 0x05 | - | - |
| | STHI | LS | 1 | 1 | 0x06 | - | - |
| | LDI | LS | 5 | 1 | 0x08 | - | - |
| | STI | LS | 1 | 1 | 0x0A | - | - |
| | VPEBS | ET1 | 1 | 1 | 0x0C | - | - |
| | VPEBU | ET1 | 1 | 1 | 0x0D | - | - |
| | VPEHS | ET1 | 1 | 1 | 0x0E | - | - |
| | VPEHU | ET1 | 1 | 1 | 0x0F | - | - |
| | SYNCP | CO | - (Control instruction) | 1 | 0x10 | - | - |
| | LDLLI | LS | 5 | 1 | 0x14 | - | - |
| | STLLI | LS | 1 | 1 | 0x15 | - | - |
| | LPS | BR | 1 | 1 | 0x16 | - | - |
| | DMAWAITS | CO | - (Control instruction) | 1 | 0x17 | - | - |
| RI | ADDI | ET1 | 1 | 1 | 0xA0 | - | - |
| | SHLI | ET1 | 1 | 1 | 0xA1 | - | - |
| | SHRAI | ET1 | 1 | 1 | 0xA2 | - | - |
| | SHRLI | ET1 | 1 | 1 | 0xA3 | - | - |
| | BRCID | BR | - (Branch instruction) | 1 | 0xA4 | - | - |
| | BRTID | BR | - (Branch instruction) | 1 | 0xA5 | - | - |
| | LDLRBI | LS | 6 | 1 | 0x28 | - | - |
| | STLRBI | LS | 1 | 1 | 0x29 | - | - |
| | LDSRBI | LS | 6 | 1 | 0x2A | - | - |
| | STSRBI | LS | 1 | 1 | 0x2B | - | - |
| | SHLI8 | ES2 | 2 | 1 | 0xAC | - | SAM4 |
| | SHLI16 | ES2 | 2 | 1 | 0xAD | - | SAM4 |
| | SHAI8 | ES2 | 2 | 1 | 0xAE | - | SAM4 |
| | SHAI16 | ES2 | 2 | 1 | 0xAF | - | SAM4 |
| | ADDSI | ET1 | 1 | 1 | 0xB0 | - | - |
| | BTSTI | ET1 | 1 | 1 | 0xB3 | - | - |
| RU | ADDU | ET1 | 1 | 1 | 0x82 | - | - |
| | ADDSU | ET1 | 1 | 1 | 0x83 | - | - |
| | SUBRU | ET1 | 1 | 1 | 0x84 | - | - |
| | SUBSRU | ET1 | 1 | 1 | 0x85 | - | - |
| | SUBUR | ET1 | 1 | 1 | 0x86 | - | - |
| | SUBSUR | ET1 | 1 | 1 | 0x87 | - | - |

| Instruction Type | Mnemonic | Instruction Group | Number of Execution Steps | Number of Issuance Cycles | Code | Code2 {INST[20], INST[13]} | SAM[*1] |
|---|---|---|---|---|---|---|---|
| | MULU | ES3 | 3[*2] | 1 | 0x88 | - | SAM8 |
| | MULSU | ES3 | 3[*2] | 1 | 0x89 | - | SAM8 |
| | EORU | ET1 | 1 | 1 | 0x8A | - | - |
| | ORU | ET1 | 1 | 1 | 0x8B | - | - |
| | ANDU | ET1 | 1 | 1 | 0x8C | - | - |
| | MAXSU | ET1 | 1 | 1 | 0x8E | - | - |
| | MINSU | ET1 | 1 | 1 | 0x90 | - | - |
| | CMPU.SGT | ET1 | 1 | 1 | 0x92 | - | - |
| | CMPU.SLE | ET1 | 1 | 1 | 0x94 | - | - |
| | CMPU.SGE | ET1 | 1 | 1 | 0x96 | - | - |
| | CMPU.SLT | ET1 | 1 | 1 | 0x98 | - | - |
| | CMPU.EQ | ET1 | 1 | 1 | 0x9A | - | - |
| | CMPU.NEQ | ET1 | 1 | 1 | 0x9B | - | - |
| | MULLSU | ES3 | 3[*2] | 1 | 0x9C | - | SAM8 |
| | MOVU | ET1 | 1 | 1 | 0x9F | - | - |
| | BTSTU | ET1 | 1 | 1 | 0xB2 | - | - |
| | ASMU | ES3 | 3[*2] | 1 | 0xD2 | - | SAM4 |
| | ADDDBLU | ET1 | 1 | 1 | 0xD3 | - | |
| | FADDU | ESF3 | 3[*2] | 1 | 0x02 | - | - |
| | FSUBU | ESF3 | 3[*2] | 1 | 0x04 | - | - |
| | FMULU | ESF3 | 3[*2] | 1 | 0x08 | - | - |
| | FMAXU | ET1 | 1 | 1 | 0x0E | - | - |
| | FMINU | ET1 | 1 | 1 | 0x10 | - | - |
| | FCMPU.OGT | ET1 | 1 | 1 | 0x12 | - | - |
| | FCMPU.UGT | ET1 | 1 | 1 | 0x13 | - | - |
| | FCMPU.OLE | ET1 | 1 | 1 | 0x14 | - | - |
| | FCMPU.ULE | ET1 | 1 | 1 | 0x15 | - | - |
| | FCMPU.OEQ | ET1 | 1 | 1 | 0x16 | - | - |
| | FCMPU.UEQ | ET1 | 1 | 1 | 0x17 | - | - |
| | FCMPU.OGE | ET1 | 1 | 1 | 0x18 | - | - |
| | FCMPU.UGE | ET1 | 1 | 1 | 0x19 | - | - |
| | FCMPU.OLT | ET1 | 1 | 1 | 0x1A | - | - |

| Instruction Type | Mnemonic | Instruction Group | Number of Execution Steps | Number of Issuance Cycles | Code | Code2 {INST[20], INST[13]} | SAM[*1] |
|---|---|---|---|---|---|---|---|
| | FCMPU.ULT | ET1 | 1 | 1 | 0x1B | - | - |
| | FCMPU.ONE | ET1 | 1 | 1 | 0x1C | - | - |
| | FCMPU.UNE | ET1 | 1 | 1 | 0x1D | - | - |
| | FCMPU.O | ET1 | 1 | 1 | 0x1E | - | - |
| | FCMPU.UO | ET1 | 1 | 1 | 0x1F | - | - |
| | FMADU | ESF4 | 4[*2] | 1 | 0x20 | - | - |
| | FMSUU | ESF4 | 4[*2] | 1 | 0x21 | - | - |
| R | LDRXY | LS | 5 | 1 | 0x80 | - | - |
| | STRXY | LS | 1 | 1 | 0x81 | - | - |
| | INV | ET1 | 1 | 1 | 0x82 | - | - |
| | ABS | ET1 | 1 | 1 | 0x83 | - | - |
| | ADD | ET1 | 1 | 1 | 0x84 | - | - |
| | ADDS | ET1 | 1 | 1 | 0x85 | - | - |
| | SUB | ET1 | 1 | 1 | 0x86 | - | - |
| | SUBS | ET1 | 1 | 1 | 0x87 | - | - |
| | MUL | ES3 | 3[*2] | 1 | 0x88 | - | SAM8 |
| | MULS | ES3 | 3[*2] | 1 | 0x89 | - | SAM8 |
| | EOR | ET1 | 1 | 1 | 0x8A | - | - |
| | OR | ET1 | 1 | 1 | 0x8B | - | - |
| | AND | ET1 | 1 | 1 | 0x8C | - | - |
| | AND64 | ET1 | 1 | 1 | 0x8D | - | - |
| | I2F | ET1 | 1 | 1 | 0x8E | - | - |
| | SHUF | ES1 | 1 | 1 | 0x8F | - | - |
| | MAXS | ET1 | 1 | 1 | 0x90 | - | - |
| | GMADBUL | ES3 | 3[*2] | 1 | 0x91 | - | SAM2 |
| | MINS | ET1 | 1 | 1 | 0x92 | - | - |
| | GMADBSL | ES3 | 3[*2] | 1 | 0x93 | - | SAM2 |
| | SCSCS | ES2 | 2 | 1 | 0x94 | - | SAM8 |
| | NCSCS | ES2 | 2 | 1 | 0x95 | - | SAM8 |
| | STC | CO | - (Control instruction) | 1 | 0x96 | - | - |
| | LLROT | ES3 | 3[*2] | 1 | 0x97 | - | SAM4 |
| | LDLRB | LS | 6 | 1 | 0x18 | - | - |
| | STLRB | LS | 1 | 1 | 0x19 | - | - |
| | CNT0 | ET1 | 1 | 1 | 0x9A | - | - |
| | CNT1 | ET1 | 1 | 1 | 0x9B | - | - |
| | MOV | ET1 | 1 | 1 | 0x9C | - | - |
| | SHL | ET1 | 1 | 1 | 0x9D | - | - |
| | SHRA | ET1 | 1 | 1 | 0x9E | - | - |

| Instruction Type | Mnemonic | Instruction Group | Number of Execution Steps | Number of Issuance Cycles | Code | Code2 {INST[20], INST[13]} | SAM[*1] |
|---|---|---|---|---|---|---|---|
| | SHRL | ET1 | 1 | 1 | 0x9F | - | - |
| | L2F | ES3 | 3[*2] | 1 | 0xA0 | - | SAM8 |
| | BTST | ET1 | 1 | 1 | 0xA1 | - | - |
| | PACKH0 | ES2 | 2 | 1 | 0xA2 | - | SAM4 |
| | PACKH1 | ES2 | 2 | 1 | 0xA3 | - | SAM4 |
| | PACKB0 | ES2 | 2 | 1 | 0xA4 | - | SAM4 |
| | PACKB1 | ES2 | 2 | 1 | 0xA5 | - | SAM4 |
| | PACKB2 | ES2 | 2 | 1 | 0xA6 | - | SAM4 |
| | PACKB3 | ES2 | 2 | 1 | 0xA7 | - | SAM4 |
| | GMADBU | ES3 | 3[*2] | 1 | 0xA8 | - | SAM2 |
| | GMADBS | ES3 | 3[*2] | 1 | 0xA9 | - | SAM2 |
| | GMADHU | ES3 | 3[*2] | 1 | 0xAA | - | SAM2 |
| | GMADHS | ES3 | 3[*2] | 1 | 0xAB | - | SAM2 |
| | GADDBU | ES3 | 3[*2] | 1 | 0xAC | - | SAM4 |
| | GADDBS | ES3 | 3[*2] | 1 | 0xAD | - | SAM4 |
| | GADDHU | ES3 | 3[*2] | 1 | 0xAE | - | SAM4 |
| | GADDHS | ES3 | 3[*2] | 1 | 0xAF | - | SAM4 |
| | GSUBBU | ES3 | 3[*2] | 1 | 0xB0 | - | SAM4 |
| | GSUBBS | ES3 | 3[*2] | 1 | 0xB1 | - | SAM4 |
| | GSUBHU | ES3 | 3[*2] | 1 | 0xB2 | - | SAM4 |
| | GSUBHS | ES3 | 3[*2] | 1 | 0xB3 | - | SAM4 |
| | GSADBU | ES3 | 3[*2] | 1 | 0xB4 | - | SAM4 |
| | GSADBS | ES3 | 3[*2] | 1 | 0xB5 | - | SAM4 |
| | GSADHU | ES3 | 3[*2] | 1 | 0xB6 | - | SAM4 |
| | GSADHS | ES3 | 3[*2] | 1 | 0xB7 | - | SAM4 |
| | CSUSU | ES2 | 2 | 1 | 0xB8 | - | SAM8 |
| | CSUSS | ES2 | 2 | 1 | 0xB9 | - | SAM8 |
| | MACS | ES3 | 3[*2] | 1 | 0xBA | - | SAM8 |
| | MACSS | ES3 | 3[*2] | 1 | 0xBA | {1,0} | SAM8 |
| | MACSSRE | ES3 | 3[*2] | 1 | 0xBA | {1,1} | SAM8 |
| | GMADHUL | ES3 | 3[*2] | 1 | 0xBB | - | SAM2 |
| | MULLS | ES3 | 3[*2] | 1 | 0xBC | - | SAM8 |
| | GMADHSL | ES3 | 3[*2] | 1 | 0xBD | - | SAM2 |
| | LDBU | LS | 5 | 1 | 0xBE | - | - |
| | LDBS | LS | 5 | 1 | 0xBF | - | - |
| | LDHU | LS | 5 | 1 | 0xC0 | - | - |
| | LDHS | LS | 5 | 1 | 0xC1 | - | - |
| | LD | LS | 5 | 1 | 0xC2 | - | - |
| | ST | LS | 1 | 1 | 0xC3 | - | - |
| | STB | LS | 1 | 1 | 0xC4 | - | - |

| Instruction Type | Mnemonic | Instruction Group | Number of Execution Steps | Number of Issuance Cycles | Code | Code2 {INST[20], INST[13]} | SAM[*1] |
|---|---|---|---|---|---|---|---|
| | STH | LS | 1 | 1 | 0xC5 | - | - |
| | CMP.SGT | ET1 | 1 | 1 | 0xC6 | - | - |
| | PLSB8 | ET1 | 1 | 1 | 0xC7 | - | - |
| | CMP.SLE | ET1 | 1 | 1 | 0xC8 | - | - |
| | PLSB16 | ET1 | 1 | 1 | 0xC9 | - | - |
| | CMP.SGE | ET1 | 1 | 1 | 0xCA | - | - |
| | CMV8 | ET1 | 1 | 1 | 0xCB | - | - |
| | CMP.SLT | ET1 | 1 | 1 | 0xCC | - | - |
| | CMV16 | ET1 | 1 | 1 | 0xCD | - | - |
| | CMP.EQ | ET1 | 1 | 1 | 0xCE | - | - |
| | CMP.NEQ | ET1 | 1 | 1 | 0xCF | - | - |
| | LDRXYU | LS | 5 | 1 | 0xD0 | - | - |
| | STRXYU | LS | 1 | 1 | 0xD1 | - | - |
| | ASM | ES3 | 3 | 1 | 0xD2 | - | SAM4 |
| | ADDDBL | ET1 | 1 | 1 | 0xD3 | - | - |
| | LDLL | LS | 5 | 1 | 0xD4 | - | - |
| | STLL | LS | 1 | 1 | 0xD5 | - | - |
| | PCMPAEQ | ET1 | 1 | 1 | 0xD6 | - | - |
| | PCMPOEQ | ET1 | 1 | 1 | 0xD7 | - | - |
| | LDSRB | LS | 6 | 1 | 0xD8 | - | - |
| | STSRB | LS | 1 | 1 | 0xD9 | - | - |
| | PCMPAGT | ET1 | 1 | 1 | 0xDA | - | - |
| | PCMPOGT | ET1 | 1 | 1 | 0xDB | - | - |
| | PCMPALT | ET1 | 1 | 1 | 0xDC | - | - |
| | PCMPOLT | ET1 | 1 | 1 | 0xDD | - | - |
| | ADDSS8 | ES2 | 2 | 1 | 0xDE | - | SAM2 |
| | ADDSS16 | ES2 | 2 | 1 | 0xDF | - | SAM2 |
| | ADDUS8 | ES2 | 2 | 1 | 0xE0 | - | SAM2 |
| | ADDUS16 | ES2 | 2 | 1 | 0xE1 | - | SAM2 |
| | SUBSS8 | ES2 | 2 | 1 | 0xE2 | - | SAM2 |
| | SUBSS16 | ES2 | 2 | 1 | 0xE3 | - | SAM2 |
| | SUBUS8 | ES2 | 2 | 1 | 0xE4 | - | SAM2 |
| | SUBUS16 | ES2 | 2 | 1 | 0xE5 | - | SAM2 |
| | MULSS8 | ES3 | 3[*2] | 1 | 0xE6 | - | SAM4 |
| | MULSS16 | ES3 | 3[*2] | 1 | 0xE7 | - | SAM4 |
| | MULUS8 | ES3 | 3[*2] | 1 | 0xE8 | - | SAM4 |
| | MULUS16 | ES3 | 3[*2] | 1 | 0xE9 | - | SAM4 |
| | MULLS8 | ES3 | 3[*2] | 1 | 0xEA | - | SAM4 |
| | MULLS16 | ES3 | 3[*2] | 1 | 0xEB | - | SAM4 |
| | MULLU8 | ES3 | 3[*2] | 1 | 0xEC | - | SAM4 |

| Instruction Type | Mnemonic | Instruction Group | Number of Execution Steps | Number of Issuance Cycles | Code | Code2 {INST[20], INST[13]} | SAM[*1] |
|---|---|---|---|---|---|---|---|
| | MULLU16 | ES3 | 3[*2] | 1 | 0xED | - | SAM4 |
| | SHL8 | ES2 | 2 | 1 | 0xE6 | - | SAM4 |
| | SHL16 | ES2 | 2 | 1 | 0xEF | - | SAM4 |
| | SHA8 | ES2 | 2 | 1 | 0xF0 | - | SAM4 |
| | SHA16 | ES2 | 2 | 1 | 0xF1 | - | SAM4 |
| | CMP.SLE8 | ES2 | 2 | 1 | 0xF2 | - | SAM4 |
| | CMP.SLE16 | ES2 | 2 | 1 | 0xF3 | - | SAM4 |
| | CMP.ULE8 | ES2 | 2 | 1 | 0xF4 | - | SAM4 |
| | CMP.ULE16 | ES2 | 2 | 1 | 0xF5 | - | SAM4 |
| | CMP.SLT8 | ES2 | 2 | 1 | 0xF6 | - | SAM4 |
| | CMP.SLT16 | ES2 | 2 | 1 | 0xF7 | - | SAM4 |
| | CMP.ULT8 | ES2 | 2 | 1 | 0xF8 | - | SAM4 |
| | CMP.ULT16 | ES2 | 2 | 1 | 0xF9 | - | SAM4 |
| | CMP.EQ8 | ES2 | 2 | 1 | 0xFA | - | SAM4 |
| | CMP.EQ16 | ES2 | 2 | 1 | 0xFB | - | SAM4 |
| | CMP.NEQ8 | ES2 | 2 | 1 | 0xFC | - | SAM4 |
| | CMP.NEQ16 | ES2 | 2 | 1 | 0xFD | - | SAM4 |
| | OR64 | ET1 | 1 | 1 | 0xFE | - | - |
| | EOR64 | ET1 | 1 | 1 | 0xFF | - | - |
| | FLDRXY | LS | 5 | 1 | 0x00 | - | - |
| | FSTRXY | LS | 1 | 1 | 0x01 | - | - |
| | FABS | ET1 | 1 | 1 | 0x03 | - | - |
| | FADD | ESF3 | 3[*2] | 1 | 0x04 | - | - |
| | FSUB | ESF3 | 3[*2] | 1 | 0x06 | - | - |
| | FMUL | ESF3 | 3[*2] | 1 | 0x08 | - | - |
| | F2I | ET1 | 1 | 1 | 0x0E | - | - |
| | FMAX | ET1 | 1 | 1 | 0x10 | - | - |
| | FMIN | ET1 | 1 | 1 | 0x12 | - | - |
| | FSTC | CO | - (Control instruction) | 1 | 0x14 | - | - |
| | FFRC | ESF4 | 4[*2] | 1 | 0x1A | - | - |
| | FFLR | ESF3 | 3[*2] | 1 | 0x1B | - | - |
| | FMAD | ESF4 | 4[*2] | 1 | 0x1C | - | - |
| | FMSU | ESF4 | 4[*2] | 1 | 0x1D | - | - |
| | FMP | ESF4 | 4[*2] | 1 | 0x1E | - | - |
| | FCMP.OGT | ET1 | 1 | 1 | 0x20 | - | - |
| | FCMP.UGT | ET1 | 1 | 1 | 0x21 | - | - |

| Instruction Type | Mnemonic | Instruction Group | Number of Execution Steps | Number of Issuance Cycles | Code | Code2 {INST[20], INST[13]} | SAM*1 |
|---|---|---|---|---|---|---|---|
| | FCMP.OLE | ET1 | 1 | 1 | 0x22 | - | - |
| | FCMP.ULE | ET1 | 1 | 1 | 0x23 | - | - |
| | FCMP.OGE | ET1 | 1 | 1 | 0x24 | - | - |
| | FCMP.UGE | ET1 | 1 | 1 | 0x25 | - | - |
| | FCMP.OLT | ET1 | 1 | 1 | 0x26 | - | - |
| | FCMP.ULT | ET1 | 1 | 1 | 0x27 | - | - |
| | FCMP.OEQ | ET1 | 1 | 1 | 0x28 | - | - |
| | FCMP.UEQ | ET1 | 1 | 1 | 0x29 | - | - |
| | FCMP.ONE | ET1 | 1 | 1 | 0x2A | - | - |
| | FCMP.UNE | ET1 | 1 | 1 | 0x2B | - | - |
| | FCMP.O | ET1 | 1 | 1 | 0x2C | -- | - |
| | FCMP.UO | ET1 | 1 | 1 | 0x2D | - | - |
| | FLDRXYU | LS | 5 | 1 | 0x50 | - | - |
| | FSTRXYU | LS | 1 | 1 | 0x51 | - | - |
| | MAXS8 | ES2 | 2 | 1 | 0x60 | - | SAM2 |
| | MAXS16 | ES2 | 2 | 1 | 0x61 | - | SAM2 |
| | MAXU8 | ES2 | 2 | 1 | 0x62 | - | SAM2 |
| | MAXU16 | ES2 | 2 | 1 | 0x63 | - | SAM2 |
| | MINS8 | ES2 | 2 | 1 | 0x64 | - | SAM2 |
| | MINS16 | ES2 | 2 | 1 | 0x65 | - | SAM2 |
| | MINU8 | ES2 | 2 | 1 | 0x66 | - | SAM2 |
| | MINU16 | ES2 | 2 | 1 | 0x67 | - | SAM2 |

*1 For light CVe cluster, SAM2 and SAM4 instructions are implemented as SAM8 instructions. For the information which CVe cluster is implemented as light CVe cluster, refer to Table 58.1 of User's Manual: Hardware, section 58.1.2 Specifications of the IMP-X7 by Products.
*2 For light CVe cluster, the number of execution steps is by 1 decreased. For the information which CVe cluster is implemented as light CVe cluster, refer to Table 58.1 of User's Manual: Hardware, section 58.1.2 Specifications of the IMP-X7 by Products.

### U32.10.3 Pipeline Operation of CVengine Thread

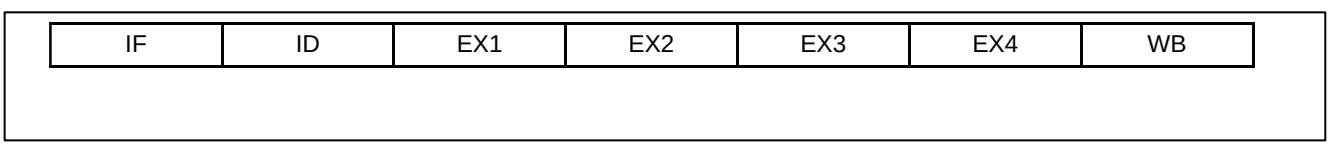



**Figure U32.10.2 Pipeline Configuration of CVengine Thread**

The CVengine thread is a microprocessor that issues single instructions to be processed in a pipeline. The pipeline is shown in the above picture. The pipeline has seven stages: instruction fetching (IF), decoding and register reading (ID),

1st execution (EX1), 2nd execution (EX2), 3rd execution (EX3), 4th execution (EX4), and write-back (WB). A single instruction is executed as a combination of pipeline stages.

### U32.10.3.1Pipeline operations for each instruction group

The pipeline operations for each instruction group are shown below.

**(1) ET1 group**

| IF | ID | EX1 | EX2 | EX3 | EX4 | WB |
|---|---|---|---|---|---|---|
| - Instruction fetch | - Instruction decode<br>- Forwarding<br>- Register read | - Operation | | | | - Write-back |

**(2) ES2 group**

| IF | ID | EX1 | EX2 | EX3 | EX4 | WB |
|---|---|---|---|---|---|---|
| - Instruction fetch | - Instruction decode<br>- Forwarding<br>- Register read | - Operation | - Operation | | | - Write-back |

**(3) ES3 group**

| IF | ID | EX1 | EX2 | EX3 | EX4 | WB |
|---|---|---|---|---|---|---|
| - Instruction fetch | - Instruction decode<br>- Forwarding<br>- Register read | - Operation | - Operation | - Operation | | - Write-back |

**(4) ESF3 group**

| IF | ID | EX1 | EX2 | EX3 | EX4 | WB |
|---|---|---|---|---|---|---|
| - Instruction fetch | - Instruction decode<br>- Forwarding<br>- Register read | - Operation | - Operation | - Operation | | - Write-back |

**(5) ESF4 group**

| IF | ID | EX1 | EX2 | EX3 | EX4 | WB |
|---|---|---|---|---|---|---|
| - Instruction fetch | - Instruction decode<br>- Forwarding<br>- Register read | - Operation | - Operation | - Operation | - Operation | - Write-back |

### (6) LS group

| IF | ID | EX1 | EX2 | EX3 | EX4 | WB |
|---|---|---|---|---|---|---|
| - Instruction fetch | - Instruction decode<br>- Forwarding<br>- Register read | - Memory data access | - Memory data access | - Memory data access | - Memory data access | - Write-back |

### (7) BR group

| IF | ID | EX1 | EX2 | EX3 | EX4 | WB |
|---|---|---|---|---|---|---|
| - Instruction fetch | - Instruction decode<br>- Forwarding<br>- Register read<br>- Branch determination | | | | | |

### (8) CO group

| IF | ID | EX1 | EX2 | EX3 | EX4 | WB |
|---|---|---|---|---|---|---|
| - Instruction fetch | - Instruction decode<br>- Forwarding | | | | | - Control |

For instructions of the ET1 and CO groups, the number of instruction steps and the number of issuance cycles are both 1, and the subsequent instruction is executed consecutively. Even if an instruction immediately following an ET1 or CO group instruction, it is dependent on that ET1 or CO group instruction, forwarding from the EX1 stage to the ID stage is performed and the subsequent instruction can be executed without stalling the pipeline. Also, even if there is an instruction that is dependent on the ET1 or CO group instruction that is multiple instructions before it, forwarding from each stage is performed and the pipeline is not stalled. However, forwarding is not done for registers used for transfer of side information like, e.g., R0 and R1 for LDRXY or R6 for uniform memory access.

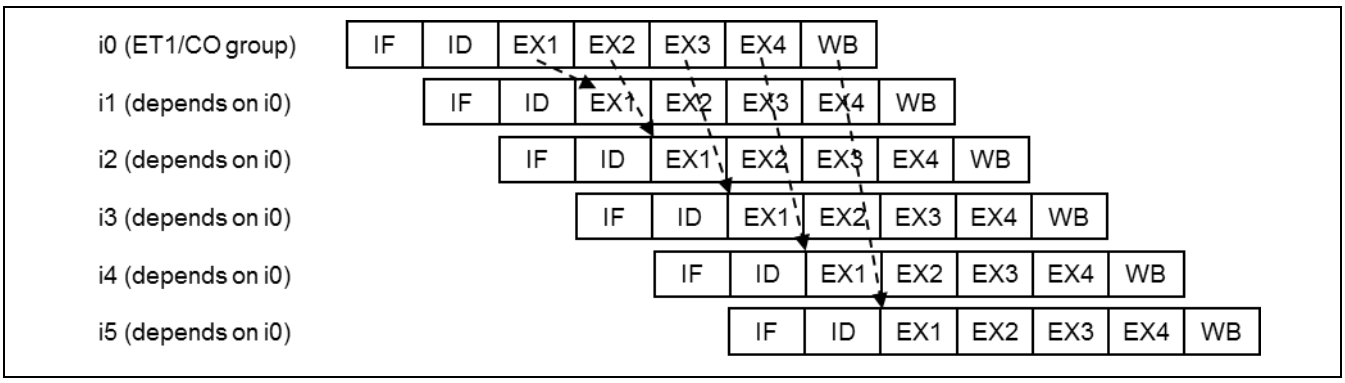



**Figure U32.10.3 Execution of an ET1/CO Group Instruction and its Dependent Instructions**

For instructions of the ES2 group, the number of instruction steps is 2 and the number of issuance cycles is 1. The subsequent instruction is executed consecutively in case it is not dependent on the ES2 group instruction. If an instruction immediately following an ES2 group instruction is dependent on that ES2 group instruction, the pipeline is stalled by one cycle. If there is an instruction that is dependent on the ES2 group instruction that is at least two instructions before it, forwarding from each stage is performed and the pipeline is not stalled. However, forwarding is not done for registers used for transfer of side information like, e.g., R0 and R1 for LDRXY or R6 for uniform memory access.

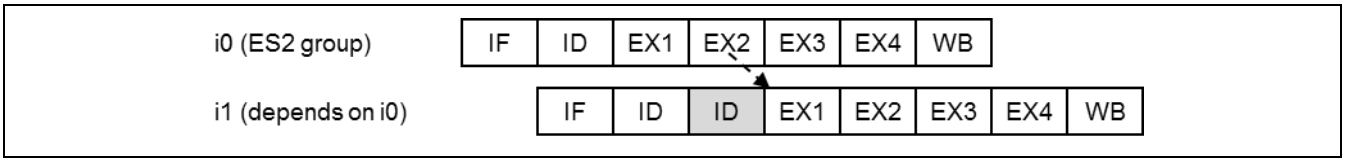



**Figure U32.10.4 Execution of an ES2 Group Instruction and its Dependent Instructions**

For instructions of the ES3/ESF3 group, the number of instruction steps is 3 and the number of issuance cycles is 1. The subsequent instruction is executed consecutively in case it is not dependent on the ES3/ESF3 group instruction. If there is an instruction that is dependent on the ES3/ESF3 group instruction that is immediately before or two instructions before it, the pipeline is stalled. If there is an instruction that is dependent on the ES3/ESF3 group instruction that is at least three instructions before it, forwarding from each stage is performed and the pipeline is not stalled. However, forwarding is not done for registers used for transfer of side information like, e.g., R0 and R1 for LDRXY or R6 for uniform memory access.

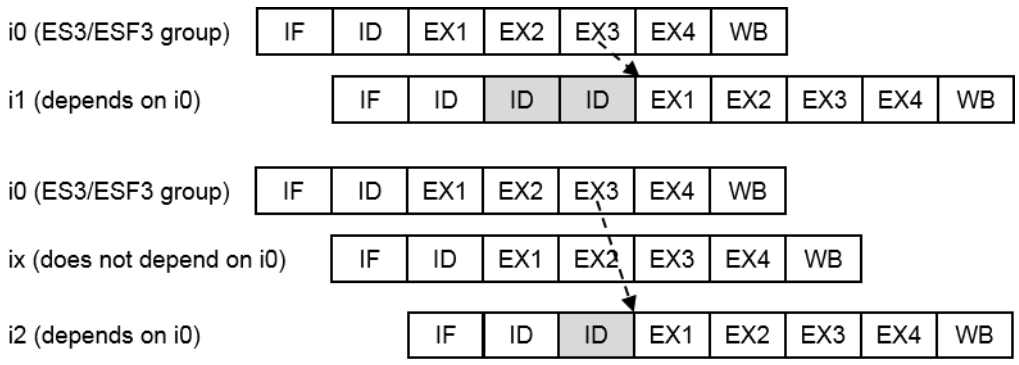



**Figure U32.10.5 Execution of an ES3/ESF3 Group Instruction and its Dependent Instructions**

For instructions of the ESF4 group, the number of instruction steps is 4 and the number of issuance cycles is 1. The subsequent instruction is executed consecutively in case it is not dependent on the ESF4 group instruction. If there is an instruction that is dependent on the ESF4 group instruction that is immediately before, two instructions before, or three instructions before it, the pipeline is stalled. If there is an instruction that is dependent on the ESF4 group instruction that is at least four instructions before it, forwarding from each stage is performed and the pipeline is not stalled. However, forwarding is not done for registers used for transfer of side information like, e.g., R0 and R1 for LDRXY or R6 for uniform memory access.

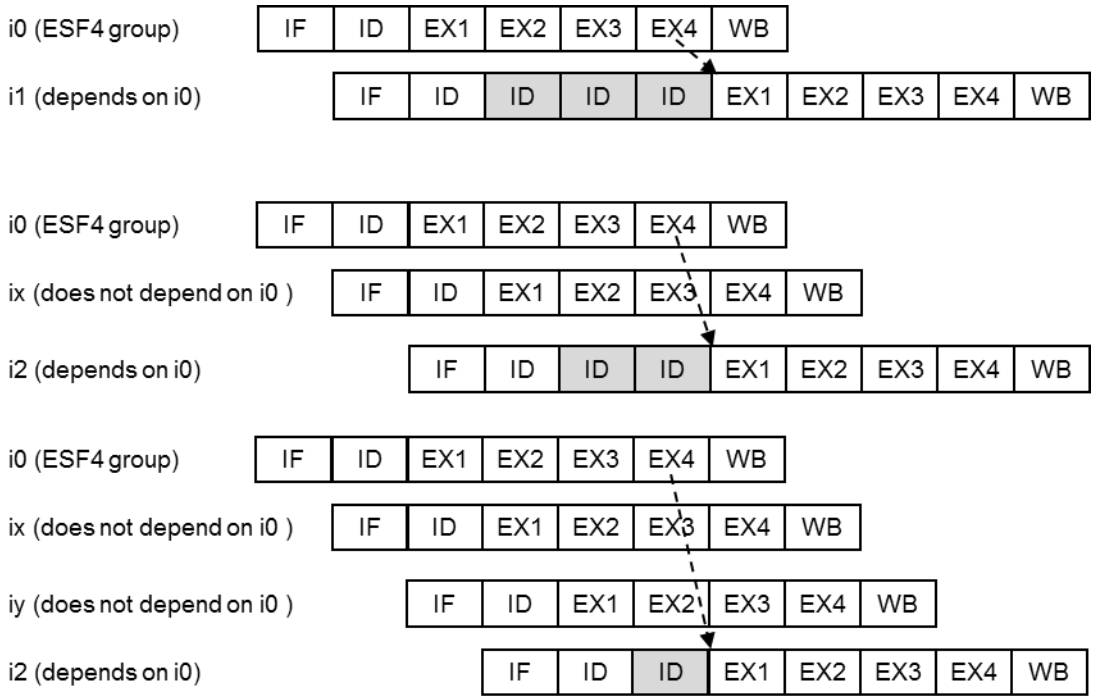



**Figure U32.10.6 Execution of an ESF4 Group Instruction and its Dependent Instructions**

For instructions of the LS group, the number of instruction steps is 5 and the number of issuance cycles is 1 for loading and the number of instruction steps and the number of issuance cycles are both 1 for storage. For a load instruction, the subsequent instruction is executed consecutively in case it is not dependent on the LS group instruction. If an instruction immediately following a load instruction is dependent on that load instruction, the pipeline is stalled by four cycles. If there is an instruction that is dependent on the LS group instruction that is at least four instructions before it, forwarding is performed, and the pipeline is not stalled. However, forwarding is not done for registers used for transfer of side information like, e.g., R0 and R1 for LDRXY or R6 for uniform memory access.

Since an LS group instruction accesses the data cache, the pipeline is stalled when a cache miss occurs. The number of stall cycles depends on the bus congestion which the CVengine is connected to and the external memory state.

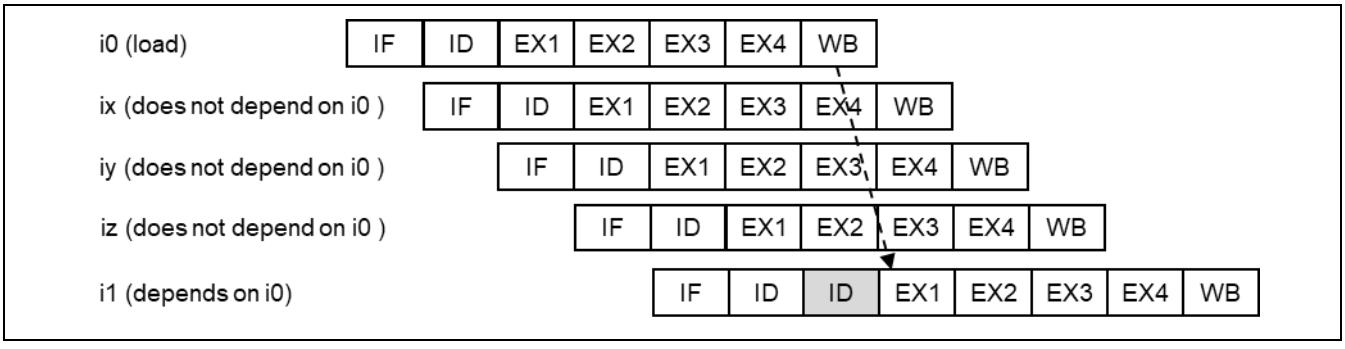



**Figure U32.10.7 Execution of a Load Instruction and its Dependent Instructions**

For storage, the subsequent instruction is always executed consecutively.

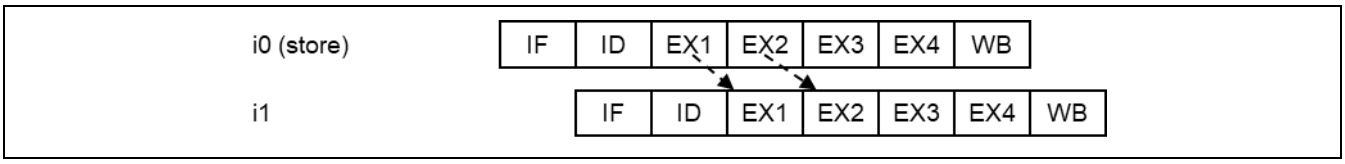



**Figure U32.10.8 Execution of a Store Instruction and its Dependent Instructions**

A BR group instruction always causes 1-cycle stall when the branch condition is satisfied.

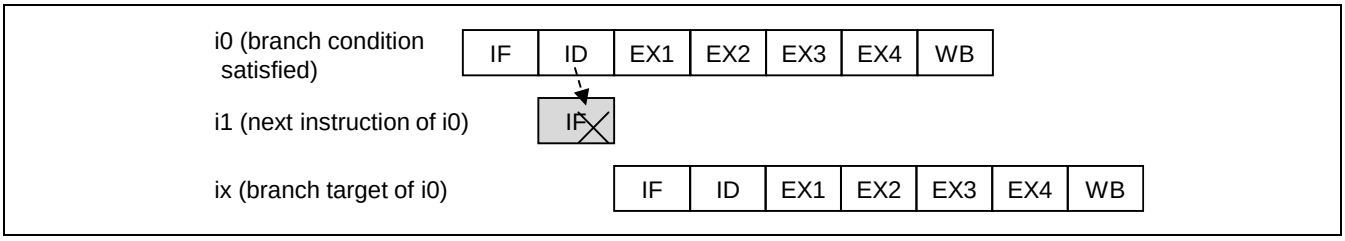



**Figure U32.10.9 Execution When the Branch Condition of a BR Group Instruction is Satisfied**

If the branch condition is not satisfied, the pipeline is not stalled.

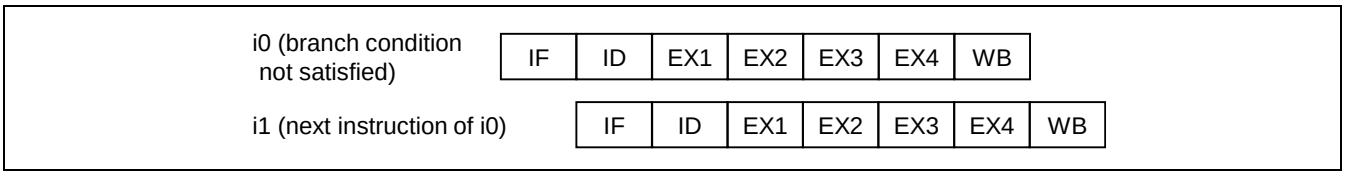



**Figure U32.10.10 Execution When the Branch Condition of a BR Group Instruction is Not Satisfied**

### U32.10.3.2Pipeline stall on arbitration

Multiple threads in the same core share the operation unit used in executing instructions of the ES2, ES3/ESF3 and ESF4 groups. Therefore, when more than one threads is requesting for the same operation unit at the same time, the pipeline may be stalled due to arbitration.

### U32.10.3.3Pipeline stall on FMAD/FMADU/FMSU/FMSUU and FADD/FADDU/FSUB/FSUBU

FMAD/FMADU/FMSU/FMSUU followed by an FADD/FADDU/FSUB/FSUBU the following instruction (e.g., FADD) is delayed by 1 clock cycle.

Example:

```
FMSU    R20, R21, R22
FADD    R26, R25, R24
```

### U32.10.3.4Forwarding with 32-bit and 64-bit instructions

A 32-bit result can be forwarded to a 64-bit source as shown in the example below:

```
ADD     R25, R17, R19
ADD     R24, R21, R23
ASMU    R24, R24, 14
```

Additionally, also for a 64-bit result and a 32-bit source, forwarding is done as shown in the example below:

```
MULLS   R16, R22, R23
```

```
ADD         R25, R17, R19
```

### U32.10.3.5Forwarding with R6 register

R6 register can be used as a general purpose register. Refer to section U32.9 Note 4 for detail. Since R6 is not a regular general purpose register, forwarding is done differently to other general purpose registers.

When LD instruction (only if accessing to Uniform) and RU type instruction are both used, conflict can occur on ID and EX2 stages so pipeline stalls 1 cycle.

```
LD          R16, R17, R3        (Accessing to Uniform)
ADD         R18, R19, R20
ADDU        R21, R22, $2        (Using R6 as offset base)
```

- Forwarding is not possible to RU type instructions when using R6 as offset base.

  ```
  ADD         R6, R16, R17
  ADDU        R18, R6, $4         (RU type)
  ```

  Forwarding is done to non-RU type instructions; no stall happens in the case below.

  ```
  ADD         R6, R16, R17
  ADD         R18, R6, R19        (non-RU type)
  ```

- Forwarding is not supported for branch instructions. The number of execution steps for branch instructions that update the general purpose register set (e.g., CALL and CALLR) is 1 clock cycle for general purpose registers. For register R6, which does not belong to the general purpose register set, no forwarding is supported, so 5 NOP cycles are added.

### U32.10.4 Number of Execution Cycles of Special Instructions of CVengine Thread

The CVengine thread has operation units dedicated to special operations, such as square root. A special operation is executed by writing to a specific register number and the result is stored in the same register number.

After these operations are started, the number of cycles shown in the following table is necessary at minimum to be able to read the result. Since these operation units are shared by multiple threads, if the same operation unit is accessed simultaneously, the number of required cycles will increase when the access right cannot be acquired due to arbitration. Also, as RCP0 (R14) and RCP1 (R15) use the same operation unit, contention occurs when operations to write to these registers happen at the same time.

**Table U32.10.3 Number of Execution Cycles of Special Instructions**

| Operation Type (Register) | Number of Execution Cycles |
|---|---|
| SIN (R8) | 5 |
| COS (R9) | 5 |
| 1/SQRT (R10) | 5 |
| EXP2 (R11) | 5 |
| LOG2 (R12) | 5 |
| SQRT (R13) | 5 |
| RCP (R14 and R15) | 6 |

When executing a read instruction that uses these register values before the result has been written, the thread will stall until the results are available.

## U32.11 Vertex Issue

The Vertex Issue is a sub module of CVengine. The Vertex Issue manages the CVengine cluster. A typical sequence of Vertex Issue behavior is the following.

1. Vertex Issue is invoked by the host processor.
2. Vertex Issue fetches command from the Command List.
3. Upon WPR commands in the Command List, Vertex Issue sets values to the Configuration Registers.
4. Upon POINT and RECT commands in the Command List, Vertex Issue throws a set of XY coordinates to a thread in idle state. Which XY coordinates are assigned to which thread is determined by Vertex Issue automatically.
5. Upon TRAP command in the Command List, the Vertex Issue returns interrupt to the host processor.

Note: The Vertex Issue has a Command List pre-fetch buffer, the size is 640 bytes.

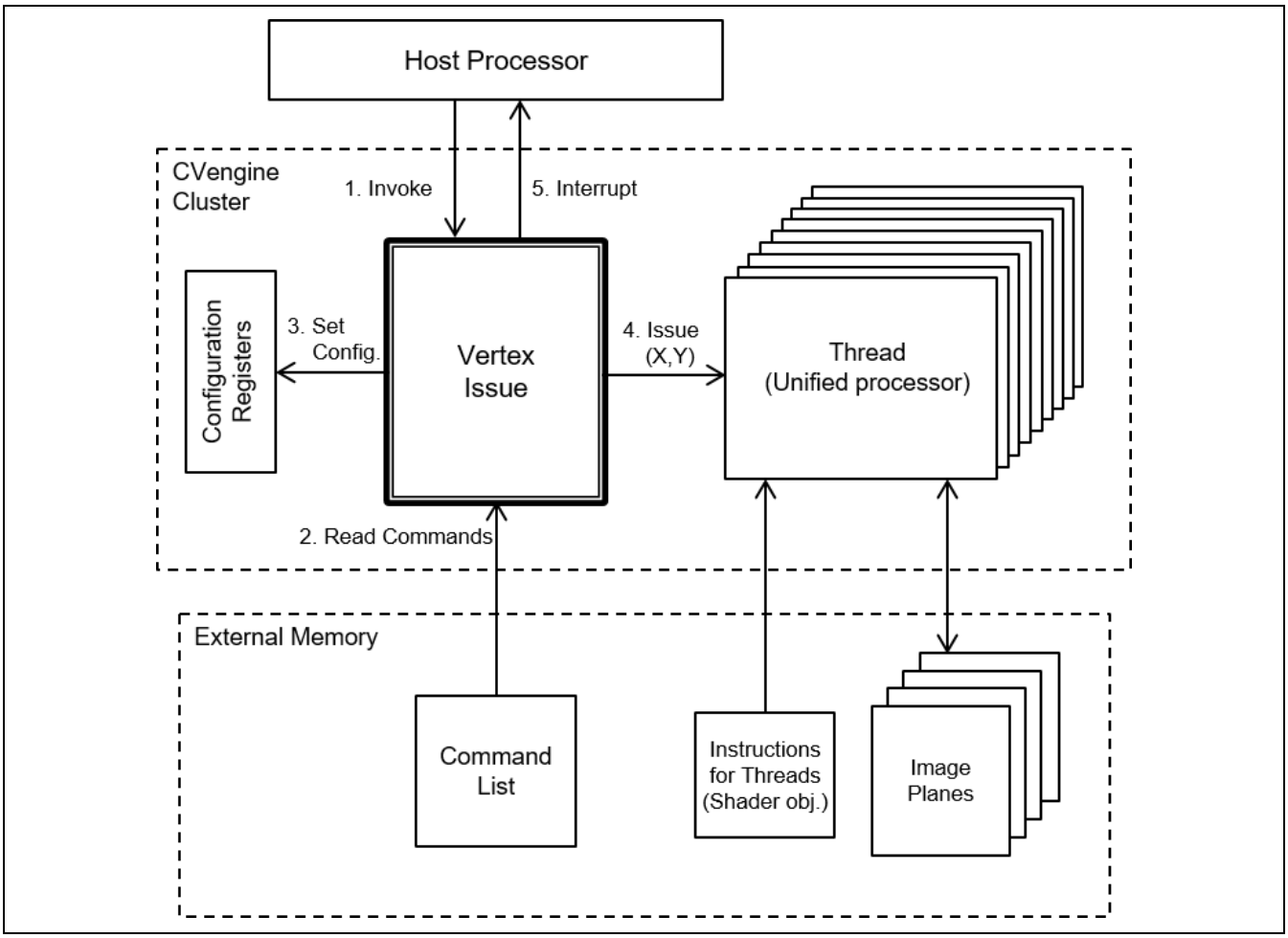



**Figure U32.11.1 Vertex Issue behavior**

### U32.11.1 Synchronization with WUP and SLP commands

To make synchronization among the cores, CNN, PSC, IMP DMAC, IMP core and CVengine core, two commands (WUP and SLP) are provided.

WUP: This command is issued by a core to wake up another core from the sleep (SLP) state.
If the WUP command is issued to a core that is operating and the core later attempts to enter the SLP state, the core will immediately wake up and cannot enter the SLP state. However, note that even if the WUP command is issued to a core two or more times, the requests are not accumulated.
When entering the SLP state, each core specifies which cores should issue the WUP command to wake it up from the SLP state. Each core wakes up only after it has received the WUP signal from all specified cores.

SLP: This command stops command fetch and decoding in the core that has issued the command and makes the core wait for the WUP signal input in the SLP state.
When issuing this command, the cores that should wake up the current core by sending the WUP signal should be specified. Upon receiving the WUP signal from all specified cores, the core wakes up from the SLP state and resumes command fetch and decoding.

The following illustrates synchronization operation.

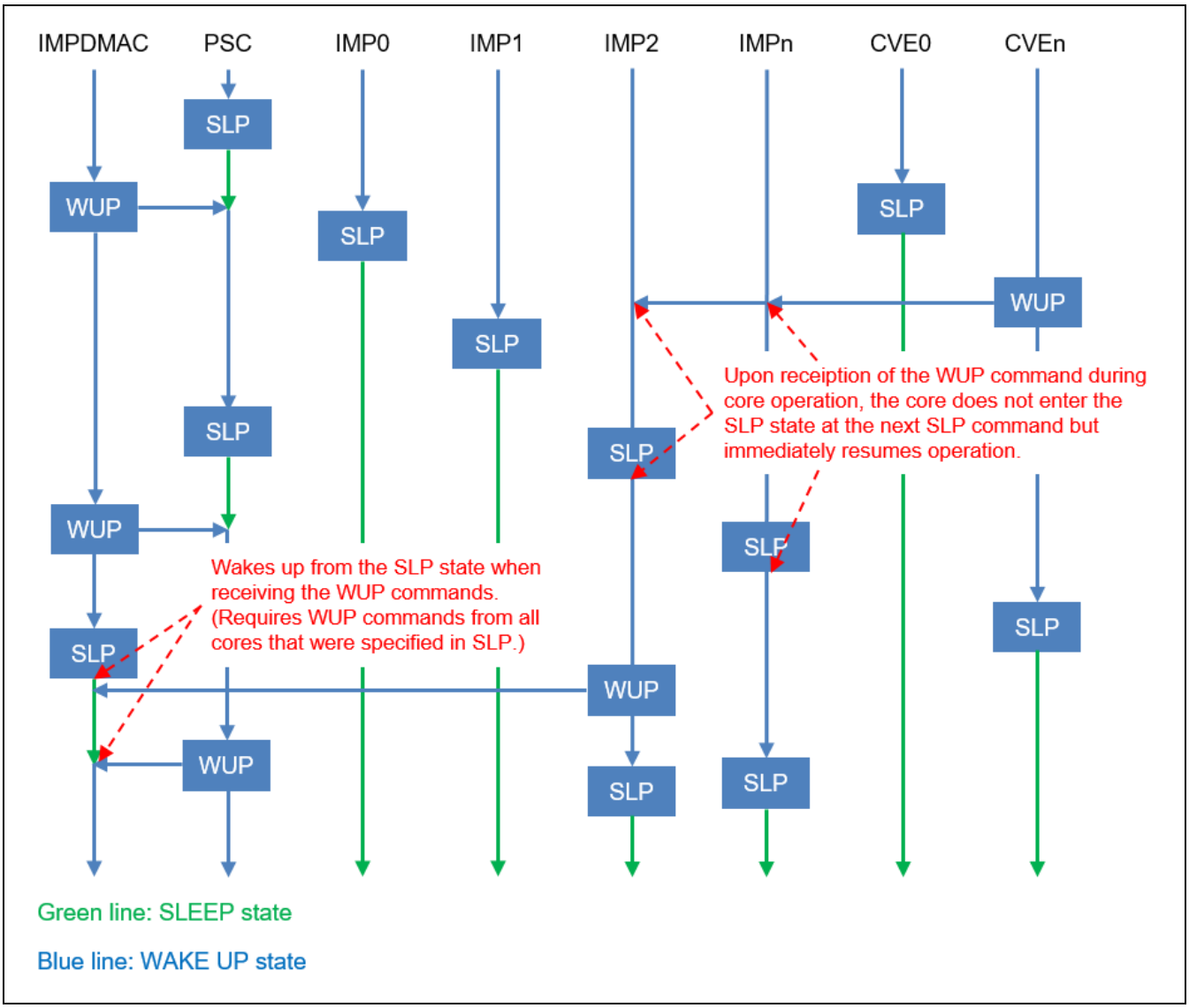



**Figure U32.11.2 Vertex Issue behavior**

## U32.12 Configuration Registers

While executing CVengine, access from Arm is prohibited except for Status Register1(SR1). Also, it is prohibited to change values using WPR command while executing threads. The read values from un-mapped addresses are undefined. Access to addresses other than those in the following lists of registers is prohibited, otherwise operation cannot be guaranteed.

The registers having address from H'780 to H'790 are duplicated to each thread and accessible from the thread program only. Host CPU cannot access them.

Value of the registers with "WPR support" checked can be set using WPR command in the Command List.

**Table U32.12.2 Register Configuration - Control Register**

| Name | Symbol | R/W | Offset | Access Size | Initial Value | Reset | WPR Support |
|---|---|---|---|---|---|---|---|
| Version Control Register 0 | VCR0 | R | H'0000 | 32 | *1 | — | — |
| ReSeT Register | RSTR | R/W | H'0008 | 32 | H'00000000 | H | — |

| Name | Symbol | R/W | Offset | Access Size | Initial Value | Reset | WPR Support |
|---|---|---|---|---|---|---|---|
| Control Register | CR | R/W | H'000C | 32 | H'00000000 | S/H | — |
| Status Register 0 | SR0 | R | H'0010 | 32 | H'00000000 | S/H | — |
| Status Register 1 | SR1 | R | H'0014 | 32 | H'00000000 | S/H | — |
| Status Clear Register 1 | SCR1 | -/W | H'0018 | 32 | H'00000000 | — | — |
| Interrupt Control Register 1 | ICR1 | R/W | H'001C | 32 | H'00000000 | S/H | — |
| Interrupt Mask Register 1 | IMR1 | R/W | H'0020 | 32 | H'00000000 | S/H | — |
| Status Register 2 | SR2 | R | H'0024 | 32 | H'00000000 | S/H | — |
| Status Clear Register 2 | SCR2 | -/W | H'0028 | 32 | H'00000000 | — | — |
| Interrupt Control Register 2 | ICR2 | R/W | H'002C | 32 | H'00000000 | S/H | — |
| Status Register 3 | SR3 | R | H'0034 | 32 | H'00000000 | S/H | — |
| Shader Core MaSK Register | SCMSKR | R/W | H'0120 | 32 | H'00000000 | S/H | — |
| Thread Group Control Regisgter0 | TGCR0 | R/W | H'012C | 32 | H'00000000 | S/H | — |
| SBO CouNT Register0 | SBOCNTR0 | R | H'0130 | 32 | H'00000000 | S/H | — |
| Descriptor List Start Address Register | DLSAR | R/W | H'0180 | 32 | H'00000000 | S/H | — |
| CLiPping MINimum Register | CLPMINR | R/W | H'0200 | 32 | H'00000000 | S/H | √ |
| CLiPping MAXimum Register | CLPMAXR | R/W | H'0204 | 32 | H'1FFF1FFF | S/H | √ |
| Shader Performance Control Register | SPCR | R/W | H'0280 | 32 | H'00000000 | S/H | √ |
| Shader Performance BuSY CouNTer Register | SPBSYCNTR | R/W | H'0284 | 32 | H'00000000 | S/H | √ |
| Shader Performance Busy Counter THReshold Register | SPBCTHRR | R/W | H'028C | 32 | H'FFFFFFFF | S/H | √ |
| Data Cache BYPass Area 0 MIN | DCBYPA0MIN | R/W | H'02A0 | 32 | H'FFFFFFFF | S/H | √ |
| Data Cache BYPass Area 0 MAX | DCBYPA0MAX | R/W | H'02A4 | 32 | H'00000000 | S/H | √ |
| Data Cache BYPass Area Enable | DCBYPAEN | R/W | H'02B0 | 32 | H'00000000 | S/H | √ |
| Data Cache BYPass Area ConTroL | DCBYPACTL | R/W | H'02B4 | 32 | H'00000000 | S/H | √ |
| Vs PC Start Address Register | VPCSAR | R/W | H'0320 | 32 | H'00000000 | S/H | √ |
| Vs SINI Start Address Register | VSINISAR | R/W | H'0324 | 32 | H'00000000 | S/H | √ |
| Ps PC Start Address Register | PPCSAR | R/W | H'0328 | 32 | H'00000000 | S/H | √ |
| Ps SINI Start Address Register | PSINISAR | R/W | H'032C | 32 | H'00000000 | S/H | √ |
| Vs I$ Base Address Register | VIBAR | R/W | H'0340 | 32 | Undefined | — | √ |
| Ps I$ Base Address Register | PIBAR | R/W | H'0344 | 32 | Undefined | — | √ |
| SBO Base Address Register 0 | SBOBAR0 | R/W | H'0360 | 32 | Undefined | — | √ |
| SBO Max Memory Size Register 0 | SBOMMSR0 | R/W | H'0364 | 32 | H'00000000 | S/H | √ |
| SBO Control Register 0 | SBOCR0 | R/W | H'0368 | 32 | H'00000000 | S/H | √ |
| DMac Control Register 0 | DMCR0 | R/W | H'03A0 | 32 | H'00003000 | S/H | √ |
| DMac EXTernal MEMory OFfSet ADDress Register | DMEXTMEMOFSADDR | R/W | H'03A4 | 32 | H'00000000 | — | √ |
| DMac GWM Base ADDress Register | DMGWMBADDR | R/W | H' 03A8 | 32 | H'00000000 | — | √ |
| DMac GWM STRide Register | DMGWMSTRR | R/W | H'03AC | 32 | H'00000000 | — | √ |
| DMac LeNGth Register | DMLNGR | R/W | H'03B0 | 32 | H'00000000 | — | √ |
| CL BReaK ADDRess Register | CLBRKADDRR | R/W | H'03F4 | 32 | H'00000000 | S/H | — |

| Name | Symbol | R/W | Offset | Access Size | Initial Value | Reset | WPR Support |
|---|---|---|---|---|---|---|---|
| CL CoNDitional GoSuB Register | CLCNDGSBR | R/W | H'03FC | 32 | H'00000000 | S/H | — |
| IMaGe Base Address Register 0 | IMGBAR0 | R/W | H'0400 | 32 | Undefined | — | √ |
| IMaGe STRide Register 0 | IMGSTR0 | R/W | H'0404 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 0 | IMGC0R0 | R/W | H'0408 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 1 | IMGBAR1 | R/W | H'040C | 32 | Undefined | — | √ |
| IMaGe STRide Register 1 | IMGSTR1 | R/W | H'0410 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 1 | IMGC0R1 | R/W | H'0414 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 2 | IMGBAR2 | R/W | H'0418 | 32 | Undefined | — | √ |
| IMaGe STRide Register 2 | IMGSTR2 | R/W | H'041C | 32 | Undefined | — | √ |
| IMaGe Control0 Register 2 | IMGC0R2 | R/W | H'0420 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 3 | IMGBAR3 | R/W | H'0424 | 32 | Undefined | — | √ |
| IMaGe STRide Register 3 | IMGSTR3 | R/W | H'0428 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 3 | IMGC0R3 | R/W | H'042C | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 4 | IMGBAR4 | R/W | H'0430 | 32 | Undefined | — | √ |
| IMaGe STRide Register 4 | IMGSTR4 | R/W | H'0434 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 4 | IMGC0R4 | R/W | H'0438 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 5 | IMGBAR5 | R/W | H'043C | 32 | Undefined | — | √ |
| IMaGe STRide Register 5 | IMGSTR5 | R/W | H'0440 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 5 | IMGC0R5 | R/W | H'0444 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 6 | IMGBAR6 | R/W | H'0448 | 32 | Undefined | — | √ |
| IMaGe STRide Register 6 | IMGSTR6 | R/W | H'044C | 32 | Undefined | — | √ |
| IMaGe Control0 Register 6 | IMGC0R6 | R/W | H'0450 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 7 | IMGBAR7 | R/W | H'0454 | 32 | Undefined | — | √ |
| IMaGe STRide Register 7 | IMGSTR7 | R/W | H'0458 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 7 | IMGC0R7 | R/W | H'045C | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 8 | IMGBAR8 | R/W | H'0460 | 32 | Undefined | — | √ |
| IMaGe STRide Register 8 | IMGSTR8 | R/W | H'0464 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 8 | IMGC0R8 | R/W | H'0468 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 9 | IMGBAR9 | R/W | H'046C | 32 | Undefined | — | √ |
| IMaGe STRide Register 9 | IMGSTR9 | R/W | H'0470 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 9 | IMGC0R9 | R/W | H'0474 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 10 | IMGBAR10 | R/W | H'0478 | 32 | Undefined | — | √ |
| IMaGe STRide Register 10 | IMGSTR10 | R/W | H'047C | 32 | Undefined | — | √ |
| IMaGe Control0 Register 10 | IMGC0R10 | R/W | H'0480 | 32 | H'00000100 | S/H | √ |
| IMaGe Base Address Register 11 | IMGBAR11 | R/W | H'0484 | 32 | Undefined | — | √ |
| IMaGe STRide Register 11 | IMGSTR11 | R/W | H'0488 | 32 | Undefined | — | √ |
| IMaGe Control0 Register 11 | IMGC0R11 | R/W | H'048C | 32 | H'00000100 | S/H | √ |
| SYNChronization Control Register 0 | SYNCCR0 | R/W | H'04C0 | 32 | H'FFFFFFFF | S/H | — |
| SYNChronization Control Register 1 | SYNCCR1 | R/W | H'04C4 | 32 | H'FFFFFFFF | S/H | — |
| SYNChronization Control Register 2 | SYNCCR2 | R/W | H'04C8 | 32 | H'FFFFFFFF | S/H | — |
| SYNChronization Control Register 3 | SYNCCR3 | R/W | H'04CC | 32 | H'FFFFFFFF | S/H | — |
| TG Shader Enable Register1 | TGSEN1 | R/W | H'0708 | 32 | H'000000FF | S/H | √ |

| Name | Symbol | R/W | Offset | Access Size | Initial Value | Reset | WPR Support |
|---|---|---|---|---|---|---|---|
| Unified Shader PerFormance Counter ConTrol Register | USPFCCTLR | R/W | H'0740 | 32 | H'00000000 | S/H | √ |
| PerFormance Counter TOTAL Register0 | PFCTOTALR0 | R/W | H'0744 | 32 | H'00000000 | S/H | √ |
| Unified Shader ConTRoL Register 0 | USCTRL0 | R/W | H'0758 | 32 | H'00010000 | S/H | √ |
| TGDMac Control Register0 | TGDMCR0 | R/W | H'0780 | 32 | H'00003000 | S/H | — |
| TGDMac EXTernal MEMory OFfSet ADDress Register | TGDMEXTMEMOFSADDR | R/W | H'0784 | 32 | H'00000000 | S/H | — |
| TGDMac LWM Base ADDress Register | TGDMLWMBADDR | R/W | H'0788 | 32 | H'00000000 | S/H | — |
| TGDMac LWM STRide Register | TGDMLWMSTRR | R/W | H'078C | 32 | H'00000000 | S/H | — |
| TGdmac LeNGth Register | TGLNGR | R/W | H'0790 | 32 | H'00000000 | S/H | — |
| TGSBO CouNT Register0 | TGSBOCNTR0 | R | H'07A0 | 32 | H'00000000 | S/H | — |
| UNiform Register n (n = 0 to 63) | UNRn | R/W | H'0C00 to H'0CFC | 32 | Undefined | S/H | √ |

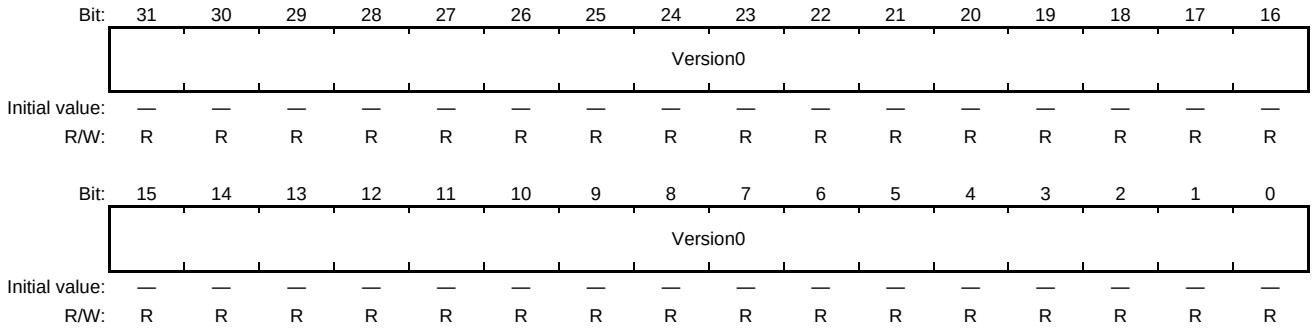
*1 Product dependent

### U32.12.1 Register Description

**[Legend]**

| | |
|---|---|
| Initial value: | Register value after a reset |
| —: | Undefined value |
| R/W: | Readable/writable. The written value can be read. |
| R: | Read-only. The write value should be the same as the read value. Only modify the target bit (Read Modify Write) because the initial value should not be changed. |
| -/W: | Write-only. The read value is undefined. |

#### U32.12.1.1Version Control Register 0 (VCR0)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | Version0 | H'00000007 | R | These bits indicate the version of this module for R-Car V4M. |

### U32.12.1.2ReSeT Register (RSTR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | SWRST |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 0 | SWRST | 0 | R/W | Software Reset<br>Software Reset is supplied to CVengine Core if this bit is "1".<br>It is required to write "0" to this bit to release the software reset before writing other registers.<br>0: Release the software reset<br>1: Supply the software reset<br>Setting this bit to 1 is prohibited when the PS bit in the CR register is 1. In that case, operation is not guaranteed. While CL, Thread or TGDMAC is being executed, the PS bit shows “1”. Data cache is invalidated after a software reset. It is prohibited to read or write the CVengine registers (accept reading this register) while software reset is supplied to the CVengine. |

### U32.12.1.3Control Register (CR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PS |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 0 | PS | 0 | R/W | Processing Start<br>If "1" is written to this bit, CVengine Core Issue starts fetching and executing CL from the address specified in DLSAR register. During the execution, "1" is read from this bit. Otherwise, "0" is read. It is prohibited to write "1" and "0" during its execution. |

### U32.12.1.4Status Register 0 (SR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INTCODE | | | | | | | | TRAPCODE | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | INTCODE | H'00 | R | INT interrupt code<br>When INT command in the command list is executed, the CODE field of the INT command that can be defined by the user is copied to this field. Refer to Section U32.13.7 . This field is updated regardless of the setting of ICR register. |
| 23 to 16 | TRAPCODE | H'00 | R | TRAP interrupt code<br>When TRAP command in the command list is executed, the CODE field of the TRAP command that can be defined by the user is copied to this field. Refer to Section U32.13.5 . This field is updated regardless of the setting of ICR register. |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |

### U32.12.1.5Status Register 1 (SR1)

Each bit of this register is only set to 1 when the corresponding bit in the interrupt control register (ICR) is 1. When any bit of this register is set to 1 while the corresponding bit in the interrupt mask register (IMR) is 0, an interrupt is generated. Each bit of this register is cleared to 0 when 1 is written to the corresponding bit in the status clearing register (SCR).

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | WUPCOVF | USINT | USIER | — | CLBRK | — | — | — | — | — | SBO0ME | PBCOVF | INT | IER | TRAP |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 15 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 14 | WUPCOVF | 0 | R | WUP Receive Counter overflow<br>Refer to U30G. IMP-X7 System, Inter module synchronization for the details when using this bit. |
| 13 | USINT | 0 | R | INT instruction interruption of thread. |
| 12 | USIER | 0 | R | Command error decode<br>This bit shows that an illegal instruction is decoded in the thread program.<br>0: An illegal instruction has not been decoded.<br>1: An illegal instruction has been decoded.<br>Additional condition the illegal instruction error takes place is described in Table U32.10.1. |
| 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 10 | CLBRK | 0 | R | CL BREAK<br>0: CL does not break<br>1: CL break<br>CL break cannot be used if CLBRKENB of ICR1 register is set to 0. Set CLBRKENB of ICR1 register to 1 to use CL break function.<br>Command list continues execution after clearing this bit using CLBRKCLR bit of SCR1 register. |
| 9 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 4 | SBO0ME | 0 | R | SBO0 memory size error |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | PBCOVF | 0 | R | Performance Busy Counter Overflow<br>0: A performance counter overflow has not been detected.<br>1: A performance counter overflow has been detected.<br>Refer to Section U32.8.8 for detail. |
| 2 | INT | 0 | R | INT command decode<br>This bit shows that INT command is decoded. Once this bit gets "1", the CL execution is stopped unless this status becomes "0".<br>0: An INT command has not been decoded.<br>1: An INT command has been decoded. |
| 1 | IER | 0 | R | Undefined command decode in CL or GOSUB overflow/underflow<br>This bit shows that undefined command is decoded. Once an undefined command is decoded, the CL execution is stopped.<br>0: An unknown command has not been decoded.<br>1: An unknown command has been decoded.<br>Reset IMP CVe by using SWRST bit of RSTR register after the IER error. |
| 0 | TRAP | 0 | R | TRAP command decode<br>This bit shows that TRAP command is decoded, and the CL execution finishes.<br>0: A TRAP command has not been decoded.<br>1: A TRAP command has been decoded. |

### U32.12.1.6Status Clear Register 1 (SCR1)

Writing 1 to each effective bit of this register clears the corresponding bit in status register 1 (SR1) to 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | WUPCOVFCLR | USINTCLR | USIERCLR | — | CLBRKCLR | — | — | — | — | — | SBO0MECLR | PBCOVFCLR | INTCLR | IERCLR | TRACLR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | -W | -/W | -/W | R | -/W | R | R | R | R | R | -/W | -/W | -/W | -/W | -/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 15 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 14 | WUPCOVFCLR | 0 | -/W | By writing "1" to this bit, the WUPCOVF bit in SR1 register is cleared. |
| 13 | USINTCLR | 0 | -/W | By writing "1" to this bit, the USINT bit in SR1 register is cleared. |
| 12 | USIERCLR | 0 | -/W | By writing "1" to this bit, the USIER bit in SR1 register is cleared. |
| 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 10 | CLBRKCLR | 0 | -/W | By writing "1" to this bit, the CLBRK bit in SR1 register is cleared.<br>Command list continues execution after clearing CLBRK bit of SR1 register. |
| 9 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 4 | SBO0MECLR | 0 | -/W | By writing "1" to this bit, the SBO0ME bit in SR1 register is cleared. |
| 3 | PBCOVFCLR | 0 | -/W | By writing "1" to this bit, the PBCOVF bit in SR1 register is cleared. |
| 2 | INTCLR | 0 | -/W | By writing "1" to this bit, the INT bit in SR1 register is cleared. |
| 1 | IERCLR | 0 | -/W | By writing "1" to this bit, the IER bit in SR1 register is cleared. |
| 0 | TRACLR | 0 | -/W | By writing "1" to this bit, the TRAP bit in SR1 register is cleared. |

### U32.12.1.7Interrupt Control Register 1 (ICR1)

When each effective bit of this register is 0, the corresponding bit in status register 1 (SR1) is not set to 1. Changing value of this register does not affect current value of SR1 register.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | WUPCOVFENB | USINTENB | USIERENB | — | CLBRKENB | — | — | — | — | — | SBO0MEENB | PBCOVFENB | INTENB | IERENB | TRAENB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | R/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 15 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 14 | WUPCOVFENB | 0 | R/W | By writing "1" to this bit, the WUPCOVF bit in SR1 register can be set. |
| 13 | USINTENB | 0 | R/W | By writing "1" to this bit, the USINT bit in SR1 register can be set. |
| 12 | USIERENB | 0 | R/W | By writing "1" to this bit, the USIER bit in SR1 register can be set. |
| 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 10 | CLBRKENB | 0 | R/W | By writing "1" to this bit, the CLBRK bit in SR1 register can be set.<br>When this bit is 0, CL break operation is not possible. |
| 9 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 4 | SBO0MEENB | 0 | R/W | By writing "1" to this bit, the SBO0ME bit in SR1 register can be set. |
| 3 | PBCOVFENB | 0 | R/W | By writing "1" to this bit, the PBCOVF bit in SR1 register can be set. |
| 2 | INTENB | 0 | R/W | By writing "1" to this bit, the INT bit in SR1 register can be set. |
| 1 | IERENB | 0 | R/W | By writing "1" to this bit, the IER bit in SR1 register can be set. |
| 0 | TRAENB | 0 | R/W | By writing "1" to this bit, the TRAP bit in SR1 register can be set. |

### U32.12.1.8Interrupt Mask Register 1 (IMR1)

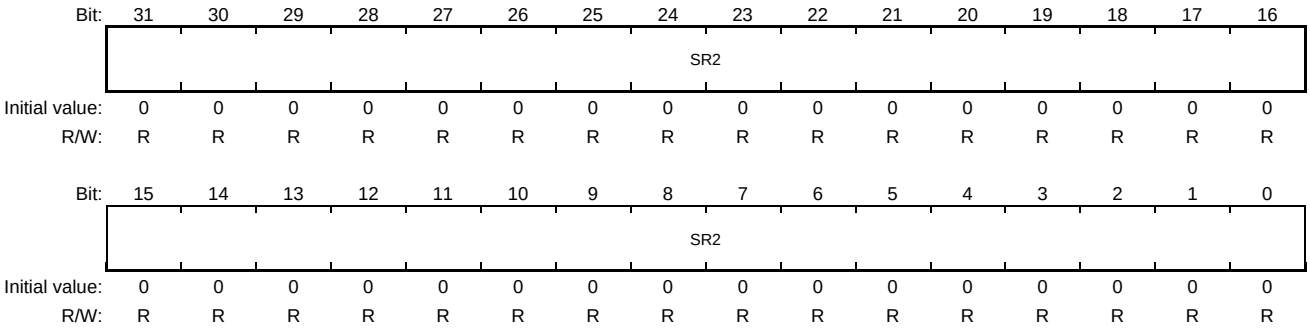
An interrupt is generated when any bit of status register 1 (SR1) is set to 1 while the corresponding bit in this register is 0.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | WUPCOVFMSK | USINTMSK | USIERMSK | — | CLBRKMSK | — | — | — | — | — | SBO0MEMSK | PBCOVFMSK | INTMSK | IERMSK | TRAMSK |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R | -/W | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W |

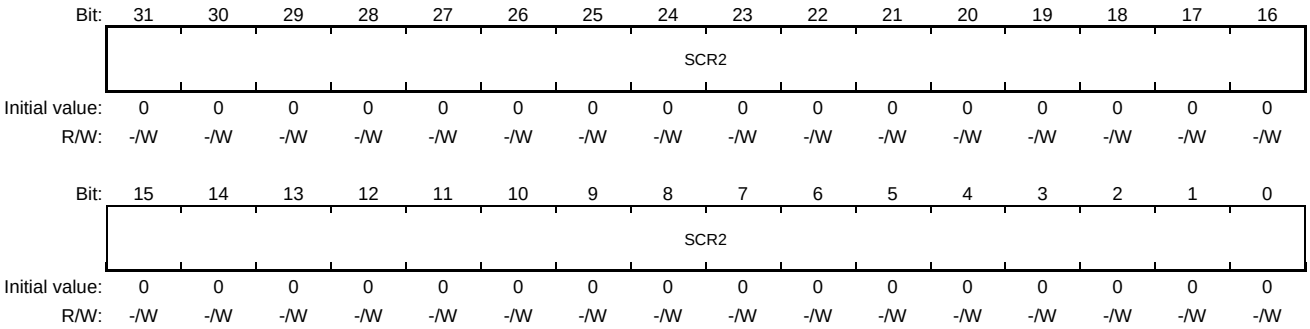
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 15 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 14 | WUPCOVFMSK | 0 | R/W | By writing "1" to this bit, the interrupt occurred by WUPCOVF bit in SR1 register is masked. |
| 13 | USINTMSK | 0 | R/W | By writing "1" to this bit, the interrupt occurred by USINT bit in SR1 register is masked. |
| 12 | USIERMSK | 0 | R/W | By writing "1" to this bit, the interrupt occurred by USIER bit in SR1 register is masked. |
| 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 10 | CLBRKMSK | 0 | -/W | By writing "1" to this bit, the interrupt occurred by CLBRK bit in SR1 register is masked. This bit is always read as 0. |
| 9 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 4 | SBO0MEMSK | 0 | R/W | By writing "1" to this bit, the interrupt occurred by SBO0ME bit in SR1 register is masked. |
| 3 | PBCOVFMSK | 0 | R/W | By writing "1" to this bit, the interrupt occurred by PBCOVF bit in SR1 register is masked. |
| 2 | INTMSK | 0 | R/W | By writing "1" to this bit, the interrupt occurred by INT bit in SR1 register is masked. |
| 1 | IERMSK | 0 | R/W | By writing "1" to this bit, the interrupt occurred by IER bit in SR1 register is masked. |
| 0 | TRAMSK | 0 | R/W | By writing "1" to this bit, the interrupt occurred by TRAP bit in SR1 register is masked. |

### U32.12.1.9Status Register 2 (SR2)



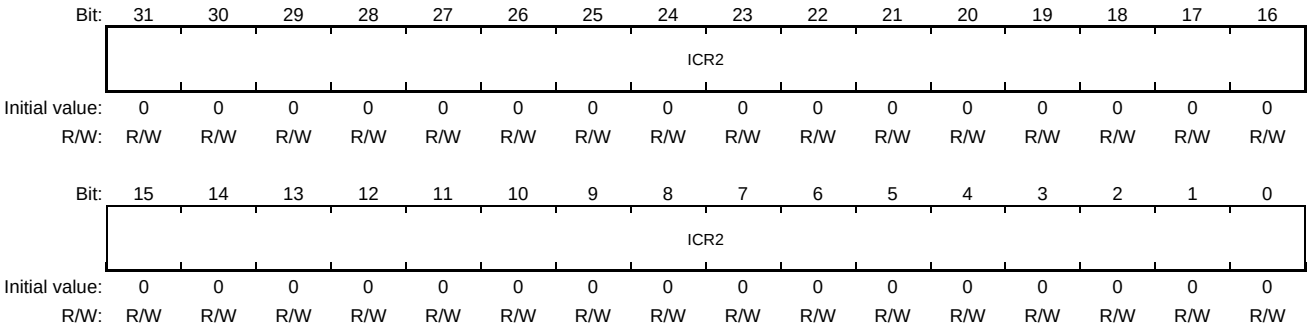

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SR2 | All 0 | R | INT command decode for thread<br>These bits show that INT command is decoded by thread. Once each bit gets "1", the CL execution is stopped unless each status becomes "0".<br>0: An INT command has not been decoded by thread.<br>1: An INT command has been decoded by thread.<br>bit31: core3.thread7, bit30: core3.thread6, bit29: core3.thread5, bit28: core3.thread4, bit27: core3.thread3, bit26: core3.thread2, bit25: core3.thread1, bit24: core3.thread0<br>bit23: core2.thread7, bit22: core2.thread6, bit21: core2.thread5, bit20: core2.thread4, bit19: core2.thread3, bit18: core2.thread2, bit17: core2.thread1, bit16: core2.thread0<br>bit15: core1.thread7, bit14: core1.thread6, bit13: core1.thread5, bit12: core1.thread4, bit11: core1.thread3, bit10: core1.thread2, bit9: core1.thread1, bit8: core1.thread0<br>bit7: core0.thread7, bit6: core0.thread6, bit5: core0.thread5, bit4: core0.thread4, bit3: core0.thread3, bit2: core0.thread2, bit1: core0.thread1, bit0: core0.thread0 |

### U32.12.1.10Status Clear Register 2 (SCR2)



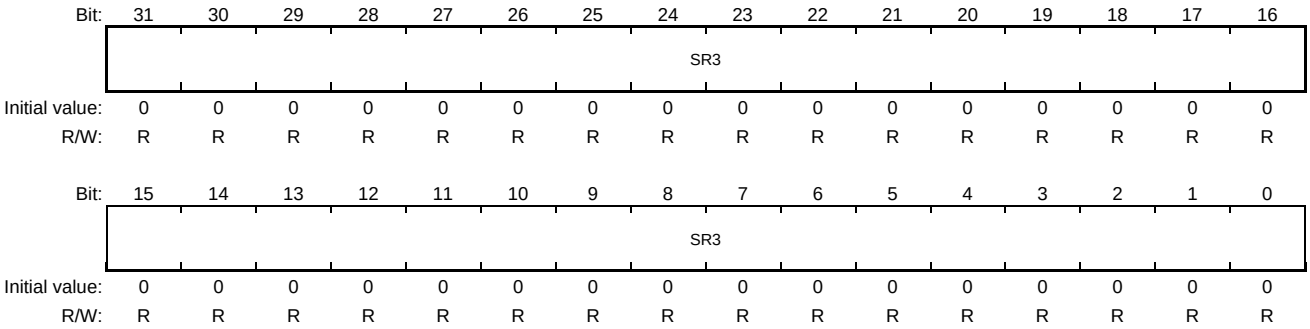

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SCR2 | All 0 | -/W | By writing "1" to each bit, each bit in SR2 register is cleared.<br>bit31: core3.thread7, bit30: core3.thread6, bit29: core3.thread5, bit28: core3.thread4, bit27: core3.thread3, bit26: core3.thread2, bit25: core3.thread1, bit24: core3.thread0<br>bit23: core2.thread7, bit22: core2.thread6, bit21: core2.thread5, bit20: core2.thread4, bit19: core2.thread3, bit18: core2.thread2, bit17: core2.thread1, bit16: core2.thread0<br>bit15: core1.thread7, bit14: core1.thread6, bit13: core1.thread5, bit12: core1.thread4, bit11: core1.thread3, bit10: core1.thread2, bit9: core1.thread1, bit8: core1.thread0<br>bit7: core0.thread7, bit6: core0.thread6, bit5: core0.thread5, bit4: core0.thread4, bit3: core0.thread3, bit2: core0.thread2, bit1: core0.thread1, bit0: core0.thread0 |

### U32.12.1.11Interrupt Control Register 2 (ICR2)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | ICR2 | All 0 | R/W | By writing "1" to a bit of ICR2, an interrupt of the corresponding thread is setting the USINT bit of SR1 when USINTENB of ICR1 is set.<br>bit31: core3.thread7, bit30: core3.thread6, bit29: core3.thread5, bit28: core3.thread4, bit27: core3.thread3, bit26: core3.thread2, bit25: core3.thread1, bit24: core3.thread0<br>bit23: core2.thread7, bit22: core2.thread6, bit21: core2.thread5, bit20: core2.thread4, bit19: core2.thread3, bit18: core2.thread2, bit17: core2.thread1, bit16: core2.thread0<br>bit15: core1.thread7, bit14: core1.thread6, bit13: core1.thread5, bit12: core1.thread4, bit11: core1.thread3, bit10: core1.thread2, bit9: core1.thread1, bit8: core1.thread0<br>bit7: core0.thread7, bit6: core0.thread6, bit5: core0.thread5, bit4: core0.thread4, bit3: core0.thread3, bit2: core0.thread2, bit1: core0.thread1, bit0: core0.thread0 |

### U32.12.1.12Status Register 3 (SR3)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SR3 | All 0 | R | Indicates whether any illegal instruction has been decoded. To clear any bit of this register, software reset is needed.<br>0: No illegal instruction has been decoded in the thread.<br>1: An illegal instruction has been decoded in the thread.<br>bit31: core3.thread7, bit30: core3.thread6, bit29: core3.thread5, bit28: core3.thread4, bit27: core3.thread3, bit26: core3.thread2, bit25: core3.thread1, bit24: core3.thread0<br>bit23: core2.thread7, bit22: core2.thread6, bit21: core2.thread5, bit20: core2.thread4, bit19: core2.thread3, bit18: core2.thread2, bit17: core2.thread1, bit16: core2.thread0<br>bit15: core1.thread7, bit14: core1.thread6, bit13: core1.thread5, bit12: core1.thread4, bit11: core1.thread3, bit10: core1.thread2, bit9: core1.thread1, bit8: core1.thread0<br>bit7: core0.thread7, bit6: core0.thread6, bit5: core0.thread5, bit4: core0.thread4, bit3: core0.thread3, bit2: core0.thread2, bit1: core0.thread1, bit0: core0.thread0 |

### U32.12.1.13Shader Core MaSK Register (SCMSKR)

When only 1 core is activated, 8 out of 32 threads work. This register is to reduce power consumption. To use debugger, remain this register in default.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | SCMSKR | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 4 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 3 to 0 | SCMSKR | All 0 | R/W | Specify if each core in a cluster is used or not.<br>[Bit3] 0: Core3 is used. 1: Core3 is not used.<br>[Bit2] 0: Core2 is used. 1: Core2 is not used.<br>[Bit1] 0: Core1 is used. 1: Core1 is not used.<br>[Bit0] 0: Core0 is used. 1: Core0 is not used. |

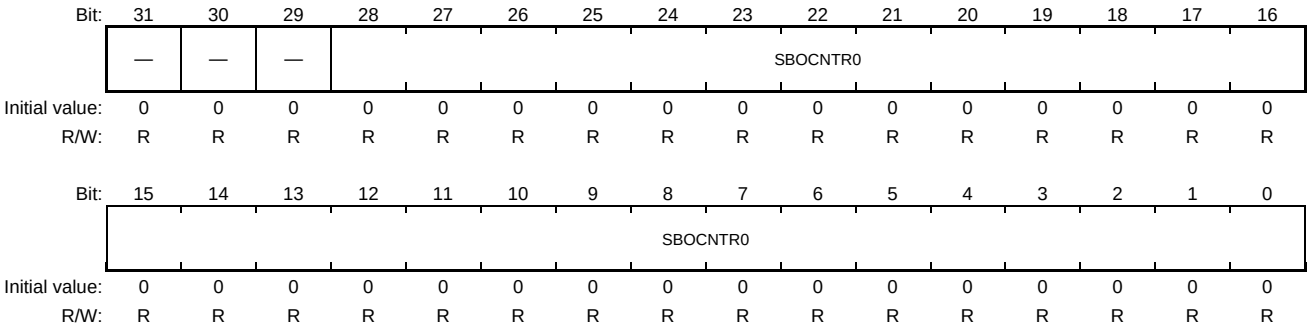
### U32.12.1.14Thread Group Control Register 0 (TGCR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | TGRSEL | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 2 to 0 | TGRSEL | All 0 | R/W | To select which core to read the performance counter.<br>0 : Read PFCTOTALR0 from Core 0<br>1 : Read PFCTOTALR0 from Core 1<br>2 : Read PFCTOTALR0 from Core 2<br>3 : Read PFCTOTALR0 from Core 3<br>This field works with SELUSPFC. Refer to U32.12.1.50 . |

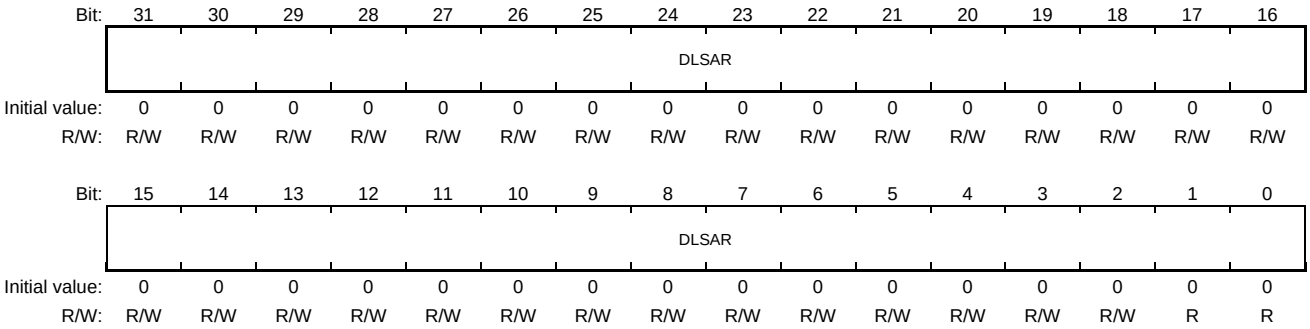
### U32.12.1.15SBO CouNT Register0 (SBOCNTR0)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 28 to 2<br>1,0 | SBOCNTR0 | All 0 | R | It expresses "SBO0 Output Number" (4byte unit).<br>These bits are cleared by reset, Processing Start, and writing 1 to the SBOCNTCLR bit in SBOCR0.<br>When SBOBAR0 is written, this register must be cleared.<br>The values of the lowest 2 bits are fixed to 0. |

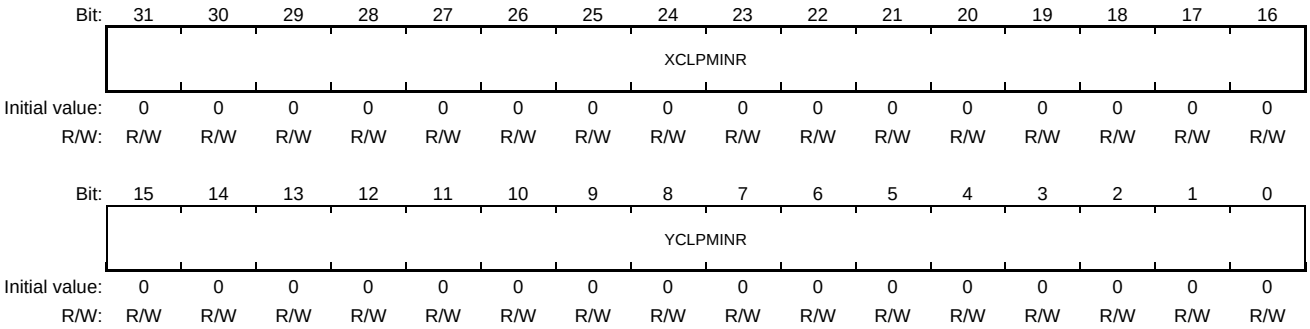
### U32.12.1.16Descriptor List Start Address Register (DLSAR)

The “Descriptor List” is an alias of “Command List”.



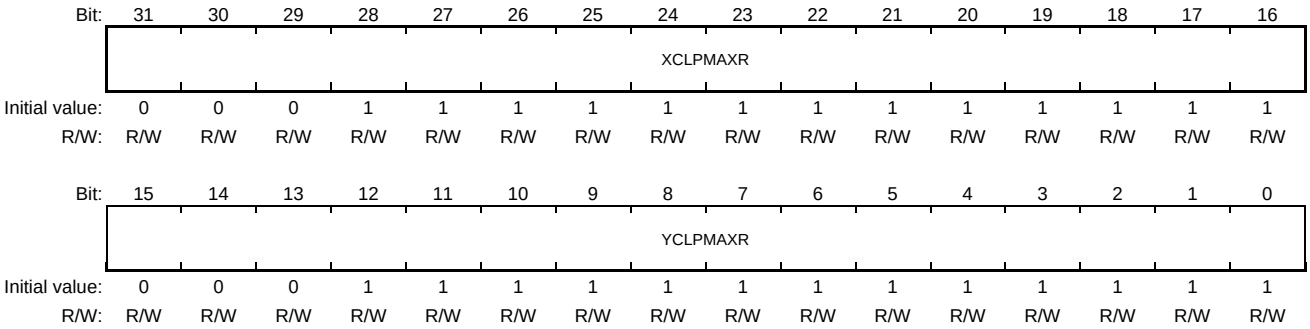

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | DLSAR | All 0 | R/W | Descriptor List start address |
| 1,0 | | | R | This register specifies the start address of CL execution. The start address should be specified as 4-byte alignment. The values of the lowest 2 bits are fixed to 0. |

### U32.12.1.17CLiPping MINimum Register (CLPMINR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | XCLPMINR | H'0000 | R/W | X's clipping minimum area (-32768 - 32767) |
| 15 to 0 | YCLPMINR | H'0000 | R/W | Y's clipping minimum area (-32768 - 32767) |

### U32.12.1.18CLiPping MAXimum Register (CLPMAXR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | XCLPMAXR | H'1FFF | R/W | X's clipping maximum area (-32768 - 32767) |
| 15 to 0 | YCLPMAXR | H'1FFF | R/W | Y's clipping maximum area (-32768 - 32767) |

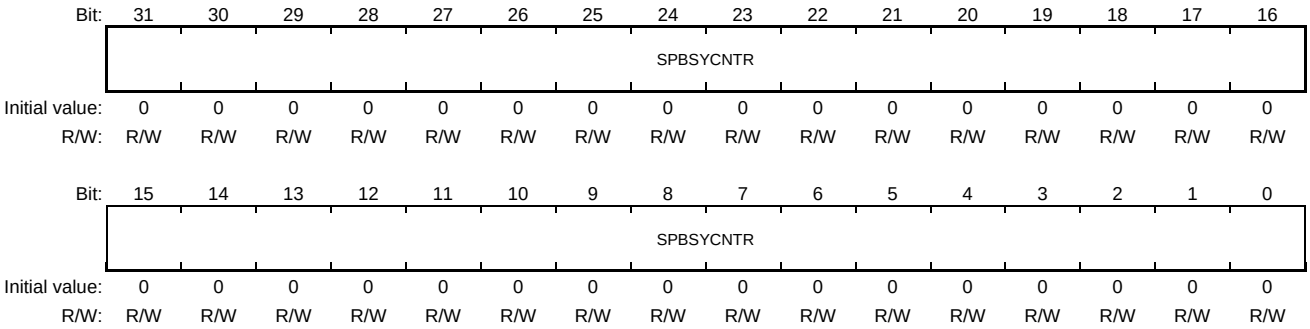
### U32.12.1.19Shader Performance Control Register (SPCR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | BSYCNTENB |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 0 | BSYCNTENB | 0 | R/W | BuSY CouNTer ENaBle<br>Counts cycle numbers of CVengine cluster is busy at cluster clock frequency. Value must be set while CVengine cluster is not working.<br>0: not counting busy cycles of CVengine cluster.<br>1: count busy cycles of CVengine cluster. |

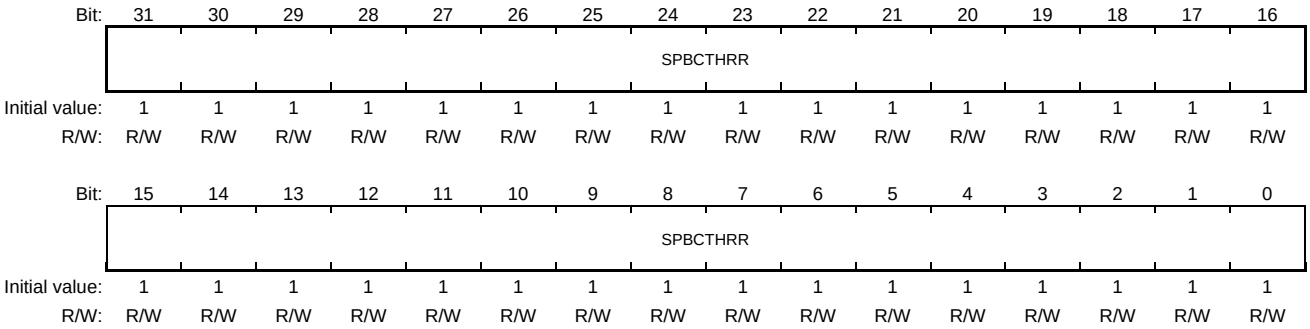
### U32.12.1.20Shader Performance BuSY CouNTer Register (SPBSYCNTR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SPBSYCNTR | All 0 | R/W | While SPCR.BSYCNTENB is 1, this register counts cycles of CVengine clusters busy. CVengine cluster is busy while the PS bit in the CR register is 1.<br>When accessing from CPU, the value of this register must be written or read while CVengine cluster is not in processing state.<br>Write 0 to this register to clear the counter.<br>Hardware does not take care of any overflow handlings, so software must take care of it. |

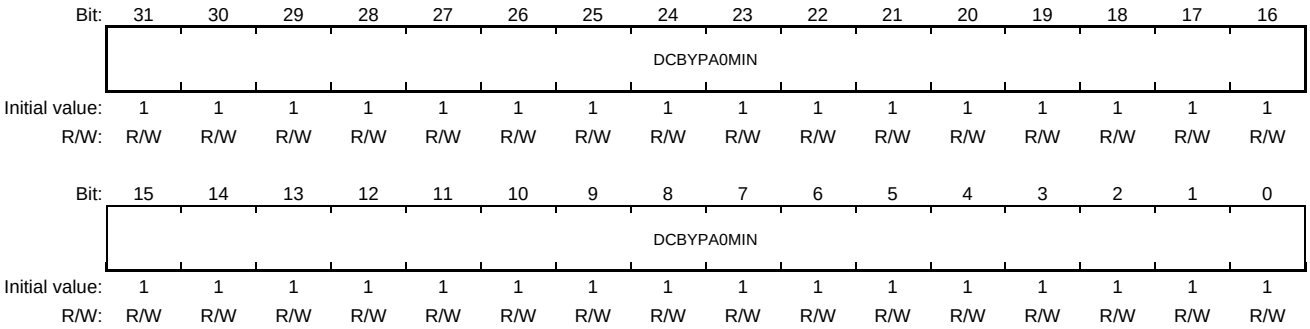
### U32.12.1.21Shader Performance Busy Counter THReshold Register (SPBCTHRR)

This register is used for the Guard Timer. Refer to Section U32.8.8 for detail.



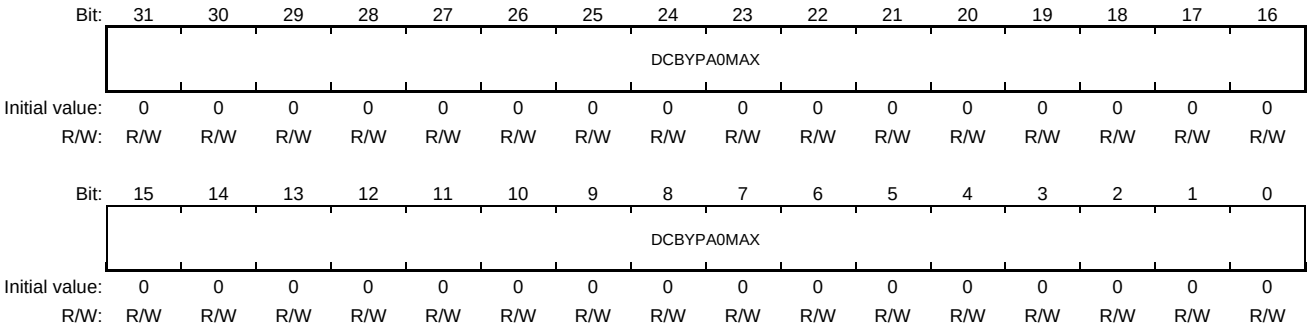

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | SPBCTHRR | All 1 | R/W | Shader Performance Busy Counter Threshold value.<br>This register is compared with SPBSYCNTR.<br>When accessing from CPU, the value of this register must be written or read while CVengine cluster is not in processing state. |

### U32.12.1.22Data Cache BYPass Area 0 MIN (DCBYPA0MIN)



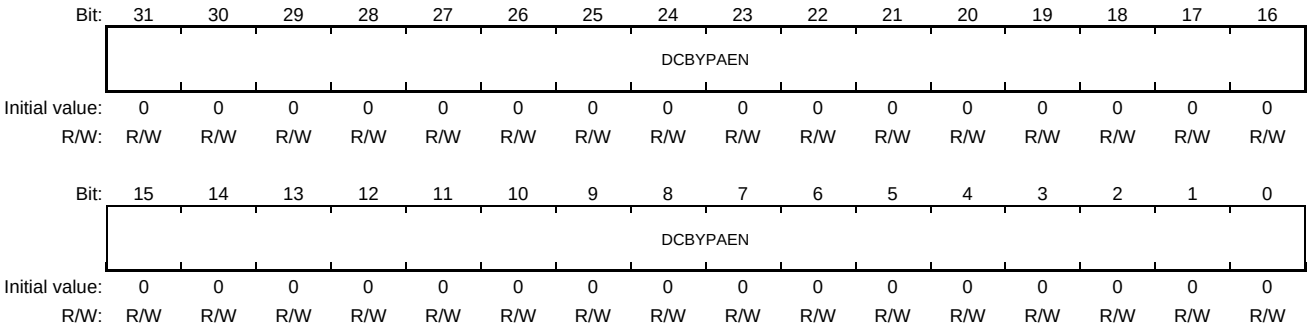

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DCBYPA0MIN | All 1 | R/W | This register holds memory address of the lower border of the D$ bypass memory area. |

### U32.12.1.23Data Cache BYPass Area 0 MAX (DCBYPA0MAX)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DCBYPA0MAX | All 0 | R/W | This register holds memory address of the higher border of the D$ bypass memory area. |

### U32.12.1.24Data Cache BYPass Area Enable (DCBYPAEN)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DCBYPAEN | All 0 | R/W | Specify the thread to enable D$ bypassing. Refer to section U32.8.9 .<br>bit31: Core3 Thread7, bit30: Core3 Thread6,<br>bit29: Core3 Thread5, bit28: Core3 Thread4,<br>bit27: Core3 Thread3, bit26: Core3 Thread2<br>bit25: Core3 Thread1, bit24: Core3 Thread0,<br>bit23: Core2 Thread7, bit22: Core2 Thread6,<br>bit21: Core2 Thread5, bit20: Core2 Thread4,<br>bit19: Core2 Thread3, bit18: Core2 Thread2<br>bit17: Core2 Thread1, bit16: Core2 Thread0,<br>bit15: Core1 Thread7, bit14: Core1 Thread6,<br>bit13: Core1 Thread5, bit12: Core1 Thread4,<br>bit11: Core1 Thread3, bit10: Core1 Thread2<br>bit9: Core1 Thread1, bit8: Core1 Thread0,<br>bit7: Core0 Thread7, bit6: Core0 Thread6,<br>bit5: Core0 Thread5, bit4: Core0 Thread4,<br>bit3: Core0 Thread3, bit2: Core0 Thread2<br>bit1: Core0 Thread1, bit0: Core0 Thread0, |

### U32.12.1.25Data Cache BYPass Area ConTroL (DCBYPACTL)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | DCBYPACTL |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 0 | DCBYPACTL | 0 | R/W | 1: Enable, 0: Disable<br>Enable Core# and Thread# replacement to the upper 16 bits of writing data for all threads which have the D$ bypassing enabled by respective bits of DCBYPAEN. Refer to section U32.8.9 . |

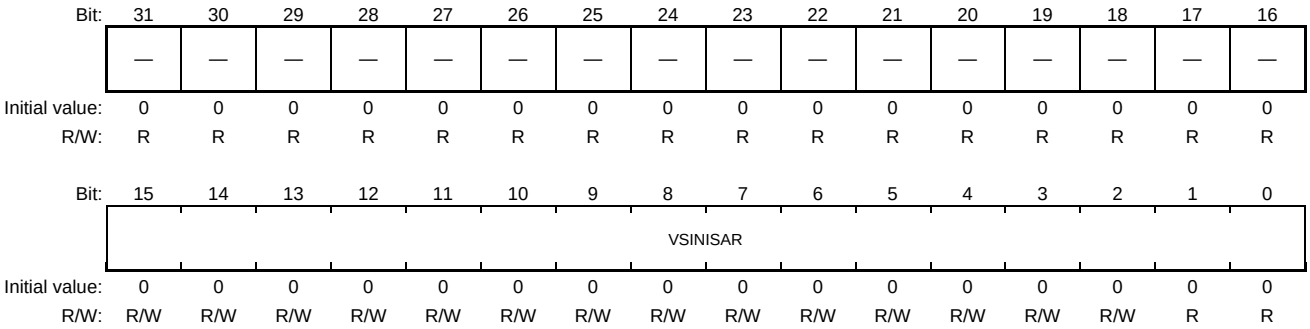
### U32.12.1.26Vs PC Start Address Register (VPCSAR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | VPCSAR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 2 | VPCSAR | H'0 0000 | R/W | Start address for master thread. |
| 1,0 | | | R | When RECT, POINT or POINTS is executed, thread (master) executes from the address of VPCSAR+VIBAR (VIBAR is the beginning of the instruction area and VPCSAR is the offset in the area). The area of the instructions must be less than 64KB.<br>The values of the lowest 2 bits are fixed to 0. |

### U32.12.1.27Vs SINI Start Address Register (VSINISAR)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 2 | VSINISAR | H'0 0000 | R/W | SINI start address for master thread. |
| 1,0 | | | R | When SINI is executed, thread (master) executes from the address of VSINISAR+VIBAR (VIBAR is the beginning of the instruction area and VSINISAR is the offset in the area). The area of the instructions must be less than 64KB.<br>The values of the lowest 2 bits are fixed to 0. |

### U32.12.1.28Ps PC Start Address Register (PPCSAR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PPCSAR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 2 | PPCSAR | H'0 0000 | R/W | Start address for slave thread. |
| 1,0 | | | R | When RECT, POINT or POINTS are executed, thread (slave) executes from the address of PPCSAR+PIBAR (PIBAR is the beginning of the instruction area and PPCSAR is the offset in the area). The area of the instructions must be less than 64KB.<br>The values of the lowest 2 bits are fixed to 0. |

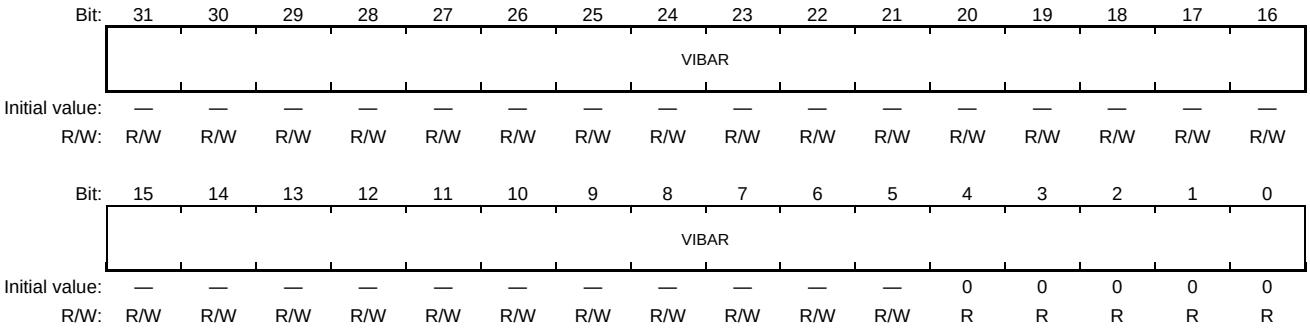
### U32.12.1.29Ps SINI Start Address Register (PSINISAR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSINISAR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R |

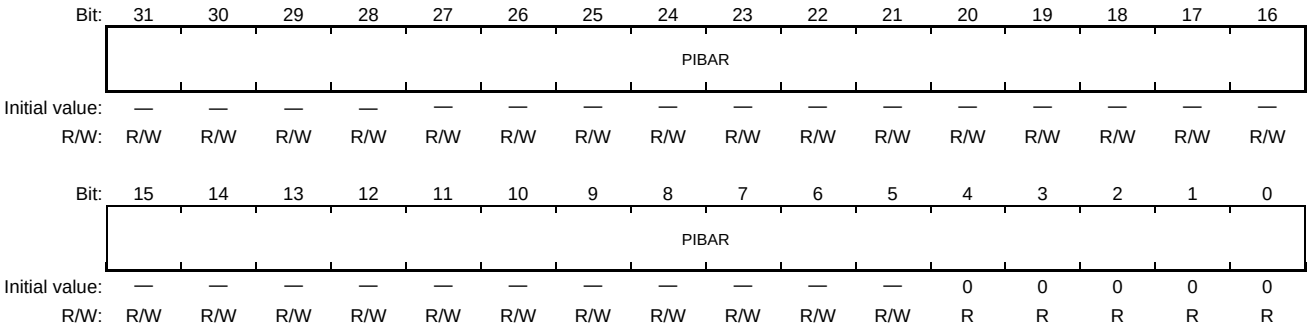
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 2 | PSINISAR | H'0 0000 | R/W | SINI start address for slave thread. |
| 1,0 | | | R | When SINI is executed, thread (slave) executes from the address of PSINISAR+PIBAR (PIBAR is the beginning of the instruction area and PSINISAR is the offset in the area). The area of the instructions must be less than 64KB.<br>The values of the lowest 2 bits are fixed to 0. |

### U32.12.1.30Vs I$ Base Address Register (VIBAR)



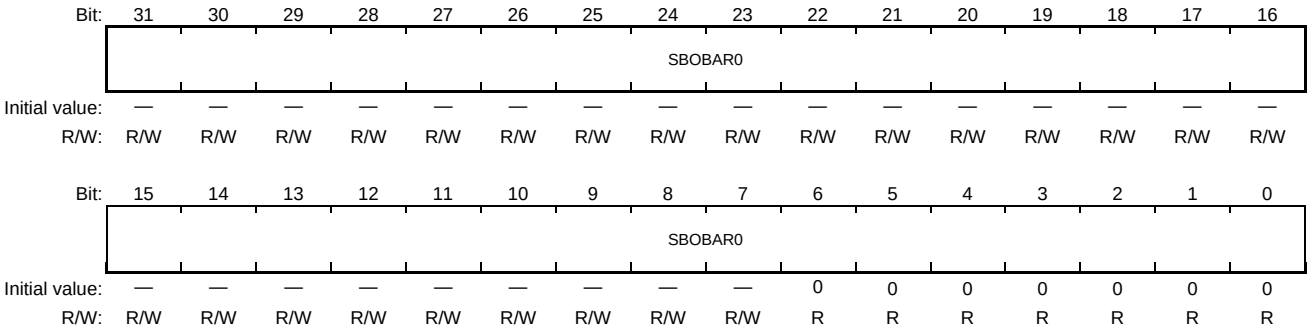

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | VIBAR | Undefined | R/W | These bits set the base address of I$ in the thread (master) |
| 4 to 0 | | All 0 | R | The address is obtained by multiplying the setting by 64. Note that bit 5 must be set to 0. Operation cannot be guaranteed if a value other than 0 is written in bit 5.<br>The values of the lowest 5 bits are fixed to 0. |

### U32.12.1.31Ps I$ Base Address Register (PIBAR)



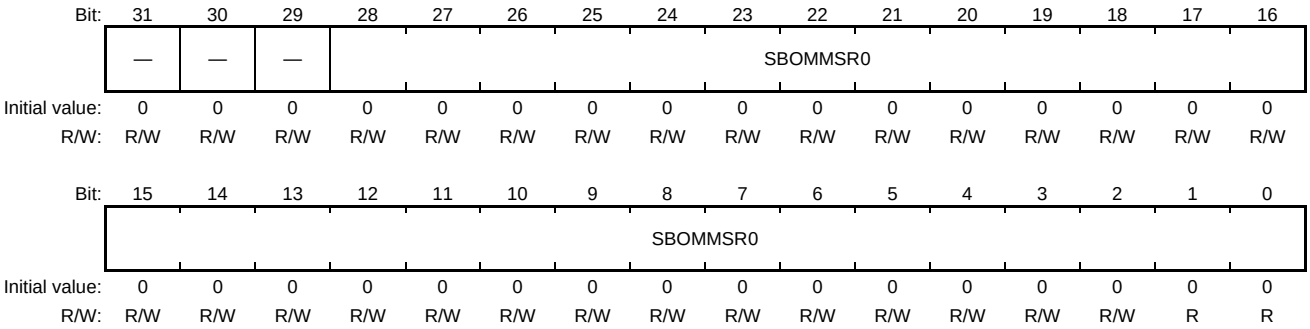

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | PIBAR | Undefined | R/W | These bits set the base address of I$ in thread (slave). |
| 4 to 0 | | All 0 | R | The address is obtained by multiplying the setting by 64. Note that bit 5 must be set to 0. Operation cannot be guaranteed if a value other than 0 is written in bit 5.<br>The values of the lowest 5 bits are fixed to 0.<br>The values must always be set same as VIBAR. |

### U32.12.1.32SBO Base Address Register 0 (SBOBAR0)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 7 | SBOBAR0 | Undefined | R/W | When this register is written, SBOCNTR0 must be cleared. |
| 6 to 0 | | All 0 | R | Note: Before rewriting this register, SBO0 must be flushed by SYNCM command.<br>The values of the lowest 7 bits are fixed to 0. |

### U32.12.1.33SBO Max Memory Size Register 0 (SBOMMSR0)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R/W | Reserved<br>These bits are always read as 0.<br>These bits must be set to 0 when these are written.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 28 to 2 | SBOMMSR0 | All 0 | R/W | This register must be set in 4-byte unit. |
| 1,0 | | | R | If it is over this limited value, the SBO0 is issued the transaction with byte-enable 0 and output the interrupt signal by setting SR1. SBO0ME bit.<br>Note: Before rewriting this register, SBO0 must be flushed by SYNCM command. Unless the interrupt status bit is cleared, no further interrupt is issued upon exceeding the limit.<br>The values of the lowest 2 bits are fixed to 0. |

### U32.12.1.34SBO Control Register 0 (SBOCR0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SBOCNTCLR | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | SIGNEDEXT | PCKMODE | XYPCK |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | SBOCNTCLR | 0 | R/W | By writing "1" to this bit, SBOCNTR0 is cleared. The read value is always 0. |
| 30 to 3 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 2 | SIGNEDEXT | 0 | R/W | By writing "1" to this bit, the SBO0 expands X and Y to 32-bit range as signed integer.<br>By writing "0" to this bit, the SBO0 expands X and Y to 32-bit range as unsigned integer. |
| 1 | PCKMODE | 0 | R/W | When XYPCK is “1”, this bit decides the way of concatenation about the coordinates.<br>0: {x[15:0], y[15:0]}<br>1: {y[15:0], x[15:0]}<br>When XYPCK is “0”, this bit has no effect. |
| 0 | XYPCK | 0 | R/W | By writing "1" to this bit, the SBO0 handles X(16-bit) and Y(16-bit) as a pair of data. |

### U32.12.1.35DMac Control Register 0 (DMCR0)

This register is for GWM-DMAC. The host processor cannot access this register during processing state, only Command list can access it.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

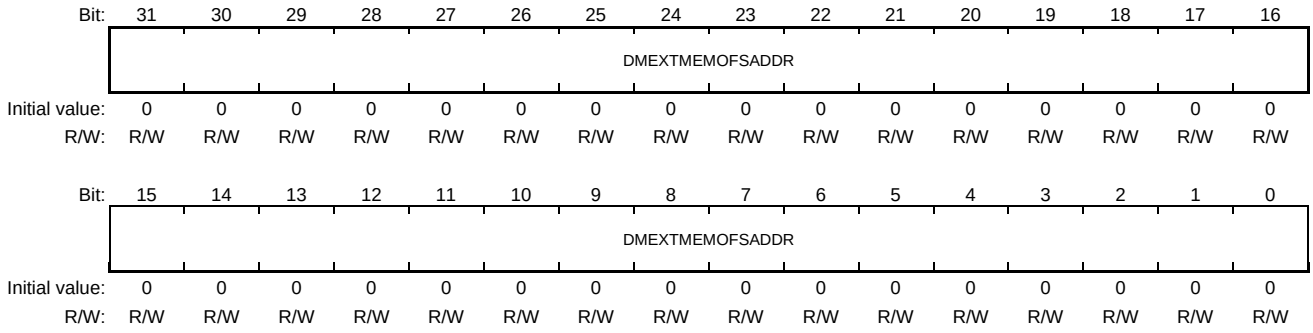
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | SELOFSADR | — | — | DIR | — | SELPLN | | | SELWAIT | — | DMACST | DMACEX |
| Initial value: | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R | R | R/W | R | R/W | R/W | R/W | R/W | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 13,12 | — | All 1 | R/W | Reserved<br>These bits are always read as 1.<br>These bits must be set to 1 when written. |
| 11 | SELOFSADR | 0 | R/W | 0: DMEXTMEMOFSADDR is used as X,Y coordinate<br>1: DMEXTMEMOFSADDR is used as the offset address in 32-bit format. |
| 10,9 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 8 | DIR | 0 | R/W | Specify the direction of DMA transfer.<br>0: GWM to the external memory<br>1: The external memory to GWM |
| 7 | — | 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 6 to 4 | SELPLN | 000 | R/W | Specify the plane number (0-7) which determines the address and the stride to calculate a base address of the external memory.<br>Addition of IMGBARn, where n is the plane number specified in this field, and DMEXTMEMOFSADDR makes the start address of the DMA transfer of the external memory. |
| 3 | SELWAIT | 0 | R/W | Specify the mode of SYNCS instruction of command list.<br>0: SYNCS exits when the DMAC transfer completes.<br>1: SYNCS exits when the DMAC startup completes. DMAC registers can be set for a next DMA transfer after the DMAC startup completes. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 2 | — | 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 1 | DMACST | 0 | R | This bit becomes "1" when DMAC is started and returns to "0" when DMAC transfer is finished. |
| 0 | DMACEX | 0 | R/W | By writing "1" to this bit, DMAC is started.<br>This bit becomes "0" when DMAC has been started.<br>Note: It is prohibited to write DMAC registers before this bit becomes "0". |

### U32.12.1.36DMac EXTernal MEMory OFfSet ADDress Register (DMEXTMEMOFSADDR)

This register is for GWM-DMAC. The host processor cannot access this register during processing state, only Command list can access it.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | DMEXTMEMOFSADDR | All 0 | R/W | SELOFSADR=0: bit31-16: X coordinate (signed) in bytes, bit15-0: Y coordinate (signed).<br>SELOFSADR=1: bit31-0: Offset address.<br>Addition of IMGBARn, where n is the plane number specified by DMCR0.SELPLN, and DMEXTMEMOFSADDR makes the start address of the DMA transfer of the external memory in bytes. |

### U32.12.1.37DMac GWM Base ADDress Register (DMGWMBADDR)

This register is for GWM-DMAC. The host processor cannot access this register during processing state, only Command list can access it.

Bit 31 to 16 of GWM address will be filled automatically with 0 by HW.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

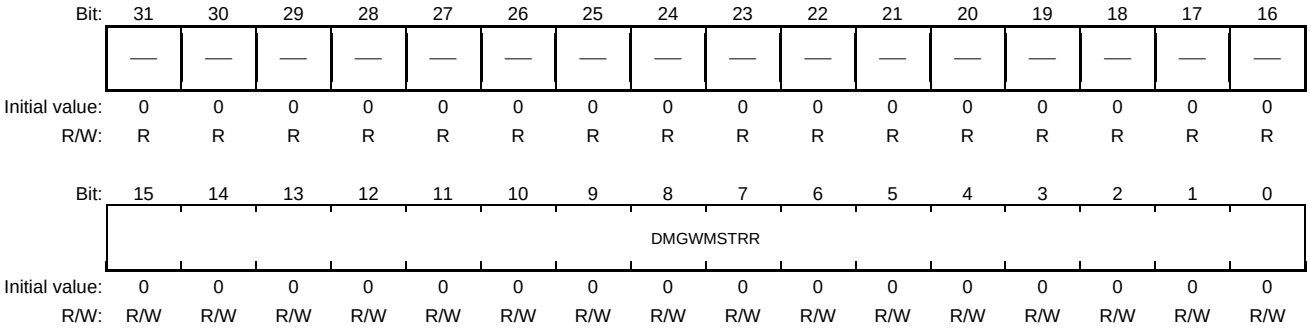
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMGWMBADDR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 0 | DMGWMBADDR | All 0 | R/W | Specify the start address for DMA transfer within GWM. |

### U32.12.1.38DMac GWM STRide Register (DMGWMSTRR)

This register is for GWM-DMAC. The host processor cannot access this register during processing state, only Command list can access it.

Bit 31 to 16 of GWM stride will be sign-extended automatically by HW.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 0 | DMGWMSTRR | All 0 | R/W | These bits set the signed stride of GWM by byte unit. |

### U32.12.1.39DMac LeNGth Register (DMLNGR)

This register is for GWM-DMAC. The host processor cannot access this register during processing state, only Command list can access it.

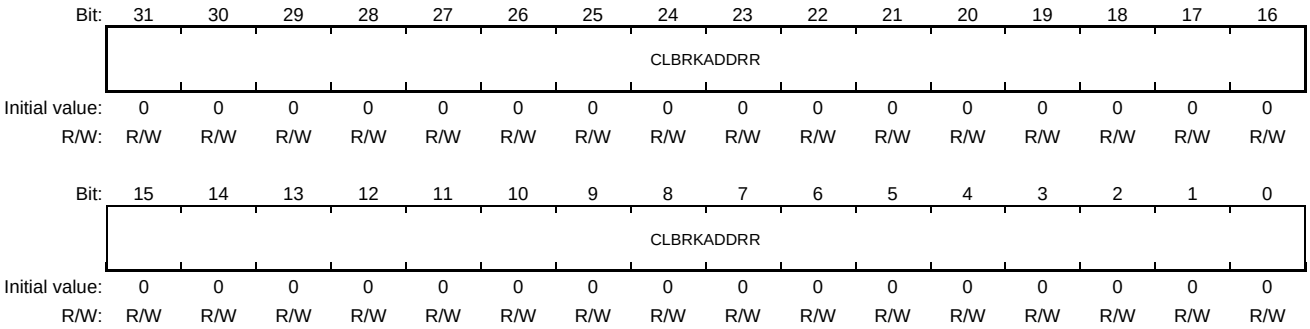
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | XLNG | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | YLNG | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 30 to 16 | XLNG | H'0000 | R/W | A width in the range of 1 to 16384 in byte unit. Value must satisfy the plane format alignment. |
| 15 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 14 to 0 | YLNG | H'0000 | R/W | A pixel height in the range of 1 to 16384. |

### U32.12.1.40CL BReaK ADDRess Register (CLBRKADDRR)

This register is for specification of the CL break address.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | CLBRKADDRR | H'00000000 | R/W | CL break address specification.<br>Specify a 32bit command list address (4byte aligned). Data address is prohibited. |

### U32.12.1.41CL CoNDitional GoSuB Register (CLCNDGSBR)

This register specifies the condition in case of a conditional GOSUB instruction.

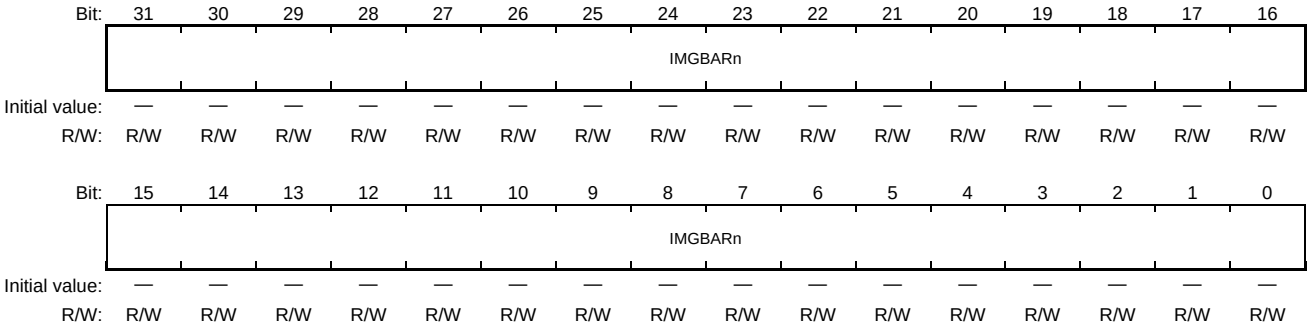
| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | CLCNDGSBR |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written. |
| 0 | CLCNDGSBR | H'0 | R/W | Condition for conditional GOSUB instruction:<br>. 1: GOSUB is executed<br>0: GOSUB is not executed, NOP is instead executed<br>(Bit 22 of the GOSUB command should be 1 for NOP operation) |

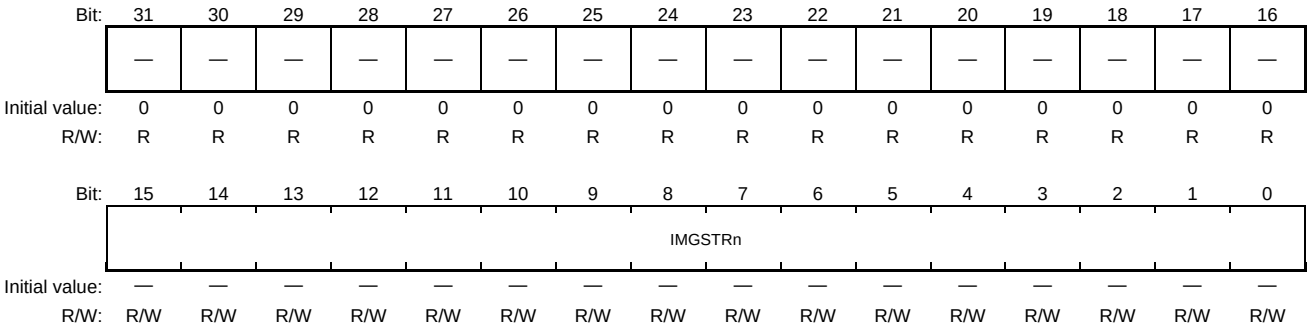
### U32.12.1.42IMaGe Base Address Register n (IMGBARn) (n = 0 to 11)

The number n (n = 0 to 11) of this register shall be set to the Control Registers CR8 and CR9 to select which plane to be read or written by a thread. Refer to section U32.6.2 .




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | IMGBARn | Undefined | R/W | Alignment must be done depending on the Pixel Format (IMGC0Rn.PFORMAT).<br>When Pixel Format is unsigned short or short, the lowest 1 bit of this register must be set to 0.<br>When Pixel Format is int or float, the lowest 2 bits of this register must be set to 0. |

### U32.12.1.43IMaGe STRide Register n (IMGSTRn) (n = 0 to 11)




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 0 | IMGSTRn | Undefined | R/W | IMGSTRn is a signed 16bit value.<br>When Pixel Format is unsigned char or char, this register must be aligned to 1-byte borders.<br>When Pixel Format is unsigned short or short, this register must be aligned to 2-byte borders.<br>When Pixel Format is int or float, this register must be aligned to 4-byte borders.<br>This register must be specified in 1-byte unit in any case. |

### U32.12.1.44IMaGe Control0 Register n (IMGC0Rn) (n = 0 to 11)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | CLPMODE | | | — | — | — | — | — | PFORMAT2 | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R | R | R/W | R/W | R/W |

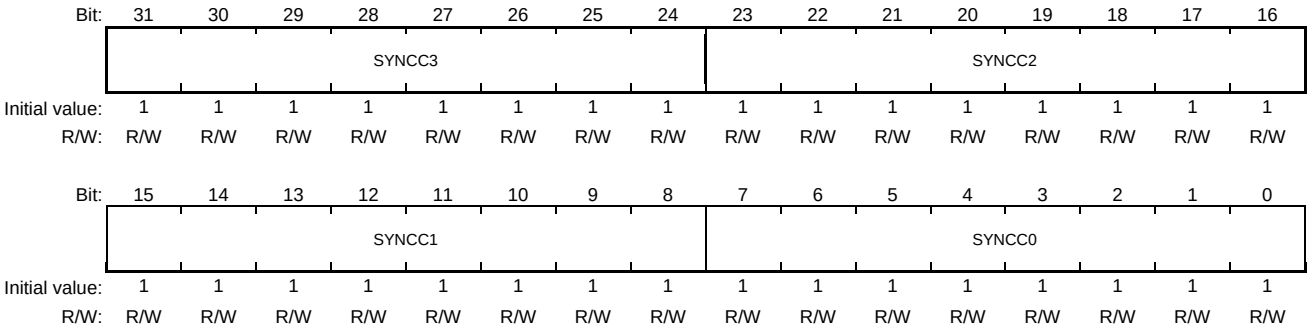
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | RNDMODE_F2I | | | — | — | — | SATENBO | — | PFORMAT | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R | R | R | R/W | R | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 27 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 26 to 24 | CLPMODE | 000 | R/W | Specify the Clipping Mode (Border Mode of OpenCV to set the extra padding to the image). Refer to section U32.8.7 .<br>3'b000: NORMAL (no border)<br>3'b001: REPLICATE<br>3'b010: REFLECT<br>3'b011: REFLECT101<br>3'b100: WRAP<br>3'b101: CONSTANT<br>3'b110: It is prohibited to write<br>3'b111: It is prohibited to write<br>Note: The number of times of the repetition is limited to one for each top/bottom/left/right of image size in the clipping (except for CONSTANT). |
| 23 to 19 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 18 to 16 | PFORMAT2 | 000 | R/W | Pixel format for thread (slave).<br>0: unsigned char<br>1: char<br>2: unsigned short<br>3: short<br>4: Reserved<br>5: int<br>6: Reserved<br>7: float |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 15 to 11 | — | 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 10 to 8 | RNDMODE_F2I | 001 | R/W | Float to Integer Format Conversion Round Mode<br>0: IEEE 754 round to nearest (Round half to even)<br>1: IEEE 754 round to zero<br>2: IEEE 754 round to positive infinity (Not supported)<br>3: IEEE 754 round to negative infinity (Not supported)<br>4: round to nearest up (Not supported)<br>5: round away from zero (Not supported)<br>6: Reserved<br>7: Reserved |
| 7 to 5 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 4 | SATENBO | 0 | R/W | Saturation Enable (Output)<br>0: Disable saturation function.<br>1: Enable saturation function.<br>If saturation occurs when the thread write some data to external memory, the output data will be treated as maximum value.<br>This function has no effect when PFORMAT is set to7 (float). |
| 3 | — | 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 2 to 0 | PFORMAT | 000 | R/W | Pixel Format for thread (master).<br>0: unsigned char<br>1: char<br>2: unsigned short<br>3: short<br>4: Reserved<br>5: int<br>6: Reserved<br>7: float |

### U32.12.1.45SYNChronization Control Register 0 (SYNCCR0)

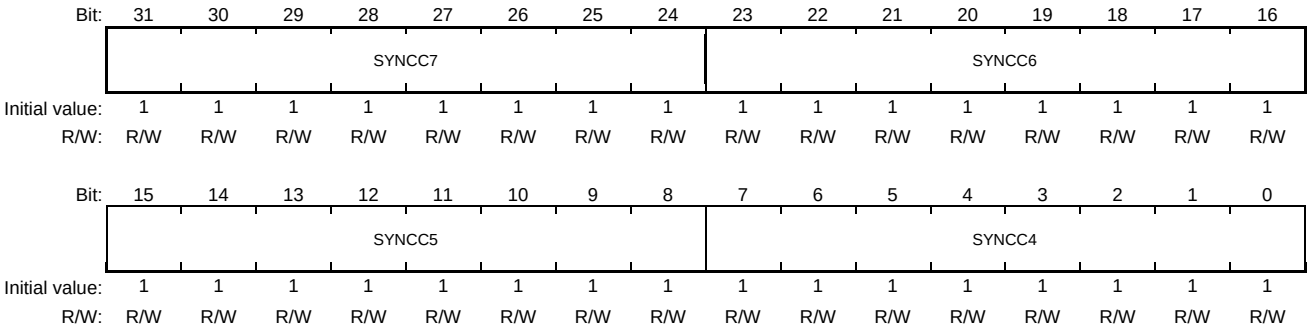
It is recommended to write this register while all the assigned cores are not running. Refer to Section U32.11.1 , Section U32.13.14 and Section U32.13.15 for details.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC3 | All 1 | R/W | Specify the core corresponding to bit 3 of CORE Number field of WUP and SLP commands.<br>[31:28] : Core Type<br>[27:24] : Core Number |
| 23 to 16 | SYNCC2 | All 1 | R/W | Specify the core corresponding to bit 2 of CORE Number field of WUP and SLP commands.<br>[23:20] : Core Type<br>[19:16] : Core Number |
| 15 to 8 | SYNCC1 | All 1 | R/W | Specify the core corresponding to bit 1 of CORE Number field of WUP and SLP commands.<br>[15:12] : Core Type<br>[11:8] : Core Number |
| 7 to 0 | SYNCC0 | All 1 | R/W | Specify the core corresponding to bit 0 of CORE Number field of WUP and SLP commands.<br>[7:4] : Core Type<br>[3:0] : Core Number |

### U32.12.1.46SYNChronization Control Register 1 (SYNCCR1)

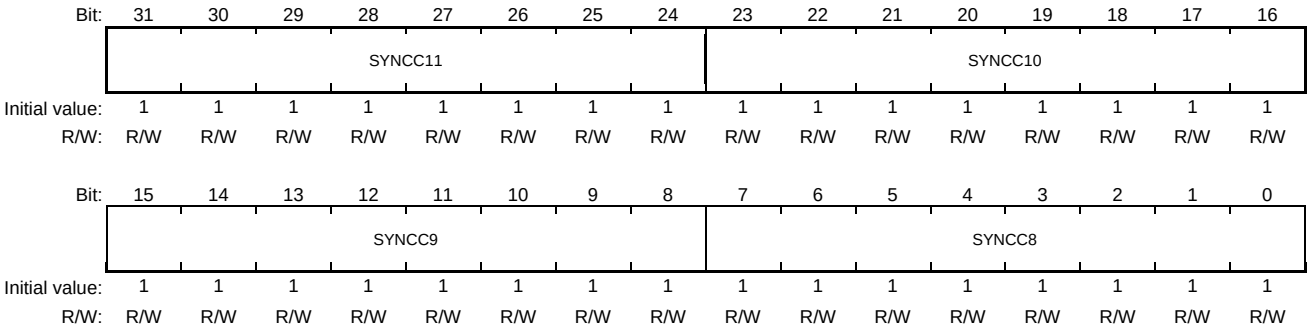
It is recommended to write this register while all the assigned cores are not running. Refer to Section U32.11.1 , Section U32.13.14 and Section U32.13.15 for details.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC7 | All 1 | R/W | Specify the core corresponding to bit 7 of CORE Number field of WUP and SLP commands.<br>[31:28] : Core Type<br>[27:24] : Core Number |
| 23 to 16 | SYNCC6 | All 1 | R/W | Specify the core corresponding to bit 6 of CORE Number field of WUP and SLP commands.<br>[23:20] : Core Type<br>[19:16] : Core Number |
| 15 to 8 | SYNCC5 | All 1 | R/W | Specify the core corresponding to bit 5 of CORE Number field of WUP and SLP commands.<br>[15:12] : Core Type<br>[11:8] : Core Number |
| 7 to 0 | SYNCC4 | All 1 | R/W | Specify the core corresponding to bit 4 of CORE Number field of WUP and SLP commands.<br>[7:4] : Core Type<br>[3:0] : Core Number |

### U32.12.1.47SYNChronization Control Register 2 (SYNCCR2)

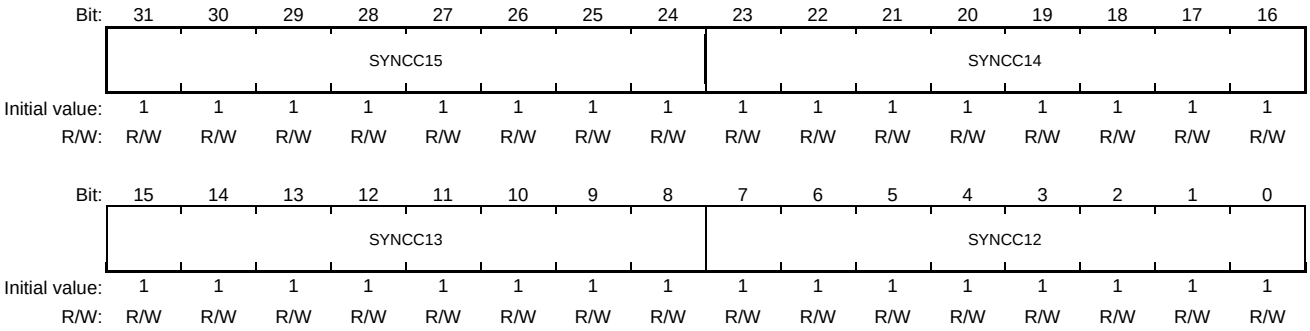
It is recommended to write this register while all the assigned cores are not running. Refer to Section U32.11.1 , Section U32.13.14 and Section U32.13.15 for details.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC11 | All 1 | R/W | Specify the core corresponding to bit 11 of CORE Number field of WUP and SLP commands.<br>[31:28] : Core Type<br>[27:24] : Core Number |
| 23 to 16 | SYNCC10 | All 1 | R/W | Specify the core corresponding to bit 10 of CORE Number field of WUP and SLP commands.<br>[23:20] : Core Type<br>[19:16] : Core Number |
| 15 to 8 | SYNCC9 | All 1 | R/W | Specify the core corresponding to bit 9 of CORE Number field of WUP and SLP commands.<br>[15:12] : Core Type<br>[11:8] : Core Number |
| 7 to 0 | SYNCC8 | All 1 | R/W | Specify the core corresponding to bit 8 of CORE Number field of WUP and SLP commands.<br>[7:4] : Core Type<br>[3:0] : Core Number |

### U32.12.1.48SYNChronization Control Register 3 (SYNCCR3)

It is recommended to write this register while all the assigned cores are not running. Refer to Section U32.11.1 , Section U32.13.14 and Section U32.13.15 for details.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | SYNCC15 | All 1 | R/W | Specify the core corresponding to bit 15 of CORE Number field of WUP and SLP commands.<br>[31:28] : Core Type<br>[27:24] : Core Number |
| 23 to 16 | SYNCC14 | All 1 | R/W | Specify the core corresponding to bit 14 of CORE Number field of WUP and SLP commands.<br>[23:20] : Core Type<br>[19:16] : Core Number |
| 15 to 8 | SYNCC13 | All 1 | R/W | Specify the core corresponding to bit 13 of CORE Number field of WUP and SLP commands.<br>[15:12] : Core Type<br>[11:8] : Core Number |
| 7 to 0 | SYNCC12 | All 1 | R/W | Specify the core corresponding to bit 12 of CORE Number field of WUP and SLP commands.<br>[7:4] : Core Type<br>[3:0] : Core Number |

### U32.12.1.49TG Shader Enable Register 1 (TGSEN1)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | SLVEN | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | MSTEN | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 23 to 16 | SLVEN | H'00 | R/W | Slave Thread Enable.<br>Specify Slave Thread threads.<br>When the SINI command is used, execution of Slave Thread starts from the PC set in the PSINISAR register. When Slave Thread has been started by Master Thread, execution starts from the PC set in the PPCSAR register.<br>[Bit23] 0: Does not use Thread 7 as Slave Thread. 1: Uses Thread 7 as Slave Thread.<br>[Bit22] 0: Does not use Thread 6 as Slave Thread. 1: Uses Thread 6 as Slave Thread.<br>[Bit21] 0: Does not use Thread 5 as Slave Thread. 1: Uses Thread 5 as Slave Thread.<br>[Bit20] 0: Does not use Thread 4 as Slave Thread. 1: Uses Thread 4 as Slave Thread.<br>[Bit19] 0: Does not use Thread 3 as Slave Thread. 1: Uses Thread 3 as Slave Thread.<br>[Bit18] 0: Does not use Thread 2 as Slave Thread. 1: Uses Thread 2 as Slave Thread.<br>[Bit17] 0: Does not use Thread 1 as Slave Thread. 1: Uses Thread 1 as Slave Thread.<br>[Bit16] 0: Does not use Thread 0 as Slave Thread. 1: Uses Thread 0 as Slave Thread.<br>Note: If both MSTEN and SLVEN are set to 0, the thread is disabled. It is prohibited to set both MSTEN and SLVEN to 1. |
| 15 to 8 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 7 to 0 | MSTEN | All 1 | R/W | Master Thread Enable.<br>Specify Master Thread.<br>When the SINI command is used, execution of Master Thread starts from the PC set in the VSINISAR register. When Master Thread has been started by Vertex Issue, execution starts from the PC set in the VPCSAR register.<br>[Bit7] 0: Does not use Thread 7 as Master Thread. 1: Uses Thread 7 as Master Thread.<br>[Bit6] 0: Does not use Thread 6 as Master Thread. 1: Uses Thread 6 as Master Thread.<br>[Bit5] 0: Does not use Thread 5 as Master Thread. 1: Uses Thread 5 as Master Thread.<br>[Bit4] 0: Does not use Thread 4 as Master Thread. 1: Uses Thread 4 as Master Thread.<br>[Bit3] 0: Does not use Thread 3 as Master Thread. 1: Uses Thread 3 as Master Thread.<br>[Bit2] 0: Does not use Thread 2 as Master Thread. 1: Uses Thread 2 as Master Thread.<br>[Bit1] 0: Does not use Thread 1 as Master Thread. 1: Uses Thread 1 as Master Thread.<br>[Bit0] 0: Does not use Thread 0 as Master Thread. 1: Uses Thread 0 as Master Thread.<br>Note: If both MSTEN and SLVEN are set to 0, the thread is disabled. It is prohibited to set both MSTEN and SLVEN to 1. |

### U32.12.1.50Unified Shader PerFormance Counter ConTrol Register (USPFCCTLR)

This register is for the performance counter which allows user to aquire detailed performance information from each thread in each core.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PFCENB | PFCTOVF | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | RW | RW | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | SELUSPFC | | | PFCMOD | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

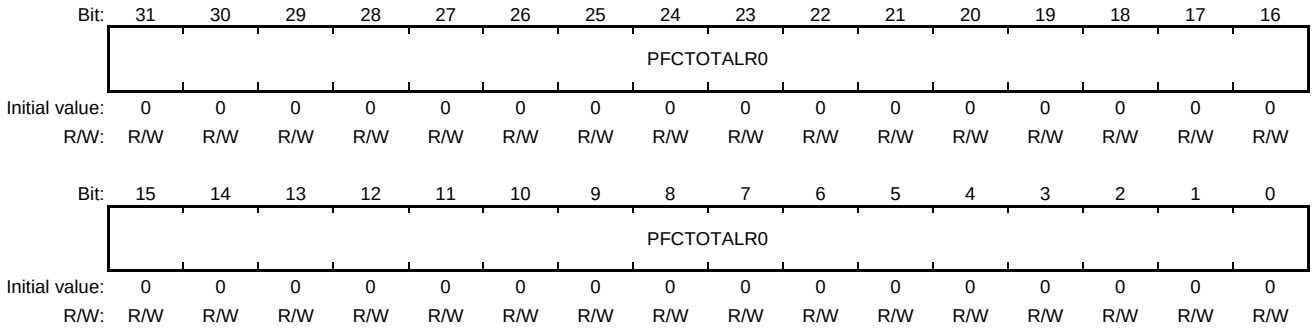
| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | PFCENB | 0 | R/W | PerFormance Counter ENaBle<br>0: Performance counter is disabled.<br>1: Performance counter is enabled.<br>Even if this bit is 1, performance counter is counted only during thread executing. But performance counter is not counted during SINI executing. |
| 30 | PFCTOVF | 0 | R/W | PerFormance Counter Total register OVerFlow<br>0: Performance Counter Total register did not overflow.<br>1: Performance Counter Total register did overflow. |
| 29 to 11 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 10 to 8 | SELUSPFC | All 0 | R/W | SELecting Unified Shader to use PerFormance Counter.<br>0: Select thread 0<br>1: Select thread 1<br>2: Select thread 2<br>3: Select thread 3<br>4: Select thread 4<br>5: Select thread 5<br>6: Select thread 6<br>7: Select thread 7<br>This field works with TGRSEL. Refer to U32.12.1.14 . |

| | | | | |
|---|---|---|---|---|
| 7 to 0 | PFCMOD | All 0 | R/W | PerFormance Counter MODe<br>0x00: Number of Thread Starts<br>0x01: Number of SINI command list instruction calls<br>0x02: Number of stall cycles due to Thread Starts<br>0x03: Number of execution cycles until TRAP<br>0x04: Number of instruction fetch cycles<br>0x05: Number of stalls during instruction fetches<br>0x06: Number of branches that were entered<br>0x07: Number of RCP0 (R14) accesses<br>0x08: Number of RCP1 (R15) accesses<br>0x09: Number of SQRT accesses<br>0x0a: Number of EXP accesses<br>0x0b: Number of LOG access cycles<br>0x0c: Number of stall cycles during RCP0 accesses<br>0x0d: Number of stall cycles during RCP1 accesses<br>0x0e: Number of stall cycles during SQRT accesses<br>0x0f: Number of stall cycles during EXP accesses<br>0x10: Number of stall cycles during LOG2 accesses<br>0x11: Number of ACTST calls<br>0x12: Number of stalls due to ACTST calls<br>0x13: Number of data dependency stall cycles<br>0x15: Number of LDW instruction execution cycles<br>0x16: Number of STW instruction execution cycles<br>0x17: Number of LWM/GWM access stall cycles<br>0x18: Number of data load cycles<br>0x19: Number of data store cycles<br>0x1a: Number of data load or store stall cycles<br>0x1f: Number of SBO (streaming buffer output) write cycles<br>0x20: Number of SBO (streaming buffer output) write stall cycles<br>0x28: Number of stall cycles caused by UDI<br>0x29: Number of execution cycles of FADD group<br>0x2a: Number of execution stall cycles of FADD group<br>0x2b: Number of execution cycles of FMUL group<br>0x2c: Number of execution stall cycles of FMUL group<br>0x2d: Number of execution cycles of MUL group<br>0x2e: Number of execution stall cycles of MUL group<br>0x2f: Number of execution cycles of ASM group<br>0x30: Number of execution stall cycles of ASM group<br>0x32: Number of execution cycles of CSCS group<br>0x33: Number of execution stall cycles of CSCS group<br>0x34: Number of execution cycles of CSUS group<br>0x35: Number of execution stall cycles of CSUS group<br>0x36: Number of execution cycles of L2F group<br>0x37: Number of execution stall cycles of L2F group<br>0x38: Number of execution cycles of MAC group<br>0x39: Number of execution stall cycles of MAC group<br>0x3a: Number of execution cycles of PACK group<br>0x3b: Number of execution stall cycles of PACK group<br>0x3c: Number of execution cycles of GASS group |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| | | | | 0x3d: Number of execution stall cycles of GASS group |
| | | | | 0x3e: Number of execution cycles of GMAD group |
| | | | | 0x3f: Number of execution stall cycles of GMAD group |
| | | | | 0x40: Number of execution cycles of LLROT group |
| | | | | 0x41: Number of execution stall cycles of LLROT group |
| | | | | 0x42: Number of execution cycles of RCPSQRT group |
| | | | | 0x43: Number of execution cycles of SIN group |
| | | | | 0x44: Number of execution cycles of COS group |
| | | | | 0x45: Number of execution stall cycles of RCPSQRT group |
| | | | | 0x46: Number of execution stall cycles of SIN group |
| | | | | 0x47: Number of execution stall cycles of COS group |
| | | | | 0x48: Number of stall cycles caused by LWM DMAC busy |
| | | | | 0x49: Number of execution stall cycles of slave busy |
| | | | | 0x4a: Number of execution cycles of SIMDABS group |
| | | | | 0x4b: Number of execution stall cycles of SIMDABS group |
| | | | | 0x4c: Number of execution cycles of SIMDADDSUB group |
| | | | | 0x4d: Number of execution stall cycles of SIMDADDSUB group |
| | | | | 0x4e: Number of execution cycles of SIMDCMP group |
| | | | | 0x4f: Number of execution stall cycles of SIMDCMP group |
| | | | | 0x50: Number of execution cycles of SIMDIMUL group |
| | | | | 0x51: Number of execution stall cycles of SIMDIMUL group |
| | | | | 0x52: Number of execution cycles of SIMDMAXMIN group |
| | | | | 0x53: Number of execution stall cycles of SIMDMAXMIN group |
| | | | | 0x54: Number of execution cycles of SIMDSHFT group |
| | | | | 0x55: Number of execution stall cycles of SIMDSHFT group |
| | | | | 0x56: Number of execution cycles of FOD group |
| | | | | 0x57: Number of execution stall cycles of FOD group |
| | | | | 0x58: Number of D$ bypass load cycles |
| | | | | 0x59: Number of D$ bypass store cycles |
| | | | | 0x5a: Number of D$ bypass load or store stall cycles |
| | | | | 0x5b: Number of blocking semaphore load cycles |
| | | | | 0x5c: Number of blocking semaphore store cycles |
| | | | | 0x5d: Number of blocking semaphore load or store stall cycles |

### U32.12.1.51PerFormance Counter TOTAL Register0 (PFCTOTALR0)

This register works with the register USPFCCTLR. Refer to U32.12.1.50 .




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | PFCTOTALR0 | All 0 | R/W | PerFormance Counter TOTAL Register0<br>This register shows the total cycles counted in the mode set by PFCMOD. Refer to U32.12.1.50 . |

### U32.12.1.52Unified Shader ConTRoL Register 0 (USCTRL0)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | PFTCHEN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W |

| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 16 | PFTCHEN | 1 | R/W | Pre-FeTCH ENable.<br>0: Disables pre fetch function of thread instruction buffer.<br>1: Enables pre fetch function of thread instruction buffer. |
| 15 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |

### U32.12.1.53TGDMac Control Register 0 (TGDMCR0)

This register is for LWM-TGDMAC. The host processor cannot access this register, only threads can access it. Each thread has its own register to control LWM-TGDMAC.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

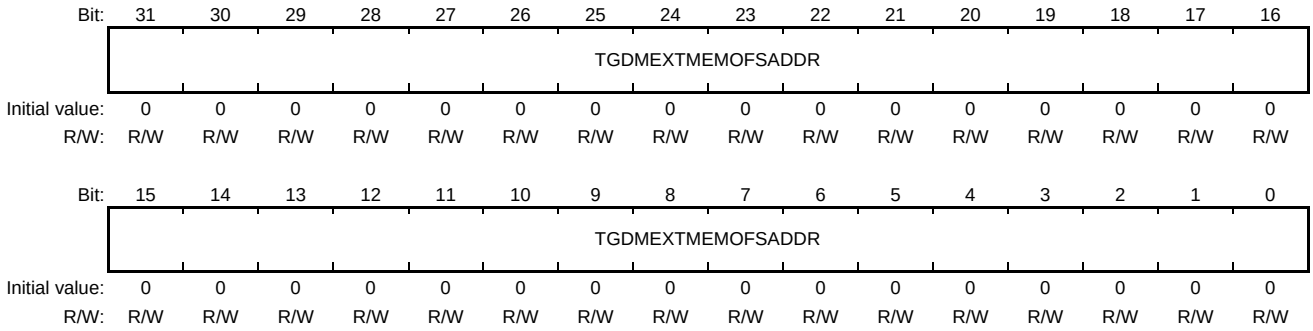
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | EXTSIZE | | SELOFSADR | — | — | DIR | — | SELPLN | | | SELWAIT | — | DMACST | DMACEX |
| Initial value: | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R/W | R/W | R/W | R | R | R/W | R | R/W | R/W | R/W | R/W | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 14 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 13,12 | EXTSIZE | 11 | R/W | Specify the transaction size of external memory.<br>For IMPC:<br>2'b00: 32 bytes<br>2'b01: 64 bytes<br>2'b11: 128 bytes<br>For DDR access, these bits must be set to 2’b11. |
| 11 | SELOFSADR | 0 | R/W | 0: TGDMEXTMEMOFSADDR is used as X,Y coordinate.<br>1: TGDMEXTMEMOFSADDR is used as offset address in 32-bit format. |
| 10,9 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 8 | DIR | 0 | R/W | Specify the direction of DMA transfer.<br>0: LWM to the external memory<br>1: The external memory to LWM |
| 7 | — | 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 6 to 4 | SELPLN | 000 | R/W | Specify the plane number (0-7) which determines the address and the stride for calculating base address of the external memory.<br>Addition of the base address of the plane number specified in this field and the external offset address makes the base address of the external memory. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 3 | SELWAIT | | R/W | Specify the mode of WAITDMAC instruction.<br>0: WAITDMAC exits when the DMAC transfer completes.<br>1: WAITDMAC exits when the DMAC startup completes. DMAC registers can be set for a next DMA transfer after the DMAC startup completes. |
| 2 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 1 | DMACST | 0 | R | This bit becomes "1" when DMAC is started and becomes "0" when DMAC transfer is finished. |
| 0 | DMACEX | 0 | R/W | By writing "1" to this bit, DMAC is started.<br>This bit becomes "0" when DMAC has been started.<br>Note: It is prohibited to write DMAC registers before this bit becomes "0". |

### U32.12.1.54TGDMac EXTernal MEMory OFfSet ADDress Register (TGDMEXTMEMOFSADDR)

This register is for LWM-TGDMAC. The host processor cannot access this register, only threads can access it. Each thread has its own register to control LWM-TGDMAC.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | TGDMEXTMEMOFSADDR | All 0 | R/W | SELOFSADR=0: bit31-16: X coordinate (signed) in bytes, bit15-0: Y coordinate (signed).<br>SELOFSADR=1: bit31-0: Offset address.<br>Addition of IMGBARn, where n is the plane number specified by DMCR0.SELPLN, and TGDMEXTMEMOFSADDR makes the start address of the DMA transfer of the external memory in bytes. |

### U32.12.1.55TGDMac LWM Base ADDress Register (TGDMLWMBADDR)

This register is for LWM-TGDMAC. The host processor cannot access this register, only threads can access it. Each thread has its own register to control LWM-TGDMAC.

Bit 31 to 16 of LWM address will be filled automatically with 0 by HW.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

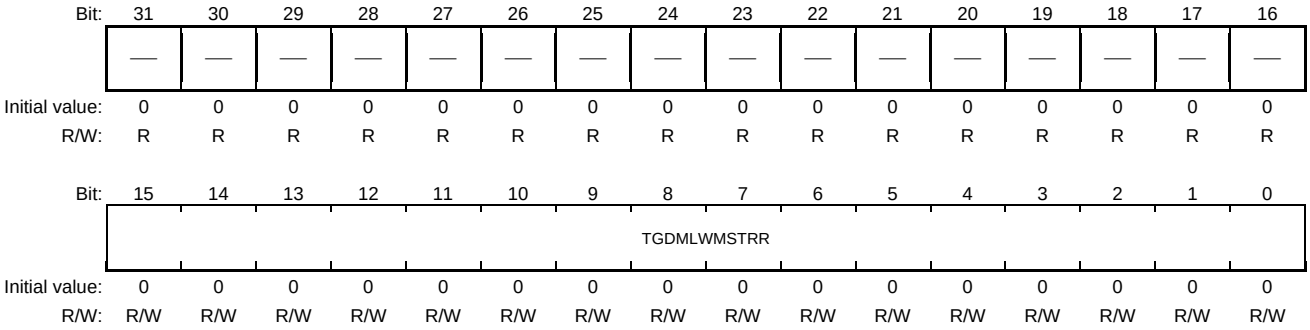
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TGDMLWMBADDR | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 0 | TGDMLWMBADDR | All 0 | R/W | Specify the start address for DMA transfer within LWM. |

### U32.12.1.56TGDmac LWM STRide Register (TGDMLWMSTRR)

This register is for LWM-TGDMAC. The host processor cannot access this register, only threads can access it. Each thread has its own register to control LWM-TGDMAC.

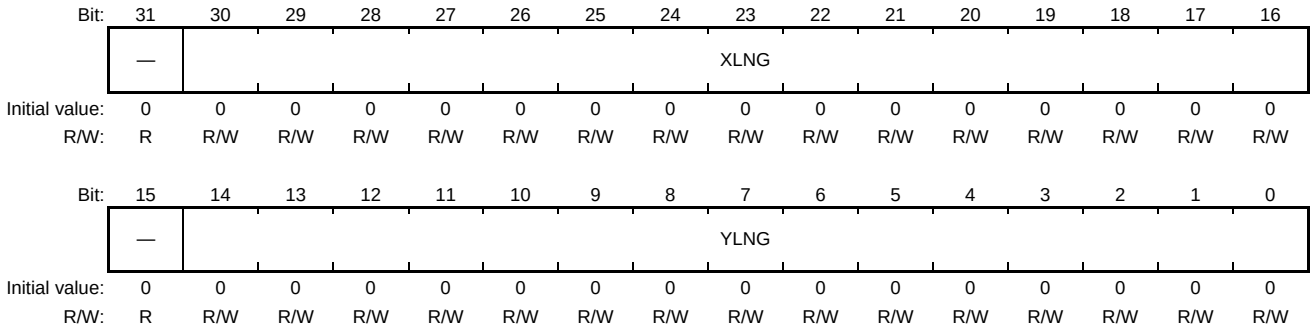
Bit 31 to 16 of LWM stride will be sign-extended automatically by HW.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 16 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 15 to 0 | TGDMLWMSTRR | All 0 | R/W | These bits set the signed stride of LWM by byte unit. |

### U32.12.1.57TGdmac LeNGth Register (TGLNGR)

This register is for LWM-TGDMAC. The host processor cannot access this register, only threads can access it. Each thread has its own register to control LWM-TGDMAC.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 30 to 16 | XLNG | H'0000 | R/W | A width in the range of 1 to 16384 in byte unit. Value must satisfy the plane format alignment. |
| 15 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 14 to 0 | YLNG | H'0000 | R/W | A height in the range of 1 to16384 in byte unit. |

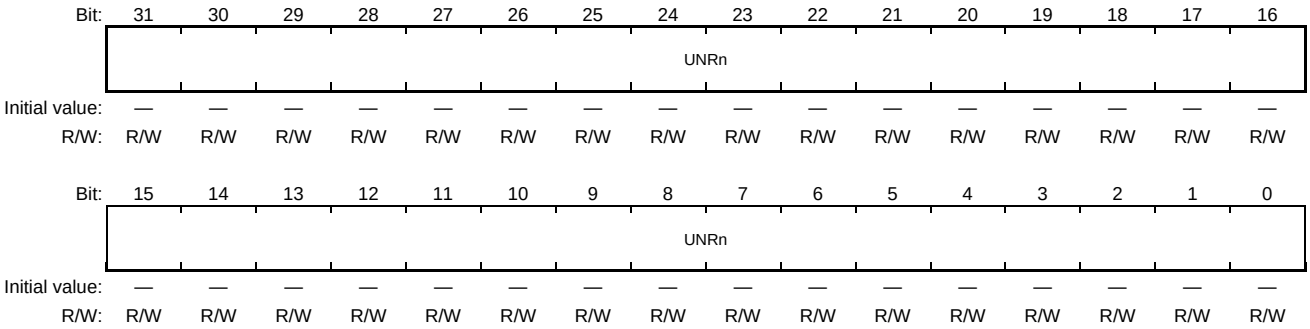
### U32.12.1.58TGSBO CouNT Register0 (TGSBOCNTR0)

This register indicates the number of write accesses to the SBO port by all 32 threads of a cluster. Even so US thread writes to SBO, it will not be reflected immediately. Also the time point the data is really written out may be later than the update of this register. Wait for the end of US thread on the VI before using it.

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 29 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |
| 28 to 2 | TGSBOCNTR0 | H'0000000 | R | Number of 4byte write accesses to the SBO port for all 32 threads of a cluster. |
| 1 to 0 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>Operation cannot be guaranteed if a value other than 0 is written here. |

### U32.12.1.59UNiform Register n (UNRn) (n = 0 to 63)

All UNR register are readable and writable from Arm. From CVengine, all UNR register are read only.




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | UNRn | Undefined | R/W | Set contents of the n-th Uniform storage |

## U32.13 Command List (CL)

It is prohibited to use instructions not listed on this section.

### U32.13.1 POINT (Point)

#### (1) Functionality

Issue coordinates specified in CL to master threads. The coordinates are placed along with the command.

The coordinate space is 65,536 × 65,536, in which -32,768 ≤ X ≤ 32,767 and -32,768 ≤ Y ≤ 32,767.

When clip bit is 1, (X, Y) coordinates are clipped according to the clipping area. Refer to section U32.8.7 .

Note: User is required to execute SYNCS instruction with bit[15:13]==b'110 before and after executing this instruction.

#### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0011_0000(H'30) | | | | | | | | Reserved (all 0) | | | | | | | clip | N(1≦N≦65535) | | | | | | | | | | | | | | | |
| X0 (signed 16-bit, -32768 - 32767) | | | | | | | | | | | | | | | | Y0 (signed 16-bit, -32768 - 32767) | | | | | | | | | | | | | | |
| X1(signed 16-bit, -32768 - 32767 | | | | | | | | | | | | | | | | Y1(signed 16-bit, -32768 - 32767) | | | | | | | | | | | | | | |

…

| X(N-1)( signed 16-bit, -32768 - 32767) | Y(N-1)( signed 16-bit, -32768 - 32767) |
|---|---|

### U32.13.2 POINTS (Point Separate)

#### (1) Functionality

Issue coordinates specified in CL to master threads. The coordinates should be placed at the address specified by the "Address" field of this command.

The coordinate space is 65,536 × 65,536, in which -32,768 ≤ X ≤ 32,767 and -32,768 ≤ Y ≤ 32,767.

When clip bit is 1, the corresponding coordinates at the addresses are clipped according to the clipping area. Refer to section U32.8.7 .

Note: User is required to execute SYNCS instruction with bit[15:13]==b'110 before and after executing this instruction.

**(2) Command Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0011_0011(H'33) | | | | | | | | Reserved (all 0) | | | | | | | clip | N(1≦N≦65535) | | | | | | | | | | | | | | | |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U32.13.3 RECT (RECTangle)

**(1) Functionality**

Generate coordinates automatically and issue the generated coordinates to master threads. The coordinates are generated in Raster-scan order.

Note: User is required to execute SYNCS instructions with bit[15:13]==b'110 before and after executing this instruction.

When clip bit is 1, (X, Y) coordinates are clipped according to the clipping area. Refer to section U32.8.7 .

From (Xs, Ys), it generates coordinates according to the parameters. The (X[i][j], Y[i][j]) coordinate is generated like the following behavior model, then it is clipped according to the clipping area if clip bit is 1.

The dX1, dY1, dX2, and dY2 are specified by signed 16-bit fixed point format. The number of the bits of decimal point is decided by sft.

Xs, Ys are specified by unsigned fixed point format. The number of the bits of decimal point is decided by sftxy. The value of sft must be more than the value of sftxy.

(Of any sftxy value, Xs and Ys work as signed 16-bit)

Note: Uses X[i][j] and Y[i][j] in the following ranges:

-32768<=X[i][j]<=32767, -32768<=Y[i][j]<=32767

< behavior model >

```
(0<= sft <= 11, 0<=sftxy<=3, sft >= sftxy)
X_ls = Xs << (sft - sftxy);
Y_ls = Ys << (sft - sftxy);
for (j=0; j<Ylen; j++) {
    X0[0][j] = X_ls;
    Y0[0][j] = Y_ls;
    X[0][j] = X0[0][j] >> sft;
    Y[0][j] = Y0[0][j] >> sft;
    for(i=1; i<Xlen; i++) {
        X0[i][j] = X0[i-1][j] + dX;
        Y0[i][j] = Y0[i-1][j] + dY2;
        X[i][j] = X0[i][j] >> sft;
        Y[i][j] = Y0[i][j] >> sft;
    }
    X_ls = X_ls + dX2;
    Y_ls = Y_ls + dY;
}
```

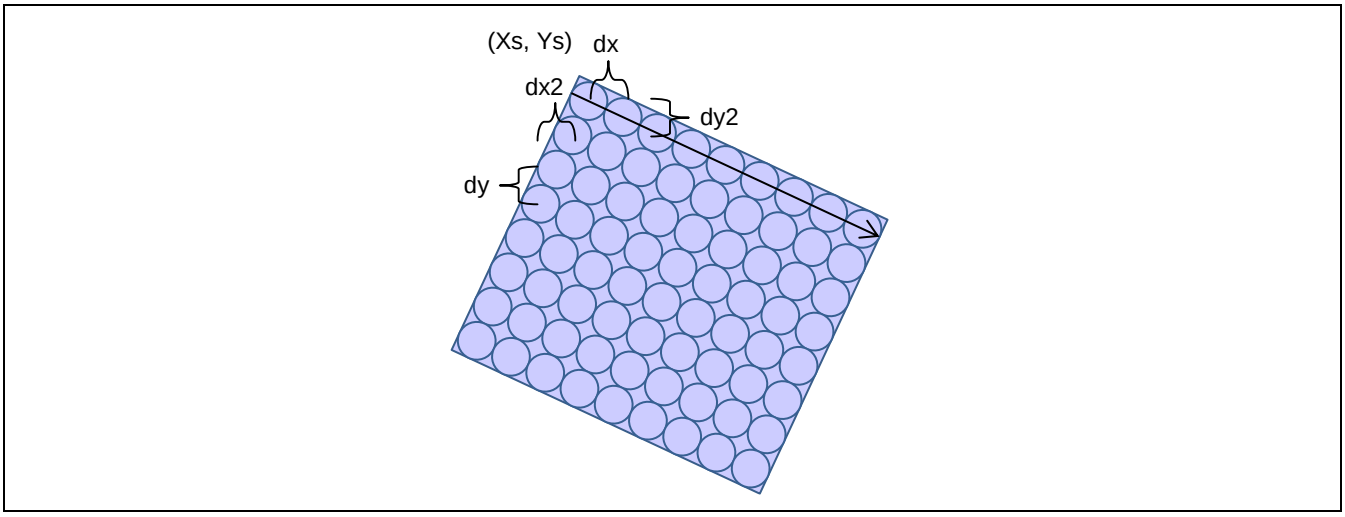



**Figure U32.13.1 RECT command parameters**

### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0010_0010(H'22) | | | | | | | | Reserved (all 0) | | | | | | | clip | Reserved (all 0) | | | | | | | | | | sftxy | | sft (0<=sft<=11) | | | |
| dX(signed 16-bit fixed point) | | | | | | | | | | | | | | | | dY(signed 16-bit fixed point) | | | | | | | | | | | | | | | |
| dX2(signed 16-bit fixed point) | | | | | | | | | | | | | | | | dY2(signed 16-bit fixed point) | | | | | | | | | | | | | | | |
| Xs(signed 16-bit,-32768<=Xs<=32767) | | | | | | | | | | | | | | | | Ys(signed 16-bit, -32768<=Ys<=32767) | | | | | | | | | | | | | | | |
| Xlen(unsigned 16-bit, 1<=Xlen<=65535) | | | | | | | | | | | | | | | | Ylen(unsigned16-bit, 1<=Ylen<=65535) | | | | | | | | | | | | | | | |

### U32.13.4 NOP(No OPeration)

#### (1) Functionality

Wait for N cycle and go to the next command.

#### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_0000(H'80) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | N(1≦N≦16) | | | | |

### U32.13.5 TRAP(TRAP)

#### (1) Functionality

Finishes CL execution and reflects it to the Status Register if permitted by Interrupt Control Register. Also, regardless of Interrupt Control Register, the value of CODE field is reflected to Status Register.

Note: User is required to execute SYNCM instruction with bit11=1, bit2=1 and bit0=1 before executing this instruction.

#### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_1000(H'88) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | CODE | | | | | | | |

### U32.13.6 WPR(Write PaRameter)

#### (1) Functionality

From the register address specified from ADD field, the successive N registers are written by DATA0 to DATA N-1 respectively. ADD[23:8] should be specified by 1Byte unit. Registers written by this command must be “WPR support” registers. Otherwise, operation cannot be guaranteed. Refer to “Offset” column on Table U32.12.2 to see the register address for ADD field and refer to “WPR support” column also.
When WPR is used after executing POINT, POINTS or RECT, SYNCS with bit15=1 and bit14=1 must be executed before executing WPR.

#### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0101_0101(H'55) | | | | | | | | ADD | | | | | | | | | | | | | | | | N(>=1) | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

…

| DATA N-1 |
|---|

### U32.13.7 INT(INTerrupt)

#### (1) Functionality

The INT bit of Status Register gets “1” if it is permitted by Interrupt Control Register. Also, regardless of Interrupt Control Register, the value of CODE field is reflected to Status Register. The CL execution is stopped till the INT bit of Status Register becomes “0”.

Before stopping CL execution, the instruction buffer in Vertex Issue and the instruction cache of the thread are cleared, and SBO and data cache of the thread are flushed. Since the pre-fetch instruction buffer is cleared with this command, CL commands can be re-written during interruption.
Note: SBO Base Address Register must be set before next CL execution when SBO is used.

**Remarks:**

- Renesas uses the INT command internally in software debug solution.
- INT codes 0xe0 to 0xff is reserved for this debug solution or Renesas internal purpose. 0x00 to 0xdf are available for users.

INT_PBCOVF is used for the Guard Timer. Refer to Section U32.8.8 for detail.

#### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_0001(H'81) | | | | | | | | Reserved (all 0) | | | | | | | | *1 | Reserved (all 0) | | | | | | | CODE | | | | | | | |

*1 INT_PBCOVF

### U32.13.8 JUMP(JUMP)

#### (1) Functionality

This is the command to jump to the specified address.

It allows jumping to the address specified in JMPADD field. Setting 1 to RLT field uses JMPADD as relative address to JUMP command address and setting 0 in RLT field uses JMPADD as absolute address of the command list. Specify 4-byte unit command address in the JMPADD field.

#### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1001_0000(H'90) | | | | | | | | RLT | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | |
| JMPADD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U32.13.9 GOSUB(GOSUB)

#### (1) Functionality

This is the command to save current CL address to the hardware buffer and jump to the specified address.

It allows jumping to the address specified in GSADD field and returning to the address next to the GOSUB command by RET command. Specify 4-byte unit command address in the GSADD field, nesting of up to 4 level is supported. Setting 1 to RLT field uses GSADD as relative address to GOSUB command address and setting 0 in RLT field uses GSADD as absolute address of the command list. Setting 1 in CON field executed GOSUB conditionally and setting 0 in the CON field executed GOSUB unconditionally. Setting 1 in CON field and 0 in CLCNDGSBR bit of CLCNDGSBR register changes GOSUB command into NOP command. After the CLCNDGSBR register is set, a SYNCS command should be added before using the GOSUB command.

#### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1001_0001(H'91) | | | | | | | | RLT | CON | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | |
| GSADD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U32.13.10 RET(RETurn)

#### (1) Functionality

Jumps to the address saved by a GOSUB command. Initial value of address buffer is not defined, so executing RET command to an address buffer which had not been saved by GOSUB command is prohibited.

#### (2) Command Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1001_1001(H'99) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |

### U32.13.11 SYNCM(SYNChronize Memory)

#### (1) Functionality

Flush or clear each cache specified in the command. After all cache operation specified in the command finishes, the next command is fetched and executed. It is prohibited to specify both flush and clear to the same cache at the same time. The command should be used after SYNCS because some threads might still be running.

If I$ CLR bit is 1, Instruction cache of the thread is cleared. If SBO CLR bit is 1, SBO is cleared, and the data left in SBO disappear. If D$ CLR bit is 1, Data cache of the thread is cleared. If SBO FLUSH bit is 1, SBO is flushed. If D$ FLUSH bit is 1, Data cache of the thread is flushed.

Upon SBO output, SBO Base Address Register is incremented to indicate the current SBO output address. And flushing or clearing SBO does not clear SBO Base Address Register. So, if SBO had been flushed or cleared, SBO Base Address Register must be set before next CL execution.

**(2) Command Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_1010(H'8A) | | | | | | | | Reserved (all 0) | | | | | | | | | | | 0 | I$ CLR | SBO CLR | 0 | D$ CLR | Reserved (all 0) | | | | | SBO FLUSH | 0 | D$ FLUSH |

## U32.13.12 SYNCS(SYNChronize Shader)

**(1) Functionality**

Wait for the end of execution of Threads. After all specified CVengine and/or DMAC's Threads get idle, the next command is executed.

**(2) Command Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_1011(H'8B) | | | | | | | | Reserved (all 0) | | | | | | | | See Below | | | Reserved (all 0) | | | | | | | | | | | | | | |

bit[15:13]==b'111 : Wait until running threads, running LWM TGDMAC and running GWM-DMAC transfers are completed

bit[15:13]==b'110 : Wait until running threads and running LWM TGDMAC transfers are completed

bit[15:13]==b'001 : Wait until running GWM-DMAC transfers are completed

bit[15:13]==else : All other settings are prohibited

## U32.13.13 SINI(Shader INItialize)

**(1) Functionality**

This command starts each enabled thread once and returns processing to the CL. The command does not wait for the threads to finish execution, for details refer to section U32.12.1.27 and U32.12.1.29 .

Note: User is required to execute SYNCS instructions with bit[15:13]==b'110 after executing this instruction.

**(2) Command Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0001_0001(H'11) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |

## U32.13.14 SLP(SLeeP)

**(1) Functionality**

This command stops execution of the succeeding commands in the CL until all specified WUP signals are received.

Specify the necessary WUP signals as the core numbers in the CORE Number field. Multiple bits can be set to 1 at the same time.

The correspondence between core numbers and actual cores can be specified through the SYNCCR0, SYNCCR1, SYNCCR2 and SYNCCR3 registers. Refer to Section U32.12.1.46 , Section U32.12.1.47 , Section U32.12.1.48 and Section U32.12.1.49 .

Each SYNCCRn register contains 4 SYNCCn fields. Each SYNCCn field is 8-bit long. Its upper 4 bits determine CORE Type and lower 4 bits determine CORE Number.

The available CORE Types are:

0: IMP Core
1: CVengine
2: IMP DMAC
3: Pyramid Scaler
4: CNN
5: LSDMAC
6: IMR
7: DTA

And possible CORE Numbers for each CORE Type are:

IMP Core: 0-4
CVengine: 0-7
DMAC: 0-3
Pyramid Scaler: 0-1
CNN: 0-2
LSDMAC: 0

All the cores must have the same CORE Type and CORE Number settings.

Using this command and the WUP command enables synchronization among cores. Refer to Section U32.11.1 .

**(2) Command Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1010_0001(H'A1) | | | | | | | | Reserved (all 0) | | | | | | | | SYNCC enable | | | | | | | | | | | | | | | |

## U32.13.15 WUP(Wake UP)

**(1) Functionality**

This command issues the WUP signal to the cores specified in this command. Multiple bits can be set to 1 at the same time.

The WUP signal triggers a wake up event at the target. The WUP command can be issued to a core even when it is not in the SLP state. However, two or more WUP requests are not accumulated. (Issuing the WUP command to a core and issuing it again to the same core before the core wakes up is not supported. i.e., the second WUP does not trigger a wake up for a second SLP at the target core.). Between 2 WUP commands at least 8 other commands must be inserted.

The correspondence between core numbers and actual cores can be specified through the SYNCCR0, SYNCCR1, SYNCCR2 and SYNCCR3 registers. Refer to Section U32.12.1.46 , Section U32.12.1.47 , Section U32.12.1.48 and Section U32.12.1.49 .

For settings of SYNCCRn register, refer to U32.13.14 .

Using this command and the SLP command enables synchronization among cores. Refer to Section U32.11.1 .

**(2) Command Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1010_0000(H'A0) | | | | | | | | Reserved (all 0) | | | | | | | | SYNCC enable | | | | | | | | | | | | | | | |

## U32.14 Appendix

This section describes the four methods for improving performance of the CVengine thread.

### U32.14.1 Effective Use of 0th Level Cache

Each thread implements the 0th level cache for accessing the D$.

The size of the 0th level cache is 256 bits. When reading data from the D$ in a single access operation, each thread reads the surrounding data that includes the accessed data within its 256-bit aligned boundary.

Data to be written from each thread is directly written to the D$. If a hit occurs in the 0th level cache, the 0th level cache is overwritten. If a miss occurs in the 0th level cache, data is written to only the D$ and 0th level cache is not changed.

As a result of monitoring programs, access to the D$ has been observed in multiple cases to be a bottleneck in good performance.

Accordingly, each thread being able to effectively use the 256-bit data in the 0th level cache is expected to improve performance.

Be noticed that the 0th level cache is only connected to D$ thus it does not work for accessing to any other memory (e.g., LWM).

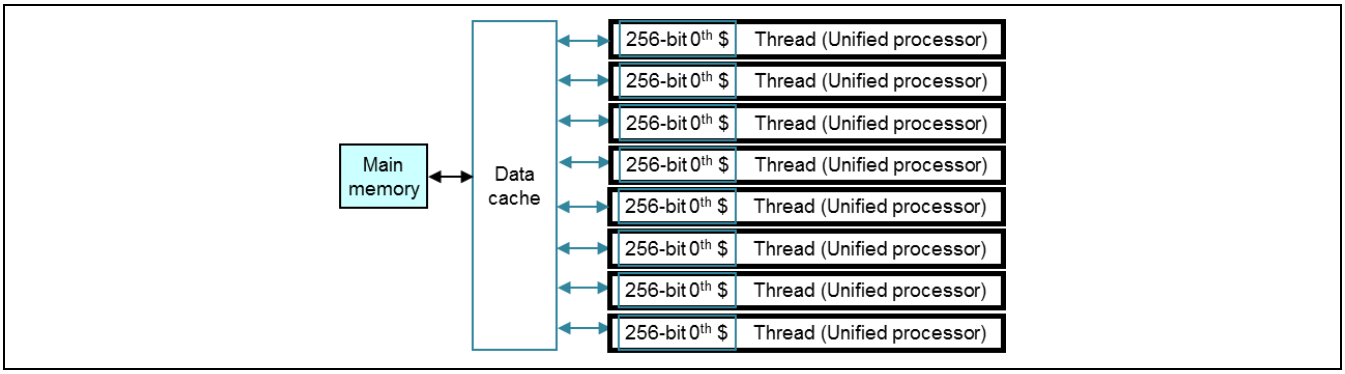



**Figure U32.14.1 Arrangement and Configuration of 0th Level Cache**

Those are two examples use cases for 256 bit alignment. In case of processing an 8-bit/pixel image, programming so that 32 (256 bits/8 bits = 32) consecutive pixels are processed by a single thread enables all of the necessary data to be accessed by a single access to the D$ and also prevents multiple threads from reading the same 256 bits of data. In case of 32-bit/pixel data, programming so that 8 (256 bits/32 bits = 8) consecutive pixels are processed by a single thread similarly enables all of the necessary data to be accessed by a single access to the D$ and also prevents multiple threads from reading the same 256 bits of data.

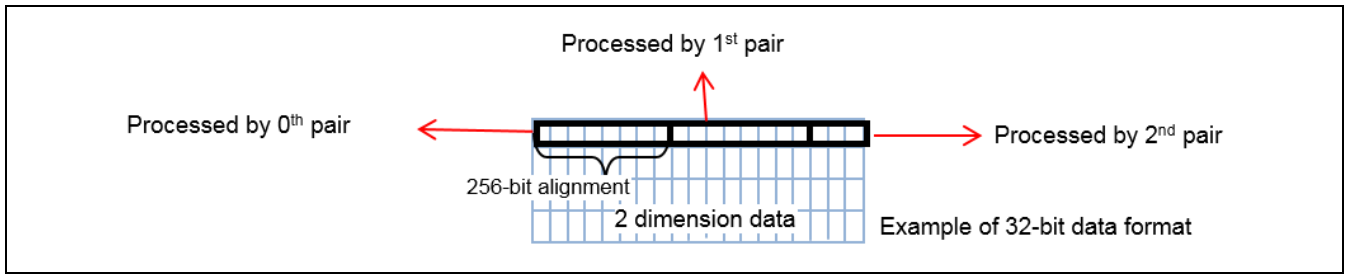



**Figure U32.14.2 Example of Efficiently Using 0th Level Cache**

### U32.14.2 Kernel Processing for Each Area

One thread needs to read multiple surrounding pixels of the center pixel in kernel processing that is specific and frequently used in image recognition. In kernel processing, a surrounding pixel that has been read once becomes the center pixel in the next kernel or can be used as a surrounding pixel. Therefore, in kernel processing, if a certain thread has processed a kernel, the same thread processing the neighboring kernels can reduce the overhead in data loading.

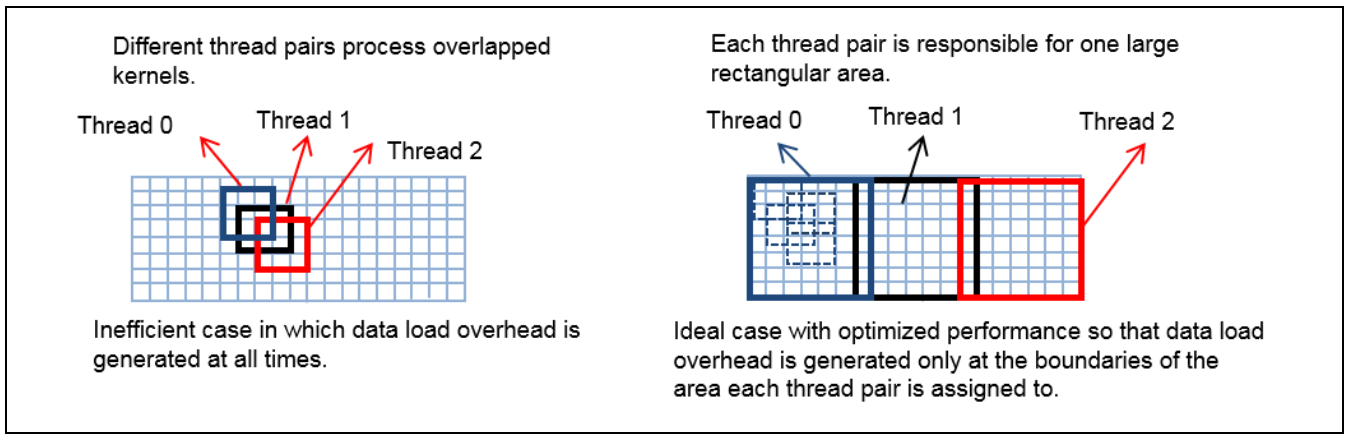



**Figure U32.14.3 Example of Efficiently Assigning Processing Areas**

### U32.14.3 Loop Unrolling

Instructions of the BR group always cause 1-cycle stall when the branch condition is satisfied. Therefore, when there is a loop in a program to generate a branch, stall cycles for the number of loops will occur. Cycles to calculate the number of loops and execute the compare instructions for determining the loop are also required.

Therefore, when there is sufficient program capacity and the number of loops is unique, optimization by loop unrolling is effective to improve the thread performance.

### U32.14.4 Effective Use of Instruction Buffers

Each thread implements buffers to hold the instructions fetched from the I$.

This level-0 instruction cache is implemented as a set of 8 buffers, with 256 bits for each of the 8 buffers. When fetching an instruction from the L1 I$ in a single access operation, each thread reads the surrounding data that includes the instruction to be fetched within its 256-bit aligned boundary.

If the instruction to be fetched is not in any of the buffers, the surrounding data that includes the instruction to be fetched is aligned at the 256-bit boundary and is stored in one buffer. The subsequent 256-bit data is stored in another buffer.

When the last instruction in a buffer X gets executed, then buffer (X + 2)%8 gets pre-loaded with 8 instructions (256 bits).

If inner loop is larger than 55 instructions but less than or equal 64 instructions, disabling pre-fetch function of the instruction buffer can improve performance. The preloading which is taken place at instruction 56 can be prevented and 64 instructions of inner loop can be hold inside instruction buffer. Refer to section U32.12.1.52 .

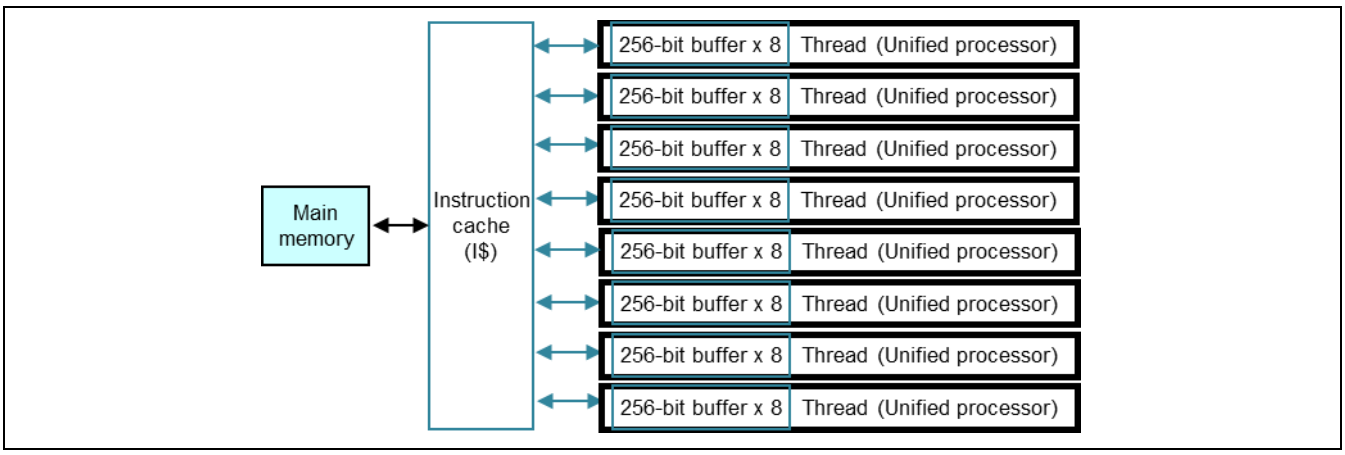



**Figure U32.14.4 Arrangement and Configuration of Instruction Buffers**

For example, if there is a loop in the program, storing the first instruction of a loop in the start address of the 256-bit aligned memory may improve the efficiency of instruction fetch. The effect is greater especially if the number of iterations of the loop is large.

### U32.14.5 Tiling Approach

Size the D$ is reduced from former R-Car SoCs, refer to U32.1.2 , it is recommended to use DMA for data loading from DDR to Scratchpad memory or LWM, for details, refer to section U32.8.3 .

## U32.15 Usage Notes

### U32.15.1 Synchronized system between modules

For synchronized system between modules, refer to the Section U34.IMP-X7_system, U34.10.5 Inter Module Synchronization of IMP-X7_System manual.

### U32.15.2 Power shutdown

For power shutdown, refer to User's Manual: Hardware, section 58.7.1, Shutting Down the Power Supply.

### U32.15.3 Module Standby Mode

For module standby mode, refer to User's Manual: Hardware, section 58.7.2, Module Standby Mode.

### U32.15.4 CPG Software Reset

For software reset of CPG, refer to User's Manual: Hardware, section 58.7.3, Software Reset by the Clock Pulse Generator (CPG).

### U32.15.5 SOU register limitation

SOU registers (R8 to R15) cannot be used as destination register of the following load instructions:

LDRCI, FLDRCI

LDRCIU, FLDRCIU

LDBUI, LDBSI, LDHUI, LDHSI, LDI

LDBU, LDBS, LDHU, LDHS, LD

LDRXY, FLDRXY

LDRXYU, FLDRXYU

If the register R8 to R15 are not used as SOU registers but as general-purpose registers, they can be used as destination register of the load instructions.

## U32.16 Safety Mechanism

### U32.16.1 EDC for CVe

#### U32.16.1.1Overview

EDC is applied to the following internal RAMs of CVe. All RAMs in CVe are protected by EDC.

- SRAM for Global Work Memory
- SRAM for Local Work Memory
- SRAM for Level 1 Instruction Cache (Data Array)
- SRAM for Level 2 Instruction Cache (Data Array)
- SRAM for Data Cache (Data Array)
- SRAM for Data Cache (Byte Enable Array)
- SRAM for Stream Buffer Output

EDC is also applied to AXI Master Bus I/F and AXI Slave Bus I/F.

#### U32.16.1.2Hardware Description

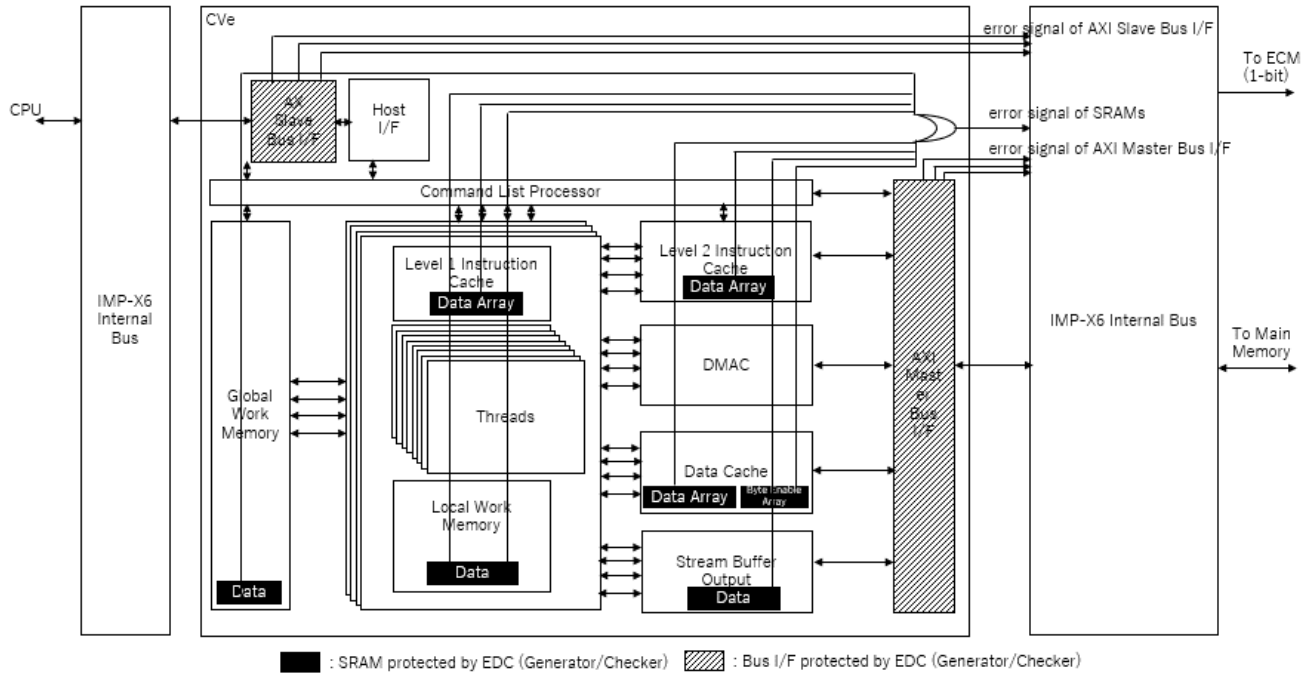



Figure U32.16.1 Block diagram of EDC protection of CVe

**(1) RAM**

When using the EDC function of memory (after every power on reset or software reset by CPG or Runtime Test or POST), it is necessary to initialize the memories.

The EDC code is generated and written to SRMs described in Section U32.16.1.1 when data is written to each SRAM. And the data and the corresponding code are compared when it is read out.

Since initial values in SRAM are undefined, it is required to initialize the contents of SRAM of Local Work Memory, Global Work Memory, Data Cache Data Array, and Data Cache Byte Enable Array since unwritten area of the SRAMs can be read out. The initialization can be done automatically by using MEMINITR register for Global Work Memory and Local Work Memory initialization. The initialization status for each SRAM can be checked by reading the status bits of MEMINITR register.

< Procedure to initialize Global Work Memory >

Execute following steps from (1) to (3) to initialize the Global Work Memory.

(1)Check gwm_state bit in MEMINITR register equal 0 (MEMINITR [16] = 1'b0). If this bit is 1, clear this bit by CPG software reset.

(2)Set gwm_init bit in MEMINITR register to 1

(3)Check gwm_state bit in MEMINITR register equal 1 (MEMINITR [16] = 1'b1)

< Procedure to initialize Local Work Memory >

Execute following steps from (1) to (3) to initialize the Local Work Memory.

(1)Check lwm3_state, lwm2_state, lwm1_state, lwm0_state, bits in MEMINITR register equal 0 (MEMINITR [23:20] = 4'b0000). If these bits are 1, clear these bits by CPG software reset.

(2)Set lwm3_init, lwm2_init, lwm1_init, lwm0_init, bits in MEMINITR register to 1

(3)Check lwm3_state, lwm2_state, lwm1_state, lwm0_state bits in MEMINITR register equal 1 (MEMINITR [23:20] = 4'b1111)

For Data Cache Data Array and Data Cache Enable Array, the memories have to be set into testmode by writing H'8006 4100 to the Memory Control Register 0 (MCR0) register to set the testmode bit and areasel bits before writing by CPU H'0000 0000 from address H'C000 to address H'DFE0 every 32byte and finally switching back to normal mode by writing H'0100 0000 to the MCR0 register.

< CPU initialization procedure for D$ Memory >

Execute following steps from (1) to (3) to initialize the D$ Memory.

(1)Write H'8006 4100 to the Memory Control Register 0 (MCR0) register to set the testmode bit and areasel bits.

(2)The CPU writes H'0000 0000 from address H'C000 to address H'DFE0 every 32byte.

(3)Write H'0100 0000 to the MCR0 register.

### (2) Bus I/F

An EDC protection is implemented in bus interface of CVe.

### (3) . Register Description

| Register Name | Abbreviation | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|---|
| Memory Initialization Register | MEMINITR | R/W | (Base Address) + H'0F8 | H'0000 0000 | 32 |
| Memory Control Register 0 | MCR0 | R/W | (Base Address) + H'040 | H'0100 0000 | 32 |

Note : Refer to the Manual of IMP-X7 about the Base Address of each channel of CVe

#### (a) MEMory INITialization Register (MEMINITR)

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | lwm3_state | lwm2_state | lwm1_state | lwm0_state | — | — | — | gwm_state |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

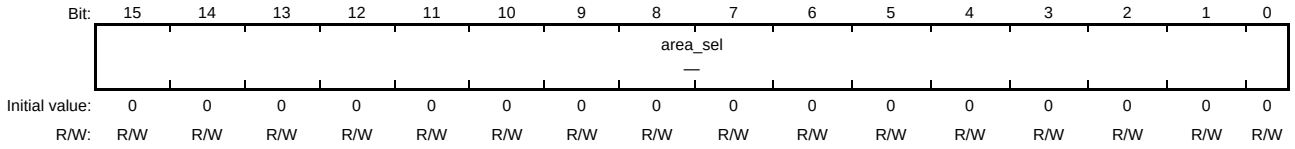
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | lwm3_init | lwm2_init | lwm1_init | lwm0_init | — | — | — | gwm_init |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R | R | R | R | R | R | R | R | RW | RW | RW | RW | R | R | R | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. |
| 23 | lwm3_state | 0 | R | 0: Local work memory in CORE3 is not initialized.<br>1: Local work memory in CORE3 is initialized at least once after power-on reset(hardware-reset) |
| 22 | lwm3_state | 0 | R | 0: Local work memory in CORE2 is not initialized.<br>1: Local work memory in CORE2 is initialized at least once after power-on reset(hardware-reset) |
| 21 | lwm3_state | 0 | R | 0: Local work memory in CORE1 is not initialized.<br>1: Local work memory in CORE1 is initialized at least once after power-on reset(hardware-reset) |
| 20 | lwm3_state | 0 | R | 0: Local work memory in CORE0 is not initialized.<br>1: Local work memory in CORE0 is initialized at least once after power-on reset(hardware-reset) |
| 19 to 17 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 16 | gwm_state | 0 | R | 0: Global work memory is not initialized.<br>1: Global work memory is initialized at least once after power-on reset(hardware-reset) |
| 15 to 8 | — | All 0 | R | Reserved.<br>These bits are always read as 0.<br>The write value should always be 0. |
| 7 | lwm3_init | 0 | R/W | Read:<br>0: Local Work Memory in CORE3 is not under initialization.<br>1: Local Work Memory in CORE3 is being initialized to 0.<br>Write:<br>0: Nothing occurs<br>1: Start initialization of Local Work Memory in CORE3 |
| 6 | lwm2_init | 0 | R/W | Read:<br>0: Local Work Memory in CORE2 is not under initialization.<br>1: Local Work Memory in CORE2 is being initialized to 0.<br>Write:<br>0: Nothing occurs<br>1: Start initialization of Local Work Memory in CORE2 |
| 5 | lwm1_init | 0 | R/W | Read:<br>0: Local Work Memory in CORE1 is not under initialization.<br>1: Local Work Memory in CORE1 is being initialized to 0.<br>Write:<br>0: Nothing occurs<br>1: Start initialization of Local Work Memory in CORE1 |
| 4 | lwm0_init | 0 | R/W | Read:<br>0: Local Work Memory in CORE0 is not under initialization.<br>1: Local Work Memory in CORE0 is being initialized to 0.<br>Write:<br>0: Nothing occurs<br>1: Start initialization of Local Work Memory in Core0 |
| 3 to 1 | — | All 0 | R | Reserved<br>These bits are always read as 0.<br>The write value should always be 0. |
| 0 | gwm_init | 0 | R/W | Read:<br>0: Global Work Memory is not under initialization.<br>1: Global Work Memory is being initialized to 0.<br>Write:<br>0: Nothing occurs<br>1: Start initialization of Global Work Memory |

**(b) Memory Control Register (MCR0)**

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | test mode | area_sel — — | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W: | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |




| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 | testmode | 0 | R/W | CPU access to internal units.<br>0: Cannot access internal units<br>1: Can access internal units |
| 30 to 0 | area_sel | H'0100 0000 | R/W | Access area selection for test mode<br>Testmode == 1<br>H'00064100: D-cache (D$)<br>Other: not supported<br>Testmode == 0<br>H'0100 0000: Initial value<br>Other: not supported |

# U33. Convolutional Neural Network (IMP CNN)

## U33.1 Overview

The Convolutional Neural Network of image processing unit is abbreviated as IMP CNN. It performs a fast execution of a Convolutional Neural Network with optional enabling additional CNN layers like RELU (Rectified Linear Units) or pooling for up to k input and m output layers with a simple instruction set. The CNNs can be used for applications like semantic segmentation for detection of lanes, pedestrians, cars, or any other object; object detection and classification to identify the vehicles, traffic signs, pedestrians, etc. Current state of the art networks reaches results which exceed human accuracy. The most important layer for a CNN is the Convolutional layer which consumes around 90% of the processing power. Other important layers are RELU, Pooling, Deconvolution, etc.

This CNN-IP can not only perform many convolutions in parallel, but also can merge additional layers into a single run. This CNN-IP provides an advanced solution for special recognition tasks within the R-Car family. In the following, first some basics about the CNN-IP are described

Compared with other designs, this setup has two major advantages:

a) Less memory consumption, because of the combined logic eliminates the need for temporal storage significantly.

b) High accuracy since the quantization-error (downshift normalization) is applied to the sum of up to k convolution results. This stands in contrast to other designs, which apply normalization on a smaller scope and have overall a bigger quantization error.

This CNN-IP is a configurable – not programmable IP. Its function is controlled by registers. These registers can be set by a command list (CL), which is processed by the command issue hardware unit. The actual data processing is done in a pipelined way as shown in Figure U33.4. That means, in each cycle a pixel is processed for each channel.

Figure U33.1 shows the result of semantic segmentation, which is a sample application of CNN. The different stages of the processing have produced a combined image showing the detection of the different items in the scene such as cars, road, and sky. This produces classification of the objects detected in the image.

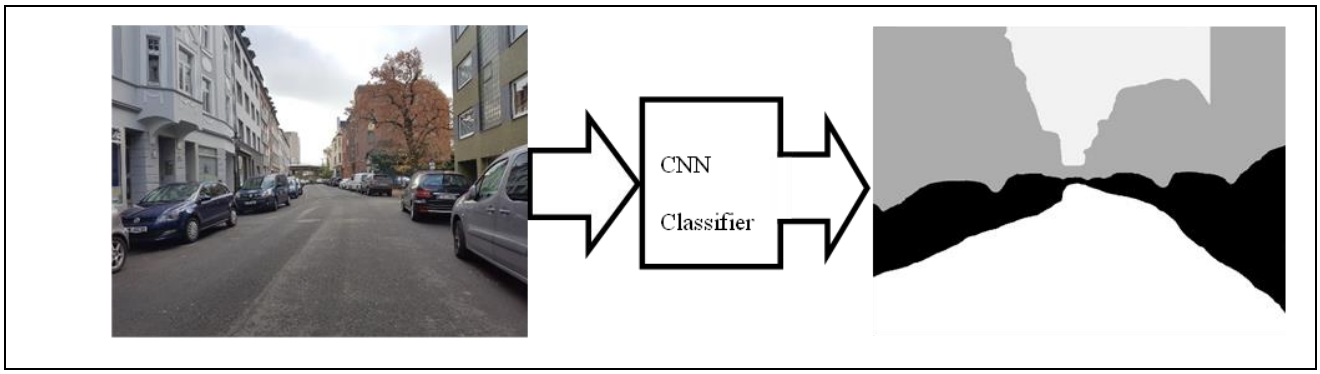



**Figure U33.1 Use case example for sematic segmentation**

Figure U33.2 shows the different layers of processing of which the convolution, (Bounded) RELU, Pooling, Deconvolution and Pixel Labelling take place. The output after processing gives the segmentation result.

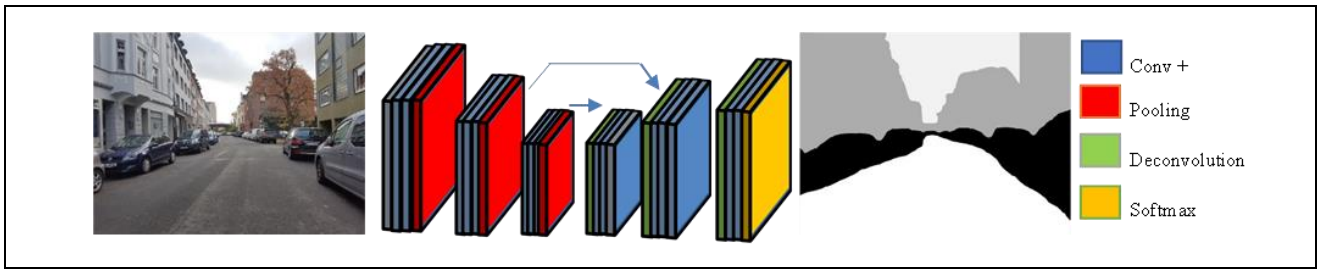



**Figure U33.2 Example network for Semantic Segmentation**

This module is treated as safety related function in case of functional safety supported product. For the details, refer to functional safety documents.

The CNN-IP module supports processing of complex CNNs basically by taking small portions, also called tiles, of an image and run the filtering process on each of them. Once the processing has been carried out on the data a result is passed to the next layer. The images are filtered with known feature patterns stored in a database to produce as output the likelihood of a successful match as shown in Figure U33.3

A dedicated CNN-IP hardware accelerator is the most efficient hardware implementation (computation vs. cost / power). The solution is pipeline based, meaning the overhead is very small compared to a software equivalent system. Furthermore, additional functions are available without the cost of delays.

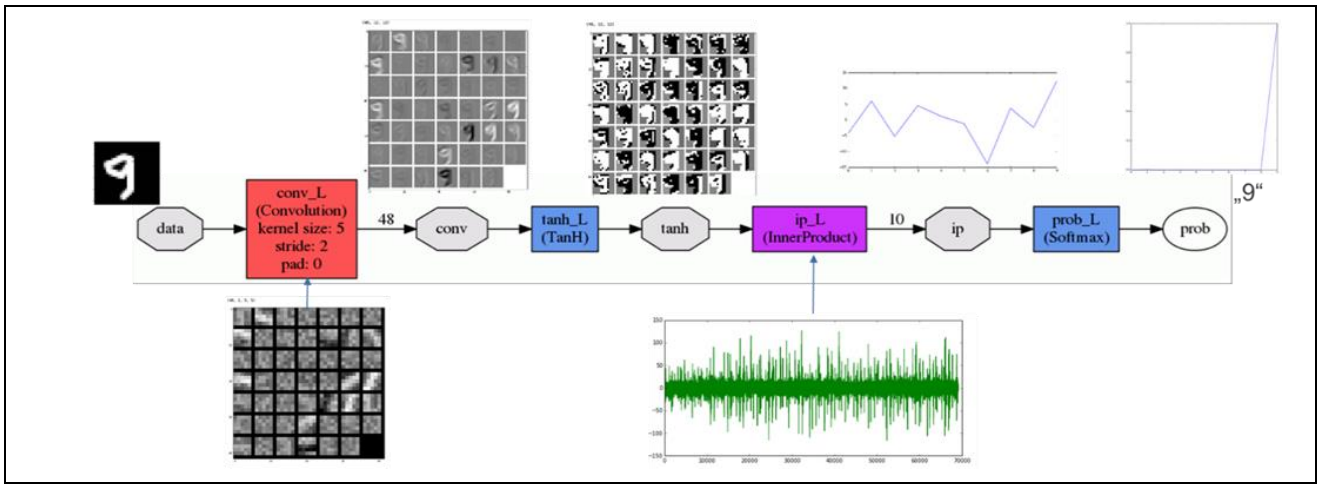



**Figure U33.3 Processing flow of image processing and likelihood weighting of a match**

### U33.1.1 Features

The following terms and symbols are used throughout the CNN specification:

**Table U33.1 The Description of terms**

| Terms | Description |
|---|---|
| CNN | Convolutional Neural Network |
| CL | Command list to control the registers in CNN-IP |
| SPMC | The dedicated memory for IMP CNN (refer U34.IMP-X7_system), including of SPMC0 and SPMC1. |
| DMAI | To load FM from SPMC0 memory |
| DMAW | To load weights from SPMC0 memory |
| CNV | The main function of the CNN-IP, doing convolution with FM and weights |
| DMA3DC | To carry the previous convolution results to ARI module, adding up to current convolution results when the channels of FM greater than number of input channels in CNN-IP |
| ARI | The sub processes Activation, Channel pool, XY Pool are abbreviated as ARI |
| DMAO | A module to store the output data to the SPMC0/SPMC1 |
| DMAx | Each DMA (DMAI/DMAW/DMA3DC/DMAO) |
| DMAxS | DMAIS/DMAWS/DMA3DC/DMAOS |
| FM | Feature Map |
| HW | Hardware |
| WT | weight |
| Pix | pixel |
| CH/ch | channel |
| ich/och | Input channel/output channel |
| s8/u8, s16/m16, | Signed 8bit/unsigned 8bit, signed 16bit/mixed 16bit |
| m32 | Mixed 32bit |
| Block channel | The set of 32 channels or 16 channels to be read or written at the same time when DMAI or DMAO is accessed by packed or planar fast type<br><br>In case of 32 channels set<br>Block channel 0 (blk-ch0): Channel 0 to 31<br>Block channel 1 (blk-ch1): Channel 32 to 63<br>Block channel 2 (blk-ch2): Channel 64 to 95<br>Block channel 3 (blk-ch3): Channel 96 to 127<br>Block channel 4 (blk-ch4): Channel 128 to 159<br>Block channel 5 (blk-ch5): Channel 160 to 191<br>Block channel 6 (blk-ch6): Channel 192 to 223<br>Block channel 7 (blk-ch7): Channel 224 to 255<br>In case of 16 channels set<br>Block channel 0 (blk-ch0): Channel 0 to 15<br>Block channel 1 (blk-ch1): Channel 16 to 31<br>Block channel 2 (blk-ch2): Channel 32 to 47<br>Block channel 3 (blk-ch3): Channel 48 to 63<br>Block channel 4 (blk-ch4): Channel 64 to 79<br>Block channel 5 (blk-ch5): Channel 80 to 95<br>Block channel 6 (blk-ch6): Channel 96 to 111<br>Block channel 7 (blk-ch7): Channel 112 to 127 |

The CNN-IP has N (16 (8bpp) / 8 (16bpp)) DMAI HW input channels. In case of 1*5x5 CNV mode, half of the DMAI HW input channels can be utilized, and each DMAI HW input channel holds 1 FM input channel, so 8 (8bpp) / 4 (16bpp) FM input channel can be processed in parallel.

In case of 2*3x3 CNV mode, 1 DMAI HW input channel holds a group of 2 FM input channels, so 32 (8bpp) / 16 (16bpp) FM input channels can be processed in parallel.

In case of 4*1x1 CNV mode, 1 DMAI HW input channel holds a group of 4 FM input channels, so 64 (8bpp) / 32 (16bpp) FM input channels can be processed in parallel.

In case of 8*1x1 CNV mode, 1 DMAI HW input channel holds a group of 8 FM input channels, so 128 (8bpp) / 64 (16bpp) FM input channels can be processed in parallel.

In case of 16*1x1 CNV mode, 1 DMAI HW input channel holds a group of 16 FM input channels, so 256 (8bpp) / 128 (16bpp) FM input channels can be processed in parallel.

The reachable performance for each mode is shown in Table U33.2.

**Table U33.2 CNN Performance**

Performance = DMAI channel * FM channel per group * OUTPUT channel * Kernel size * Frequency * Parallel pixels

| Pixel format | DMAI channel | FM channel per group | OUTPUT channel | Kernel size | Frequency | Parallel pixels | Performance |
|---|---|---|---|---|---|---|---|
| 8bpp | 8 | 1 | 32 | 5x5 | 800MHz | 1 | 5.1TMACs |
| 16bpp | 4 | 1 | 16 | 5x5 | 800MHz | 1 | 1.3 TMACs |
| 8bpp | 16 | 2 | 32 | 3x3 | 800MHz | 1 | 7.4 TMACs |
| 16bpp | 8 | 2 | 16 | 3x3 | 800MHz | 1 | 1.8 TMACs |
| 8bpp | 16 | 4 | 32 | 1x1 | 800MHz | 1 | 1.6 TMACs |
| 16bpp | 8 | 4 | 16 | 1x1 | 800MHz | 1 | 0.4 TMACs |
| 8bpp | 16 | 8 | 32 | 1x1 | 800MHz | 1 | 3.2 TMACs |
| 16bpp | 8 | 8 | 16 | 1x1 | 800MHz | 1 | 0.8 TMACs |
| 8bpp | 16 | 16 | 32 | 1x1 | 800MHz | 1 | 6.5 TMACs |
| 16bpp | 8 | 16 | 16 | 1x1 | 800MHz | 1 | 1.6 TMACs |

The following features are supported with the CNN-IP, for details and some limitations of features, please refer U33.3 Operation/ U33.4 Command list

- General feature:
  - Command list support: a list of supported instructions to control the CNN module is available
  - CL debugging support: Help users debug their programs by setting the break point address of CL
  - GOSUB/JUMP support: Absolute and relative address handing
  - 1 pixel processing per clock cycle
  - Double Buffering of DMAI, DMA3DC and DMAO parameters is supported.
- DMAI feature:
  - DMAI enable: enable/disable
  - Magnification mode of pixel repeat/skip (up/downscale): 0 to 4 for x, y direction
  - Magnification mode of pixel repeat (upscale) filling: repeat constant or repeat identity
  - DMAI data type: u8, s8, s16
  - DMAI transfer mode: planar or planar fast or packed
  - DMAI x length: 1 to 128 pixels
  - DMAI y length: 1 to 2048 rows
  - DMAI convolutional mode : 1*5x5, 2*3x3, 4*1x1, 8*1x1，16*1x1
  - Padding mode: Constant, identity (North/South: 0 to 7, West/East: 0 to 16)
  - Convolution stride (x-dir/y-dir): 1 to 3
  - Dilated convolution (x-dir): 0 to 3
  - Start position on FM(x-dir/y-dir): x-dir: 0 to 15, y-dir: 0 to 7
  - Start address: DMAI SPMC0 start address
- DMAW feature:
  - Start address: DMAW SPMC0 start address
  - Weight mode: depthwise mode,1*5x5, 2*3x3, 4*1x1, 8*1x1，16*1x1
  - Weight data type: s8, s16
  - DMAW transfer mode: packed
  - Weight preset mode: Setting N, VAL bits of CLW command via CL to select which mode to be used
- CNV feature:
  - Input channel processing: N (16bpp: 8, 8bpp: 16)
    - 2*3x3: 2*N (16bpp: 16, 8bpp: 32)
    - 4*1x1: 4*N (16bpp: 32, 8bpp: 64)
    - 8*1x1: 8*N (16bpp: 64, 8bpp: 128)

  - 16*1x1: 16*N (16bpp: 128, 8bpp: 256)
  - 1*5x5: (16pp: 4, 8bpp: 8)
- Output channel processing: (16bpp: 16, 8bpp: 32)
- Deconvolution is also supported (in case of semantic segmentation): Magnification + Convolution

- ARI feature:
  - ARI enable: enable/disable
  - ARI data type: s8, s16
  - ARI x length: 1 to 128 pixels
  - ARI y length: 1 to 2048 rows
  - DMA3DC value addition
  - Bias addition 16bit and 32bit: enable/disable
  - 16bit upscaler
  - Downshift of fractional position for MAC(ARI/DMAO) output channels: -47 to 0 (“-” means downshift)
  - Activation functions: RELU, BRELU, Leaky ReLU, Sigmoid, ELU, Tanh, H-Swish, Swish, Mish, Exp
  - Channel Pooling: Max, Rank, Label
  - X/Y pooling: XY MAX, XY average, Upper left XY

- DMA3DC feature:
  - Start address: DMA3DC SPMC1 start address
  - DMA3DC enable: enable/disable
  - DMA3DC data type: s8, s16, m16, m32
  - DMA3DC transfer mode: packed
  - Upshift of fractional position for all DMA3DC channel: 0 to 31
  - DMA3DC x length: 1 to 128 pixels
  - DMA3DC y length: 1 to 1024 rows
- DMAO feature :
  - Start address: DMAO SPMC0/SPMC1 start address
  - DMAO data type: s8, s16, m16, m32
  - DMAO x length: 1 to 128 pixels
  - DMAO y length: 1 to 2048 rows
  - DMAO transfer mode: planar fast/packed
  - DMAO enable: enable/disable
- Statistics generation:
  - Global SUM: sum of DMAO channel x (x: 0 to 31)
  - MAX: holding the maximal value of channel x (x: 0 to 31)
  - MIN: holding the minimum value of channel x (x: 0 to 31)
- The CNN-IP performs the following tasks in parallel:
  - DMAI for loading FM
  - Convolutional layer
    - 1*5x5 or 2*3x3 or 4*1x1 or 8*1x1 or 16*1x1convolution on the fly
  - Addition using DMA3DC:
    - Many channels should be processed (greater than number of input channels in CNN-IP)
    - Element-wise sum up: sum up the same pixel position over all channels.
  - Activation layer
  - Channel Pooling layer
  - XY Pooling layer
  - DMAO for storing the new FM for next layer

### U33.1.2 CNN Block Diagram

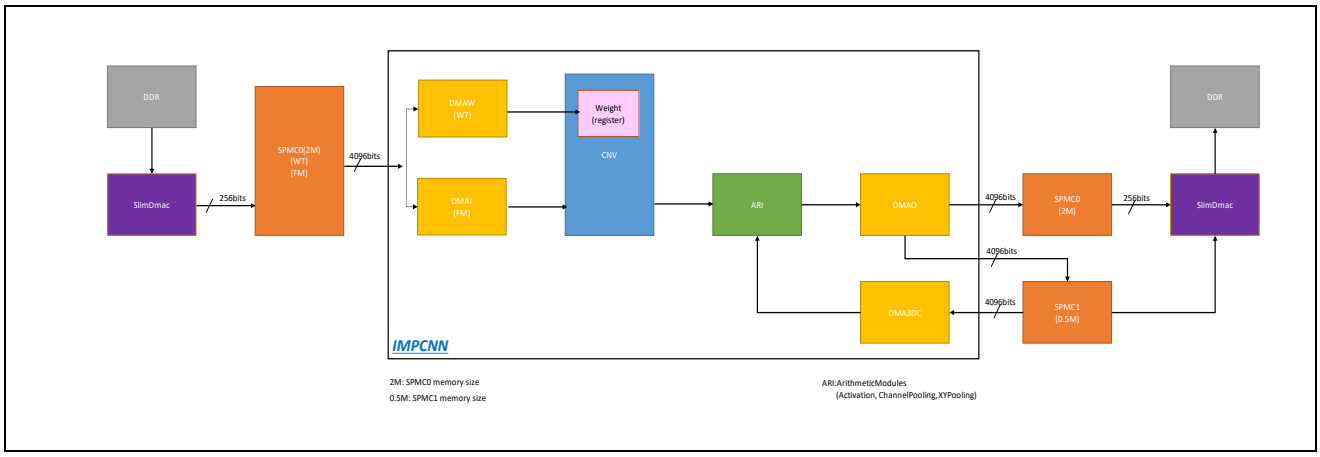



**Figure U33.4 CNN Block Diagram**

CNN IP includes the following main functional modules: DMAW is used to load weights. DMAI is responsible for loading FM from SPMC0 memory in planar type or planar fast type or packed format by setting the DMAIS_MODE bit of DMAIS register. CNV is responsible to do convolution based on kernel mode by setting DMAIS_TYPE bit of DMAIS register. Activation, Channel Pool, XY Pool are done in ARI by setting ARICSR register. DMAO is used to output the convolution results into SPMC0 memory in planar fast type or packed format (planar type is not supported in DMAO) by setting DMAOS_PLANAR bit of DMAOS register. Further, it is also used to output the temporary convolution results into SPMC1 memory. DMA3DC returns the previous run results to the current ARI module to add them, and the loop iterates through the execution until the last run.

### U33.1.3 CNN Register Configuration

Accessing to addresses not listed in this register list is prohibited. Operation cannot be guaranteed when accessing to an address, which is not listed. Access size is 32-bit. Operation cannot be guaranteed when accessing by other than 32-bit. Operation cannot be guaranteed when writing to a register from CPU during IMP CNN execution.

**Table U33.3 CNN Register Configuration Table for 4th Generation R-Car Series Products R-Car V4H/V4M**

| Register Name | Symbol | Access Possible From CP-R/ CP-W/ CL-R/ CL-W*1 | Value after Reset | Trigger | Offset Address*4 | RESET *2 |
|---|---|---|---|---|---|---|
| Version Control Register 0 | VCR0 | 1/0/0/0 | 0x00040100 | - | 0x0000 | - |
| Software reset | SWRST | 1/1/0/0 | 0 | - | 0x0008 | H |
| Multi-purpose reset | MPRST | 1/1/0/1 | 0 | - | 0x000C | S/H |
| Status Register | SR | 1/0/0/0 | 0 | - | 0x0010 | - |
| Status Register Enable | SRE | 1/1/0/0 | 0 | - | 0x0014 | S/H |
| Status Register Clear | SRC | 0/1/0/0 | 0 | - | 0x0018 | S/H |
| Status Register Mask | SRM | 1/1/0/0 | 0 | - | 0x001C | S/H |
| Index Register 0 | INDEX0 | 1/1/0/0 | 0 | - | 0x0020 | - |
| Index Register 1 | INDEX1 | 1/1/0/0 | 0 | - | 0x0024 | S/H |
| Index Register 2 | INDEX2 | 1/1/0/0 | 0 | - | 0x0028 | S/H |
| Index Register 3 | INDEX3 | 1/1/0/0 | 0 | - | 0x002C | S/H |
| Index Register 4 | INDEX4 | 1/1/0/0 | 0 | - | 0x0030 | S/H |
| Index Register 5 | INDEX5 | 1/1/0/0 | 0 | - | 0x0034 | S/H |
| Index Register 6 | INDEX6 | 1/1/0/0 | 0 | - | 0x0038 | S/H |
| Index Register 7 | INDEX7 | 1/1/0/0 | 0 | - | 0x003C | S/H |
| CL AXI PORT | CL_AXI_PORT | 1/1/0/1 | 0 | - | 0x0040 | S/H |
| CL program counter | CLPC | 1/0/0/0 | *3 | - | 0x0100 | S/H |
| Start address for Command List processing | SACL | 1/1/0/0 | 0 | - | 0x0104 | S/H |
| Start Command List Processing | SCLP | 1/1/0/0 | 0 | - | 0x0108 | S/H |
| CL executed instruction register | CLEIR | 1/0/0/0 | 0 | - | 0x010C | S/H |
| Start Layer Set Processing | SLSP | 1/1/0/1 | 0 | ARI trigger | 0x0110 | S/H |
| CL Break Address Register | CLBRKADDRR | 1/1/0/0 | 0 | - | 0x0114 | S/H |
| CL Conditional GOSUB Register | CLCNDGSBR | 1/1/0/1 | 0 | - | 0x0118 | S/H |
| V3U compatible mode | V3U_COMP_MODE | 1/1/0/0 | 0 | - | 0x011C | S/H |
| DMAI CONFIGuration register | DMAI_CONFIG | 1/1/0/1 | 0 | - | 0x0240 | S/H |
| DMAW CONFIGuration register | DMAW_CONFIG | 1/1/0/1 | 0 | - | 0x0250 | S/H |
| Global channel SUM | GLOB_SUM | 1/0/0/0 | 0 | - | 0x0800 – 0x087C | S/H |
| Channel maximum and channel minimum | MAX_MIN | 1/1/0/0 | 0x0000 FFFF | - | 0x0880 – 0x08BC | S/H |
| Performance counter 0 CTRL register | PERFCNT0CTRL | 1/1/0/1 | 0 | - | 0x08C0 | S/H |
| Performance counter 1 CTRL register | PERFCNT1CTRL | 1/1/0/1 | 0 | - | 0x08C4 | S/H |
| Performance counter 2 CTRL register | PERFCNT2CTRL | 1/1/0/1 | 0 | - | 0x08C8 | S/H |

| Register Name | Symbol | Access Possible From CP-R/ CP-W/ CL-R/ CL-W*1 | Value after Reset | Trigger | Offset Address*4 | RESET *2 |
|---|---|---|---|---|---|---|
| Performance counter 3 CTRL register | PERFCNT3CTRL | 1/1/0/1 | 0 | - | 0x08CC | S/H |
| Performance counter 0 | PERFCNT0 | 1/1/0/1 | 0 | - | 0x08D0 | S/H |
| Performance counter 1 | PERFCNT1 | 1/1/0/1 | 0 | - | 0x08D4 | S/H |
| Performance counter 2 | PERFCNT2 | 1/1/0/1 | 0 | - | 0x08D8 | S/H |
| Performance counter 3 | PERFCNT3 | 1/1/0/1 | 0 | - | 0x08DC | S/H |
| Synchronize Information register field 0-3 | SYNCC_00_03 | 1/1/0/1 | 0xFFFFFFFF | - | 0x0900 | S/H |
| Synchronize Information register field 4-7 | SYNCC_04_07 | 1/1/0/1 | 0xFFFFFFFF | - | 0x0904 | S/H |
| Synchronize Information register field 8-11 | SYNCC_08_11 | 1/1/0/1 | 0xFFFFFFFF | - | 0x0908 | S/H |
| Synchronize Information register field 12-15 | SYNCC_12_15 | 1/1/0/1 | 0xFFFFFFFF | - | 0x090C | S/H |
| ARI Configuration setup register | ARICSR | 1/1/0/1 | 0 | ARI | 0x1000 | S/H |
| ARI Channel Enable | ARIE | 1/1/0/1 | 0 | ARI | 0x1004 | S/H |
| Common padding, convolutional stride, startpos and dilated convolution value for all input channel | PADD_STRIDE_STARTPOS | 1/1/0/1 | 0x0005 0000 | DMAI | 0x1008 | S/H |
| ARI length | ARI_LEN | 1/1/0/1 | 0 | DMAI/ARI | 0x100C | S/H |
| SFTM zero point | SFTM_ZERO_POINT | 1/1/0/1 | 0 | ARI | 0x1010 | S/H |
| Multiplier post shift value for all channel used in activation unit | ACTIVATION_MULT_POST_SFT | 1/1/0/1 | 0 | ARI | 0x1014 | S/H |
| Multiplier value for all channel used in activation unit | ACTIVATION_MULT_VAL | 1/1/0/1 | 0x0000 0001 | ARI | 0x1018 | S/H |
| Scaler multiplier value | SCALER_MULT_VAL | 0/1/0/1 | 0 | ARI | 0x101C | - |
| Bias for all channel | BIAS | 0/1/0/1 | 0 | ARI | 0x1020 | - |
| Padding constant value | PADD_CONST_VAL | 1/1/0/1 | 0 | DMAI | 0x1024 | S/H |
| Parameter used at activation output | ACTIVATION_OUTPUT_PARAM | 1/1/0/1 | 0x0000 0001 | ARI | 0x1028 | S/H |
| Padding size configuration register for input channel 0 and 1 | PADD00_01 | 1/1/0/1 | 0 | DMAI | 0x1100 | S/H |
| Padding size configuration register for input channel 4 and 5 | PADD04_05 | 1/1/0/1 | 0 | DMAI | 0x1104 | S/H |
| Padding size configuration register for input channel 8 and 9 | PADD08_09 | 1/1/0/1 | 0 | DMAI | 0x1108 | S/H |
| Padding size configuration register for input channel 12 and 13 | PADD12_13 | 1/1/0/1 | 0 | DMAI | 0x110C | S/H |
| Bias lower 16bit register for channel 0 & 1 | BIAS00_01 | 1/1/0/1 | 0 | ARI | 0x1138 | S/H |
| Bias lower 16bit register for channel 2 & 3 | BIAS02_03 | 1/1/0/1 | 0 | ARI | 0x113C | S/H |
| Bias lower 16bit register for channel 4 & 5 | BIAS04_05 | 1/1/0/1 | 0 | ARI | 0x1140 | S/H |
| Bias lower 16bit register for channel 6 & 7 | BIAS06_07 | 1/1/0/1 | 0 | ARI | 0x1144 | S/H |
| Bias lower 16bit register for channel 8 & 9 | BIAS08_09 | 1/1/0/1 | 0 | ARI | 0x1148 | S/H |
| Bias lower 16bit register for channel 10 & 11 | BIAS10_11 | 1/1/0/1 | 0 | ARI | 0x114C | S/H |
| Bias lower 16bit register for channel 12 & 13 | BIAS12_13 | 1/1/0/1 | 0 | ARI | 0x1150 | S/H |
| Bias lower 16bit register for channel 14 & 15 | BIAS14_15 | 1/1/0/1 | 0 | ARI | 0x1154 | S/H |

| Register Name | Symbol | Access Possible From CP-R/ CP-W/ CL-R/ CL-W*1 | Value after Reset | Trigger | Offset Address*4 | RESET *2 |
|---|---|---|---|---|---|---|
| Bias lower 16bit register for channel 16 & 17 | BIAS16_17 | 1/1/0/1 | 0 | ARI | 0x1158 | S/H |
| Bias lower 16bit register for channel 18 & 19 | BIAS18_19 | 1/1/0/1 | 0 | ARI | 0x115C | S/H |
| Bias lower 16bit register for channel 20 & 21 | BIAS20_21 | 1/1/0/1 | 0 | ARI | 0x1160 | S/H |
| Bias lower 16bit register for channel 22 & 23 | BIAS22_23 | 1/1/0/1 | 0 | ARI | 0x1164 | S/H |
| Bias lower 16bit register for channel 24 & 25 | BIAS24_25 | 1/1/0/1 | 0 | ARI | 0x1168 | S/H |
| Bias lower 16bit register for channel 26 & 27 | BIAS26_27 | 1/1/0/1 | 0 | ARI | 0x116C | S/H |
| Bias lower 16bit register for channel 28 & 29 | BIAS28_29 | 1/1/0/1 | 0 | ARI | 0x1170 | S/H |
| Bias lower 16bit register for channel 30 & 31 | BIAS30_31 | 1/1/0/1 | 0 | ARI | 0x1174 | S/H |
| Bias upper 16bit register for channel 0 & 1 | BIAS00_01_H | 1/1/0/1 | 0 | ARI | 0x1178 | S/H |
| Bias upper 16bit register for channel 2 & 3 | BIAS02_03_H | 1/1/0/1 | 0 | ARI | 0x117C | S/H |
| Bias upper 16bit register for channel 4 & 5 | BIAS04_05_H | 1/1/0/1 | 0 | ARI | 0x1180 | S/H |
| Bias upper 16bit register for channel 6 & 7 | BIAS06_07_H | 1/1/0/1 | 0 | ARI | 0x1184 | S/H |
| Bias upper 16bit register for channel 8 & 9 | BIAS08_09_H | 1/1/0/1 | 0 | ARI | 0x1188 | S/H |
| Bias upper 16bit register for channel 10 & 11 | BIAS10_11_H | 1/1/0/1 | 0 | ARI | 0x118C | S/H |
| Bias upper 16bit register for channel 12 & 13 | BIAS12_13_H | 1/1/0/1 | 0 | ARI | 0x1190 | S/H |
| Bias upper 16bit register for channel 14 & 15 | BIAS14_15_H | 1/1/0/1 | 0 | ARI | 0x1194 | S/H |
| Bias upper 16bit register for channel 16 & 17 | BIAS16_17_H | 1/1/0/1 | 0 | ARI | 0x1198 | S/H |
| Bias upper 16bit register for channel 18 & 19 | BIAS18_19_H | 1/1/0/1 | 0 | ARI | 0x119C | S/H |
| Bias upper 16bit register for channel 20 & 21 | BIAS20_21_H | 1/1/0/1 | 0 | ARI | 0x11A0 | S/H |
| Bias upper 16bit register for channel 22 & 23 | BIAS22_23_H | 1/1/0/1 | 0 | ARI | 0x11A4 | S/H |
| Bias upper 16bit register for channel 24 & 25 | BIAS24_25_H | 1/1/0/1 | 0 | ARI | 0x11A8 | S/H |
| Bias upper 16bit register for channel 26 & 27 | BIAS26_27_H | 1/1/0/1 | 0 | ARI | 0x11AC | S/H |
| Bias upper 16bit register for channel 28 & 29 | BIAS28_29_H | 1/1/0/1 | 0 | ARI | 0x11B0 | S/H |
| Bias upper 16bit register for channel 30 & 31 | BIAS30_31_H | 1/1/0/1 | 0 | ARI | 0x11B4 | S/H |
| Scaler multiplier value for channel 0 & 1 | SCALER_MULT_00_01 | 1/1/0/1 | 0x00010001 | ARI | 0x11C0 | S/H |
| Scaler multiplier value for channel 2 & 3 | SCALER_MULT_02_03 | 1/1/0/1 | 0x00010001 | ARI | 0x11C4 | S/H |
| Scaler multiplier value for channel 4 & 5 | SCALER_MULT_04_05 | 1/1/0/1 | 0x00010001 | ARI | 0x11C8 | S/H |
| Scaler multiplier value for channel 6 & 7 | SCALER_MULT_06_07 | 1/1/0/1 | 0x00010001 | ARI | 0x11CC | S/H |
| Scaler multiplier value for channel 8 & 9 | SCALER_MULT_08_09 | 1/1/0/1 | 0x00010001 | ARI | 0x11D0 | S/H |
| Scaler multiplier value for channel 10 & 11 | SCALER_MULT_10_11 | 1/1/0/1 | 0x00010001 | ARI | 0x11D4 | S/H |

| Register Name | Symbol | Access Possible From CP-R/ CP-W/ CL-R/ CL-W*1 | Value after Reset | Trigger | Offset Address*4 | RESET *2 |
|---|---|---|---|---|---|---|
| Scaler multiplier value for channel 12 & 13 | SCALER_MULT_12_13 | 1/1/0/1 | 0x00010001 | ARI | 0x11D8 | S/H |
| Scaler multiplier value for channel 14 & 15 | SCALER_MULT_14_15 | 1/1/0/1 | 0x00010001 | ARI | 0x11DC | S/H |
| Scaler multiplier value for channel 16 & 17 | SCALER_MULT_16_17 | 1/1/0/1 | 0x00010001 | ARI | 0x11E0 | S/H |
| Scaler multiplier value for channel 18 & 19 | SCALER_MULT_18_19 | 1/1/0/1 | 0x00010001 | ARI | 0x11E4 | S/H |
| Scaler multiplier value for channel 20 & 21 | SCALER_MULT_20_21 | 1/1/0/1 | 0x00010001 | ARI | 0x11E8 | S/H |
| Scaler multiplier value for channel 22 & 23 | SCALER_MULT_22_23 | 1/1/0/1 | 0x00010001 | ARI | 0x11EC | S/H |
| Scaler multiplier value for channel 24 & 25 | SCALER_MULT_24_25 | 1/1/0/1 | 0x00010001 | ARI | 0x11F0 | S/H |
| Scaler multiplier value for channel 26 & 27 | SCALER_MULT_26_27 | 1/1/0/1 | 0x00010001 | ARI | 0x11F4 | S/H |
| Scaler multiplier value for channel 28 & 29 | SCALER_MULT_28_29 | 1/1/0/1 | 0x00010001 | ARI | 0x11F8 | S/H |
| Scaler multiplier value for channel 30 & 31 | SCALER_MULT_30_31 | 1/1/0/1 | 0x00010001 | ARI | 0x11FC | S/H |
| Shift for all 3DC channels | SFTC | 0/1/0/1 | 0 | ARI | 0x2004 | - |
| Downshift MAC | SFTM | 0/1/0/1 | 0 | ARI | 0x2008 | - |
| Zero point inside activation unit | ACTIVATION_ZERO_POINT | 1/1/0/1 | 0 | ARI | 0x20D8 | S/H |
| Boundaries for BReLU | ARI_BND | 1/1/0/1 | 0x8000 7FFF | ARI | 0x20DC | S/H |
| Downshift of MAC output channel 0 to 3 | SFTM00_03 | 1/1/0/1 | 0 | ARI | 0x2500 | S/H |
| Downshift of MAC output channel 3 to 7 | SFTM04_07 | 1/1/0/1 | 0 | ARI | 0x2504 | S/H |
| Downshift of MAC output channel 8 to 11 | SFTM08_11 | 1/1/0/1 | 0 | ARI | 0x2508 | S/H |
| Downshift of MAC output channel 12 to 15 | SFTM12_15 | 1/1/0/1 | 0 | ARI | 0x250C | S/H |
| Downshift of MAC output channel 16 to 19 | SFTM16_19 | 1/1/0/1 | 0 | ARI | 0x2510 | S/H |
| Downshift of MAC output channel 20 to 23 | SFTM20_23 | 1/1/0/1 | 0 | ARI | 0x2514 | S/H |
| Downshift of MAC output channel 24 to 27 | SFTM24_27 | 1/1/0/1 | 0 | ARI | 0x2518 | S/H |
| Downshift of MAC output channel 28 to 31 | SFTM28_31 | 1/1/0/1 | 0 | ARI | 0x251C | S/H |
| Downshift of 3DC channel 0 to 3 | SFTC00_03 | 1/1/0/1 | 0 | ARI | 0x2520 | S/H |
| Downshift of 3DC channel 3 to 7 | SFTC04_07 | 1/1/0/1 | 0 | ARI | 0x2524 | S/H |
| Downshift of 3DC channel 8 to 11 | SFTC08_11 | 1/1/0/1 | 0 | ARI | 0x2528 | S/H |
| Downshift of 3DC channel 12 to 15 | SFTC12_15 | 1/1/0/1 | 0 | ARI | 0x252C | S/H |
| Downshift of 3DC channel 16 to 19 | SFTC16_19 | 1/1/0/1 | 0 | ARI | 0x2530 | S/H |
| Downshift of 3DC channel 20 to 23 | SFTC20_23 | 1/1/0/1 | 0 | ARI | 0x2534 | S/H |
| Downshift of 3DC channel 24 to 27 | SFTC24_27 | 1/1/0/1 | 0 | ARI | 0x2538 | S/H |
| Downshift of 3DC channel 28 to 31 | SFTC28_31 | 1/1/0/1 | 0 | ARI | 0x253C | S/H |
| DMAI wrap around start address 7 | DMAIWPSA7 | 1/1/0/1 | 0 | DMAI | 0x3FE0 | S/H |
| DMAI wrap around parameter 7 | DMAIWPA7 | 1/1/0/1 | H'1000 0000 | DMAI | 0x3FE4 | S/H |
| DMAI wrap around start address 6 | DMAIWPSA6 | 1/1/0/1 | 0 | DMAI | 0x3FE8 | S/H |
| DMAI wrap around parameter 6 | DMAIWPA6 | 1/1/0/1 | H'1000 0000 | DMAI | 0x3FEC | S/H |

| Register Name | Symbol | Access Possible From CP-R/ CP-W/ CL-R/ CL-W*1 | Value after Reset | Trigger | Offset Address*4 | RESET *2 |
|---|---|---|---|---|---|---|
| DMAI wrap around start address 5 | DMAIWPSA5 | 1/1/0/1 | 0 | DMAI | 0x3FF0 | S/H |
| DMAI wrap around parameter 5 | DMAIWPA5 | 1/1/0/1 | H'1000 0000 | DMAI | 0x3FF4 | S/H |
| DMAI wrap around start address 4 | DMAIWPSA4 | 1/1/0/1 | 0 | DMAI | 0x3FF8 | S/H |
| DMAI wrap around parameter 4 | DMAIWPA4 | 1/1/0/1 | H'1000 0000 | DMAI | 0x3FFC | S/H |
| DMAI wrap around start address 3 | DMAIWPSA3 | 1/1/0/1 | 0 | DMAI | 0x4000 | S/H |
| DMAI wrap around parameter 3 | DMAIWPA3 | 1/1/0/1 | H'1000 0000 | DMAI | 0x4004 | S/H |
| DMAI wrap around start address 2 | DMAIWPSA2 | 1/1/0/1 | 0 | DMAI | 0x4008 | S/H |
| DMAI wrap around parameter 2 | DMAIWPA2 | 1/1/0/1 | H'1000 0000 | DMAI | 0x400C | S/H |
| DMAI wrap around start address 1 | DMAIWPSA1 | 1/1/0/1 | 0 | DMAI | 0x4010 | S/H |
| DMAI wrap around parameter 1 | DMAIWPA1 | 1/1/0/1 | H'1000 0000 | DMAI | 0x4014 | S/H |
| DMAI wrap around parameter | DMAIWPA | 1/1/0/1 | H'1000 0000 | DMAI | 0x4018 | S/H |
| DMAI channel offset | DMAICO | 1/1/0/1 | 0 | DMAI | 0x401C | S/H |
| DMAI internal memory start address | DMAISA | 1/1/0/1 | 0 | DMAI | 0x4020 | S/H |
| DMAI length | DMAIL | 1/1/0/1 | 0 | DMAI | 0x4024 | S/H |
| DMAI parameter | DMAIPARAM | 1/1/0/1 | 0 | DMAI | 0x4028 | S/H |
| DMAI internal memory offset address | DMAIOA | 0/1/0/1 | 0 | - | 0x402C | - |
| DMAI channel enable | DMAIE | 1/1/0/1 | 0 | DMAI | 0x4030 | S/H |
| DMAI start | DMAIS | 1/1/0/1 | 0 | DMAI trigger | 0x4034 | S/H |
| DMAW wrap around parameter | DMAWWA | 1/1/0/1 | 0 | DMAW | 0x4040 | S/H |
| DMAW start address | DMAWSA | 1/1/0/1 | 0 | DMAW | 0x4044 | S/H |
| DMAW offset address | DMAWOA | 101/0/1 | 0 | - | 0x4048 | - |
| DMAW start | DMAWS | 1/1/0/1 | 0 | DMAW trigger | 0x404C | S/H |
| DMAI internal memory start address, input channel 0 | DMAISA00 | 1/1/0/1 | 0 | DMAI | 0x4100 | S/H |
| DMAI internal memory start address, input channel 1 | DMAISA01 | 1/1/0/1 | 0 | DMAI | 0x4104 | S/H |
| DMAI internal memory start address, input channel 4 | DMAISA04 | 1/1/0/1 | 0 | DMAI | 0x4108 | S/H |
| DMAI internal memory start address, input channel 5 | DMAISA05 | 1/1/0/1 | 0 | DMAI | 0x410C | S/H |
| DMAI internal memory start address, input channel 8 | DMAISA08 | 1/1/0/1 | 0 | DMAI | 0x4110 | S/H |
| DMAI internal memory start address, input channel 9 | DMAISA09 | 1/1/0/1 | 0 | DMAI | 0x4114 | S/H |
| DMAI internal memory start address, input channel 12 | DMAISA12 | 1/1/0/1 | 0 | DMAI | 0x4118 | S/H |
| DMAI internal memory start address, input channel 13 | DMAISA13 | 1/1/0/1 | 0 | DMAI | 0x411C | S/H |
| DMA3DC internal memory start address | DMA3DCSA | 1/1/0/1 | 0 | DMA3DC | 0x5000 | S/H |
| DMA3DC length | DMA3DCL | 1/1/0/1 | 0 | DMA3DC | 0x5004 | S/H |
| DMA3DC parameter | DMA3DCPARAM | 1/1/0/1 | 0 | DMA3DC | 0x5008 | S/H |
| DMA3DC internal memory offset address | DMA3DCOA | 0/1/0/1 | 0 | - | 0x500C | - |
| DMA3DC channel enable | DMA3DCE | 1/1/0/1 | 0 | DMA3DC | 0x5010 | S/H |
| DMA3DC start | DMA3DCS | 1/1/0/1 | 0 | DMA3DC trigger | 0x5014 | S/H |
| DMAO wrap around parameter | DMAOWA | 1/1/0/1 | H'1000 0000 | DMAO | 0x6000 | S/H |

| Register Name | Symbol | Access Possible From CP-R/ CP-W/ CL-R/ CL-W*1 | Value after Reset | Trigger | Offset Address*4 | RESET *2 |
|---|---|---|---|---|---|---|
| DMAO channel offset | DMAOCO | 1/1/0/1 | 0 | DMAO | 0x6004 | S/H |
| DMAO internal memory start address | DMAOSA | 1/1/0/1 | 0 | DMAO | 0x6008 | S/H |
| DMAO length | DMAOL | 1/1/0/1 | 0 | DMAO | 0x600C | S/H |
| DMAO parameter | DMAOPARAM | 1/1/0/1 | 0 | DMAO | 0x6010 | S/H |
| DMAO internal memory offset address | DMAOOA | 0/1/0/1 | 0 | - | 0x6014 | - |
| DMAO channel enable | DMAOE | 1/1/0/1 | 0 | DMAO | 0x6018 | S/H |
| DMAO start | DMAOS | 1/1/0/1 | 0 | DMAO trigger | 0x601C | S/H |
| DMA Register Dump External Start Address | DMARDESA | 1/1/0/1 | 0 | DMAO | 0x6020 | S/H |

Notes:

1. CP = central processor, CL = command list, R = read, W = write
2. H: Power on reset, S: Software reset by the SWRST bit of the SWRST register
3. Undefined value.
4. In the CNN register address area, offset addresses larger than 0x7000 are forbidden to access for read and write direction by WPR instruction and other IP.

### U33.1.4 CNN Connected Module

The modules connected to CNN and their function descriptions are shown in Table U33.4.

**Table U33.4 Connected Modules**

| Connected Modules | Function of the Connected Modules |
|---|---|
| CPG | Reset and Clock Stop Signal Supplier |
| INTTOP | Interrupt Controller |
| ECM | FUSA error Interrupt Controller |

Note: FUSA (Road Vehicle Functional Safety)

## U33.2 Register Description

Initial value: Register value after a reset
—: Undefined value
R/W: Readable/writable. The written value can be read.
R: Read-only. The write value should be the same as the read value.
-/W: Write-only. The read value is undefined.

Refer to R-CarV4M_UM_U33_IMP_CNN_r0p70.xlsx

## U33.3 Operation

### U33.3.1 SPMC storage format

This section describes data format stored in SPMC (Scratchpad memory).

**Table U33.5 Data format stored in SPMC0/1**

| Format in SPMC0/1 | | DMAI/DMAW | DMA3DC | DMAO |
|---|---|---|---|---|
| Planar format (FM) | Planar type (s8, u8, s16) | √ | | |
| | Planar fast type (s8, u8, s16) | √ | | √ |
| Packed format (FM) | Packed s8/u8 type | √ | √ | √ |
| | Packed s16 type | √ | √ | √ |
| | Packed m16 type | | √ | √ |
| | Packed m32 type | | √ | √ |
| Packed format (Weight) | 8*1x1 s8 | √ | | |
| | 8*1x1 s16 | √ | | |
| | 4*1x1 s8 | √ | | |
| | 4*1x1 s16 | √ | | |
| | 2*3x3 s8 | √ | | |
| | 2*3x3 s16 | √ | | |
| | 1*5x5 s8 | √ | | |
| | 1*5x5 s16 | √ | | |
| | 16*1x1 s8 | √ | | |
| | 16*1x1 s16 | √ | | |
| | Depthwise s8 | √ | | |
| | Depthwise s16 | √ | | |

The WT output is defined as ich{n}-och{m}-w{p}h{q} (WT) and the FM output as ich{n} (FM), and each symbol has the meaning shown in Table U33.6.

**Table U33.6 Symbol in Block Diagram**

| Data Type | Symbol | Descriptions |
|---|---|---|
| WT | ich{n} | WT for input channel{n} (n = 0 to 255) |
| | och{m} | WT for output channel{m} (m = 0 to 31) |
| | w{p} | WT for kernel width{p} (p = 0 to 4) |
| | h{q} | WT for kernel height{q} (q = 0 to 4) |
| FM | ich{n} | FM for input channel{n} (n = 0 to 255) (*1) |

The ranges of {n}, {m}, {p}, and {q} above depend on the Kernel mode. The ranges of ich, och, and wh for each kernel are listed in Table U33.7. For details of each Kernel, refer to section (2).

(*1) The index of ich is converted by the part function in Figure U33.20.

**Table U33.7 Value range of ich, och, wh per kernel**

| Kernel | n (Ich{n}) | m (och{m}) | p (w{p}} | q(h{q}) |
|---|---|---|---|---|
| 16*1x1 | 0 to 255 | 0 to 31 | 0 | 0 |
| 8*1x1 | 0 to 127 | | 0 | 0 |
| 4*1x1 | 0 to 63 | | 0 | 0 |
| 2*3x3 | 0 to 31 | | 0 to 2 | 0 to 2 |

| | | | | |
|---|---|---|---|---|
| 1*5x5 | 0 to 7 | | 0 to 4 | 0 to 4 |

**(1) Planar Format**

**(a) Planar type**

In planar type, one channel's data is stored continuously in SPMC0. An example of ChN (channel N) data stored in SPMC0 is illustrated in Figure U33.5. Data in planar type is 2D, it is possible to set the line stride separately from the 1-line size setting (x-size). The storage of 8bpp/16bpp data in SPMC0 is little-endian, 16bpp data must be stored in 2byte alignment.

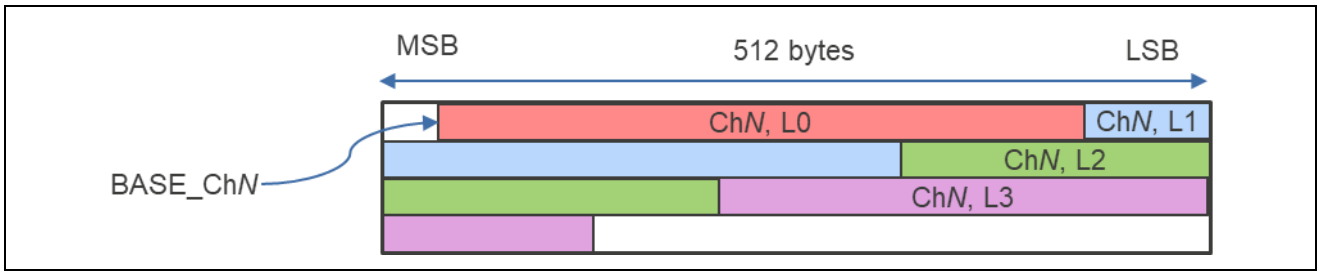



**Figure U33.5 Planar type**

Planar type for DMAI is selected by specifying the DMAIS_MODE bit of DMAIS register as 0. Note that planar type cannot be selected for DMAO.

### (b) Planar fast type

When FM is transferred by planar fast type, multiple channels (32ch *16pix) can be read from SPMC0 simultaneously, so that the data to be provided to the CNV module is transmitted earlier than by planar type. Planar fast type can be selected for the input data by specifying the DMAIS_MODE bit of DMAIS register as 1 and the DMAOS_PLANAR bit of DMAOS register as 1 for the output data, respectively.

To enable planar fast type, the BASE_ChN (the start address of channel N) must be 1 pixel aligned (no magnification) or 16byte aligned (magnification used). The line stride must be 16 pixels aligned. Assuming that one channel's size is M byte, BASE_ChN, is calculated as follows.

8bit: BASE_ChN = BASE_Ch0 + (16 * Skip_factor + FLOOR(M,512)) * ChN
Skip_factor is an odd number from 1 to 33. If the channel size is M byte,
Skip_factor = 2 * (FLOOR ((M - FLOOR(M,512) + 15) / 32,1)) + 1,
If all channels's size is M byte, the channel stride is a constant value.

16bit: BASE_ChN = BASE_Ch0 + (32 * Skip_factor + FLOOR(M,512)) * ChN
Skip_factor is an odd number from 1 to 17. If the channel size is M byte,
Skip_factor = 2 * (FLOOR ((M - FLOOR(M,512) + 31) / 64,1)) + 1,
If all channels' size is M byte, the channel stride is a constant value.

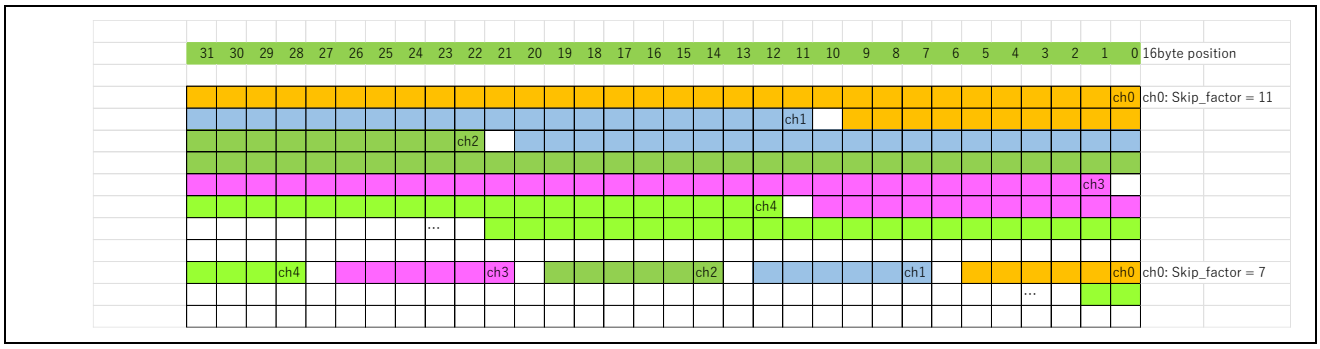



**Figure U33.6 Placement of each channel of planar fast type in SPMC0**

### (2) Packed format (FM)

In packed format, 32 channels' data with 32 channels/1pixels (8bpp) or 16 channels' data with 16 channels/1 pixel (16bpp) is packed and stored in SPMC0. Since multiple channels (32 channels or 16 channels) can be read from SPMC0 at the same time, data can be supplied to the CNV module at an earlier time point than in the planar type. There are four types of packed formats (a) to (d).

### (a) Packed s8/u8 type

32 channels/1 pixel data is stored in raster scan order. 32 channels*16 pixels*8bpp of 512byte alignment is the element unit. The horizontal stride is DMAxST(stride) and the vertical size is DMAxLY. The horizontal stride DMAxST must be specified as multiple of 16 pixels. If the channel number to be accessed is C (C up to 32 channels. In case of larger number of channels, channels get divided in different groups of up to 32 channel and are separately stored.), the horizontal position is X, and the vertical position is Y, then the Offset from the base address in SPMC0 is
Offset =( Y* DMAxST + X ) *32+C
The above is independent of kernel size, and all have the same offset. Figure U33.7 shows an example with DMAxST=48.

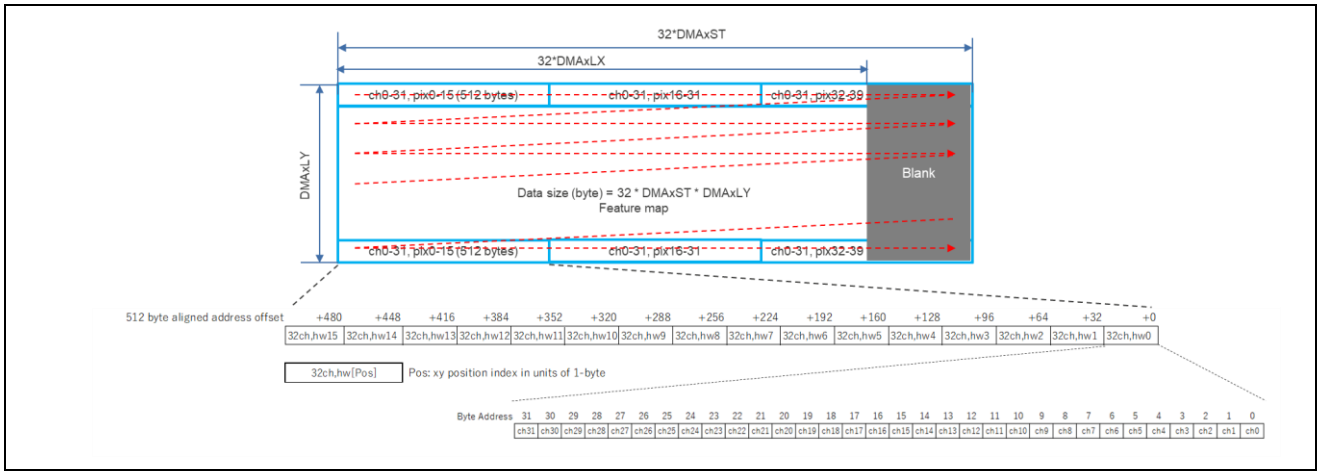



**Figure U33.7 Packed s8/u8 type**

### (b) Packed s16 type

16 channels/1 pixels data is stored in raster scan order. 16 channels*16 pixels*16bpp of 512byte alignment is the element unit. The horizontal stride is DMAxST(stride) and the vertical size is DMAxLY. The horizontal stride DMAxST must be specified as multiple of 16 pixels. If the channel number to be accessed is C (C up to 16 channels. In case of larger number of channels, channels get divided in different groups of up to 16 channel and are separately stored.), the horizontal position is X and the vertical position is Y, the Offset from the base address in SPMC0 is
Offset = (Y * DMAxST + X) * 16 * 2 + C * 2
The above is independent of kernel size, and all have the same offset.
Figure U33.8 shows an example with DMAxST=48.

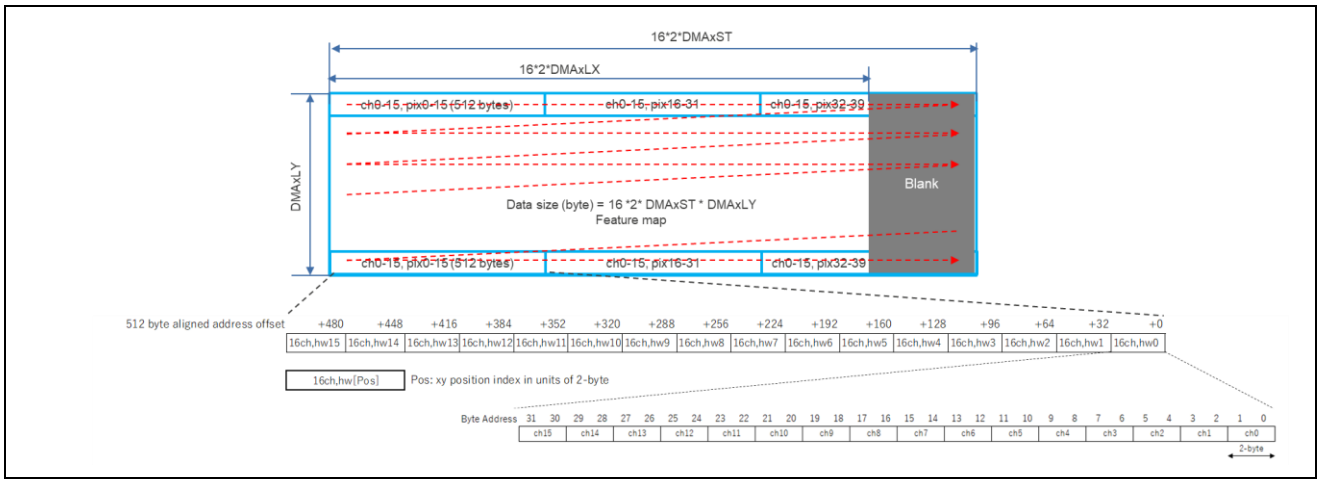



**Figure U33.8 Packed s16 type**

### (c) Packed m16 type

Data for 32 channels is stored in raster scan order. 32 channels*8 pixels*16bpp of 512byte align is the element unit. The horizontal stride is DMAxST(stride) and the vertical size is DMAxLY. The horizontal stride DMAxST must be specified as multiple of 8 pixels If the channel number to be accessed is C(C up to 32 channels. In case of larger number of channels, channels get divided in different groups of up to 32 channel and are separately stored.), the horizontal position is X and the vertical position is Y, the Offset from the base address in SPMC1 memory is
Offset = (Y * DMAxST + X) * 32 * 2 +C*2
The above is independent of kernel size, and all have the same offset. Figure U33.9 shows an example with DMAxST=24.

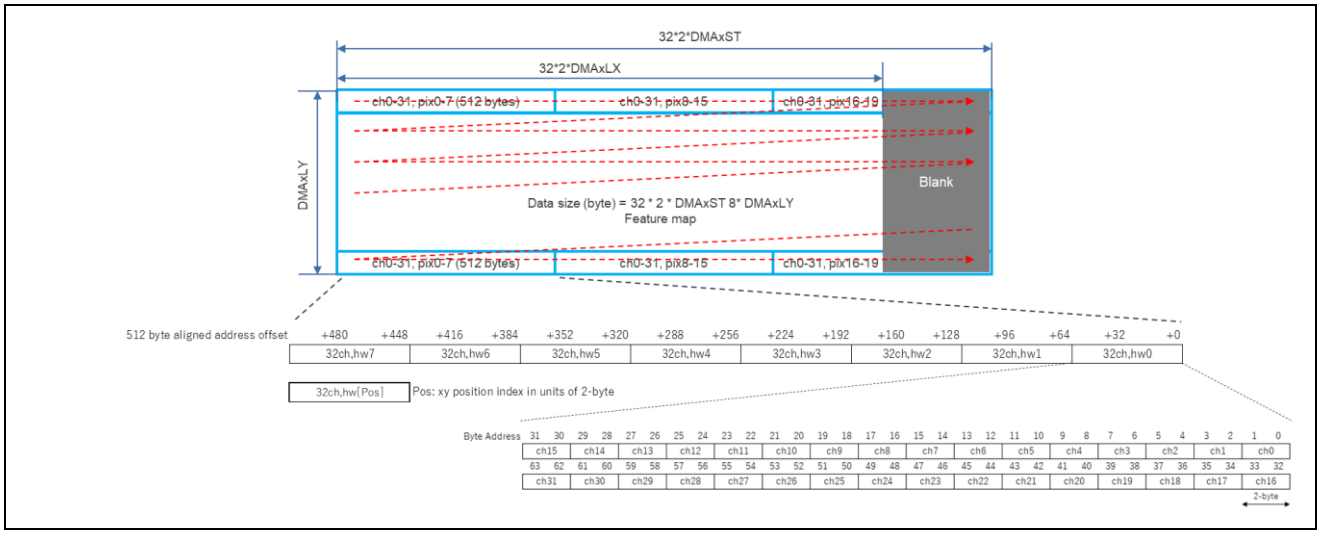



**Figure U33.9 Packed m16 type**

### (d) Packed m32 type

Data for 32 channels is stored in raster scan order. 32 channels*4 pixels*32bpp of 512byte align is the element unit. The horizontal stride is DMAxST(stride) and the vertical size is DMAxLY. The horizontal stride DMAxST must be specified as multiple of 4 pixels. If the channel number to be accessed is C (C up to 32 channels. In case of larger number of channels, channels get divided in different groups of up to 32 channel and are separately stored.), the horizontal position is X and the vertical position is Y, the Offset from the base address in SPMC1 is

Offset = (Y * DMAxST + X) * 32 * 4 + C * 4

The above is independent of kernel size, and all have the same offset. Figure U33.10 shows an example with DMAxST=12.

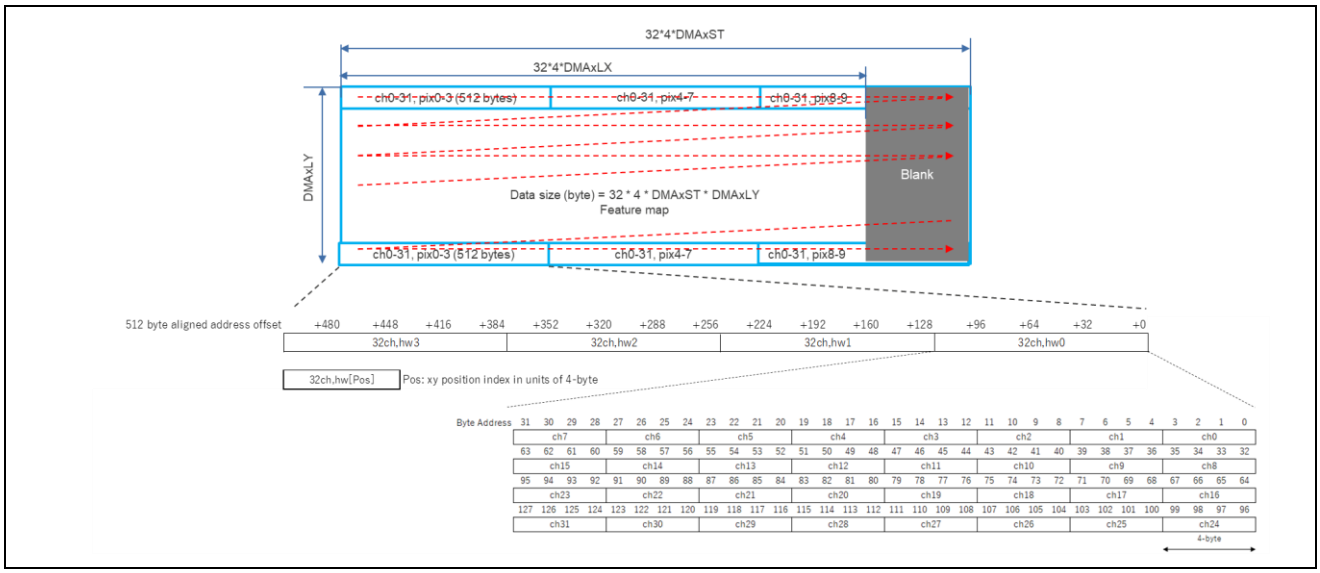



**Figure U33.10 Packed m32 type**

### (3) Packed format (weight)

DMAI uploads weights from SPMC0 and transfers them to the CNV module. The weight formats corresponding to each kernel mode and data formats are shown in (a) to (f). Please note, weight area size must be set in multiple of 512byte.

### (a) 16*1x1

Weight area size is 8KB (s8) or 4KB (s16) in 16*1x1 kernel mode. The data order of input channels within 32byte is LSB first (Little Endian).Weight coefficients of unused channels must be filled with 0 data.

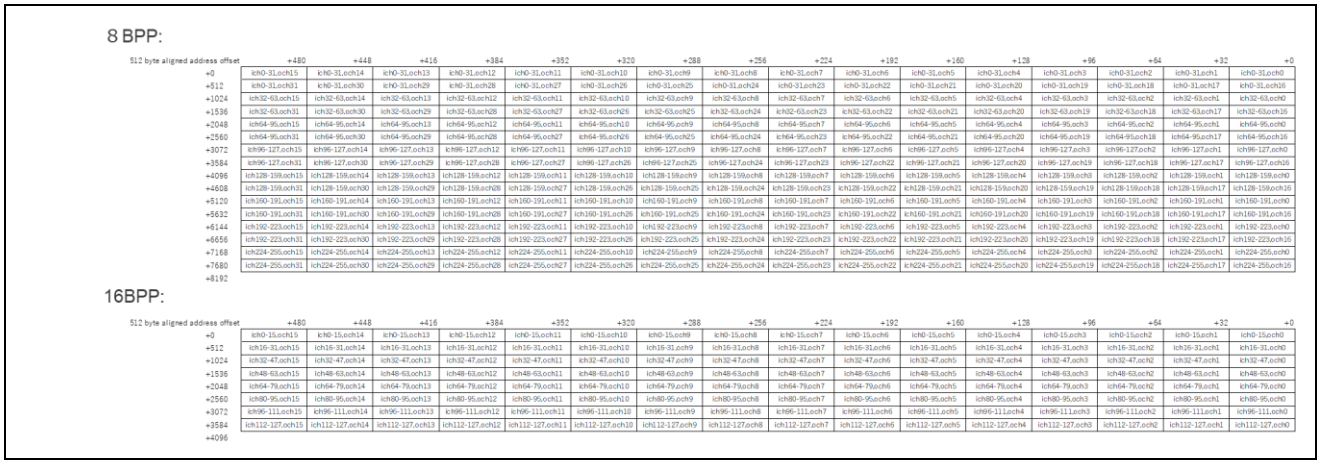



**Figure U33.11 Weight 16*1x1**

### (b) 8*1x1

Weight area size is 4KB (s8) or 2KB (s16) in 8*1x1 kernel mode. The data order of input channels within 32byte is LSB first (Little Endian).Weight coefficients of unused channels must be filled with 0 data.

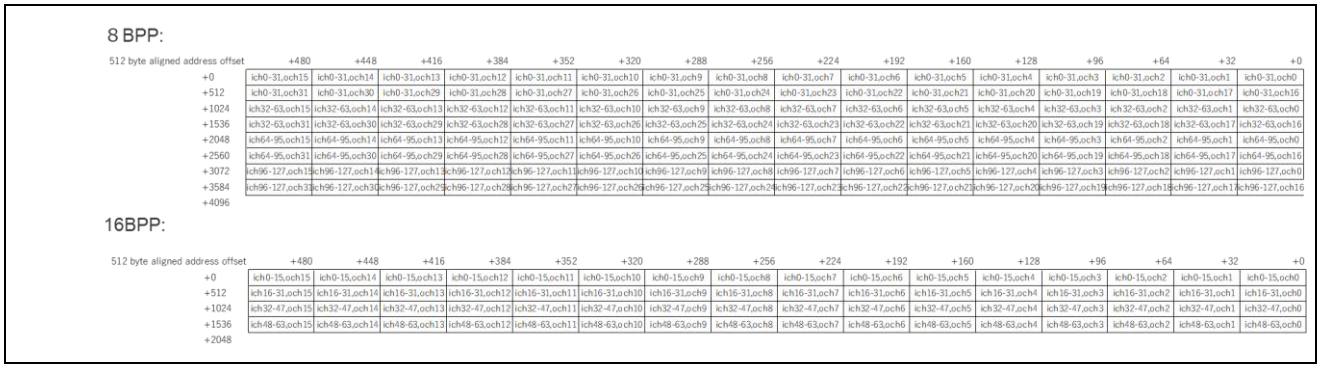



**Figure U33.12 Weight 8*1x1**

### (c) 4*1x1

Weight area size is 2KB (s8) or 1KB (s16) in 4*1x1 kernel mode. The data order of input channels within 32byte is LSB first. Weight coefficients of unused channels must be filled with 0 data.

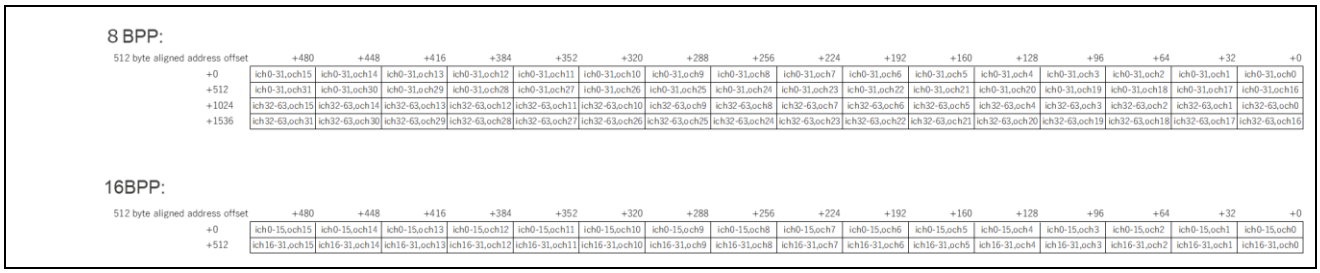



**Figure U33.13 Weight 4*1x1**

### (d) 2*3x3

Weight area size is 9KB (s8) or 4.5KB (s16) in 2*3x3 kernel mode, weights are stored in raster order. The data order of input channels within 32byte is LSB first. Weight coefficients of unused channels must be filled with 0 data.

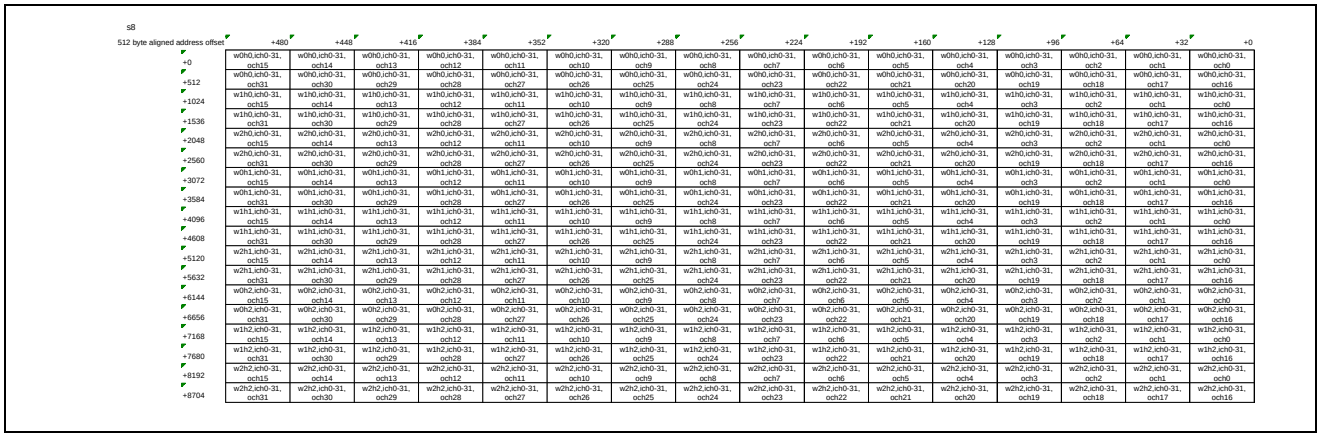



**Figure U33.14 Weight 2*3x3 (s8)**

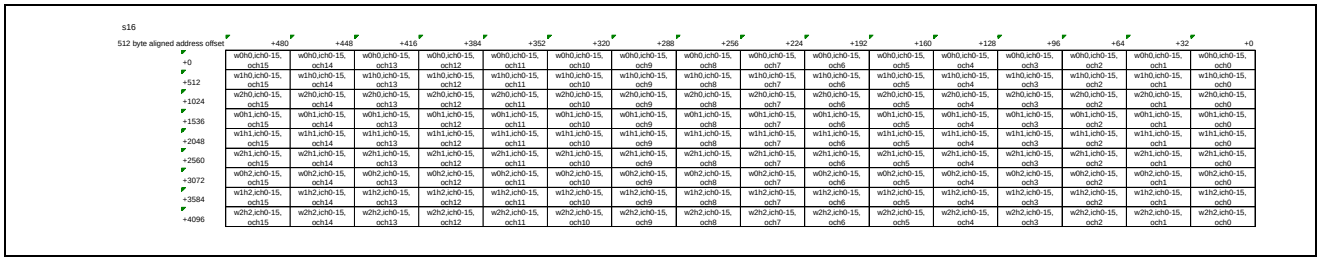



**Figure U33.15 Weight 2*3x3 (s16)**

### (e) 1*5x5

Weight area size is 6.5KB (s8) or 3.5KB (s16) in 1*5x5 kernel mode, weights are stored in raster order. The loop structure within 32byte is like other Kernel modes, with input channels using inner loop followed by output channels and kernel elements. The data order of och within 32byte and ich within 8byte are both LSB first. Weight coefficients of unused channels must be filled with 0 data.

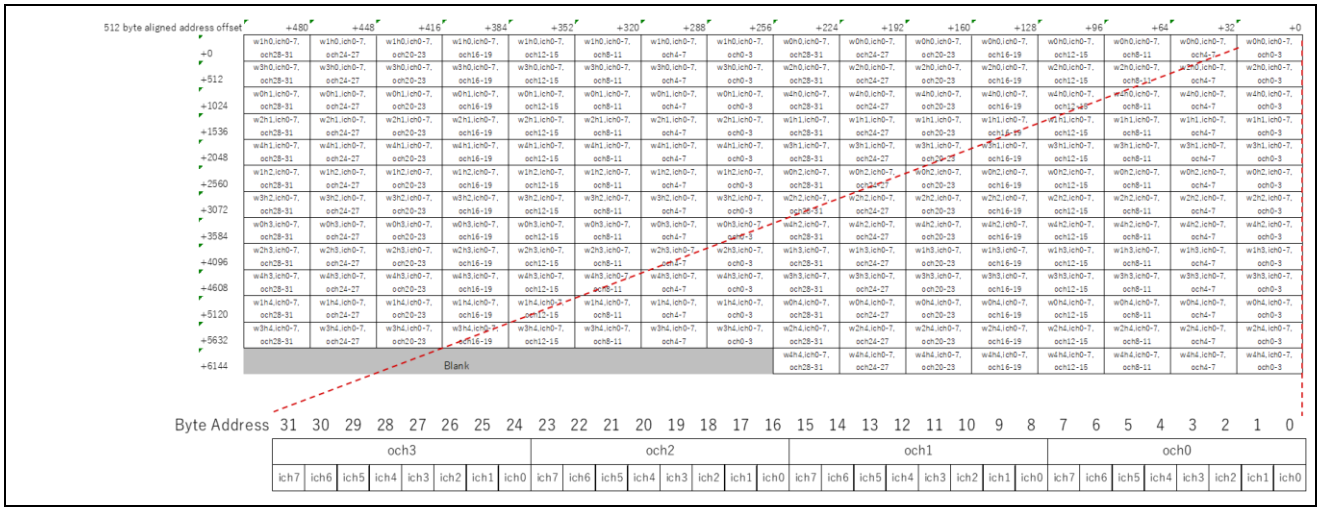



**Figure U33.16 Weight 1*5x5 (s16)**

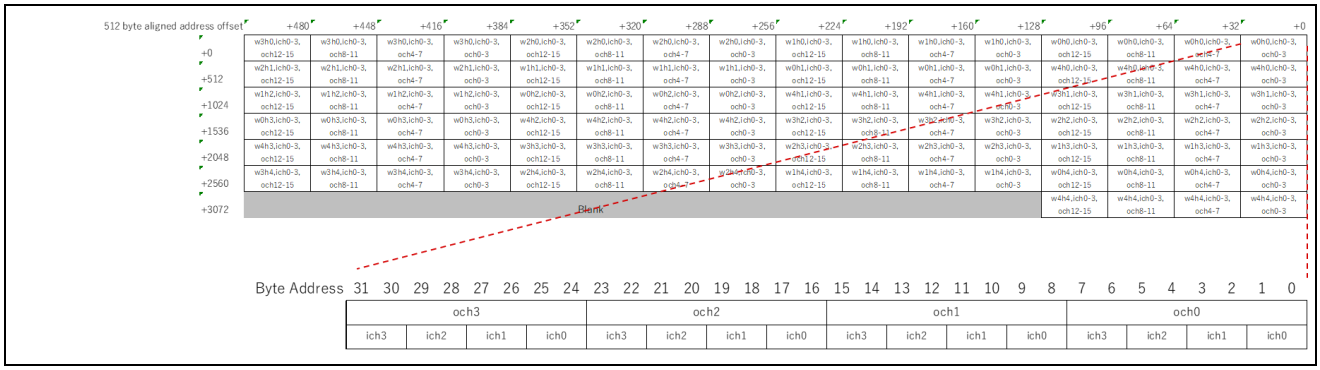



**Figure U33.17 Weight 1*5x5 (s8)**

### (f) Depthwise

Weight area size is 0.5KB (s8 or s16) in 2*3x3 kernel depthwise mode, weights are stored in raster order. The data order of ich within 32byte is LSB first (Little Endian). Weight coefficients of unused channels must be filled with 0 data.

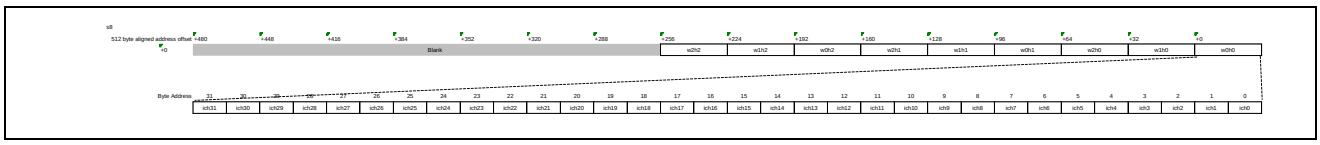


**Figure U33.18 Weight depthwise (s8)**

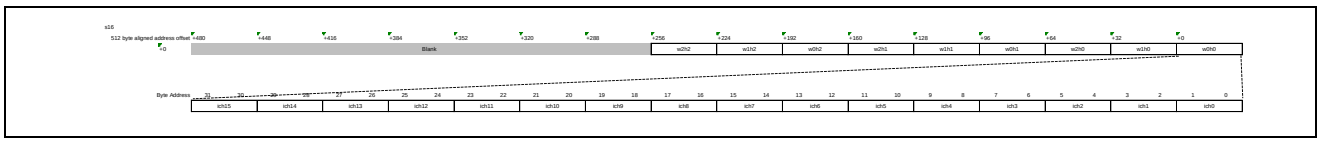


**Figure U33.19 Weight depthwise (s16)**

### U33.3.2 DMAI module

#### (1) DMAI module block diagram

DMAI module loads weights from SPMC0 to transfer them into WT buffer, and loads FM from SPMC0 after finishing weight transmission to transfer them into each line memory.
While loading FM from SPMC0, each required FM element is read from Line Memory and sent to CNV module.

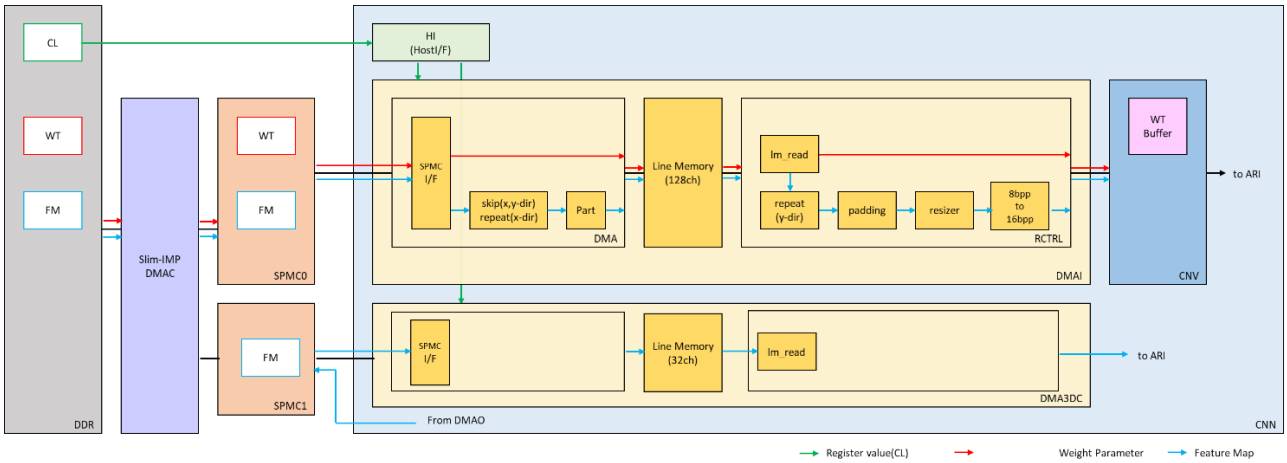



**Figure U33.20 DMAI/DMA3DC block diagram**

**Table U33.8 Block name in DMAI block diagram**

| Block name | Input | Output | Function |
|---|---|---|---|
| DDR | - | CL/FM/WT | Main memory stored CL, FM and WT. |
| Silm-IMP DMAC | FM/WT | FM/WT | DMAC for transfer data between DDR and SPMC0/1. |
| SPMC0 | - | FM/WT | SRAM for FM and WT 2MB.<br>These data are transferred from DDR by Slim-IMP DMAC. |
| SPMC1 | Temp data | Temp data | SRAM to store temp processed data for 3DC 0.5MB. |
| HI | CL | reg. value | Host I/F. Reads CL and decodes register values. |
| DMAI.DMA | FM/WT | FM/WT | Direct Memory Access to SPMC0 for input FM and WT |
| SPMC I/F | FM/WT | FM/WT | SPMC Interface |
| Skip(x,y-dir)/Repeat(x-dir) | FM | FM | Skip and Repeat function |
| Part | FM | FM | Partially selects channels and converts the channel id |
| LineMemory | FM/WT | FM/WT | LineMemory. 128ch.<br>Internal Memory for FM and WT. It is switched for each application. |
| DMAI.RCTRL | FM/WT | FM/WT | Read data controller.<br>Adjusts output image size and controls sub-modules within DMAI. |
| lm_read | FM/WT | FM/WT | LineMemory Reader. |
| Repeat(y-dir) | FM | FM | Repeat FM in Y direction. |
| Padding | FM | FM | Padding the specified value to the area where FM is not available at the edge of the picture. |
| Resizer | FM | FM | Transforms an input FM to the size specified by ARI_LEN register. |
| DMA3DC | FM | FM | DMAC for 3DC data.<br>Transfers 3DC data from SPMC1 to ARI. |
| CNV | FM | - | Convolution engine. |
| WT Buffer | WT | - | Internal buffer for WT in CNV. |
| ARI | - | - | Arithmetic engine. |

### (2) DMAI kernel mode

DMAI has 1*5x5, 2*3x3, 4*1x1, 8*1x1 and 16*1x1 kernel modes. The functions that can be realized for each kernel mode are different (shown in Table U33.9). Also, the combination of each function is restricted as shown in Table U33.10. Each kernel mode is explained in section (a) to (d).

**Table U33.9 DMAI supported function per each kernel mode**

| Function | Direction | 1x1 | 3x3 | 5x5 |
|---|---|---|---|---|
| Convolution Striding | X | | ✓ (1-3) | ✓ (1-3) |
| | Y | | ✓ (1-3) | ✓ (1-3) |
| Dilated Convolution | X | | | ✓ (0-3) |
| | Y | | | |
| Magnification (skip) | X | ✓ (0-1) | | ✓ (0-4) |
| | Y | ✓ (0-1) | | ✓ (0-4) |
| Magnification (repeat) | X | | ✓ (0-4) | ✓ (0-4) |
| | Y | | ✓ (0-4) | ✓ (0-4) |
| Flexible start position | X | | ✓ (0-15) | ✓ (0-15) |
| | Y | | ✓ (0-7)*1 | ✓ (0-7) |
| Padding (Constant) | North / South | | ✓ (0-2) | ✓ (0-7) |
| | West / East | | ✓ (0-2) | ✓ (0-16) |
| Padding (Identity) | North / South | | ✓ (0-2) | ✓ (0-7) |
| | West / East | | ✓ (0-2) | ✓ (0-16) |

✓ Supported: Inside () is range of setting

**Table U33.10 Combination of DMAI supported function**

| | **Convolutional Striding** | **Dilated Convolution** | **Magnification (skip)** | **Magnification (repeat)** | **Flexible start position** | **Padding** |
|---|---|---|---|---|---|---|
| Convolutional striding | | | | | ✓ | ✓ |
| Dilated convolution | | | | | ✓ | ✓ |
| Magnification (skip) | | | | | ✓ | ✓ |
| Magnification (repeat) | | | | | ✓ | ✓ |
| Flexible start position | ✓ | ✓ | ✓ | ✓ | | ✓ |
| Padding | ✓ | ✓ | ✓ | ✓ | ✓ | |

✓ Supported

### (a) Mode 1*5x5

The 1*5x5 mode is the base mode of this architecture. It is selected when not setting any of the special modes in the DMAIS register. In 1*5x5 kernel mode, DMAI transfers up to 8 channels of FM for 8bpp and 4 channels of FM for 16bpp to the CNV module. It is supported in common and individual setup for start address and padding. The method of setup is shown under point (6) of this sub chapter. The 1*5x5 kernel mode is performing one 5 pixels times 5 pixels

convolution per active channel. Smaller convolutions, like 2*3x3 or 4*1x1 also 8*1x1 can be mapped onto the 1*5x5 convolution by setting the weights for the not used convolution elements to zero.
In 1*5x5 kernel mode, DMAI reads WT parameter in the format shown in section U33.3.1(3)(e)(8-ich x 32-och for a total of 256 sets of 5x5 WT parameters for 8bpp, 4-ich x 16-och for a total of 64 sets of 5x5 WT parameters for 16bpp). The correspondence between the 5x5 WT parameters and pixel positions for each ich/och combination is shown in Figure U33.21. Figure U33.21 shows the correspondence between Pixel and WT for output pixel positions of x4-y4, but the same relationship applies to other output pixel positions. 1*5x5 kernel mode supports convolutional striding and dilated Convolution, where the output pixel position and the pixel position of kernel are skipped, respectively. For convolutional striding and dilated convolution, refer to point (15) and (16) of this sub chapter. To use 1*5x5, set B'0000 to DMAIS_TYPE bit.

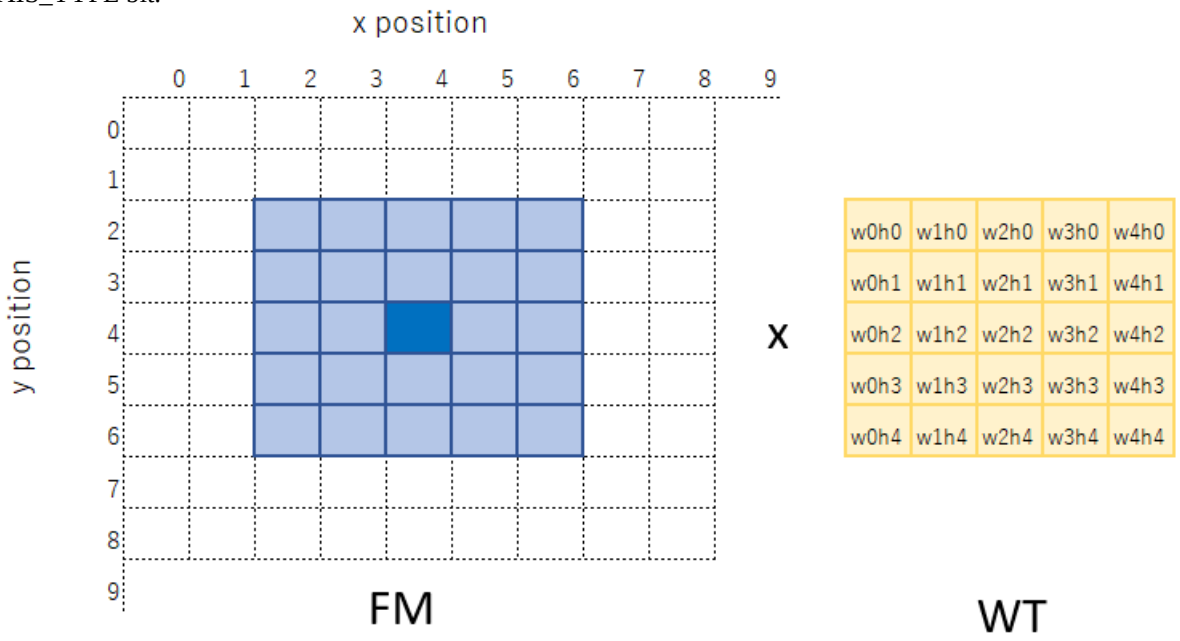



**Figure U33.21 Correspondence between WT and FM for 1*5x5 kernel mode**

### (b) Mode 2*3x3

To speed up the convolution process when performing smaller sized convolutions, like e.g., 3x3 convolution, a 2*3x3 mode has been added. It is selected by setting the DMAIS_TYPE bit of the DMAIS register. In 2*3x3 kernel mode, DMAI transfers up to 32 channels of FM for 8bpp and 16 channels of FM for 16bpp to the CNV module. In 2*3x3 kernel mode, DMAI reads WT parameter in the format shown in section U33.3.1(3)(d) (32-ich x 32-och for a total of 1024 sets of 3x3 WT parameters for 8bpp, 16-ich x 16-och for a total of 256 sets of 3x3 WT parameters for 16bpp). The correspondence between the 3x3 WT parameters and pixel positions for each ich/och combination is shown in Figure U33.22. Figure U33.22 shows the correspondence between Pixel and WT for output pixel positions of x4-y4, but the same relationship applies to other output pixel positions. 2*3x3 kernel mode supports convolutional striding, where the output pixel positions are skipped. For convolutional striding, refer to point (15) of this sub chapter. To use 2*3x3, set B'0100 to DMAIS_TYPE bit.

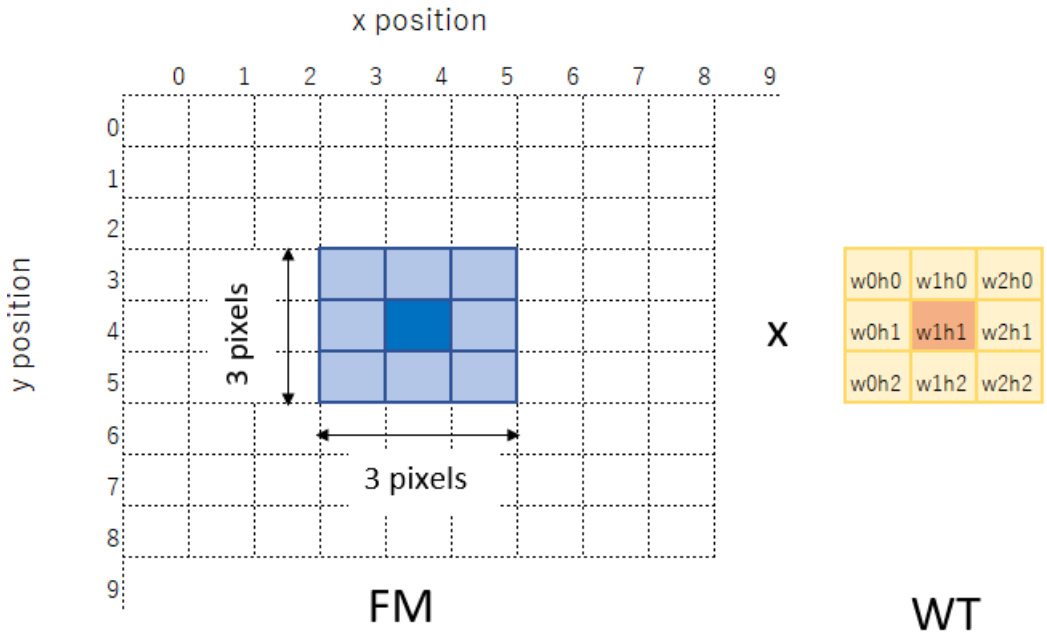



**Figure U33.22 Correspondence between WT and FM for 2*3x3 kernel mode**

### (c) Mode 4*1x1

To speed up the convolution process for the convolution with very small 1x1 convolution matrix, a 4*1x1 mode has been added. It is selected by setting the DMAIS_TYPE bit of the DMAIS register. In 4*1x1 kernel mode, DMAI transfers up to 64 channels of FM for 8bpp and 32 channels of FM for 16bpp to the CNV module. DMAI also has 8*1x1 kernel mode, but 4*1x1 kernel mode has higher line memory tolerance. Therefore, it is better to use 4*1x1 kernel mode when the number of input channels is 64ch or less.
However, as shown under point (d) of this sub chapter, 8*1x1 kernel mode supports up to 128 input channels. Therefore, if the number of input channels is larger than 64, it is better to select 8*1x1 to reduce the number of CNN activation, and the processing performance per input channel will be higher.
In 4*1x1 kernel mode, DMAI reads WT parameter in the format shown in section U33.3.1(3)(c)(64-ich x 32-och for a total of 2048 sets of 1x1 WT parameters for 8bpp, 32-ich x 16-och for a total of 512 sets of 1x1 WT parameters for 16bpp). To use 4*1x1, set B'0010 to DMAIS_TYPE bit.

### (d) Mode 8*1x1

To speed up the convolution process for the convolution with very small 1x1 convolution matrix, a 8*1x1 mode has been added. It is selected by setting the DMAIS_TYPE bit of the DMAIS register. In 8*1x1 kernel mode, DMAI transfers up to 128 channels of FM for 8bpp and 64 channels of FM for 16bpp to the CNV module.
In 8*1x1 kernel mode, DMAI reads WT parameter in the format shown in section U33.3.1(3)(b)(128-ich x 32-och for a total of 4096 sets of 1x1 WT parameters for 8bpp, 64-ich x 16-och for a total of 1024 sets of 1x1 WT parameters for 16bpp). To use 8*1x1, set B'1000 to DMAIS_TYPE bit.

### (e) Mode 16*1x1

To speed up the convolution process for the convolution with very small 1x1 convolution matrix, a 16*1x1 mode has been added. It is selected by setting the DMAIS_TYPE bit of the DMAIS register. In 16*1x1 kernel mode, DMAI transfers up to 256 channels of FM for 8bpp and 128 channels of FM for 16bpp to the CNV module. As shown in Table U33.12, the horizontal size supported by 16*1x1 is half that of other kernel modes. However, MAC activation rate for 16*1x1 is twice that of 8*1x1 and four times that of 4*1x1. Therefore, when the number of input channels is larger than 128, it is more advantageous to use 16*1x1 instead of 8*1x1 in performance.
In 16*1x1 kernel mode, DMAI reads WT parameter in the format shown in section U33.3.1(3)(a) (256-ich x 32-och for a total of 8192 sets of 1x1 WT parameters for 8bpp, 128-ich x 16-och for a total of 2048 sets of 1x1 WT parameters for 16bpp). To use 16*1x1, set B'1100 to DMAIS_TYPE bit. 16*1x1 kernel supports only packed format, set 2'B10 to DMAIS_MODE to use 16*1x1 kernel.

### (3) DMAI Format

DMAI can read three formats as FM. The register settings for selecting each format are as follows.

- Planar type
  - ✧ Set 0 to DMAIS_MODE
- Planar fast type
  - ✧ Set 1 to DMAIS_MODE
- Packed
  - ✧ Set 2 to DMAIS_MODE

As section U33.3.1(2)(a) and U33.3.1(2)(b) show, 1 block channel in packed format consists of 32 channels in 8bpp and 16 channels in 16bpp. Maximum number of input channels for 4*1x1 kernel mode is 64 channels (8bpp) or 32 channels (16bpp), so DMAI reads 2 block channels for 4*1x1 kernel mode. Maximum number of input channels for 8*1x1 kernel mode is 128 channels (8bpp) or 64 channels (16bpp), so DMAI reads 4 block channels for 8*1x1 kernel mode. Maximum number of input channels for 16*1x1 kernel mode is 256 channels (8bpp) or 128 channels (16bpp), so DMAI reads 8 block channels for 16*1x1 kernel mode. Refer to section (5) for how to set start address of each block channel.
Table U33.11 shows the format that each kernel supports.

Table U33.11 Supported format per kernel

| Kernel | Supported format | | |
|---|---|---|---|
| | Planar type | Planar fast type | Packed |
| 1*5x5 | ✓ | ✓ | ✓ |
| 2*3x3 | ✓ | ✓ | ✓ |
| 4*1x1 | ✓ | ✓ | ✓ |
| 8*1x1 | ✓ | ✓ | ✓ |
| 16*1x1 | | | ✓ |

### (4) Picture start address, width, height, and stride

The tile dimension is specified in pixel and rows. The size in x-direction is specified in pixel, set the pixel size in DMAILX. The size in y-direction is specified in rows, set row size in DMAILY. The allowed value for the supported tile dimension depends on the utilized magnification parameters. As result, after magnification operation has been performed, the width must not be larger than 128 pixels while the height must not be larger than 2048 rows. This results for the tile dimension depending on the used magnification parameters shown in Table U33.9 to the allowed value ranges for x and y as shown in Table U33.12 and Table U33.13.

**Table U33.12 Allowed tile dimensions depending on magnification parameter setting (x direction)**

(1) 8*1x1, 4*1x1, 2*3x3 and 1*5x5 case

| DMAIMM | DMAIPMV | Allowed dimension x |
|---|---|---|
| 00 | any | 0 < DMAILX <= 128 |
| 01 | 4 | 0 < DMAILX <= 640 |
| 01 | 3 | 0 < DMAILX <= 512 |
| 01 | 2 | 0 < DMAILX <= 384 |
| 01 | 1 | 0 < DMAILX <= 256 |
| 01 | 0 | 0 < DMAILX <= 128 |
| 10, 11 | 0 | 0 < DMAILX <= 128 |

| | | |
|---|---|---|
| 10, 11 | 1 | 0 < DMAILX <= 64 |
| 10, 11 | 2 | 0 < DMAILX <= 42 |
| 10, 11 | 3 | 0 < DMAILX <= 32 |
| 10, 11 | 4 | 0 < DMAILX <= 25 |

(2) 16*1x1 case

| DMAIMM | DMAIPMV | Allowed dimension x |
|---|---|---|
| 00 | any | 0 < DMAILX <= 64 |
| 01 | 1 | 0 < DMAILX <= 128 |
| 01 | 0 | 0 < DMAILX <= 64 |
| 10, 11 | 0 | 0 < DMAILX <= 64 |

**Table U33.13 Allowed tile dimensions depending on magnification parameter setting (y direction)**

| DMAIMM | DMAIRMV | Allowed dimension y |
|---|---|---|
| 00 | any | 0 < DMAILY <= 2048 |
| 01 | 4 | 0 < DMAILY <= 10240 |
| 01 | 3 | 0 < DMAILY <= 8192 |
| 01 | 2 | 0 < DMAILY <= 6144 |
| 01 | 1 | 0 < DMAILY <= 4096 |
| 01 | 0 | 0 < DMAILY <= 2048 |
| 10, 11 | 0 | 0 < DMAILY <= 2048 |
| 10, 11 | 1 | 0 < DMAILY <= 1024 |
| 10, 11 | 2 | 0 < DMAILY <= 682 |
| 10, 11 | 3 | 0 < DMAILY <= 512 |
| 10, 11 | 4 | 0 < DMAILY <= 409 |

Start address and line stride of the FM in SPMC0 are set by DMAISA and DMAIST, respectively; DMAISA is set in byte unit and DMAIST is set in pixel unit. Packed format and Planar format are shown in section U33.3.1; DMAI can cut out any rectangle of FM stored in SPMC0. The blue rectangles in Figure U33.23 and Figure U33.24 show the FM cutout from SPMC0 in the planar case and the Packed case, respectively. The blue rectangles are the cutout image: 32byte/pix for packed (32-ch pack * 8bpp, 16-ch pack * 16bpp), 1byte/pix for planar-8bpp, and 2byte/pix for planar-16bpp, for line stride and horizontal size x32, x1, and x2, respectively, for the settings. SPMC0 is 2MB of scratchpad memory shared by FM and WT. The value set in line stride register (DMAIST) must be greater than or equal to the value in horizontal size register (DMAILX).

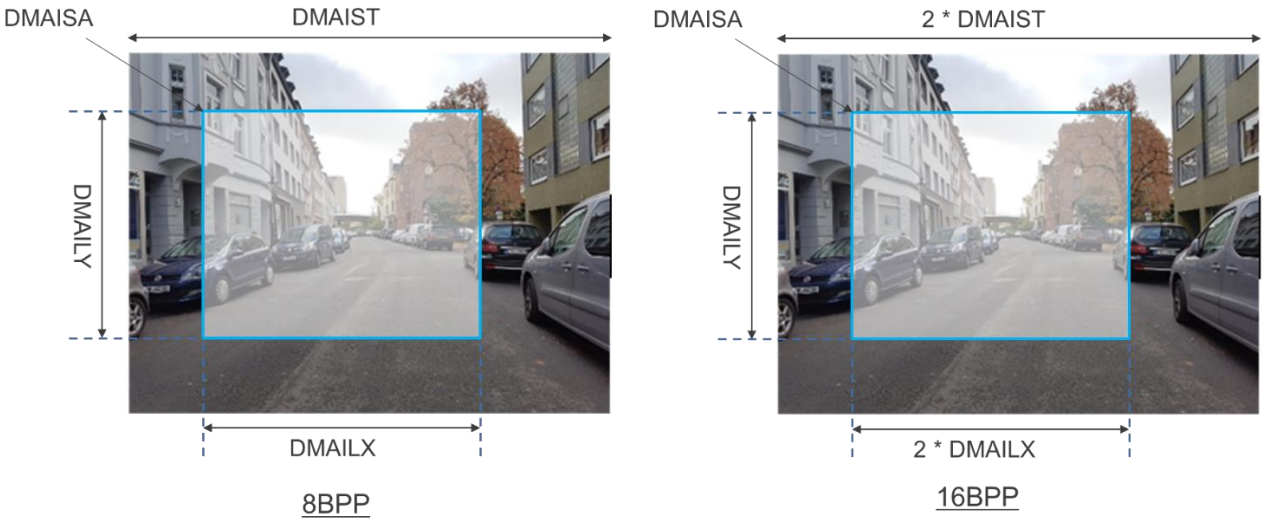



**Figure U33.23 Partial FM read out from SPMC0 (planar)**

**Figure U33.24 Partial FM read out from SPMC0 (packed)**

When planar type or packed is selected, DMAI can cut out FM at any pixel position. When planar fast type is selected, DMAI can cut out FM at any pixel position when Magnification is not used, but when Magnification is used, the cut-out position is restricted to 16byte align. Restriction of DMAISA is shown in Table U33.14.

**Table U33.14 Restriction of DMAI start address**

| Combination of register value | | | Restriction of DMAI start address | Note |
|---|---|---|---|---|
| Format (DMAIS [10:9]) | Magnification (*1) 0: No magnification 1: Magnification | bpp (DMAIPARAM [17:16]) | DMAISA | |
| 0 (planar type) | In any case | 8bpp (0, 1) | None | 1 pixel aligned (8bpp is 1byte/pixel) |
| | In any case | 16bpp (2) | DMAISA [0] = 0 | 1 pixel aligned (16bpp is 2byte/pixel) |
| 1 (planar fast type) | 0 | 8bpp (0, 1) | None | 1 pixel aligned (8bpp is 1byte/pixel) |
| | | 16bpp (2) | DMAISA [0] = 0 | 1 pixel aligned (16bpp is 2byte/pixel) |
| | 1 | In any case | DMAISA [3:0] = 0 | 16byte aligned in case that magnification is used. |
| 2 (packed) | In any case | In any case | DMAISA [4:0] = 0 | 1 pixel aligned (1 pixel is corresponding to 32byte) |

(*1) Magnification
0 for DMAIMM = 0 or (DMAIPMV=0 and DMAIRMV=0)
1 for DMAIMM != 0 and (DMAIPMV!=0 or DMAIRMV!=0)

Restriction of line stride (DMAIST) is shown in Table U33.15.
When planar fast type is selected, DMAIST is restricted to 16 pixels units to satisfy the condition of memory allocation described in section U33.3.1(1)(b). When packed format is selected, as explained in section U33.3.1(2), the format is 16 pixels units in the horizontal direction, and invalid data is stored in such a way that if it is not 16 pixels units, it is rounded up to 16 pixels. Therefore, line stride is restricted to 16 pixels units.

**Table U33.15 Restriction of DMAI line stride**

| Classification of registers value | Restriction of line stride | Note |
|---|---|---|
| DMAIS_MODE (DMAIS [10:9]) | DMAIPARAM [15:0] (DMAIST [15:0]) | |
| 0 (planar type) | None | 1 pixel aligned |
| 1 (planar fast type) | DMAIST [3:0] = 0 | 16 pixels aligned |
| 2 (packed) | DMAIST [3:0] = 0 | 16 pixels aligned |

### (5) Channel offset

Channel offset specifies the difference between the start addresses of 2 neighbored input channels. It is set to the DMAICO register. DMAICO is set by pixel unit. DMAICO is effective except in the 1*5x5 DMAISA individual setup (part (6) of this sub chapter) and in Wrap-direct (part (8) of this sub chapter). DMAICO must be set to 0 when using DMAISA individual setup.

By using this channel offset, it can be made sure that each channel can work on its own data set. Since only 1 offset is used, there exists less flexibility for the start address specification of each single channel than in the individual channel setup case. Further to mention is that inactive channels are still counting for the distance calculation, so, e.g., if byte pixels are processed and channel 0 has as start address H'1_0000 and the offset is specified as H'2_0000, then channel 2 is starting at address H'5_0000, independent of whether channel 1 is active or not.
Restriction of channel offset (DMAICO) is shown in Table U33.16.

**Table U33.16 Restriction of DMAI channel offset**

| Format (DMAIS [10:9]) | Method of DMAISA setup | Restriction of DMAICO | Note |
|---|---|---|---|
| 0 (planar type) | Individual setup | Set 0 to DMAICO | - |
| | Common setup | None | 1 pixel unit |
| 1 (planar fast type) | Common setup | Set 5H'10 to DMAICO [4:0] | 16 pixels unit and bit 4 is 1 |
| 2 (packed) | Common setup | Set 4H'0 to DMAICO [3:0] | 16 pixels unit |

Table U33.17 shows the Start address of each channel using channel offset. The addresses in Table U33.17 are for common DMAISA setup; refer to part (6) of this sub chapter for the start addresses of each channel in individual DMAISA setup.

**Table U33.17 Start address of each channel for common DMAISA setup**

| Format (DMAIS [10:9]) | bpp (DMAFM [1]) | Start Address of each channel (byte unit) |
|---|---|---|
| 0, 1 (planar) | 0 (8bpp) | DMAISA + DMAICO x ch-idx (n=0 to 127) |
| | 1 (16bpp) | DMAISA + 2 * DMAICO x ch-idx (n=0 to 63) |
| 2 (packed) | Any | DMAISA + 32 * DMAICO x blkch-idx (n=0 to 7) |

ch-idx: Channel index for planar indicating channel index before channel conversion by part function (see part (11) of this sub chapter)
blkch-idx: Block channel index for packed

### (6) Common and individual channel setup

The architecture is supporting an individual and common setup. For the common setup, always same parameter is used for all channels, and register setting can be realized at high speed. For the individual setup, each parameter which supports individual setup can be specified for each channel individually, which offers the largest flexibility, but comes with the longest setup time. This individual setup is only supported for the 1*5x5 kernel mode and planar type format. Individual setup is possible only when 1*5x5 planar type is used, and the target registers are DMAISA and PADD registers. The common DMAISA setup register is DMAISA and the individual DMAISA setup registers are DMAISA00, DMAISA01, DMAISA04, DMAISA05, DMAISA08, DMAISA09, DMAISA12 and DMAISA13. Refer to Table U33.25 for the common PADD setup register and individual PADD setup registers. If write-access occurs to the common DMAISA setup register, the same value is overwritten in all individual DMAISA registers; the same applies to the PADD register.
Since the common setup register and the individual setup registers are supposed to be used by exclusive write access, do not write both the common and individual registers during the same frame's register setup period. Also, always use the common setup register except for 1*5x5 and planar type format. If violated, operation is not ensured. The relationship between the start address and register for each channel when using the individual DMAISA setup is shown in Table U33.18. Set 0 to DMAICO in case of individual DMAISA setup.
Individual DMAISA setup registers (DMAISA00,01,04,05,08,09,12,13) have same restriction as common DMAISA setup register (planar type restriction in Table U33.14)

**Table U33.18 Start address of each channel for individual DMAISA setup**

| **Format (DMAIS [10:9])** | **bpp (DMAFM [1])** | **Start Address of each channel (byte unit)** |
|---|---|---|

| | | |
|---|---|---|
| 0 (planar type) | 0 (8bpp) | DMAISA00 (Ch0)<br>DMAISA01 (Ch1)<br>DMAISA04 (Ch2)<br>DMAISA05 (Ch3)<br>DMAISA08 (Ch4)<br>DMAISA09 (Ch5)<br>DMAISA12 (Ch6)<br>DMAISA13 (Ch7) |
| | 1 (16bpp) | DMAISA00 (Ch0)<br>DMAISA04 (Ch1)<br>DMAISA08 (Ch2)<br>DMAISA12 (Ch3) |

Ch id inside (): Indicates channel index before channel conversion by part function (see part (11) of this sub chapter)

### (7) Weight loading

The register settings for the WT format in section U33.3.1(3) is shown in Table U33.19. Start address should be set to DMAWSA; start address is restricted to 512byte align, so the lower 9bit of DMAWSA should be set to 0. When loading weight in depthwise mode (DMAWS.DMAWS_TYPE=B'0001), the CLW command with N field=B'00000 must be executed before starting the transfer by setting DMAWS.DMAWS_START to 1.

**Table U33.19 Format register settings (DMAWS) and WT size**

| Packed format (weight) | | DMAWS [5:1] | | Area Size (Kbyte) |
|---|---|---|---|---|
| Kernel mode | s8/s16 | DMAWS_FM | DMAWS_TYPE[3:0] | |
| 16*1x1 | s8 | 0 | B'1100 | 8 |
| | s16 | 1 | B'1100 | 4 |
| 8*1x1 | s8 | 0 | B'1000 | 4 |
| | s16 | 1 | B'1000 | 2 |
| 4*1x1 | s8 | 0 | B'0010 | 2 |
| | s16 | 1 | B'0010 | 1 |
| 2*3x3 | s8 | 0 | B'0100 | 9 |
| | s16 | 1 | B'0100 | 4.5 |
| 1*5x5 | s8 | 0 | B'0000 | 6.5 |
| | s16 | 1 | B'0000 | 3.5 |
| Depthwise | s8 | 0 | B'0001 | 0.5 |
| | s16 | 1 | B'0001 | 0.5 |

### (8) Wrap around area

DMAI supports Wrap-around. Wrap-around is supported only for the packed type format (DMAIS_MODE=B'10). Wrap-around start and end addresses shall be 512byte aligned.
Wrap-around has two options; Wrap-ico (set by DMAIS_WRAP=1, DMAIS_WRAP_MODE=0) and Wrap-direct (set by DMAIS_WRAP=1, DMAIS_WRAP_MODE=1).
Figure U33.25 shows a conceptual diagram when Wrap-ico is selected, using 8*1x1 mode as an example. Set the start address in DMAISA, Wrap-around start address in DMAIWA_START, and Wrap-around end address in DMAIWA_END for block channel0 (ch0-31 plane). Each address of block channel 1 to 7 is calculated by adding DMAICO to each address of block channel 0. Wrap-around start address and Wrap-around end address are set in 512byte units.
Figure U33.26 shows a conceptual diagram when Wrap-direct is selected, using 8*1x1 mode as an example. When Wrap-direct is selected, the start address, Wrap-around start address, and Wrap-around end address of block channels 0 to 7 can be set individually and directly.
Table U33.20 shows the relationship between the start address, Wrap-around start address, Wrap-around end address and registers settings for each block channel for Wrap-ico and Wrap-direct, respectively.
The relationship between the offset address of FM and the access address of SPMC0 is shown below; for offset, refer to section U33.3.1(2).

- Access Address has not wrap-arounded yet (DMAISA + offset < DMAIWA_END*512)
  - SPMC0 access address: DMAISA + offset
- Access Address has already wrap-arounded (DMAISA + offset >= DMAIWA_END*512)
  - SPMC0 access address: DMAISA + offset - (DMAIWA_END - DMAIWA_START) * 512

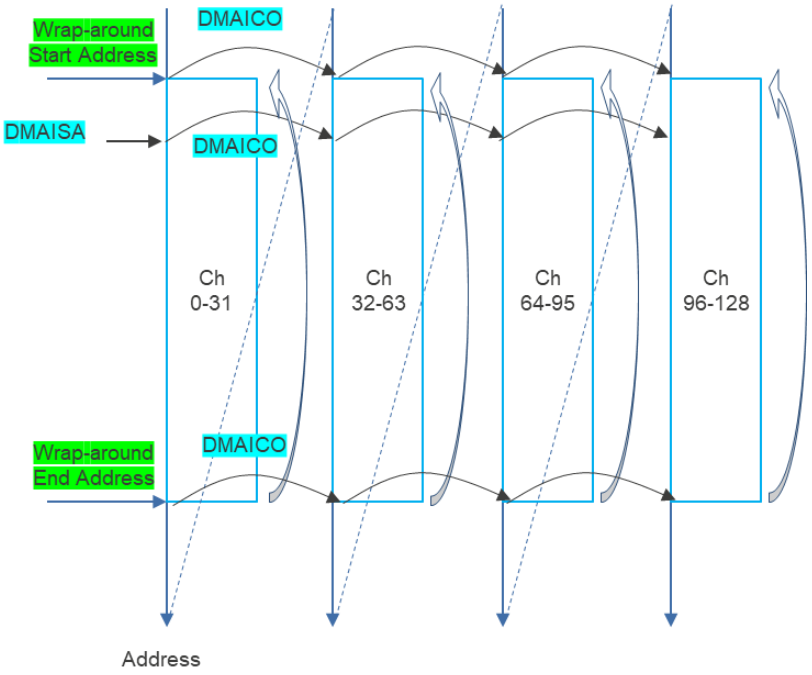



**Figure U33.25 FM wrap around ico (Wrap-ico)**

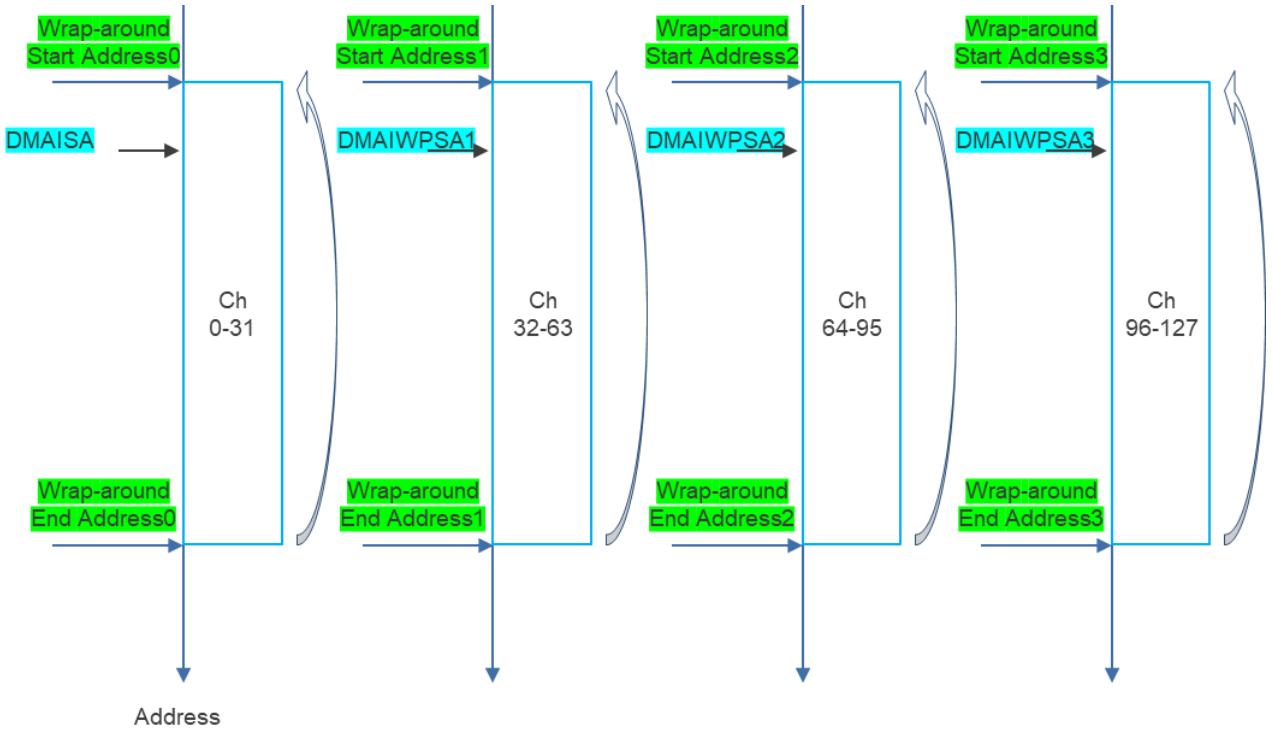



**Figure U33.26 FM wrap around direct (Wrap-direct)**

**Table U33.20 FM Wrap-around address**

| Wrap mode | Block channel ID | Start address | Wrap start address | Wrap end address |
|---|---|---|---|---|
| Wrap-ico | 0 (ch0-31) | DMAISA | DMAIWA_START * 512 | DMAIWA_END * 512 |
| | 1 (ch32-63) | DMAISA + 32 * 1 * DMAICO | DMAIWA_START * 512 + 32 * 1 * DMAICO | DMAIWA_END * 512 + 32 * 1 * DMAICO |
| | 2 (ch64-95) | DMAISA + 32 * 2 * DMAICO | DMAIWA_START * 512 + 32 * 2 * DMAICO | DMAIWA_END * 512 + 32 * 2 * DMAICO |
| | 3 (ch96-127) | DMAISA + 32 * 3 * DMAICO | DMAIWA_START * 512 + 32 * 3 * DMAICO | DMAIWA_END * 512 + 32 * 3 * DMAICO |
| | 4 (ch128-159) | DMAISA + 32 * 4 * DMAICO | DMAIWA_START * 512 + 32 * 4 * DMAICO | DMAIWA_END * 512 + 32 * 4 * DMAICO |
| | 5 (ch160-191) | DMAISA + 32 * 5 * DMAICO | DMAIWA_START * 512 + 32 * 5 * DMAICO | DMAIWA_END * 512 + 32 * 5 * DMAICO |
| | 6 (ch192-223) | DMAISA + 32 * 6 * DMAICO | DMAIWA_START * 512 + 32 * 6 * DMAICO | DMAIWA_END * 512 + 32 * 6 * DMAICO |
| | 7 (ch224-255) | DMAISA + 32 * 7 * DMAICO | DMAWA_START * 512 + 32 * 7 * DMAICO | DMAIWA_END * 512 + 32 * 7 * DMAICO |
| Wrap-direct | 0 (ch0-31) | DMAISA | DMAIWA_START * 512 | DMAIWA_END * 512 |
| | 1 (ch32-63) | DMAIWPSA1 | DMAIWPA1_START * 512 | DMAIWPA1_END * 512 |
| | 2 (ch64-95) | DMAIWPSA2 | DMAIWPA2_START * 512 | DMAIWPA2_END * 512 |
| | 3 (ch96-127) | DMAIWPSA3 | DMAIWPA3_START * 512 | DMAIWPA3_END * 512 |
| | 4 (ch128-159) | DMAIWPSA4 | DMAIWPA4_START * 512 | DMAIWPA4_END * 512 |
| | 5 (ch160-191) | DMAIWPSA5 | DMAIWPA5_START * 512 | DMAIWPA5_END * 512 |
| | 6 (ch192-223) | DMAIWPSA6 | DMAIWPA6_START * 512 | DMAIWPA6_END * 512 |
| | 7 (ch224-255) | DMAIWPSA7 | DMAIWPA7_START * 512 | DMAIWPA7_END * 512 |

The following is a list of restrictions when using Wrap-around.

1. Start address must be set within the Wrap-around range.
   DMAWA_START * 512 <= DMAISA < DMAWA_END * 512 (for Wrap-ico and Wrap-direct)
   DMAIWPA1_START * 512 <= DMAIWPSA1 < DMAIWPA1_END * 512 (for Wrap-direct)
   DMAIWPA2_START * 512 <= DMAIWPSA2 < DMAIWPA2_END * 512 (for Wrap-direct)
   DMAIWPA3_START * 512 <= DMAIWPSA3 < DMAIWPA3_END * 512 (for Wrap-direct)
   DMAIWPA4_START * 512 <= DMAIWPSA4 < DMAIWPA4_END * 512 (for Wrap-direct)
   DMAIWPA5_START * 512 <= DMAIWPSA5 < DMAIWPA5_END * 512 (for Wrap-direct)
   DMAIWPA6_START * 512 <= DMAIWPSA6 < DMAIWPA6_END * 512 (for Wrap-direct)
   DMAIWPA7_START * 512 <= DMAIWPSA7 < DMAIWPA7_END * 512 (for Wrap-direct)

2. The Wrap around area must be same size with FM size or larger
   (DMAIWA_END – DMAIWA_START) * 512 >= 32 * DMAIST * DMALY (for Wrap-ico and Wrap-direct)
   (DMAIWPA1_END – DMAIWPA1_START) * 512 >= 32 * DMAIST * DMAILY (for Wrap-direct)
   (DMAIWPA2_END – DMAIWPA2_START) * 512 >= 32 * DMAIST * DMAILY (for Wrap-direct)
   (DMAIWPA3_END – DMAIWPA3_START) * 512 >= 32 * DMAIST * DMAILY (for Wrap-direct)
   (DMAIWPA4_END – DMAIWPA4_START) * 512 >= 32 * DMAIST * DMAILY (for Wrap-direct)
   (DMAIWPA5_END – DMAIWPA5_START) * 512 >= 32 * DMAIST * DMAILY (for Wrap-direct)
   (DMAIWPA6_END – DMAIWPA6_START) * 512 >= 32 * DMAIST * DMAILY (for Wrap-direct)
   (DMAIWPA7_END – DMAIWPA7_START) * 512 >= 32 * DMAIST * DMAILY (for Wrap-direct)

3. Wrap-direct start address registers (DMAIWPSA1 to DMAIWPSA7) have same restriction as DMAISA register (packed restriction in Table U33.14)

### (9) Magnification (repeat and skip)

The magnification operation is responsible for resizing the picture dimension, allowing a resizing from 1/5 to 5 of the FM width and height. It is set over the DMAIPARAM register.
For the repeat mode, 2 filling options exist as shown in Figure U33.27, the constant filling (DMAIMM=B'10) and identity filling (DMAIMM=B'11) option. The constant filling option adds the constant value specified by the PADD_CONST_VAL register to the newly added pixel. 0-filling is also possible by setting 0 to the PADD_CONST_VAL register.
The identity filling option is repeating the last pixel value. The magnification operation is supported in x and y direction.
For skip mode, rounding up is implemented.

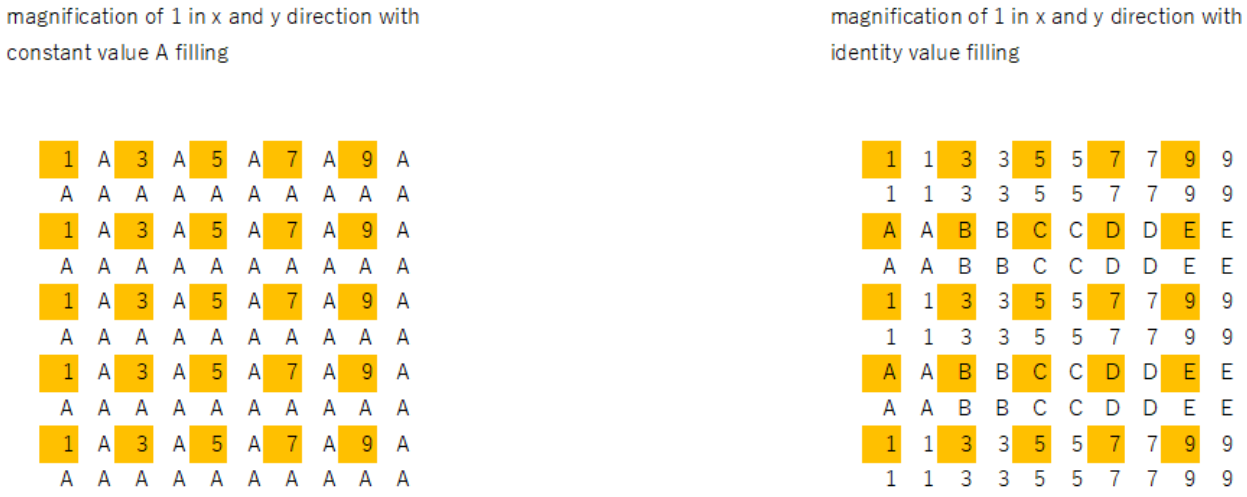



**Figure U33.27 Supported magnification mode examples**

### (10) Channel enable

DMAI channels can be enabled/disabled by the DMAIE register in units of the number of FM channels determined for each kernel mode, and power consumption can be reduced by disabling them.
The relationship between the number of FM input channels and the bits in the DMAIE register is shown in Table U33.21.
The FM input channel shown in the Table U33.21 is corresponding to FM(ich') in Section (11) of this chapter, so it is the channel number after part function conversion.

**Table U33.21 Relation between FM input channel and DMAIE register bits**

| DMAIS Register | ARICSR Register | The FM input channel assigned to the DMAIE register bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMAIS_TYPE[3:0] | DMAIFM bit setting | bit 15 | bit 14 | bit 13 | bit 12 | bit 11 | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| B'0000 (1*5x5) (*1) | s8/u8 | 7 | 6 | 7 | 6 | 5 | 4 | 5 | 4 | 3 | 2 | 3 | 2 | 1 | 0 | 1 | 0 |
| | s16 | - | 3 | - | 3 | - | 2 | - | 2 | - | 1 | - | 1 | - | 0 | - | 0 |
| B'0010 (4*1x1) | s8/u8 | 60~63 | 56~59 | 52~55 | 48~51 | 44~47 | 40~43 | 36~39 | 32~35 | 28~31 | 24~27 | 20~23 | 16~19 | 12~15 | 8~11 | 4~7 | 0~3 |
| | s16 | - | 28~31 | - | 24~27 | - | 20~23 | - | 16~19 | - | 12~15 | - | 8~11 | - | 4~7 | - | 0~3 |
| B'0100 (2*3x3) | s8/u8 | 30~31 | 28~29 | 26~27 | 24~25 | 22~23 | 20~21 | 18~19 | 16~17 | 14~15 | 12~13 | 10~11 | 8~9 | 6~7 | 4~5 | 2~3 | 0~1 |
| | s16 | - | 14~15 | - | 12~13 | - | 10~11 | - | 8~9 | - | 6~7 | - | 4~5 | - | 2~3 | - | 0~1 |
| B'1000 (8*1x1) | s8/u8 | 120~127 | 112~119 | 104~111 | 96~103 | 88~95 | 80~87 | 72~79 | 64~71 | 56~63 | 48~55 | 40~47 | 32~39 | 24~31 | 16~23 | 8~15 | 0~7 |
| | s16 | - | 56~63 | - | 48~55 | - | 40~47 | - | 32~39 | - | 24~31 | - | 16~23 | - | 8~15 | - | 0~7 |
| B'1100 (16*1x1) | s8/u8 | 240~255 | 224~239 | 208~223 | 192~207 | 176~191 | 160~175 | 144~159 | 128~143 | 112~127 | 96~111 | 80~95 | 64~79 | 48~63 | 32~47 | 16~31 | 0~15 |
| | s16 | - | 112~127 | - | 96~111 | - | 80~95 | - | 64~79 | - | 48~63 | - | 32~47 | - | 16~31 | - | 0~15 |

(*1) When DMAI is executed in 1*5x5 kernel mode, bit 0 and bit 2, bit 1 and bit 3, bit 4 and bit 6, bit 5 and bit 7, bit 8 and bit 10, bit 9 and bit 11, bit 12 and bit 14, bit 13 and bit 15 of the DMAIE register must be set to enabled/disabled as a set.

Notes:
1.It is prohibited to run CNN with the DMAIE register set to 16'h0000. At least one channel must be enabled by setting DMAIE register.
2.When the DMAI format conversion function is enabled and FM is converted from u8/s8 to s16, DMAIE should be specified according to the converted format (s16). Therefore, only even bits of DMAIE are available, and odd bits should be set to 0. For details on the DMAI format conversion function, see (12).

### (11) Channel Part

The Part function converts ch-id of FM input channel in the SPMC. Ch-id of the FM input channel (indicated by FM(ich) in Figure U33.28) in the SPMC0 is converted for CNV module by the Part function. (FM input channel after Part function is indicated by FM(ich') in Figure U33.28). The conversion formula is shown in Table U33.22. The coefficients with each ich index number in WT-Format (shown in section U33.3.1(3)) correspond to FM input channels with the same ich index after ch-id conversion by Part function (indicated by FM(ich') in Figure U33.28). In the case of enabling 8bpp to 16bpp conversion, FM format is in 8bpp format and the WT format is in 16bpp format.
For packed format, channel index of FM(ich) corresponds to index number with symbol ch in the FM format in section U33.3.1(2). In case of 4*1x1, 8*1x1, 16*1x1 kernel, there are multiple block channels and the relationship between ch-id and each block channel is as follows.

- 8bpp
  - FM(ich0)-FM(ich31) : ch0-31 in blk-ch0
  - FM(ich32)- FM(ich63) : ch0-31 in blk-ch1
  - FM(ich64)- FM(ich95) : ch0-31 in blk-ch2
  - FM(ich96)- FM(ich127) : ch0-31 in blk-ch3
  - FM(ich128)- FM(ich155) : ch0-31 in blk-ch4
  - FM(ich160)- FM(ich191) : ch0-31 in blk-ch5
  - FM(ich192)- FM(ich223) : ch0-31 in blk-ch6
  - FM(ich224)- FM(ich255) : ch0-31 in blk-ch7
- 16bpp
  - FM(ich0)- FM(ich15) : ch0-15 in blk-ch0
  - FM(ich16)- FM(ich31) : ch0-15 in blk-ch1
  - FM(ich32)- FM(ich47) : ch0-15 in blk-ch2
  - FM(ich48)- FM(ich63) : ch0-15 in blk-ch3

- FM(ich64)- FM(ich79) : ch0-15 in blk-ch4
- FM(ich80)- FM(ich95) : ch0-15 in blk-ch5
- FM(ich96)- FM(ich111) : ch0-15 in blk-ch6
- FM(ich112)- FM(ich127) : ch0-15 in blk-ch7

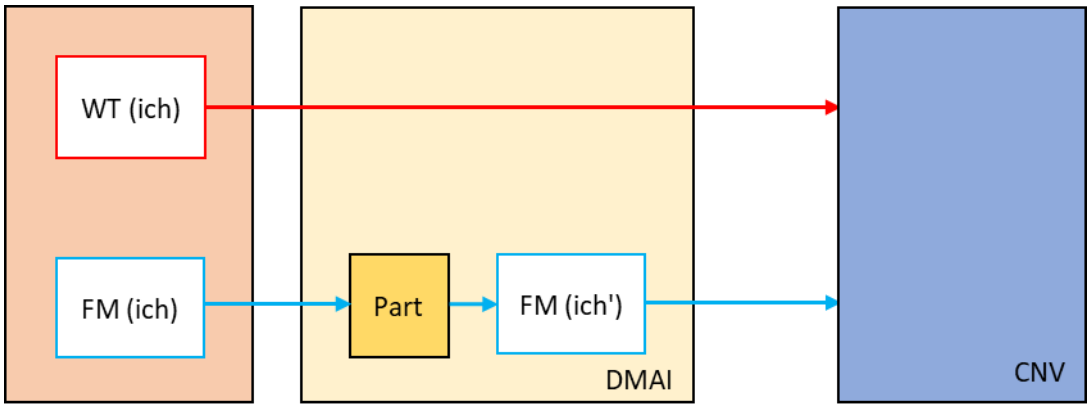



**Figure U33.28 Part function**

**Table U33.22 Part function ch-id conversion formula**

| Kernel mode | Input bpp (DMAIFM [1]) | DMAIS_ 8TO16BIT_ CONV | Output bpp | DMAIS_ PART | Pack | Planar (*2) | Conversion from ch to ich' by Part function |
|---|---|---|---|---|---|---|---|
| 1*5x5 | 8bpp (0) | 0 | 8bpp | 0 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 7) |
| 1*5x5 | 8bpp (0) | 0 | 8bpp | 1 | ✓ | | ich'{n} = ich{n+8} (n= 0 to 7) |
| 1*5x5 | 8bpp (0) | 0 | 8bpp | 2 | ✓ | | ich'{n} = ich{n+16} (n= 0 to 7) |
| 1*5x5 | 8bpp (0) | 0 | 8bpp | 3 | ✓ | | ich'{n} = ich{n+24} (n= 0 to 7) |
| 1*5x5 | 16bpp (1) | 0 | 16bpp | 0 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 3) |
| 1*5x5 | 16bpp (1) | 0 | 16bpp | 1 | ✓ | | ich'{n} = ich{n+4} (n= 0 to 3) |
| 1*5x5 | 16bpp (1) | 0 | 16bpp | 2 | ✓ | | ich'{n} = ich{n+8} (n= 0 to 3) |
| 1*5x5 | 16bpp (1) | 0 | 16bpp | 3 | ✓ | | ich'{n} = ich{n+12} (n= 0 to 3) |
| 1*5x5 | 8bpp (0) | 1 | 16bpp | 4 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 3) |
| 1*5x5 | 8bpp (0) | 1 | 16bpp | 5 | ✓ | | ich'{n} = ich{n+4} (n= 0 to 3) |
| 1*5x5 | 8bpp (0) | 1 | 16bpp | 6 | ✓ | | ich'{n} = ich{n+8} (n= 0 to 3) |
| 1*5x5 | 8bpp (0) | 1 | 16bpp | 7 | ✓ | | ich'{n}= ich{n+12} (n= 0 to 3) |
| 1*5x5 | 8bpp (0) | 1 | 16bpp | 8 | ✓ | | ich'{n} = ich{n+16} (n= 0 to 3) |
| 1*5x5 | 8bpp (0) | 1 | 16bpp | 9 | ✓ | | ich'{n} = ich{n+20} (n= 0 to 3) |
| 1*5x5 | 8bpp (0) | 1 | 16bpp | 10 | ✓ | | ich'{n} = ich{n+24} (n= 0 to 3) |
| 1*5x5 | 8bpp (0) | 1 | 16bpp | 11 | ✓ | | ich'{n} = ich{n+28} (n= 0 to 3) |
| 2*3x3 | 8bpp (0) | 0 | 8bpp | 0 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 31) |
| 2*3x3 | 16bpp (1) | 0 | 16bpp | 0 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 15) |
| 2*3x3 | 8bpp (0) | 0 | 8bpp | 2 | ✓ | ✓ | ich'{2*n} = ich{n} (n= 0 to 15)(*1) |
| 2*3x3 | 8bpp (0) | 0 | 8bpp | 3 | ✓ | | ich'{2*n} = ich{n+16} (n= 0 to 15)(*1) |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2*3x3 | 8bpp (0) | 1 | 16bpp | 2 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 15) |
| 2*3x3 | 8bpp (0) | 1 | 16bpp | 3 | ✓ | | ich'{n} = ich{n+16} (n= 0 to 15) |
| 2*3x3 | 16bpp (1) | 0 | 16bpp | 2 | ✓ | ✓ | ich'{2*n} = ich{n} (n= 0 to 7)(*1) |
| 2*3x3 | 16bpp (1) | 0 | 16bpp | 3 | ✓ | | ich'{2*n} = ich{n+8} (n= 0 to 7)(*1) |
| 2*3x3 | 8bpp (0) | 0 | 8bpp | 4 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 7) |
| 2*3x3 | 16bpp (1) | 0 | 16bpp | 4 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 3) |
| 2*3x3 | 8bpp (0) | 0 | 8bpp | 5 | ✓ | | ich'{n} = ich{n+8} (n= 0 to 7) |
| 2*3x3 | 16bpp (1) | 0 | 16bpp | 5 | ✓ | | ich'{n} = ich{n+4} (n= 0 to 3) |
| 2*3x3 | 8bpp (0) | 0 | 8bpp | 6 | ✓ | | ich'{n} = ich{n+16} (n= 0 to 7) |
| 2*3x3 | 16bpp (1) | 0 | 16bpp | 6 | ✓ | | ich'{n} = ich{n+8} (n= 0 to 3) |
| 2*3x3 | 8bpp (0) | 0 | 8bpp | 7 | ✓ | | ich'{n} = ich{n+24} (n= 0 to 7) |
| 2*3x3 | 16bpp (1) | 0 | 16bpp | 7 | ✓ | | ich'{n} = ich{n+12} (n= 0 to 3) |
| 4*1x1 | 8bpp (0) | 0 | 8bpp | 0 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 63) |
| 4*1x1 | 16bpp (1) | 0 | 16bpp | 0 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 31) |
| 4*1x1 | 8bpp (0) | 1 | 16bpp | 2 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 31) |
| 8*1x1 | 8bpp (0) | 0 | 8bpp | 0 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 127) |
| 8*1x1 | 16bpp (1) | 0 | 16bpp | 0 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 63) |
| 8*1x1 | 8bpp (0) | 1 | 16bpp | 2 | ✓ | ✓ | ich'{n} = ich{n} (n= 0 to 63) |
| 16*1x1 | 8bpp (0) | 0 | 8bpp | 0 | ✓ | | ich'{n} = ich{n} (n= 0 to 255) |
| 16*1x1 | 16bpp (1) | 0 | 16bpp | 0 | ✓ | | ich'{n} = ich{n} (n= 0 to 127) |

(*1) Set 0 in WT coefficient of missing ich.
Ex: In case of 2x3x3 && DMIAS_PART=2 && DMAIFM [1] && DMAIS_8TO16BIT_CONV=0, odd ich (ich1, ich3, ich5, ich7, ... ich31) is unused, therefore, set 0 in the WT coefficient of those ich
(*2) Planar here indicates planar type or planar fast type

### (12) DMAI Format conversion

CNV 16bit convolution can be performed by converting the u8/s8 FM to s16 FM. If user want to do this operation, please set DMAIS. DMAIS_8TO16BIT_CONV as shown in Table U33.23.

**Table U33.23 Register setting for DMAI format conversion**

| Format conversion<br>(FM format to ARI format) | DMAIPARAM.DMAIFM | DMAIS.<br>DMAIS_8TO16BIT_CONV | ARICSR.DMAIFM |
|---|---|---|---|
| s8 to s16 | B00 | B1 | B10 |
| u8 to s16 | B01 | B1 | B10 |
| s16 to s16(NOT convert) | B10 | B0 | B10 |

Setting DMAIS.DMAIS_8TO16BIT_CONV to 1'b1 changes the format of PADD_CONST_VAL by changing DMAIPARAM.DMAIFM as follows.

In case of DMAIPARAM.DMAIFM=2'b00, PADD_CONST_VAL is treated as signed 8bit, and the sign is extended to treat it as 16bit data. On the other hand, DMAIPARAM.DMAIFM=2'b01, PADD_CONST_VAL is treated as unsigned 8bit and zero is extended to treat it as 16bit data.

### (13) Offset address

Offset addresses to FM and weight parameter start addresses can be set using the DMAIOA and DMAWOA registers.

To quickly change between input data sets of consecutively processed input channels of the same layer with the same dimensions, by specifying an offset address, the next start address is calculated automatically by adding this offset address to the current start address, which is afterwards replaced by the newly calculated start address.
When writing to the DMAIOA register, both DMAISA and DMAISAxx are updated. When writing to the DMAWOA register, DMAWSA is updated. The DMAIOA and DMAWOA registers are write-only registers, and updates by these registers can be obtained by reading the corresponding starting address registers.

For example, if 0x80000 is set to DMAISA and then 0x1000 is written to DMAIOA, DMAISA becomes 0x81000.

### (14) Padding

The padding parameters define an extra area around the loaded tile. Supported padding function is shown in Table U33.24.

If choosing common setup, the same value is used for all input channels defined by the PADDN, PADDS, PADDW and PADDE bits of the PADD_STRIDE_STARTPOS register, while if choosing individual setup, each channel can be setup with a different value using the PADDxxN, PADDxxS, PADDxxW and PADDxxE bits of the PADDxx_yy registers.

There exist 2 different padding modes, called the constant padding and identity padding as shown in Figure U33.29.

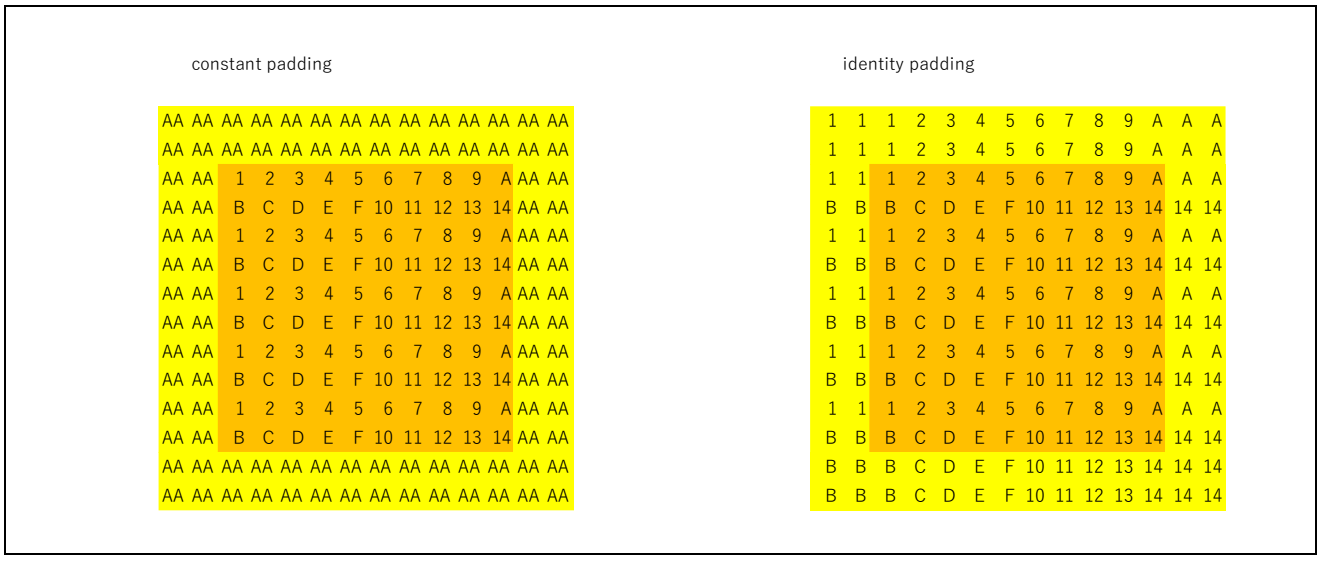



**Figure U33.29 Supported padding mode examples**

Figure U33.29 is showing an example padding of 2 pixels in each direction. On left side, the constant padding is shown, which is adding constant value set in the PADD_CONST_VAL register for the padded pixels. 0-padding is possible by setting PADD_CONST_VAL register to 0. On the right side, the identity padding is shown, which is duplicating the nearest corner pixel for each padded pixel.

The padding on the left is called west padding, the right is east padding, the top is named north padding, and the below is south padding. The number of Padding pixels for each padding is set by the registers shown in Table U33.25**.**

**Table U33.24 Supported padding function**

| Padding Mode | Direction | | The number of pixels | | |
|---|---|---|---|---|---|
| | | | Kernel size | | |
| | | | 1x1 | 3x3 | 5x5 |
| Constant | X(Hrizontal) | West | Not supported | 0 to 2 | 0to 16 |

| | | | | | |
|---|---|---|---|---|---|
| | | East | Not supported | 0 to 2 | 0to 16 |
| | Y(Vertical) | North | Not supported | 0 to 2 | 0 to 7 |
| | | South | Not supported | 0 to 2 | 0 to 7 |
| Identity | X(Hrizontal) | West | Not supported | 0 to 2 | 0to 16 |
| | | East | Not supported | 0 to 2 | 0to 16 |
| | Y(Vertical) | North | Not supported | 0 to 2 | 0 to 7 |
| | | South | Not supported | 0 to 2 | 0 to 7 |

**Table U33.25 Register for padding setting**

| | | |
|---|---|---|
| Common setup | North | PADD_STRIDE_STARTPOS.PADDN |
| | South | PADD_STRIDE_STARTPOS.PADDS |
| | West | PADD_STRIDE_STARTPOS.PADDW |
| | East | PADD_STRIDE_STARTPOS.PADDE |
| Individual setup<br>(Only usable for planar type and 1*5x5 kernel mode) | North | PADD00_01.PADD00N<br>PADD00_01.PADD01N<br>PADD04_05.PADD04N<br>PADD04_05.PADD05N<br>PADD08_09.PADD08N<br>PADD08_09.PADD09N<br>PADD12_13.PADD12N<br>PADD12_13.PADD13N |
| | South | PADD00_01.PADD00S<br>PADD00_01.PADD01S<br>PADD04_05.PADD04S<br>PADD04_05.PADD05S<br>PADD08_09.PADD08S<br>PADD08_09.PADD09S<br>PADD12_13.PADD12S<br>PADD12_13.PADD13S |
| | West | PADD00_01.PADD00W<br>PADD00_01.PADD01W<br>PADD04_05.PADD04W<br>PADD04_05.PADD05W<br>PADD08_09.PADD08W<br>PADD08_09.PADD09W<br>PADD12_13.PADD12W<br>PADD12_13.PADD13W |
| | East | PADD00_01.PADD00E<br>PADD00_01.PADD01E<br>PADD04_05.PADD04E<br>PADD04_05.PADD05E<br>PADD08_09.PADD08E<br>PADD08_09.PADD09E<br>PADD12_13.PADD12E<br>PADD12_13.PADD13E |

### (15) Convolutional striding

The convolutional striding defines the pixel movement steps in x and y direction of the convolution operation. If the convolutional striding parameter is set in x and y direction to 1, e.g., which is the default value, for every pixel a convolution operation is calculated.

By increasing the parameter in x/y direction, convolution operations for pixels can be skipped. If the convolutional striding parameter is set in x and y direction to 2, e.g., every other column/row is processed. Allowed values for the convolutional striding parameter are 1 till 3 for x and y direction. Convolutional striding is not supported at the same time with dilated convolution or magnification.Supported convolutional striding function is shown in Table U33.26.

The same value is used for all input channels defined by STRX bits for x-direction and STRY bits for y-direction of the PADD_STRIDE_STARTPOS register. The register setting is shown in Table U33.27.

**Table U33.26 Convolutional striding function**

| Direction | Convolutional striding parameter | | |
|---|---|---|---|
| | Kernel size | | |
| | 1x1 | 3x3 | 5x5 |
| X (Horizontal) | 1 | 1 to 3 | 1 to 3 |
| Y (Vertical) | 1 | 1 to 3 | 1 to 3 |

**Table U33.27 Register for convolutional stride setting**

| Direction | Register Name |
|---|---|
| X | PADD_STRIDE_STARTPOS.STRX |
| Y | PADD_STRIDE_STARTPOS.STRY |

### (16) Dilated convolution

The dilated convolution parameter DIL_CONV is defining the number of skipped pixels in x direction between 2 taken pixels for the calculation of a convolution matrix.

If the dilated convolution setting in x direction is 1, e.g., then for a 5x5 matrix which needs 5 pixels in x direction, instead of the pixels 2,3,4,5,6 (Figure U33.30 (1)) the pixels 0,2,4,6,8 are taken when the center of convolution matrix is 4. (Figure U33.30 (2))

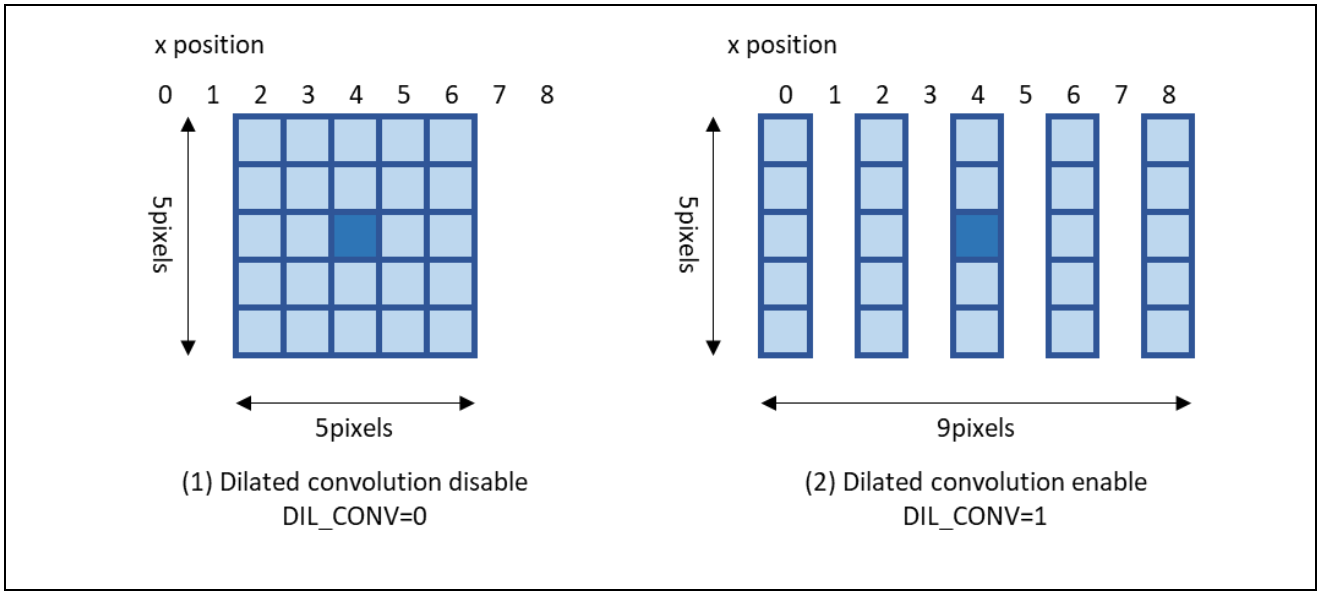



**Figure U33.30 Dilated convolution setting examples**

Allowed values for the dilated convolution parameter are 0 to 3 for x direction. A dilated convolution setting for y direction is not supported. Dilated convolution is not supported at the same time with convolutional striding or magnification. The supported dilated convolution function is shown in Table U33.28.

The same value is always used for all input channels defined by DIL_CONV bits of PADD_STRIDE_STARTPOS register.

**Table U33.28 Supported dilated convolution**

| Direction | Dilated convolution parameter | | |
|---|---|---|---|
| | Kernel size | | |
| | 1x1 | 3x3 | 5x5 |
| X(Horizontal) | Not supported | Not supported | 0 to 3 |
| Y(Vertical) | Not supported | Not supported | Not supported |

**Table U33.29 Register for dilated convolution setting**

| Direction | Register Name |
|---|---|
| X | PADD_STRIDE_STARTPOS.DIL_CONV |

### (17) Start position

The start position defined by START_POS_X and START_POS_Y defines the setting of the start position in x and y direction, from where values are read out and given to the convolution unit.
The supported start position function values are shown in Table U33.30.
The same value defined by START_POS_X bits for x-direction and START_POS_Y bits for y-direction of the PADD_STRIDE_STARTPOS register is used for all input channels.

Please note the following restriction:
CEILING( DMAIL.DMAILY * Magnification(1, 2, 3, 4, 5, 1/2, 1/3, 1/4, 1/5) ) >= PADD_STRIDE_STARTPOS.START_POS_Y+1

**Table U33.30 Supported start address**

| Direction | Start position | | |
|---|---|---|---|
| | Kernel size | | |
| | 1x1 | 3x3 | 5x5 |
| X(Horizontal) | Not supported | 0 to 15 | 0 to 15 |
| Y(Vertical) | Not supported | 0 to 7 | 0 to 7 |

**Table U33.31 Register for start position setting**

| Direction | Register Name |
|---|---|
| X | PADD_STRIDE_STARTPOS. START_POS_X |
| Y | PADD_STRIDE_STARTPOS. START_POS_Y |

### U33.3.3 DMA3DC module

#### (1) DMA3DC module block diagram

DMA3DC reads the intermediate convolution result data (up to 32 channels) of convolution by the CNV module and supplies it to the ARI module. Intermediate data is stored in SPMC1 (0.5Mbyte). DMA3DC block diagram is shown in Figure U33.20. The data formats of intermediate result data are the four types shown in section U33.3.1(2).

#### (2) Picture start address, width, height, and stride

The tile dimension is specified in pixel and rows. Pixel is the size in x-direction, set the pixel size in DMA3DCLX (up to 128 pixels). Row is the size in y-direction, set row size in DMA3DCLY (up to 1024 lines).
Start address and line stride of the FM in SPMC1 are set by DMA3DCSA and DMA3DCST, respectively; DMA3DCSA is set in byte unit and DMA3DCST is set in pixel unit. SPMC1 is 0.5MB of scratchpad memory. DMA3DCSA must be set 512byte aligned. DMA3DCST must be set in 16 pixels units for s8 and s16 format, 8 pixels units for m16 format, and 4 pixels units for m32 format. As can be understood in the light of chapter U33.3.1(2), the stride setting is aligned to 512 byte. The value set in line stride register (DMA3DCST) must be 0 or greater than or equal to the value in horizontal size register (DMA3DCLX).

#### (3) Offset address

Offset address to 3DC data start address can be set using the DMA3DCOA register.

To quickly change between input data sets of consecutively processed 3DC channels of the same layer with the same dimensions, by specifying an offset address, the next start address is calculated automatically by adding this offset address to the current start address, which is afterwards replaced by the newly calculated start address.
When writing to the DMA3DCOA register, DMA3DCSA is updated.

The DMA3DCOA register is write-only registers, and updates by this register can be obtained by reading the DMA3DCSA register.

For example, if 0x80000 is set to DMA3DCSA and then 0x1000 is written to DMA3DCOA, DMA3DCSA becomes 0x81000.

### (4) Channel enable

DMA3DC channels can be enabled/disabled by the DMA3DCE register, and power consumption can be reduced by disabling it.

The relationship between the number of 3DC data channels and the bits in the DMA3DCE register is shown in Table U33.32.

**Table U33.32 Relationship of DMA3DC channel to DMA3DCE register bit depending on selected data type**

| DMA3DCPA RAM Register DMA3DCFM bit setting | The FM input channel assigned to the DMA3DCE register bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bit 15 | bit 14 | bit 13 | bit 12 | bit 11 | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| s8 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| s16 | - | 7 | - | 6 | - | 5 | - | 4 | - | 3 | - | 2 | - | 1 | - | 0 |
| m16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| m32 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| DMA3DCPA RAM Register DMA3DCFM bit setting | The FM input channel assigned to the DMA3DCE register bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bit 31 | bit 30 | bit 29 | bit 28 | bit 27 | bit 26 | bit 25 | bit 24 | bit 23 | bit 22 | bit 21 | bit 20 | bit 19 | bit 18 | bit 17 | bit 16 |
| s8 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| s16 | - | 15 | - | 14 | - | 13 | - | 12 | - | 11 | - | 10 | - | 9 | - | 8 |
| m16 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| m32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

When DMA3DC is not used (DMA3DCS is not set), the DMA3DCE register must be set to 32'h0000_0000.
When DMA3DC is used, at least one bit from bit0 to bit7 of the DMA3DCE register must be set.
If a bit in the ARIE register is set to 0, the corresponding bit in the DMA3DCE register must also be set to 0 (exception: ELTWISE MULT 16bit).

### (5) Value shift

A further operation on the 3DC data is the data value shifter which is supporting a shift of values [0..31], support of 32bit (8bpp, m16, m32) and 41bit (16bpp) for shifted output value.

Either each SFTCxx register can be defined with a different value or, when all values are same, the SFTC address can be used to write the same value to all SFTCxx registers.

Here is to mention that an overflow check is not performed by hardware, so the user is responsible for any data loss occurring during 3DC data shift.

#### (6) 3DC input usage

The 3DC data for 3DC addition and Eltwise Mult processing can be specified by the register ARICSR.3DC_INPUT_USAGE. The setting for ARICSR.3DC_INPUT_USAGE depends on the value of register ARICSR. MIXED_MODE and ARICSR. MIXED_MODE2.

When the value of ARICSR. MIXED_MODE is 0 and ARICSR. MIXED_MODE2 is 0, for 32 channels addition, user can specify 32 channels input with 3DC format 8bpp (s8), 16bpp (m16), 32bpp (m32). While for 16 channels even addition, user can specify 16 channels input with 3DC format 16bpp (s16) or specify the first/last 16 channels input with 3DC format 16bpp (m16), 32bpp (m32).

When the value of ARICSR. MIXED_MODE is 0 and ARICSR. MIXED_MODE2 is 1, user can specify 32 channels input with 3DC format 16bpp (m16) for 32 channels Eltwise Mult processing. While for 16 channels even Eltwise Mult processing, user can specify the first/last 16 channels input with 3DC format 16bpp(m16).

For other combinations of ARICSR. MIXED_MODE and ARICSR. MIXED_MODE2, user should use the default value ARICSR.3DC_INPUT_USAGE is 0 to specify 32 channels input with 3DC format 8bpp (s8), 16bpp (m16), 32bpp (m32) for 32 channels addition.

### U33.3.4 CNV module

#### (1) 8bit mode

For 8bit FM modes, signed convolution is supported. The input data for the convolution is either interpreted as 8bit signed values with a range from -128 to 127 or as 8bit unsigned values with the range from 0 to 255, while the weight data is always interpreted as 8bit signed values with a range from -128 to 127.

If you set 0x0 or 0x1 to DMAIPARAM.DMAIFM register, 8bit modes are enabled. In the same way, DMA3DC and ARI formats can be set in DMA3DCPARAM.DMA3DCFM, ARICSR.ARIFM and ARICSR.DMA3DCFM registers. (Please refer to register list for each value setting)

While always 32 output channels can be processed at the same time, the number of simultaneously processable FM input channels has the limitation as follows:
- -1*5x5 kernel => up to 8 FM input channels.
- -2*3x3 kernel => up to 32 FM input channels.
- -4*1x1 kernel => up to 64 FM input channels.
- -8*1x1 kernel => up to 128 FM input channels.
- -16*1x1 kernel => up to 256 FM input channels.

When processing 1x5x5 kernel, set 1 to the ARICSR.CONV_MODE register.

#### (2) 16bit mode

16bit mode can be set using the same registers as 8bit mode.

In 16bit mode, 16bit signed convolution is supported. For 16bit signed convolution, the input data for the convolution as well as the weight data is interpreted as 16bit signed values with a range from -32768 to 32767.

While always 16 output channels can be processed at the same time, the number of simultaneously processable FM input channels has the limitation as follows:
- -1*5x5 kernel => up to 4 FM input channels.
- -2*3x3 kernel => up to 16 FM input channels.
- -4*1x1 kernel => up to 32 FM input channels. s
- -8*1x1 kernel => up to 64 FM input channels.
- -16*1x1 kernel => up to 128 FM input channels.

When processing 1x5x5 kernel, set 1 to the ARICSR.CONV_MODE register.

### (3) Supported FM and weight data type combination

The combinations of FM and weight data type supported by CNV are listed in Table U33.33.

**Table U33.33 Supported combinations of FM and WT data type**

| | | Data type of weight data (DMAWS.DMAWS_FM and ARICSR, ARIFM register setting (*2)) | | |
|---|---|---|---|---|
| | | s8 | s16 | |
| **Data type of FM** (DMAIPARAM, DMAIFM register setting) | u8 | ✓ | ✓ | (*1) |
| | s8 | ✓ | ✓ | (*1) |
| | s16 | - | ✓ | |

When performing convolution with s8/u8 FM and s16 WT, set 0x1 in the DMAIS.DMAIS_8TO16BIT_CONV register to enable 8bpp to 16bpp conversion for FM. If DMAIS.DMAIS_8TO16BIT_CONV is 1, DMAIE can be set only for even bits; odd bits must be set to 0. ARICSR.DMAIFM register must be set to 0x2 (s16).

(*1) 8bpp => 16bpp format conversion is required in FM settings
(*2) When using DMAI to load weight data, the DMAWS. DMAWS_FM register must be set to the same value as the ARICSR. ARIFM register.

### U33.3.5 ARI module

#### (1) SUMC

#### (a) 3DC value and BIAS addition

After all convolution results have been added up for an output channel, optionally a 3DC value and a BIAS value can be added. The 3DC value is individual channel value sent from the DMA3DC input channel. The 3DC value is usually used to input the convolution result of the previous run of the same layer when the convolution result of more than the maximum FM input channels that can be processed in one run must be added to get the result of one output channel. The BIAS value can be a common offset value specified by the BIAS register and used for all output channels, or it can be an individual offset value specified by the BIASxx_yy and BIASxx_yy_H register and used for only the specified output channel. The addition of BIAS values is enabled when the ARICSR.BIAS register is set to 1.

When ARICSR.BIAS_32bit is set to 1, a signed 32bit BIAS value can be specified with the BIASxx_yy and BIASxx_yy_H registers. When ARICSR.BIAS_32bit is set to 0, a signed 16bit BIAS value can be specified by the BIASxx_yy register.

#### (b) Scaler multiplier, zero-point adder and downshifter

To increase accuracy of recognition, 16bits scaler multiplier is prepared for each ARI channel before SFTM shifter. The SCALER value can be a common value specified by the SCALER_MULT_VAL register and used for all output channels, or it can be an individual value specified by the SCALER_MULTxx bits of the SCALER_MULT xx_yy register and used for only the specified output channel.

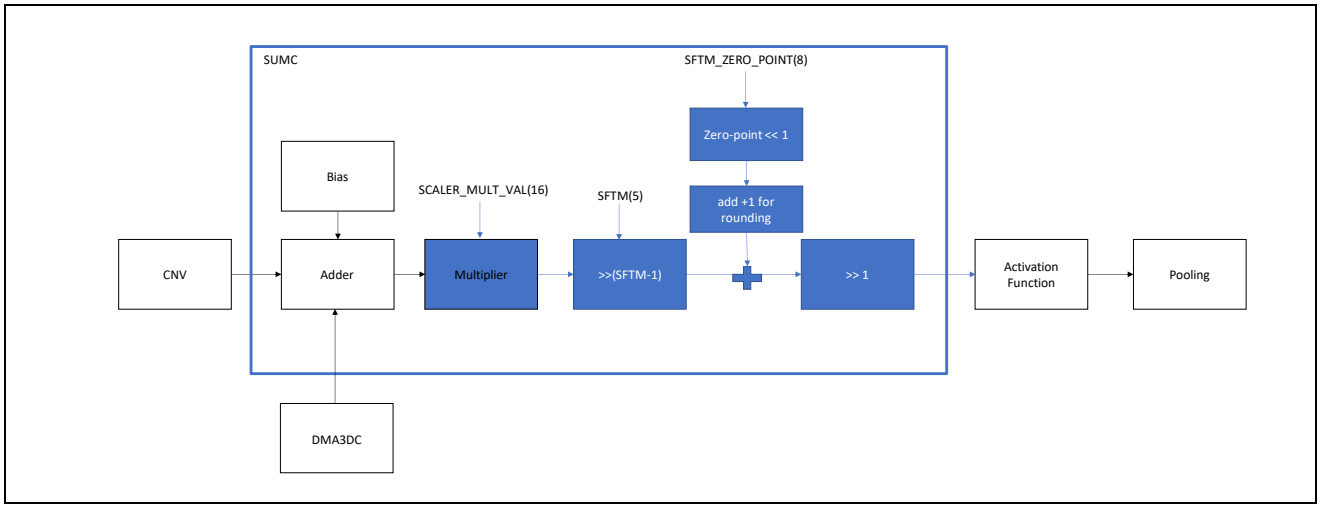



**Figure U33.31 The flow of Scaler multiplier, zero-point adder and downshifter**

After convolution, 3DC and BIAS addition and optional scaler multiplication, a zero-point adder defined over SFTM_ZERO_POINT register and a downshifter defined over SFTM value can be performed to transfer the intermediate value into the chosen output range.
For this downshifter, the rounding function “round to nearest, half towards +infinity” is available.

The SFTM value can be a common value specified by the SFTM register and used for all output channels, or it can be an individual value specified by the SFTMxx bits of the SFTM xx_yy register and used for only the specified output channel.

### (2) Activation

Table U33.34 is showing the supported activation functions and their implementation. They can be selected by using the ACT_FUNC bits in the ARICSR register. The activation function is only available for 8bit and 16bit format setting and for m16 format setting in case, the register bit ARICSR.M16_VAL_AFTER_ACT is set to 1, in case of m32 format setting, the activation function cannot be used.

**Table U33.34 Activation functions**

| Mode | Activation function | 16bit output channels | 8bit output channels |
|---|---|---|---|
| | | Post processing | Post processing |
| 0x0 | No activation | - | - |
| 0x1 | RELU | - | - |
| 0x2 | BRELU | - | - |
| 0x3 | Leakage RELU | Support | Support |
| 0x4 | Sigmoid | Support | Support |
| 0x5 | ELU | Support | Support |
| 0x6 | Tanh | Support | Support |
| 0x7 | H-Swish | Support | Support |
| 0x8 | Swish | Support | Support |
| 0x9 | Mish | Support | Support |
| 0xa | Exp | Support | Support |

In the following sub chapters, each activation function is described in more detail.

### (a) RELU

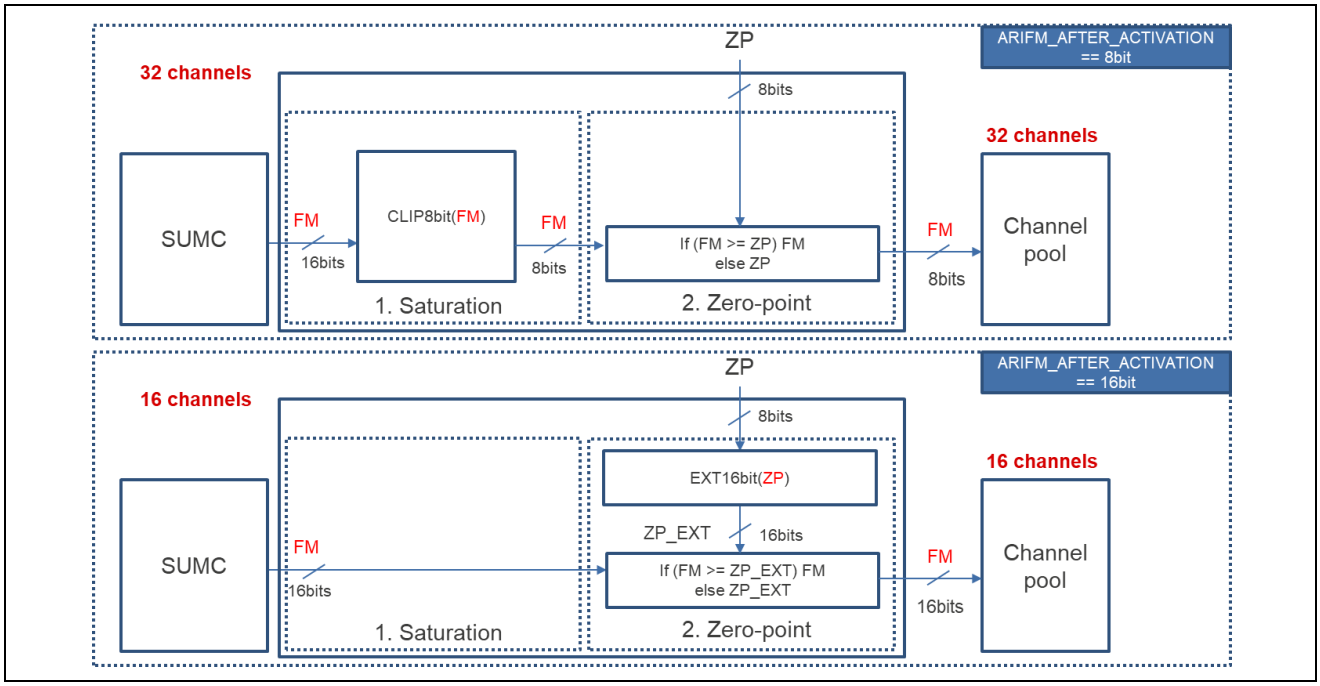



**Figure U33.32 RELU activation function flow**

**Table U33.35 RELU activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| ZP | SFTM_ZERO_POINT | Zero-point used for 8bit after SFTM function<br>User must set zero-point to 8'h0 when ARIFM_AFTER_ACTIVATION is 16bit. |
| ZP_EXT | - | Zero-point after signed extension from 8bits to 16bits |
| CLIP8bit(FM) | - | Clipping 8bits signed for input value FM:<br>CLIP8bit(FM) {<br>If (FM < -128)<br>{y = -128}<br>else if (FM >127)<br>{y = 127}<br>else {y = FM}<br>return y<br>} |
| EXT16bit(ZP) | - | Signed extension from 8bit signed to 16bit signed for ZP.<br>EXT16bit(ZP){<br>y= (int16) ZP<br>return y<br>} |

The rectified linear unit (RELU) activation function is suppressing all values smaller than zero-point value specified by the SFTM_ZERO_POINT register and replacing them with zero-point value while passing values greater than the zero-point value unchanged. When using 16bit mode, user must set the SFTM_ZERO_POINT register to zero.

For 8bit mode, the rectified linear unit (RELU) activation function is suppressing all values smaller than zero-point value specified by the SFTM_ZERO_POINT register and replacing them with zero-point value while the values greater than zero-point value get passed through unchanged. The input data and zero-point will be clipped to 8bit signed range by function CLIP8bit().

For 16bit mode, the rectified linear unit (RELU) activation function is suppressing all negative values and replacing them with 0 while positive values get passed through unchanged.

**(b) BRELU**

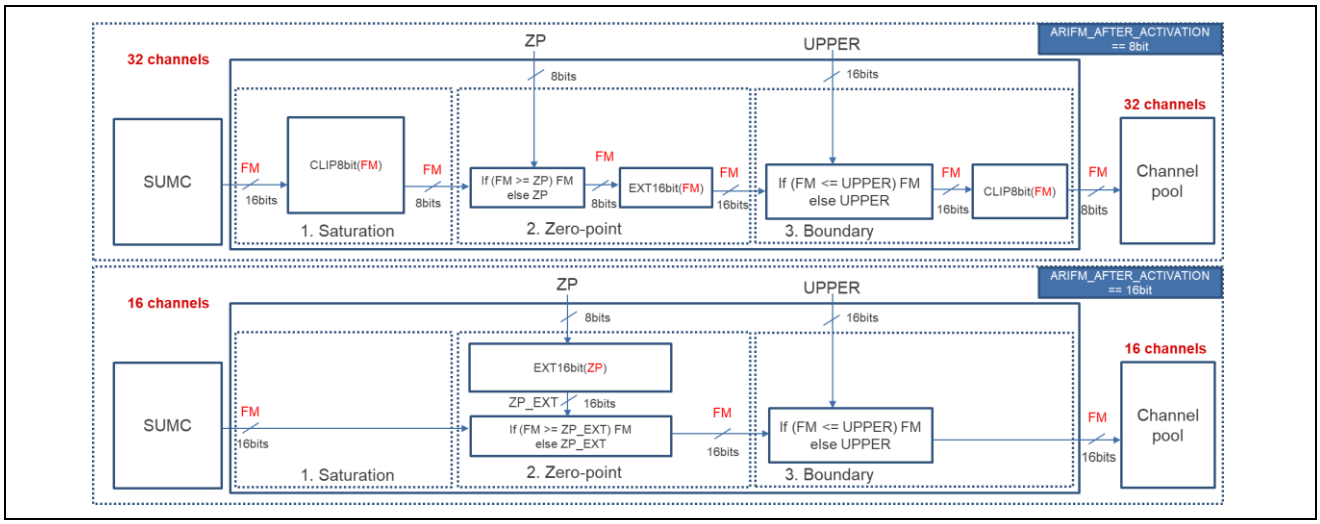



**Figure U33.33 BRELU activation function flow**

**Table U33.36 BRELU activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| ZP | SFTM_ZERO_POINT | Zero-point used for 8bit after SFTM function<br>User must set zero-point to 8'h0 when ARIFM_AFTER_ACTIVATION is 16bit. |
| ZP_EXT | - | Zero-point after signed extension from 8bits to 16bits |
| UPPER | ARI_BND. ARI_UPPER_BND | ARI upper boundary |
| CLIP8bit(FM) | - | Clipping 8bits signed for input value FM:<br>CLIP8bit(FM) {<br>If (FM < -128)<br>{y = -128}<br>else if (FM >127)<br>{y = 127}<br>else {y = FM}<br>return y<br>} |
| EXT16bit(ZP) | - | Signed extension from 8bit signed to 16bit signed for ZP.<br>EXT16bit(ZP){<br>y= (int16) ZP<br>return y<br>} |
| EXT16bit(FM) | - | Signed extension from 8bit signed to 16bit signed for FM.<br>EXT16bit(FM){<br>y= (int16) FM<br>return y<br>} |

For 8bit mode, the bounded rectified linear unit (BRELU) activation function is suppressing all values smaller than zero-point value specified by the SFTM_ZERO_POINT register and replacing them with zero-point value.
The values greater than zero-point value are compared with an upper boundary specified by the ARI_UPPER_BND bits of the ARI_BND register. In case the value is larger than the specified upper boundary, the value gets replaced by the upper boundary, otherwise, the value gets passed through unchanged. When using 16bit mode, user must set SFTM_ZERO_POINT register to zero. The input data and zero-point will be clipped to 8bit signed range by function CLIP8bit().

For 16bit mode, the bounded rectified linear unit (BRELU) activation function is suppressing all negative values and replacing them with 0. Positive values are compared with an upper boundary specified by the ARI_UPPER_BND bits of

the ARI_BND register. In case a positive value is larger than the specified upper boundary, the value gets replaced by the upper boundary, otherwise, the positive value gets passed through unchanged.

### (c) Leakage RELU

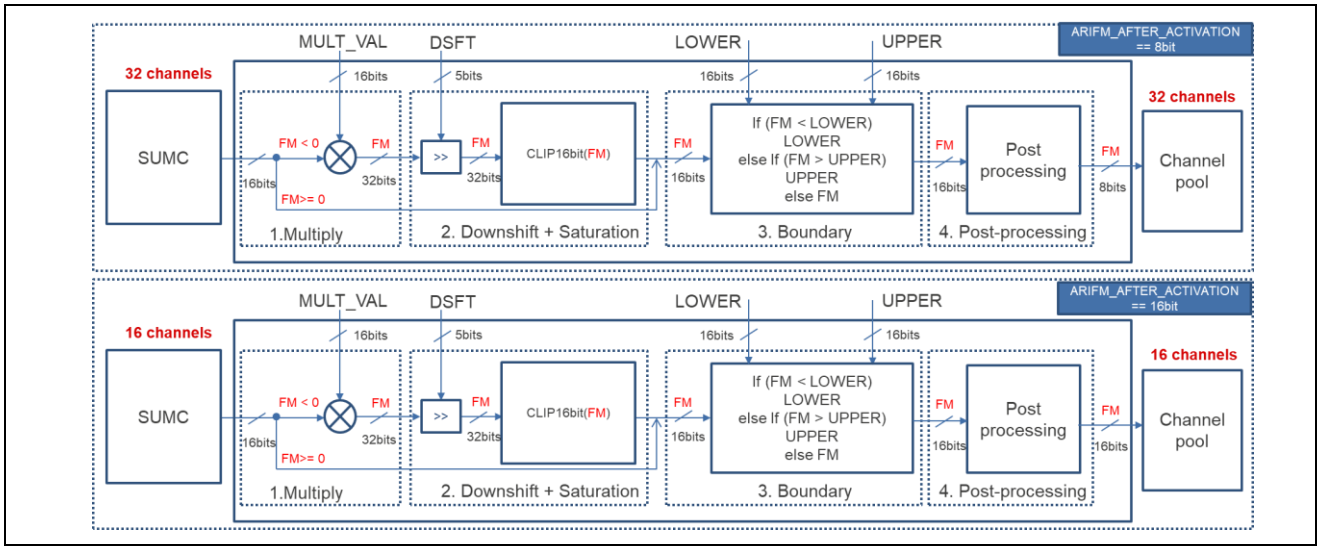



**Figure U33.34 LEAKY RELU activation function flow**

**Table U33.37 Leakage RELU activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| MULT_VAL | ACTIVATION_MULT_VAL | Factor for MULT operation |
| DSFT | ACTIVATION_MULT_POST_SFT | Downshift after MULT operation |
| UPPER | ARI_BND.ARI_UPPER_BND | ARI upper boundary |
| LOWER | ARI_BND.ARI_LOWER_BND | ARI lower boundary |
| CLIP16bit(FM) | - | Clipping 16bits signed for input value FM:<br>CLIP16bit(FM) {<br>If (FM < -32768)<br>{y = -32768}<br>else if (FM >32767)<br>{y = 32767}<br>else {y = FM}<br>return y<br>} |

For positive values, the value is defined between the ARI_LOWER_BND and ARI_UPPER_BND bits of the ARI_BND register. If the value is larger than the upper boundary specified by the ARI_UPPER_BND bits of the ARI_BND register, the value gets replaced by the upper boundary. If the value is smaller than the lower boundary specified by the ARI_LOWER_BND bits of the ARI_BND register, the value gets replaced by the lower boundary.

For negative values, a multiplication with a positive number defined over the ACTIVATION_MULT_VAL register is performed, followed by an arithmetic down shift defined over the ACTIVATION_MULT_POST_SFT register. The result after downshift is saturated to signed 16bit range by function CLIP16bit(). After that, the value is defined between the ARI_LOWER_BND and ARI_UPPER_BND bits of the ARI_BND register like positives value.

### (d) Sigmoid

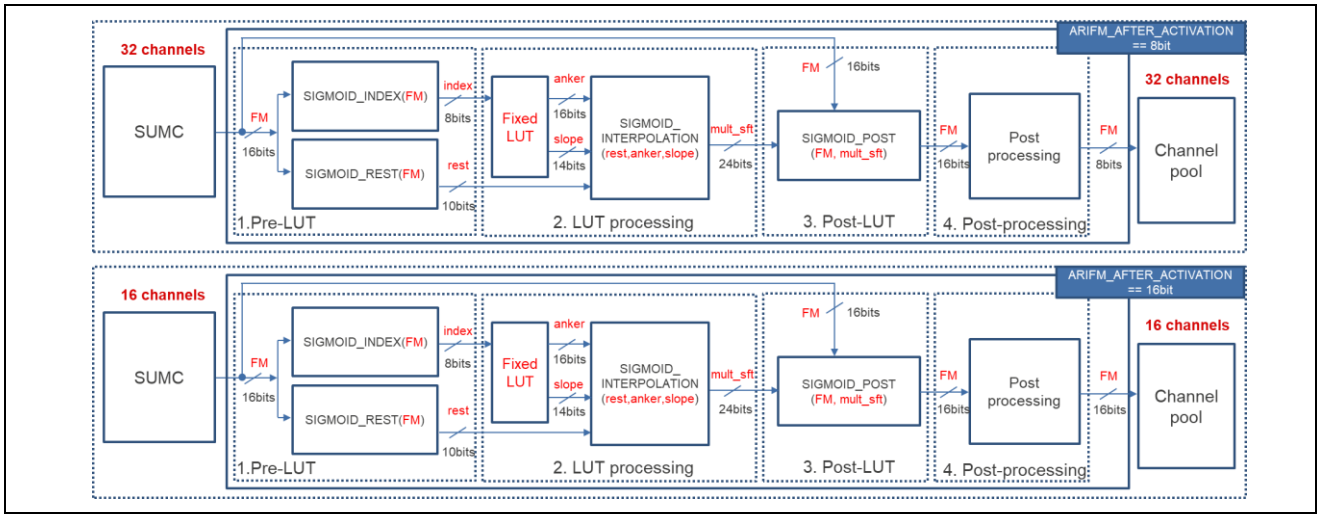



**Figure U33.35 SIGMOID activation function flow**

**Table U33.38 SIGMOID activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| SIGMOID_INDEX(FM) | - | Calculate Index value for input value FM:<br>SIGMOID_INDEX(FM) {<br>abs_x = ( FM < 0 ) ? -FM : FM;<br>y = (abs_x / 256)<br>return y<br>} |
| SIGMOID_REST(FM) | - | Calculate Rest value for input value FM:<br>SIGMOID_REST(FM) {<br>abs_x = ( FM < 0 ) ? -FM : FM;<br>y = mod(abs_x ,256)<br>return y<br>} |
| SIGMOID_INTERPOLATION<br>(rest, anker, slope) | - | Calculate LUT processing value for Sigmoid:<br>SIGMOID_INTERPOLATION(rest, anker, slope){<br>mult_val = rest * slope;<br>y = (mult_val / 512) + anker;<br>return y<br>} |
| SIGMOID_POST<br>(FM, intercept) | - | Calculate post-LUT processing value for input value FM:<br>SIGMOID_POST (FM, intercept) {<br>y= ( FM < 0 ) ? (32768 - intercept) : intercept;<br>return y<br>} |

Sigmoid functions are implemented for positive and negative values by using the LUT, while for 16bit input data the input ranges from -8 to 8 is covered by the LUT with size 128 elements. The input data to LUT is calculated by function SIGMOID_INDEX() and SIGMOID_REST() and the output data after LUT is calculated by function SIGMOID_POST().
After LUT, the value is continuously calculated at “Post processing” logic.

**(e) ELU**

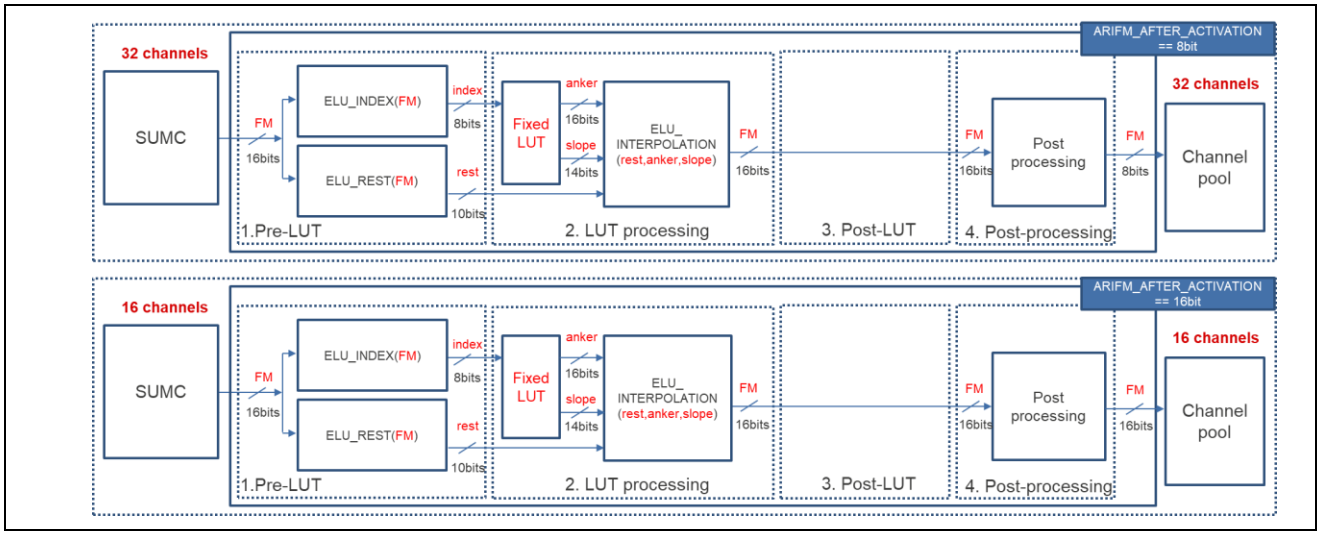



**Figure U33.36 ELU activation function flow**

**Table U33.39 ELU activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| ELU_INDEX(FM) | - | Calculate Index value for input value FM:<br>ELU_INDEX(FM) {<br>y = ((FM + 32768) / 256)<br>return y<br>} |
| ELU_REST(FM) | - | Calculate Rest value for input value FM:<br>ELU_REST(FM) {<br>tmp = mod(FM,256)<br>y = mod(tmp + 256,256)<br>return y<br>} |
| ELU_INTERPOLATION<br>(rest, anker, slope) | - | Calculate LUT processing value for ELU:<br>ELU_INTERPOLATION(rest, anker, slope){<br>mult_val = rest * slope;<br>y = (mult_val / 4096) + anker;<br>return y<br>} |

For the exponential linear unit (ELU) activation function, the positive values get passed through unchanged.
For negative values, an exponential function implemented using an LUT is used to produce small negative values which allows them to push mean unit activations closer to zero. While for 16bit input data the input ranges from -8 to 0 is covered by the LUT with size 256 elements. The input data to LUT is calculated by function ELU_INDEX() and ELU_REST().
After LUT, the value is continuously calculated at “Post processing” logic.

### (f) Tanh

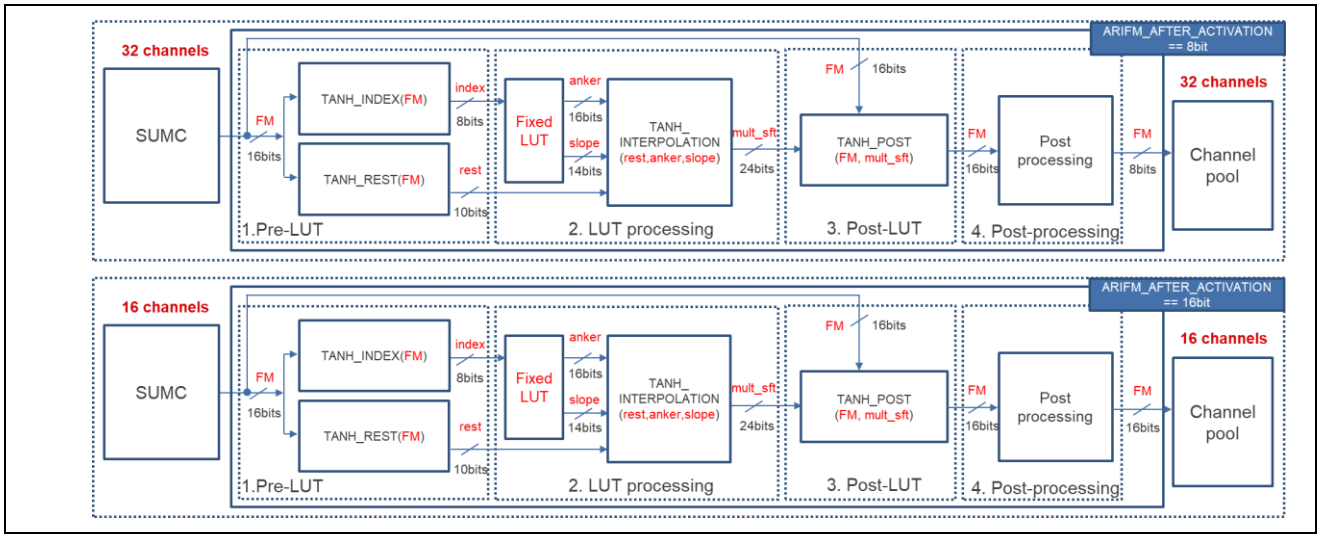



**Figure U33.37 TANH activation function flow**

**Table U33.40 TANH activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| TANH_INDEX(FM) | - | Calculate Index value for input value FM:<br>TANH_INDEX(FM) {<br>abs_x = ( FM < 0 ) ? -FM : FM;<br>y = (abs_x / 512)<br>return y<br>} |
| TANH_REST(FM) | - | Calculate Rest value for input value FM:<br>TANH_REST(FM) {<br>abs_x = (FM < 0 ) ? -FM : FM;<br>y = mod(abs_x ,512)<br>return y<br>} |
| TANH_INTERPOLATION (rest, anker, slope) | - | Calculate LUT processing value for TANH:<br>TANH_INTERPOLATION(rest, anker, slope){<br>mult_val = rest * slope;<br>y = (mult_val / 1024) + anker;<br>return y<br>} |
| TANH_POST(FM, intercept) | - | Calculate post-LUT processing value for input value FM:<br>TANH_POST (FM, intercept) {<br>y= (FM < 0 ) ? -intercept : intercept;<br>return y<br>} |

Tanh function is implemented for positive and negative values by using the LUT as done for the ELU activation function, while for 16bit input data the input ranges from -4 to 4 is covered by the LUT with size 64 elements. The input data to LUT is calculated by function TANH_INDEX() and TANH_REST() and the output data after LUT is calculated by function TANH_POST().
After LUT, the value is continuously calculated at “Post processing” logic.

### (g) H-Swish

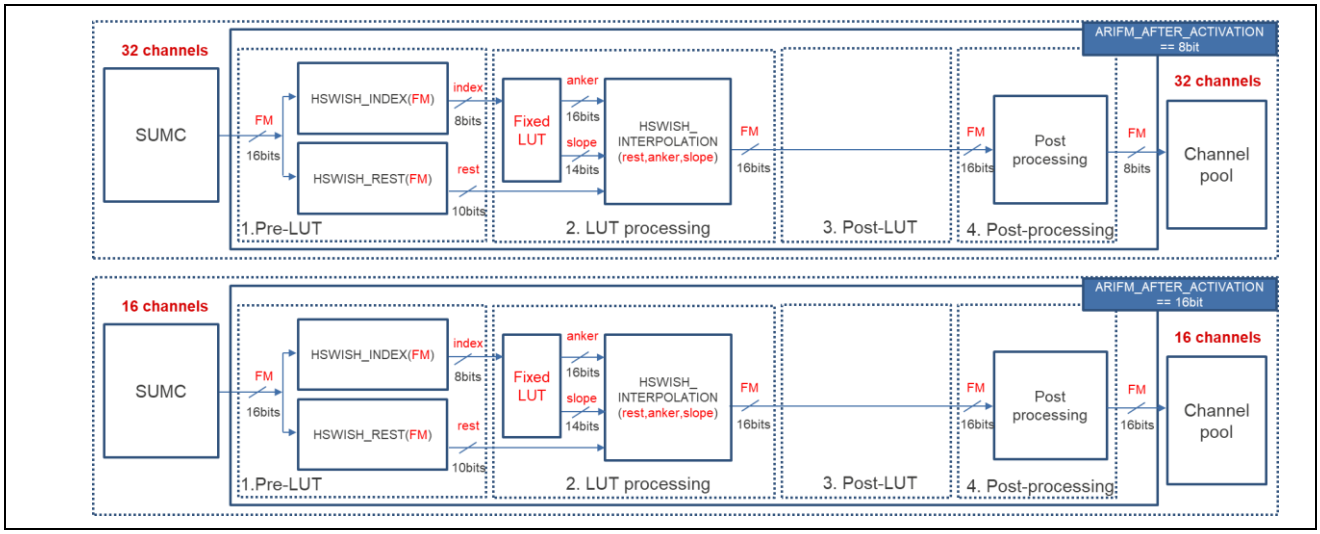



**Figure U33.38 H-SWISH activation function flow**

**Table U33.41 H-SWISH activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| HSWISH_INDEX(FM) | - | Calculate Index value for input value FM:<br>HSWISH_INDEX(FM) {<br>y = ((FM + 32768) / 512)<br>return y<br>} |
| HSWISH_REST(FM) | - | Calculate Rest value for input value FM:<br>HSWISH_REST(FM) {<br>tmp = mod(FM,512)<br>y = mod(tmp + 512, 512)<br>return y<br>} |
| HSWISH_INTERPOLATION (rest, anker, slope) | - | Calculate LUT processing value for HSWISH:<br>HSWISH_INTERPOLATION(rest, anker, slope){<br>mult_val = rest * slope;<br>y = (mult_val / 4096) + anker;<br>return y<br>} |

H-Swish function is implemented for 16bit input data the input ranges from -8 to 8, if the input value greater than 4, the value gets passed through unchanged. The input data to LUT is calculated by function HSWISH_INDEX() and HSWISH_REST().

The values smaller than 4 are covered by the LUT with size 128 elements and the value after LUT is continuously calculated at “Post processing” logic.

### (h) Swish

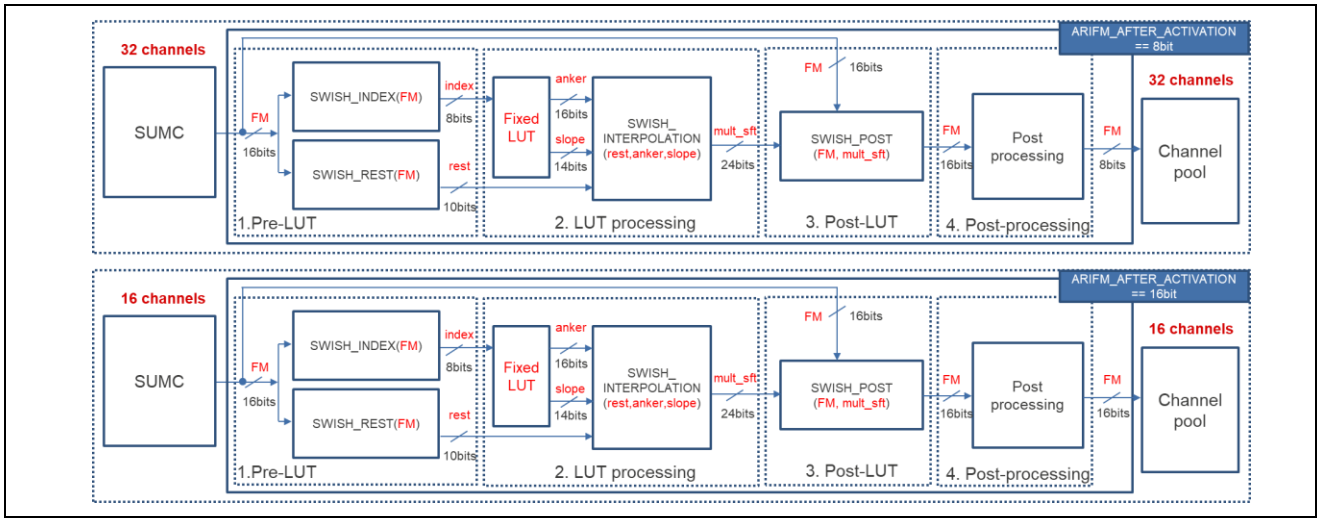



**Figure U33.39 SWISH activation function flow**

**Table U33.42 SWISH activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| SWISH_INDEX(FM) | - | Calculate Index value for input value FM:<br>SWISH_INDEX(FM) {<br>abs_x = (FM < 0 ) ? -FM : FM;<br>y = (abs_x / 256)<br>return y<br>} |
| SWISH_REST(FM) | - | Calculate Rest value for input value FM:<br>SWISH_REST(FM) {<br>abs_x = (FM < 0 ) ? -FM : FM;<br>y = mod(abs_x ,256)<br>return y<br>} |
| SWISH_INTERPOLATION (rest, anker, slope) | - | Calculate LUT processing value for SWISH:<br>SWISH_INTERPOLATION(rest, anker, slope){<br>mult_val = rest * slope;<br>y = (mult_val / 512) + anker;<br>return y<br>} |
| SWISH_POST(FM, intercept) | - | Calculate post-LUT processing value for input value FM:<br>SWISH_POST (FM, intercept) {<br>temp_y= ( FM < 0 ) ? (32768 - intercept) : intercept;<br>y= temp_y * FM>>15;<br>return y<br>} |

Swish function is implemented for positive and negative values by using the LUT as done for the Sigmoid activation function, while for 16bit input data the input ranges from -8 to 8 is covered by the LUT with size 128 elements. The input data to LUT is calculated by function SWISH_INDEX() and SWISH_REST() and the output data after LUT is calculated by function SWISH_POST().
After LUT, the value is continuously calculated at “Post processing” logic.

**(i) Mish**

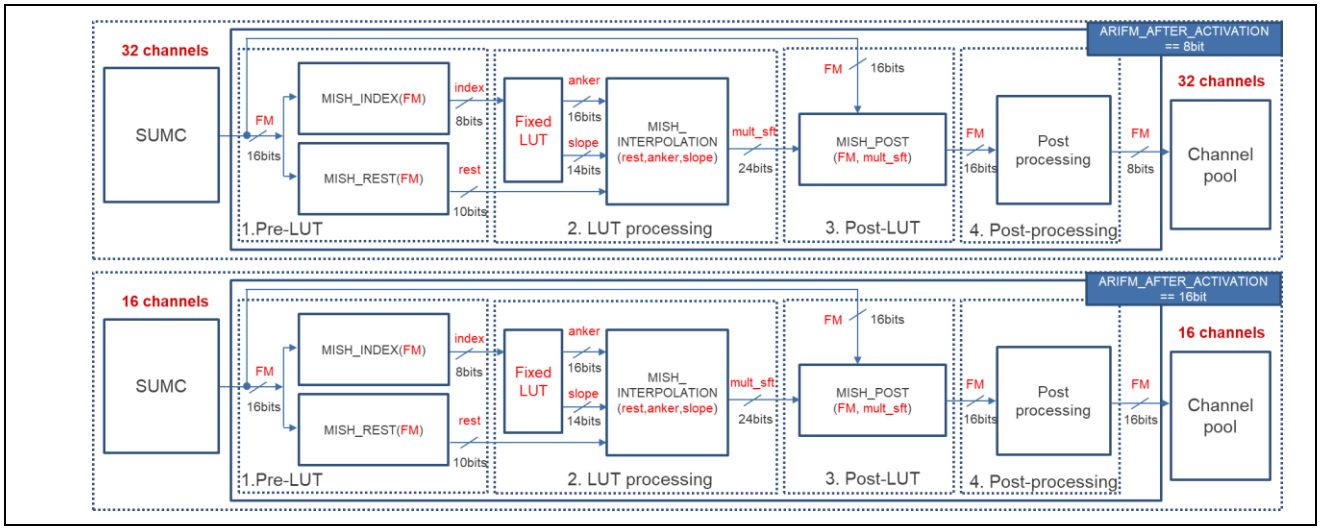



**Figure U33.40 MISH activation function flow**

**Table U33.43 MISH activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| MISH_INDEX(FM) | - | Calculate Index value for input value FM:<br>MISH_INDEX(FM) {<br>y = ((FM + 32768) / 512)<br>return y<br>} |
| MISH_REST(FM) | - | Calculate Rest value for input value FM:<br>MISH_REST(FM) {<br>tmp = mod(FM,512)<br>y = mod(tmp + 512, 512)<br>return y<br>} |
| MISH_INTERPOLATION (rest, anker, slope) | - | Calculate LUT processing value for MISH:<br>MISH_INTERPOLATION(rest, anker, slope){<br>mult_val = rest * slope;<br>y = (mult_val / 512) + anker;<br>return y<br>} |
| MISH_POST(FM, intercept) | - | Calculate post-LUT processing value for input value FM:<br>MISH_POST (FM, intercept) {<br>y= intercept * FM>>15;<br>return y<br>} |

Mish function is implemented for positive and negative values by using the LUT as done for the Sigmoid activation function, while for 16bit input data the input ranges from -8 to 8 is covered by the LUT with size 128 elements. The input data to LUT is calculated by function MISH_INDEX() and MISH_REST() and the output data after LUT is calculated by function MISH_POST().
After LUT, the value is continuously calculated at “Post processing” logic.

### (j) Exp

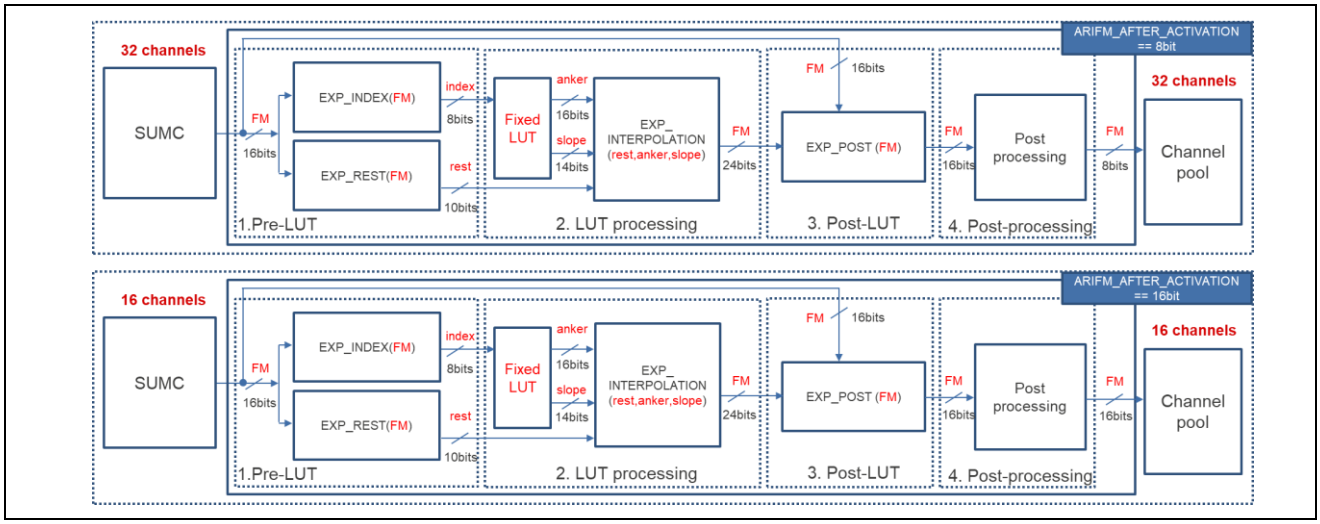



**Figure U33.41 EXP activation function flow**

**Table U33.44 EXP activation function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| EXP_INDEX(FM) | - | Calculate Index value for input value FM:<br>EXP_INDEX(FM) {<br>y = ((FM + 32768) / 512)<br>return y<br>} |
| EXP_REST(FM) | - | Calculate Rest value for input value FM:<br>EXP_REST(FM) {<br>y = mod(FM,512)<br>return y<br>} |
| EXP_INTERPOLATION<br>(rest, anker, slope) | - | Calculate LUT processing value for EXP:<br>EXP_INTERPOLATION(rest, anker, slope){<br>mult_val = rest * slope;<br>y = (mult_val / 4096) + anker;<br>return y<br>} |
| EXP_POST(FM) | - | Calculate post-LUT processing value for input value FM:<br>EXP_POST (FM) {<br>y= FM + 4095;<br>return y<br>} |

Exp function is implemented for negative values by using the LUT as done for the ELU activation function, user needs guarantee the input value is always below 0. The input data to LUT is calculated by function EXP_INDEX() and EXP_REST() and the output data after LUT is calculated by function EXP_POST().
After LUT, the value is continuously calculated at “Post processing” logic.

### (k) Post processing

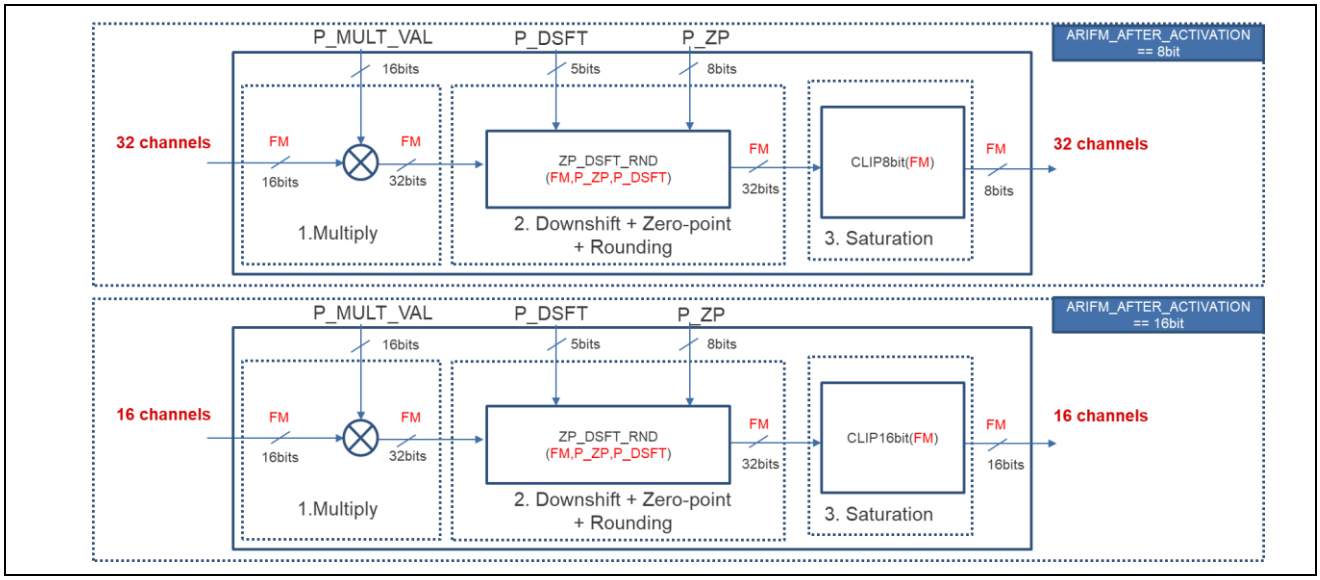



**Figure U33.42 Post processing flow**

**Table U33.45 Post processing function symbols**

| Symbol | Register bit | Description |
|---|---|---|
| FM | - | Feature Map |
| ARIFM_AFTER_ACTIVATION | ARICSR.ARIFM_AFTER_ACTIVATION | Output data format of Activation function. |
| P_ZP | ACTIVATION_ZERO_POINT | Zero Point used for 8bit activation functions at post processing<br>User must set zero-point to 8'h0 when ARIFM_AFTER_ACTIVATION is 16bit. |
| P_MULT_VAL | ACTIVATION_OUTPUT_PARAM.<br>ACTIVATION_OUTPUT_MULT | Factor for MULT operation of post processing |
| P_DSFT | ACTIVATION_OUTPUT_PARAM.<br>ACTIVATION_OUTPUT_SFT | Downshift after MULT operation of post processing |
| ZP_DSFT_RND<br>(FM,P_ZP,P_DSFT) | - | Downshift + Zero-point + Rounding function<br>ZP_DSFT_RND(FM,P_ZP,P_DSFT) {<br>y = (((FM >> (P_DSFT-1)) + P_ZP + 1) >> 1<br>return y<br>} |
| CLIP8bit(FM) | - | Clipping 8bits signed for input value FM:<br>CLIP8bit(FM) {<br>If (FM < -128)<br>{y = -128}<br>else if (FM >127)<br>{y = 127}<br>else {y = FM}<br>return y<br>} |
| CLIP16bit(FM) | - | Clipping 16bits signed for input value FM:<br>CLIP16bit(FM) {<br>If (FM < -32768)<br>{y = -32768}<br>else if (FM >32767)<br>{y = 32767}<br>else {y = FM}<br>return y<br>} |

Post processing implemented a multiplication with a positive number defined over ACTIVATION_OUTPUT_MULT bits of the ACTIVATION_OUTPUT_PARAM register, followed by zero-point adder defined over ACTIVATION_ZERO_POINT register and an arithmetic down shift defined over the ACTIVATION_OUTPUT_SFT bits of the ACTIVATION_OUTPUT_PARAM register.

For this downshifter, the rounding function "round to nearest, half towards +infinity" is available. The value itself is handled on output side with the ARIFM_AFTER_ACITIVATION.

### (l) Format dependent data flow

Post processing is supported on output side of activation functions. When user set no activation or RELU or BRELU functions, post processing is not supported.

For other activation modes above, post processing is supported. While for ARIFM_AFTER_ACITIVATION equal to 8bit mode, user can define zero-point over ACTIVATION_ZERO_POINT register, for ARIFM_AFTER_ACITIVATION equal to 16bit mode, ACTIVATION_ZERO_POINT register must be set to zero.

## (3) Channel Pool

The channel pooling function is only available for 8bit and 16bit format setting, in case of m16 or m32 format setting, the channel pooling function cannot be used. As channel pooling functions, the channel MAX, channel RANK and channel LABEL functions are supported which are described in the following in more detail.

### (a) Channel MAX

The channel MAX function is selected by setting the CPOOL_MODE info bits inside the ARICSR register to 0x1. It searches for each x/y position the maximum value of all channels inside a group and using for the lowest group channel output this value. All other channels of the group get disabled and do not hold valid data afterwards. Supported are the group settings defined in Table U33.46.

**Table U33.46 Channel MAX group division**

| CPOOL_INFO | Group division |
|---|---|
| 0x0 | 1 group of 32 channels |
| 0x1 | 2 groups of each 16 channels |
| 0x2 | 4 groups of each 8 channels |
| 0x3 | 8 groups of each 4 channels |
| 0x4 | 16 groups of each 2 channels |
| 0x5 | 32 groups of each 1 channel |
| other | Not supported |

### (b) Channel RANK

The Channel RANK function is selected by setting the CPOOL_MODE info bits inside the ARICSR register to 0x2. The function is sorting all 32 channels (8bit) or 16 channels (16bit) starting with largest value in the channel with lowest channel number. Inactivated channels are sorted to the end in the channels with highest channel number.

The user can additionally define how many lower data bits should be used for the information of the original channel number by specifying the number of bits over CPOOL_INFO bits of ARICSR register. Values from 0 till 5 are here supported. If specifying 0, e.g., then all data bits are used for the data information and the original channel information before sorting is getting lost. If using 32 channels in ARI 8bit mode and all channel information is needed, the user must specify 5 for CPOOL_INFO. In that case, all original channel information is stored out over DMAO replacing the lower 5 bits of the data information.

For the special case of sorting 32 channels and operating in 16bit mode, 2 runs with 16 channels become necessary. To distinguish later between the 2 runs, the user must specify the user information CPOOL_UI in the ARICSR register, which is replacing bit number 5 of the output data.

### (c) Channel LABEL

The Channel LABEL function is similar working as the Channel RANK function, however, it is storing only the channel information out over the DMAO channels, the data information is getting lost after the sorting process.

The Channel LABEL function is selected by setting the CPOOL_MODE bits inside the ARICSR register to 0x3. The function is sorting all 32 channels (8bit) or 16 channels (16bit) starting with largest value in the channel with lowest channel number. Inactivated channels are sorted to the end in the channels with highest number.

The user can additional define how many lower bits should be used for the information of the original channel number by specifying the number of bits over CPOOL_INFO bits of ARICSR register. Values from 0 till 5 are here supported. If specifying e.g. 0, then the original channel information before sorting is getting lost. If using 32 channels in ARI 8bit mode and all channel information is needed, the user must specify 5 for the CPOOL_INFO bits. In that case, all original channel information is stored out over DMAO using the lower 5 bits.

For the special case of sorting 32 channels and operating in 16bit mode, 2 runs with 16 channels become necessary. To distinguish later between the 2 runs, the user must specify the user information in the CPOOL_UI bit of the ARICSR register, which is sent out over bit number 5 of the DMAO channel data.

#### (4) XY Pool

The XY pooling function is only available for 8bit and 16bit format setting, in case of m16 or m32 format setting, the XY pooling function cannot be used. With the XYPOOL_MODE bit of the ARICSR register the XYPOOL mode can be selected, which enables the user to perform a pooling over a defined x/y area of a channel. The output of the selected XY POOL function is replacing the selected area inside the DMAO channel, so that when using an area larger 1x1, a down shrinking of the input size is performed. The supported pooling size is defined by the XPOOL bits of the ARICSR register for x direction and by the YPOOL bits of the ARICSR register for y direction. XPOOL and YPOOL can take values from 0 till 3 causing values for the pooling size in x and y direction from 1 till 4.

#### (a) XY MAX

In case of a XY MAX pooling function selected by setting the XYPOOL_MODE bits of the ARICSR register to 2b01, the maximal value is searched inside the defined pooling area and replacing the pooling area inside the DMAO output channel.

#### (b) XY Average

In case of a XY Average pooling function selected by setting the XYPOOL_MODE bits of the ARICSR register to 1b10, the average of all values inside the defined pooling area is calculated and replacing the pooling area inside the DMAO output channel. After the XY Average operation, an ARI format dependent saturation function is supported.
When using XY average pooling, XPOOL and YPOOL bits of the ARICSR register must be the same.

#### (c) XY upper left

In case of a XY upper left pooling function selected by setting the XYPOOL_MODE bits of the ARICSR register to 2b11, the first value inside the defined pooling area is kept and replacing the pooling area inside the DMAO output channel.

### U33.3.6 DMAO module

The DMAO_OUT module, also called as DMAO module, is responsible to store the output data from the 16 (for 16bit) or 32 (for 8bit, mixed mode 16bit/32bit) DMA output buffers to the SPMC. Figure U33.43 is showing a block diagram of this module.

### (1) DMAO module block diagram

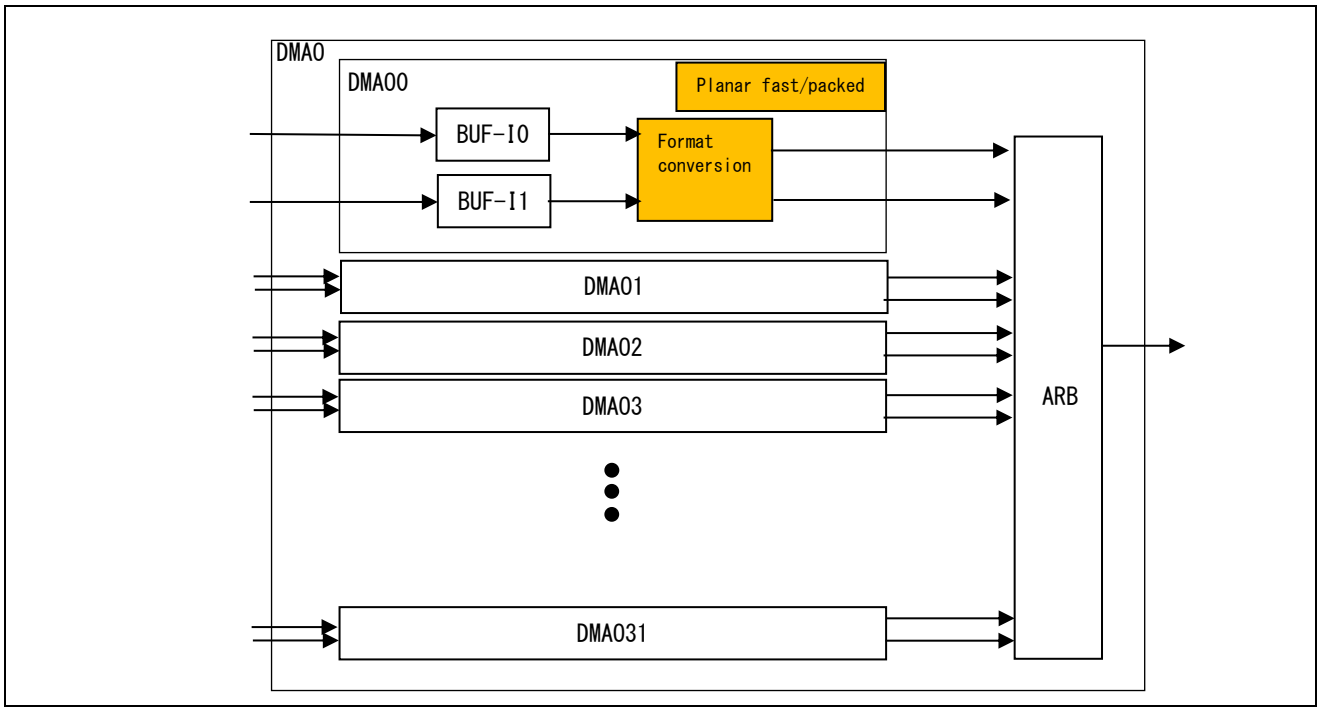



**Figure U33.43 DMAO BLOCK**

The DMAO module consists of 32 sub modules DMAO0 till DMAO31 and an arbiter.
The arbiter is responsible for the data arbitration between single external channel and 32 internal channels selecting the data from the 32 submodules DMAO0 till DMAO31. Each DMAO sub module has the same structure. It consists of the 2 data buffers and on output side the format conversion unit and the planar fast/packed decision unit.
The format conversion unit on output size is responsible for converting the data from 16bit signed ARI format to 8bit signed DMAO format only.

The planar fast/packed decision unit is specifying the data destination and arrangement of the outgoing data. In planar fast type, every 16 pixels for all channels are transferred at the same time. The pixels for each channel are stored here separately inside the IMPC memory.

In packed mode, first for each channel 1 pixel is sent, before the next pixel for each channel is sent. In this channel interleaved transfer mode, it is required to enable all output channels, so that the data for 1 channel is always received with the same distance. For the case of 16 16bit channels as well as for the case of 32 8bit channels, the distance is 32byte. For the case of 32 channels mixed mode, the distance is 64byte (mixed mode 16bit) and 128byte (mixed mode 32bit). For the Format conversion function in Figure U33.43, please refer to Table U33.48.

### (2) Picture start address, width, height, and stride settings

As external start addresses, an address of the SPMC0/SPMC1 address range must be specified. For packed mode, the start address of the channels is defined by using the DMAOSA register, while for planar fast type, the start addresses are defined by using the DMAOSA register and the DMAOCO register. The address must be aligned 16 pixels (s8/s16) for planar fast type. For packed format, the address must be aligned dependent 1 pixel (s8/s16) when output to SPMC0 and dependent on the pixel width to 16 pixels (s8, s16), 8 pixels (m16), 4 pixels (m32) when output to SPMC1.
The tile dimension is specified in pixels and rows. Here, the width defined by the DMAOLX bits of the DMAOL register must not be larger than 128 pixels while the height defined by the DMAOLY bits of the DMAOL register must be smaller than 2048 rows.

The stride defined by the DMAOST register defines the distance in pixel between 2 rows and the stride is specified in pixel. For planar type, the stride is specified multiple of 16 pixels when output to SPMC0. The value set in line stride register (DMAOST) must be greater than or equal to the value in horizontal size register (DMAOLX).

For packed mode, the stride is set as multiple of 16 pixels (8bi/16bit), multiple of 8 pixels (mixed mode 16bit), multiple of 4 pixels (mixed mode 32bit).

If the specified output size is not enough to output the ARI channel data size defined by the ARI_XLEN bits and ARI_YLEN bits of the ARI_LEN register, the rest is thrown away.

**Table U33.47 DMAO parameter support**

| Format in SPMC0/1 | SPMC memory | Data type | Start address | Channel offset | Offset address | Stride | Length | Channel part | Wrap-around address |
|---|---|---|---|---|---|---|---|---|---|
| Planar fast type of planar format | SPMC0 | s8 | 16pix aligned | All channels | Support | Multiple of 16pix | 0 < x <= 128<br>0 < y <= 2048 | Support | - |
| | | s16 | 16pix aligned | Only even channels | Support | Multiple of 16pix | 0 < x <= 128<br>0 < y <= 2048 | Support | - |
| Packed | SPMC0 | s8 | 1pix aligned | - | Support | Multiple of 16pix | 0 < x <= 128<br>0 < y <= 2048 | Support | Multiple of 512byte |
| | | s16 | 1pix aligned | - | Support | Multiple of 16pix | 0 < x <= 128<br>0 < y <= 2048 | Support | Multiple of 512byte |
| | SPMC1 | s8 | 16pix aligned | - | Support | Multiple of 16pix | 0 < x <= 128<br>0 < y <= 1024 | Support | - |
| | | s16 | 16pix aligned | - | Support | Multiple of 16pix | 0 < x <= 128<br>0 < y <= 1024 | Support | - |
| | | m16 | 8pix aligned | - | Support | Multiple of 8pix | 0 < x <= 128<br>0 < y <= 1024 | - | - |
| | | m32 | 4pix aligned | - | Support | Multiple of 4pix | 0 < x <= 128<br>0 < y <= 1024 | - | - |

### (3) DMAO Format conversion

The ARI format defined by the ARIFM/ ARIFM_AFTER_ACTIVATION bits of the ARICSR register can be set differently from ARIFM_AFTER_ACTIVATION bit of the DMAOPARAM and the DMAO format defined by the DMAOFM bits of the DMAOPARAM, only the combinations shown in Table U33.48 between ARI format and DMAO format are supported format conversion.

**Table U33.48 ARI to DMAO format conversion implementation**

| ARIFM of ARICSR | ARIFM_AFTER_ACTIVATION of ARICSR | ARIFM_AFTER_ACTIVATION of DMAOPARAM | DMAOFM Of DMAOPARAM | Used formula |
|---|---|---|---|---|
| s8 | s8 | s8 | s8 | DMAO_val = ARI_val[7:0] |
| s8 | s16 | s8 | s8 | DMAO_val = ARI_val[7:0] |
| s8 | s16 | s16 | s8 | DMAO_val = (ARI_val[15:7] + 1) >>1 |
| s8 | s16 | s16 | s16 | DMAO_val = ARI_val[15:0] |
| s16 | s16 | s8 | s8 | DMAO_val = ARI_val[7:0] |
| s16 | s16 | s16 | s8 | DMAO_val = (ARI_val[15:7] + 1) >>1 |
| s16 | s16 | s16 | s16 | DMAO_val = ARI_val[15:0] |

### (4) Wrap around area

Wrap-around function for address supports packed format only and can be enabled/disabled by setting the bit DMAOS_WRAP of DMAOS register. The start and end addresses of output areas can be set by DMAOWA_START and DMAOWA_END of DMAOWA register. The start and end addresses shall be 512byte aligned. Initial address defined by DMAOSA register must be set within the range.

### (5) Offset address

To fastly change between output data sets of consecutively processed DMAO channels of the same layer with the same dimensions, by specifying an offset address, the next start address is calculated automatically by adding this offset address to the current start address, which is afterwards replaced by the newly calculated start address.

### (6) Channel offset

The channel offset is only used for the planar fast type. Here it is specifying the difference between the start addresses of 2 neighbored output channels. It is set over the DMAOCO register. By using this channel offset, it can be made sure that each channel can work on its own data set.

Further to mention is that inactive channels are still counting for the distance calculation, so, e.g., if byte pixels are processed and channel 0 has as start address 0x0000 0000 and the offset is specified as 0x2010, then channel 2 is starting at address 0x0000 4020, independent of whether channel 1 is active or not.

### (7) Channel enable

In planar fast type and packed mode, each channel can be individually enabled or disabled by using the bits of the DMAOE register, by disabling them, energy can be saved. However, bit 0 of DMAOE register is mandatory enabled. The disabled channels are counting when e.g., calculating a channel start address for planar fast type.

When the function read-modify-write (specified by bit DMAO_RMW of DMAOS register) is disabled, if the channel is disabled (DMAOE is 0), 0 is written out.

In planar fast and packed mode, if DMAO is not used (DMAOS is not set), DMAOE bits must be set all to zero.

The relationship between the number of DMAO data channels and the bits in the DMAOE register is shown in Table U33.49

There are some limitations, please refer to U33.5.1.

**Table U33.49 Relationship of DMAO channel to DMAO register bit depending on selected data type**

| DMAOPARAM Register DMAOFM bit setting | The output channel assigned to the DMAOE register bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bit 15 | bit 14 | bit 13 | bit 12 | bit 11 | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| s8 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| s16 | - | 7 | - | 6 | - | 5 | - | 4 | - | 3 | - | 2 | - | 1 | - | 0 |
| m16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| m32 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| DMAOPARAM Register DMAOFM bit setting | The output channel assigned to the DMAOE register bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bit 31 | bit 30 | bit 29 | bit 28 | bit 27 | bit 26 | bit 25 | bit 24 | bit 23 | bit 22 | bit 21 | bit 20 | bit 19 | bit 18 | bit 17 | bit 16 |
| s8 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| s16 | - | 15 | - | 14 | - | 13 | - | 12 | - | 11 | - | 10 | - | 9 | - | 8 |
| m16 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| m32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

Note:
It is prohibited to run CNN with the DMAOE register set to 32'h0000. At least channel 0 must be enabled by setting DMAOE register.

### U33.3.7 Statistics

#### (1) Channel statistics

In case of 8bit and 16bit processing, there exist 3 statistic data which can be generated on request for each channel, the channel max, channel min and channel sum, it is not available for mixed mode (m16 and m32) format settings.
For the channel max and channel min statistics, the maximal and minimal processed value of the channel are stored inside the upper half and lower half of the registers MAX_MIN_x. When the generation of this statistic is enabled by the bit CHANNEL_MAX_MIN of the register ARICSR, the value is each time replaced when a larger/smaller value is processed inside the channel. The register value can be reset or initialized to a special value by writing from CPU a new value inside. Additionally, in the MPRST register there is the MAX_MIN_16S bit for resetting MAX and MIN values for the s16-bit ARI format and the MAX_MIN_8S bit for resetting MAX and MIN values for the s8-bit ARI format. In case of 16bit ARI processing, MAX_MIN_x is holding the maximal and minimal value of channel x, in case of 8bit ARI processing, MAX_MIN_x is holding the maximal and minimal values of channel 2*x and channel 2*x+1.
The channel sum statistic is storing the sum of the processed values of a channel inside the register GLOB_SUM_x. When the generation of this statistic is enabled by the GLOB_SUM_MODE bit of the register ARICSR, the currently processed value of a channel is each time added to the already stored value for the channel. The register value can be reset by using the GLOB_SUM_RESET bit of the register MPRST.

#### (2) Performance Counter

As performance counter, the architecture supports 4 performance counter registers PERFCNT0 till PERFCNT3 including their control registers PERFCNT0CTRL till PERFCNT3CTRL. The number of cycles provided by these counters are not proven to be working correctly and are not accurate. Renesas does not offer any guarantee for the validity of the cycle counts provided by these counters. The users shall use these counters at their own risk.

The structure of the 4 performance counter control registers is same and is explained in the following for the register PERFCNT0CTRL. The PERFCNTNO bits of the register PERFCNT0CTRL are defining the entity for which the counter is working. For DMAI, 0 till 15 (8bit) or 0 till 7 (16bit) can be defined, while for DMA3DC and DMAO 0 till 31 (8bit, mixed 16bit, mixed 32bit) or 0 till 15 (16bit) can be defined. Alternatively, to count for all HW channels, the value 32 can be set. If CL or ARI is selected, only 0 can be specified. If PERFCNTNO is set to 64, all channels of all units (DMAI, DMA3DC, DMAO, ARI, CL) are selected.
Over the PERFCNTFUNC bits of the register PERFCNT0CTRL the function of the performance counter can be selected as shown inside Table U33.50.

**Table U33.50 Supported performance counter functions**

| PERFCNTFUNC value | PERFCNTFUNC function |
|---|---|
| H'0 | DMAI busy cycles |
| H'1 | DMAI stall cycles |
| H'2 | DMAO busy cycles |
| H'3 | DMAO stall cycles |
| H'4 | DMA3DC busy cycles |
| H'5 | DMA3DC stall cycles |
| H'6 | ARI busy cycles |
| H'7 | ARI stall cycles |
| H'8 | CL busy cycles |
| H'9 | CL stall cycles |

If bit 0 of PERFCNTFUNC is set to 1, the stall reason must be set in the PERFCNTSTRSN bits.
Table U33.51 lists the possible stall reasons to select.

**Table U33.51 Supported Measured Stall Reasons**

| | |
|---|---|
| Stall reason for DMAI, DMA3DC (when PERFCNT0FUNC is 1 or 5) | |
| H'2 | Stalls caused by wait as data source for data from DMAI/DMA3DC buffer. |
| Stall reason for DMAO (when PERFCNT0FUNC is 3) | |
| H'1 | Stalls caused by wait as data sink for data from output buffer. |
| H'2 | Stalls caused by wait as data source for data to memory. |
| Stall reason for ARI (when PERFCNT0FUNC is 7) | |
| H'0 | All stalls |
| H'1 | Stalls caused by wait as data sink for data from input buffer. |
| H'2 | Stalls caused by wait as data source for data to output buffer. |
| H'3 | Stalls caused by wait as data sink for data from 3DC buffer. |
| Stall reason for CL (when PERFCNT0FUNC is 9) | |
| H'1 | Stalls caused by the cache miss. |
| H'2 | Stalls caused by SYNCS. |
| H'3 | Stalls caused by the bus not being able to receive data requests to memory. |

Further, each performance counter can be enabled individually, and an overflow bit is available which is set if the counter overflows.

### (3) Register dump

To store statistics from CNN-IP (global sum, max and min) as well as performance counter values quickly to SPMC memory (only SPMC1 is available), additional to the transfer by CPU a DMA Register Dump unit is available, the DMARD unit. For storing the register values, only the start addresses in external memory (aligned to 512byte) must be specified and it must be guaranteed that at least 1 DMAO channel is enabled. Additionally, before setting the start bit, user must guarantee the register values are available. Table U33.52 shows the registers for which the register dump function is implemented, while Figure U33.44 is showing the structure of the dumped 512 bytes.

**Table U33.52 Supported registers for dump using DMARD**

| Offset address | Register name | Description |
|---|---|---|
| 0x0800 - 0x087C | GLOB_SUM_00 - 31 | Global sum of DMAO channel 0 - 31 |
| 0x0880 - 0x08BC | MAX_MIN_00 - 15 | MAX and MIN of DMAO channel 0-1 |
| 0x08D0 - 0x08DC | PERFCNT0 - 3 | Performance counter 0-3 |

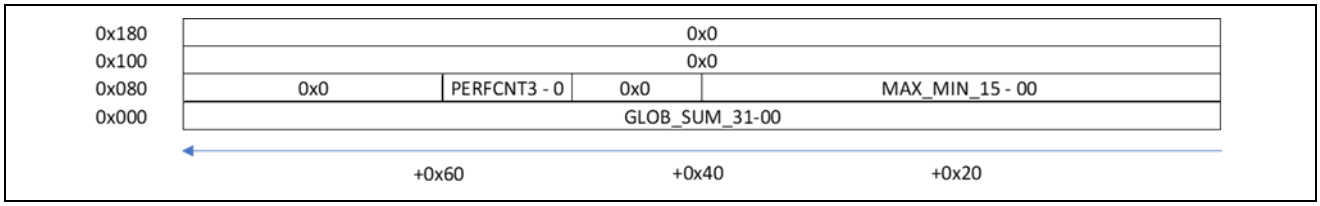



**Figure U33.44 Structure of the dumped 512 bytes**

The design shares the two busy bits SYNCS.DMAO and SYNCS.BUS between DMAO and DMARD. If the user wants to wait only till DMARD finishes the output of the data before starting the next layer, the user only has to wait until the busy bit SYNCS.DMAO becomes 0. If the user wants to confirm also that the write transaction has finished, the user also has to wait until the busy bit SYNCS.BUS becomes 0.

In the special case, where DMAO and DMARD are writing data out to different SPMC memories, e.g., first DMAO outputs to SPMC0 and then DMARD outputs to SPMC1, the user has to wait until the busy bit SYNCS.BUS becomes 0.

### U33.3.8 Other Information

#### (1) Status register

Status registers are available to store some CNN states to easily make them accessible from CPU. Some of them are only readable, some of them may have additional a possibility to trigger an interrupt to CPU. For the signals which can trigger an interrupt, an enable, mask, and clear bit are additionally provided. This is done to have the chance to disable the update inside the status register SR, to mask the interrupt triggering and to clear the signal status inside SR.

#### (2) WUP/SLP

The CNN unit supports the WUP/SLP functionality to wake-up other modules or to get woken-up by other modules after entering the sleep state. Over the SYNCC registers, the WUP/SLP connection network is configured. The WUP command wake-ups modules which are specified by the given parameter, while the SLP command defines by the given parameter the modules by which it wants to be woken-up after the transfer into the sleep state. Between 2 wake-up signals to the same module a SLP command must be scheduled for handshaking to the same one. Further, be sure to insert 8 or more non-WUP instructions between two WUP instructions in the CL.

#### (3) CL debugging

To use the CL break function, following setting is required.
1. Specify the break point address of command list. Make sure that break point is command address of command list.
2. Set CLBE bit in SRE register to 1 and if interrupt is required CLBM bit in SRM register to 0 to use CL break function.

3. Check the CLB status in SR register.
4. To continue execution after CL break, set the CLBC bit of SRC register to 1.

### (4) CL index addressing

CL index addressing is supported by adding an address offset provided by CL to a selectable index address from a set of internal registers (index registers, e.g., INDEX[0...7]). The index registers are write-able by CPU.

When using CL index addressing, only 13bits of internal register address is supported. The user also can use compatible mode 16bits of internal register address by setting V3U_COMP_MODE bit of V3U_COMP_MODE register to 1'b1 to disable CL index addressing. The INDEX[0] is fixed set to 0 and cannot be changed.

### (5) Double Buffering

To support overlapping execution for multiple layers, double buffering of DMAI, DMA3DC and DMAO and ARI parameters are supported for all registers except DMA start registers and all virtual registers (refer Table U33.3). So, the next layer can be setup before the previous layer has finished.

### (6) Bandwidth usage

All DMAs support a bandwidth of 512byte/cycle when accessing SPMC0/1. DMAI/DMAW can access SPMC0 only. DMA3DC can access SPMC1 only. DMAO can access both SPMC0/1. If 2 DMA want to access at the same time to the same SPMC, 1 DMA has to wait while the other DMA can access, so the access is done interleaved in time.

### (7) Automatic weight loading

Next to using the DMAW for loading the weights, the CLW instruction can be used when operating with 2x3x3 kernel or 1x5x5 kernel to load predefined values for special purposes. Therefore, the parameter N of the CLW instruction must be set accordingly. Possible values of N with their weight loading purpose are shown in Table U33.54. In the following sub chapters, it will be shown how the non-zero weight elements are set for each selection of N, zero values are blank. It is to mention, that the shown channels in the figures (in-ch and out-ch) present 8bit/16bit HW channels.

#### (a) Identity matrix

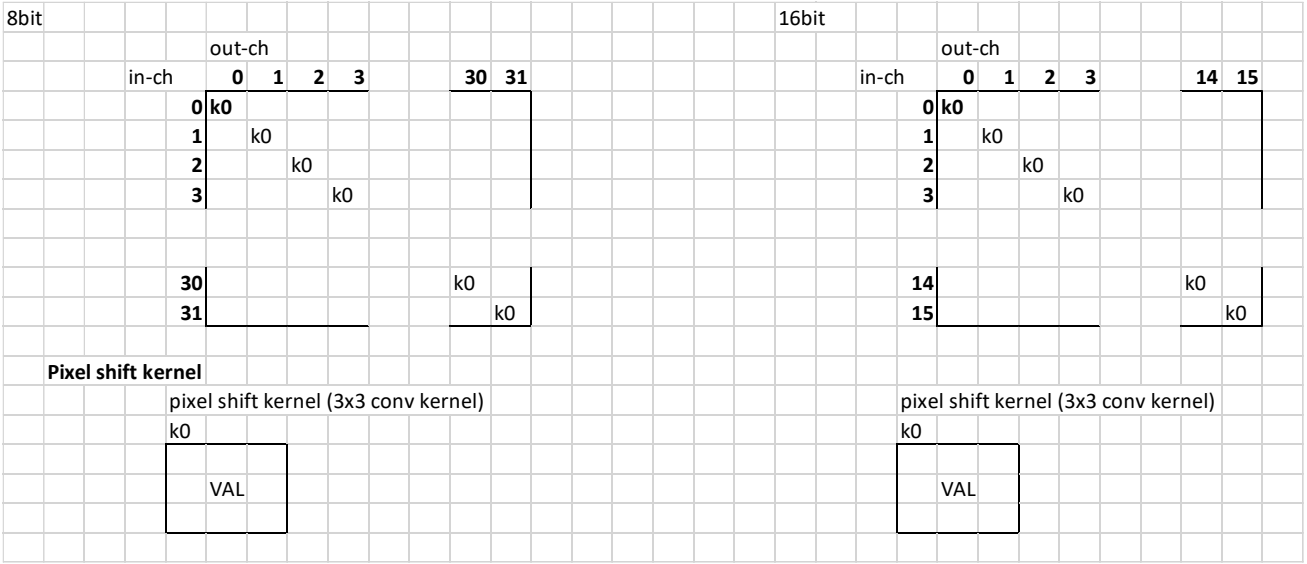



**Figure U33.45 Weight loading for identity matrix**

**Figure U33.45** shows the setting of the non-zero weight elements in case of setting N to 23 (8bit) or 24 (16bit). For the case of the identity matrix, the value specified in the VAL field of the CLW instruction is used as weight.

### (b) ELTWISE MAX reorder

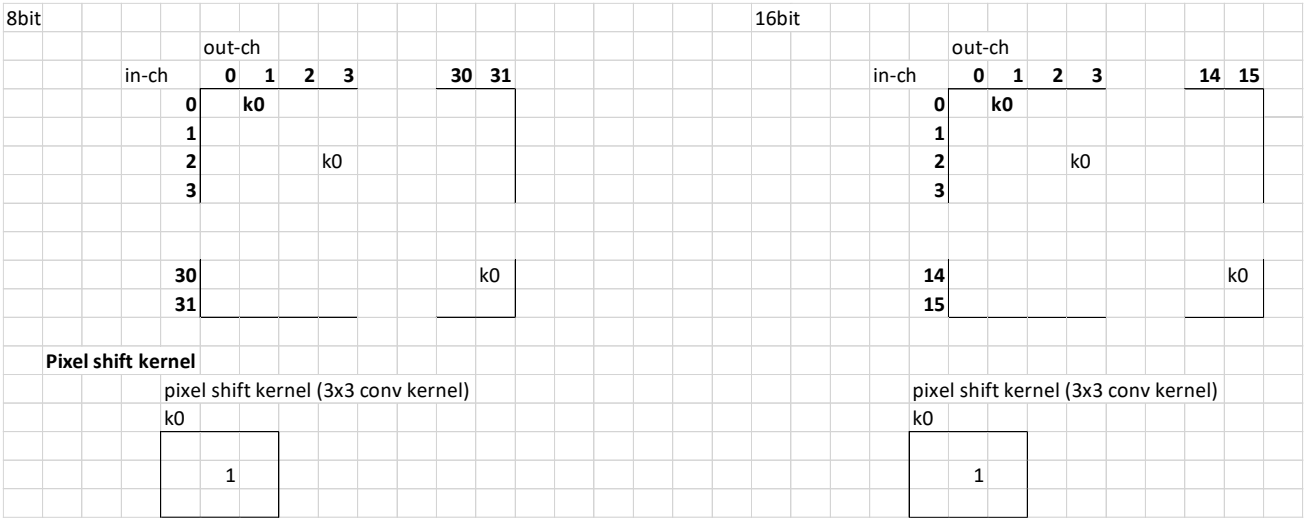



**Figure U33.46 Weight loading for ELTWISE MAX reorder**

**Figure U33.46** shows the setting of the non-zero weight elements in case of setting N to 1 (8bit) or 2 (16bit). For the case of ELTWISE MAX reorder, the fixed value 1 is used as weight.

### (c) ELTWISE MAX through

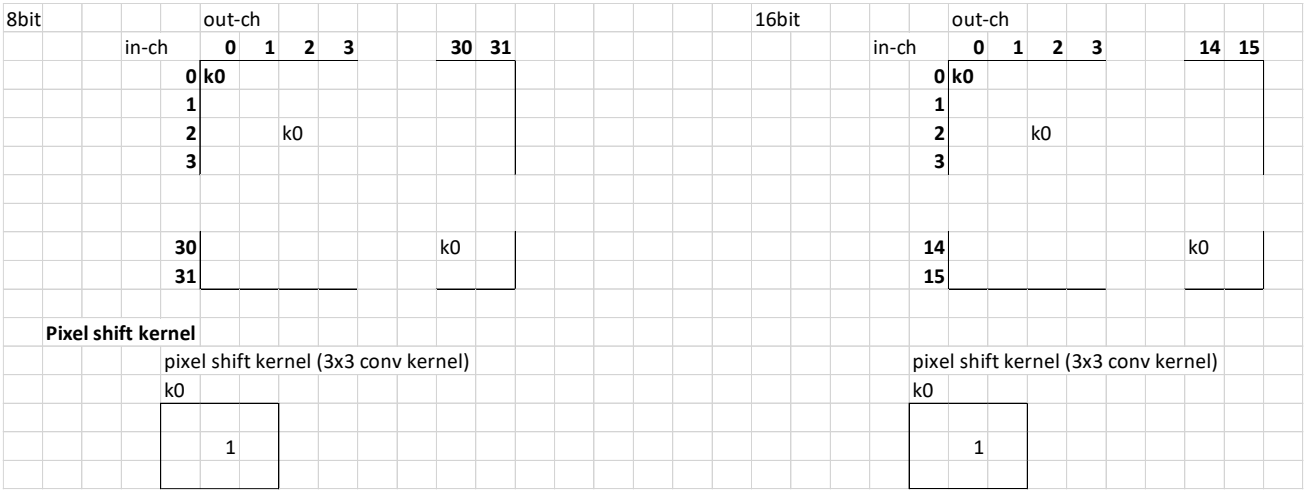



**Figure U33.47 Weight loading for ELTWISE MAX through**

**Figure U33.47** shows the setting of the non-zero weight elements in case of setting N to 3 (8bit) or 4 (16bit). For the case of ELTWISE MAX through, the fixed value 1 is used as weight.

### (d) Pool 3x3 stride2 and pool 2x2 stride 1

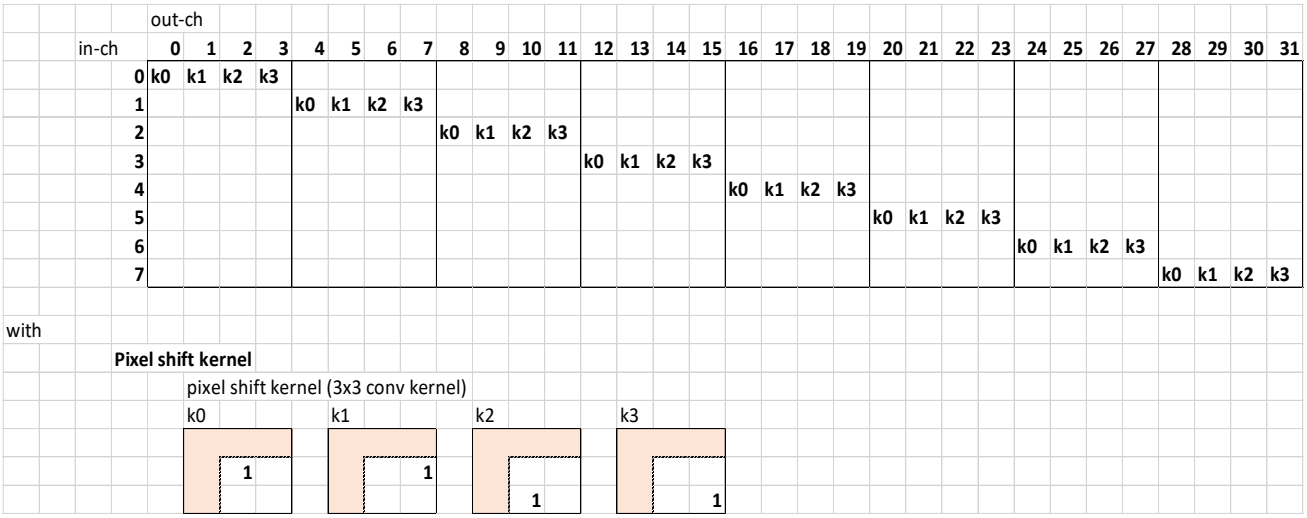



**Figure U33.48 Weight setting for 8bit pool 3x3 stride2 and pool 2x2 stride 1**

**Figure U33.48** shows the setting of the non-zero weight elements in case of setting N to 5. In this case, the weights are set for emulating 8bit overlapped pooling with 3x3 pooling kernel and stride 2 or 2x2 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

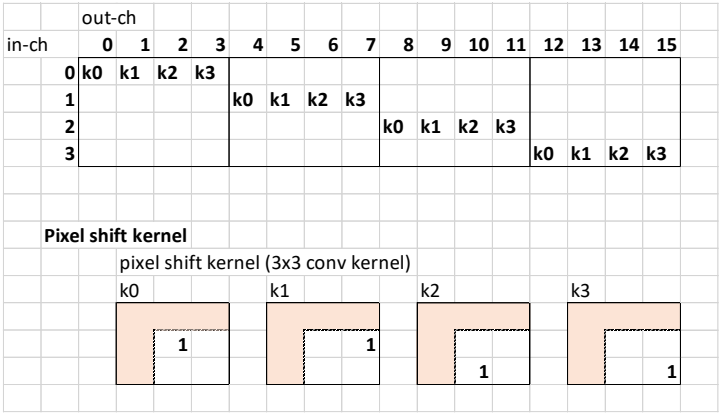



**Figure U33.49 Weight setting for 16bit pool 3x3 stride2 and pool 2x2 stride 1**

**Figure U33.49** shows the setting of the non-zero weight elements in case of setting N to 6. In this case, the weights are set for emulating 16bit overlapped pooling with 3x3 pooling kernel and stride 2 or 2x2 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

**(e) Pool 3x1 stride 1**

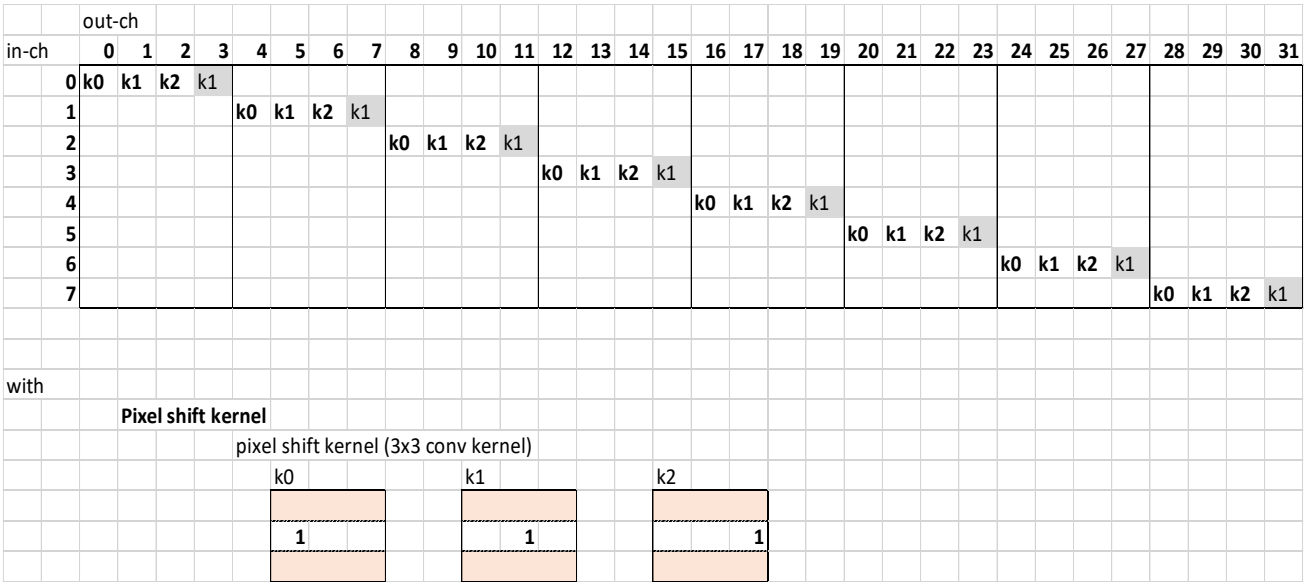



**Figure U33.50 Weight setting for 8bit pool 3x1 stride 1**

**Figure U33.50** shows the setting of the non-zero weight elements in case of setting N to 7. In this case, the weights are set for emulating 8bit overlapped pooling with 3x1 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

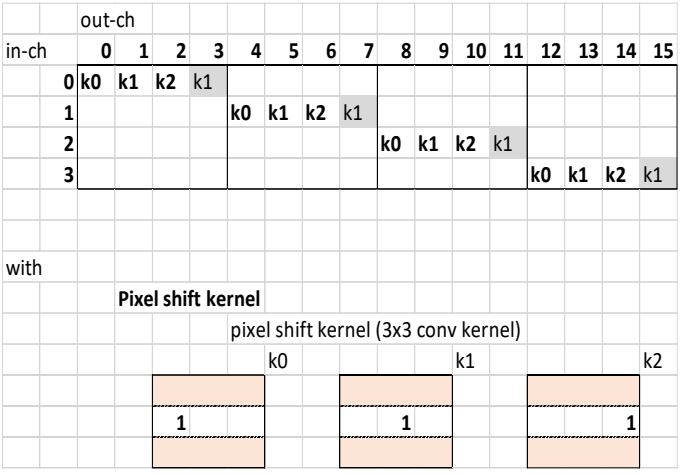



**Figure U33.51 Weight setting for 16bit pool 3x1 stride 1**

**Figure U33.51** shows the setting of the non-zero weight elements in case of setting N to 8. In this case, the weights are set for emulating 16bit overlapped pooling with 3x1 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

**(f) Pool 1x3 stride 1**

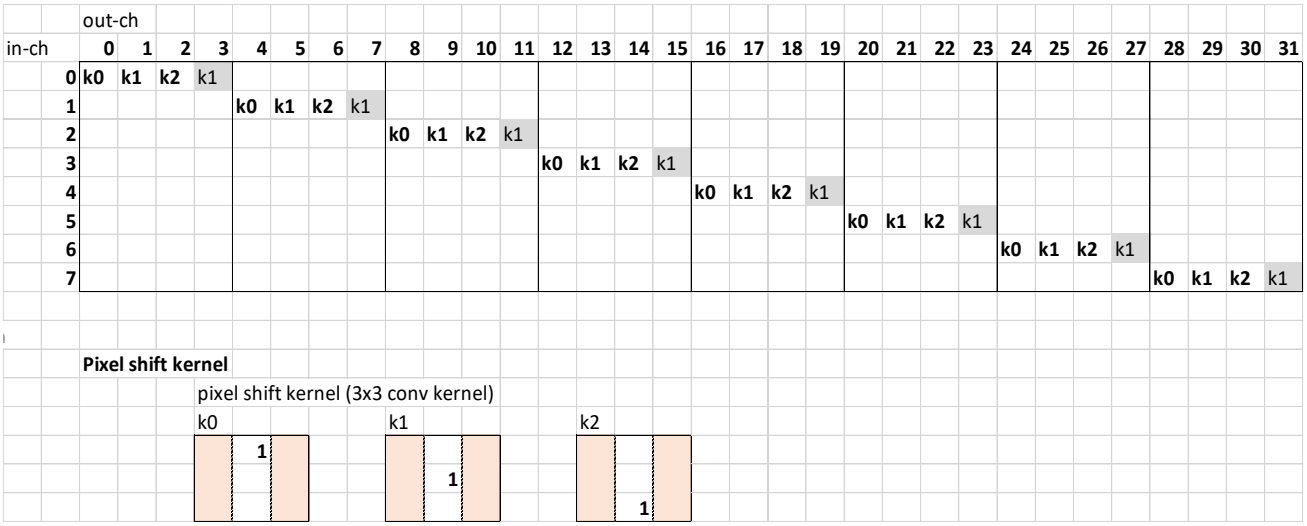



**Figure U33.52 Weight setting for 8bit pool 1x3 stride 1**

**Figure U33.52** shows the setting of the non-zero weight elements in case of setting N to 9. In this case, the weights are set for emulating 8bit overlapped pooling with 1x3 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

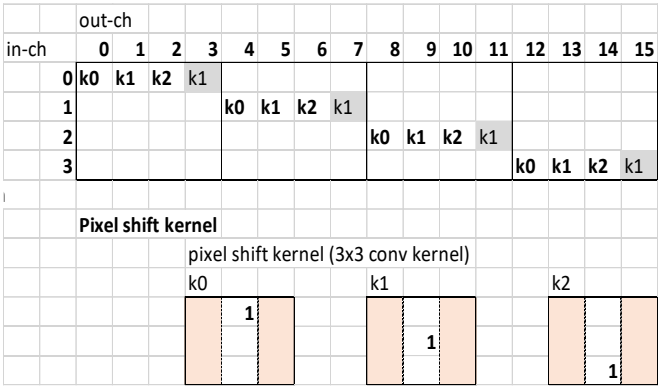



**Figure U33.53 Weight setting for 16bit pool 1x3 stride 1**

**Figure U33.53** shows the setting of the non-zero weight elements in case of setting N to 10. In this case, the weights are set for emulating 16bit overlapped pooling with 1x3 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

### (g) Pool 2x1 stride 1

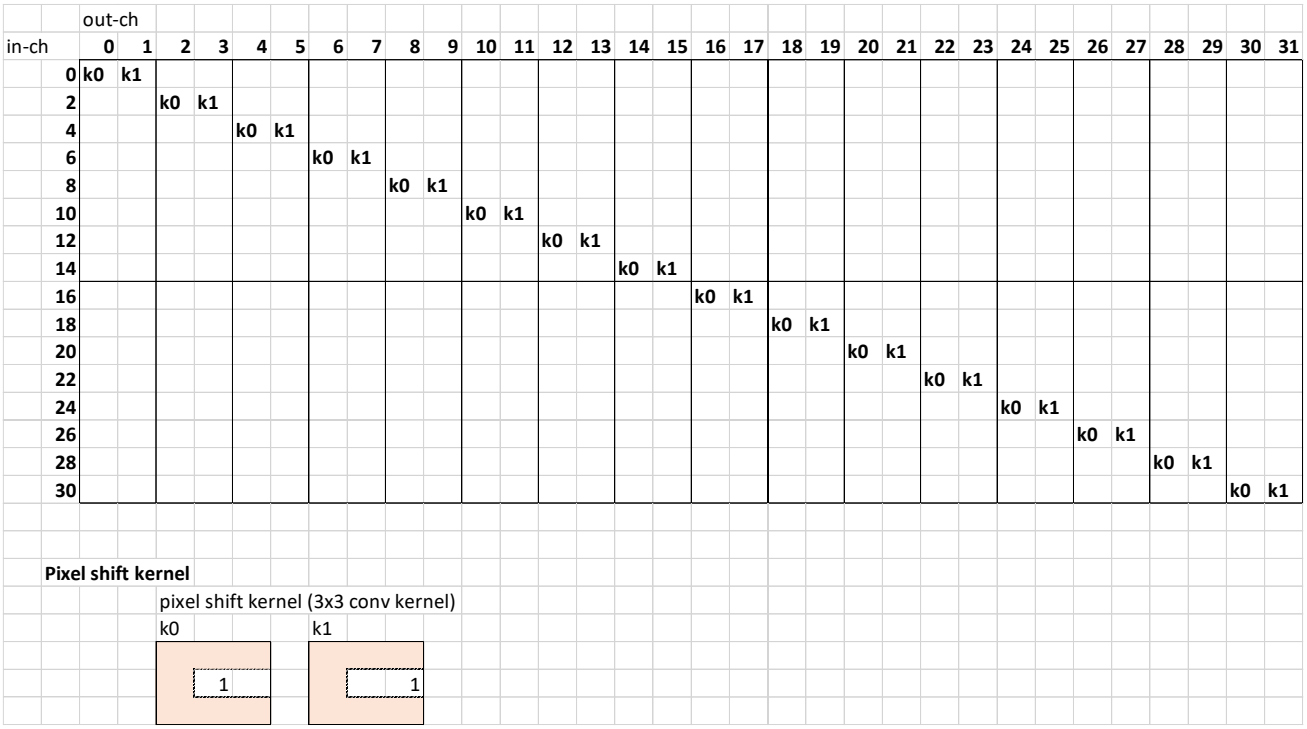



**Figure U33.54 Weight setting for 8bit pool 2x1 stride 1**

**Figure U33.54** shows the setting of the non-zero weight elements in case of setting N to 11. In this case, the weights are set for emulating 8bit overlapped pooling with 2x1 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

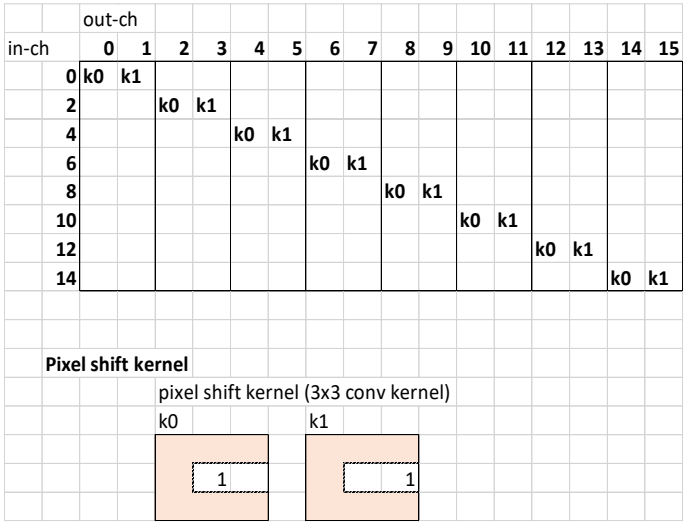



**Figure U33.55 Weight setting for 16bit pool 2x1 stride 1**

**Figure U33.55** shows the setting of the non-zero weight elements in case of setting N to 12. In this case, the weights are set for emulating 16bit overlapped pooling with 2x1 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

### (h) Pool 1x2 stride 1

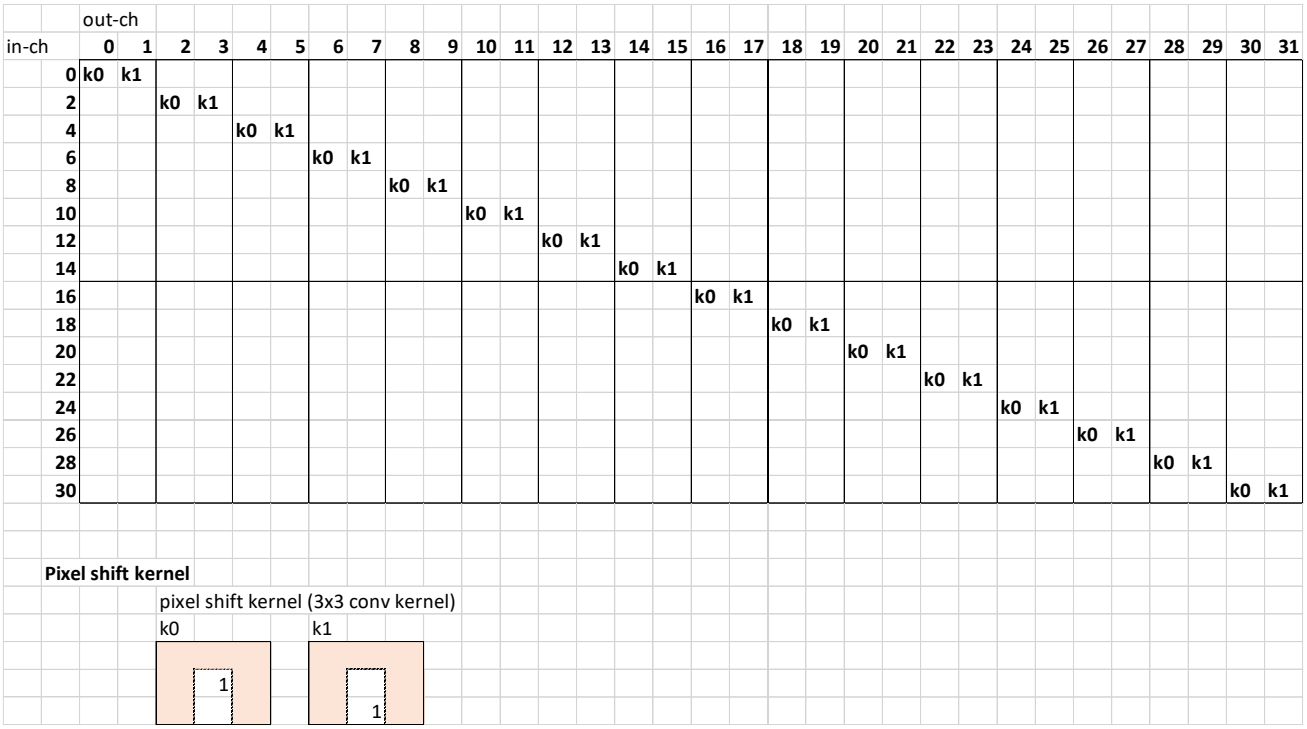



**Figure U33.56 Weight setting for 8bit pool 1x2 stride 1**

**Figure U33.56** shows the setting of the non-zero weight elements in case of setting N to 13. In this case, the weights are set for emulating 8bit overlapped pooling with 1x2 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

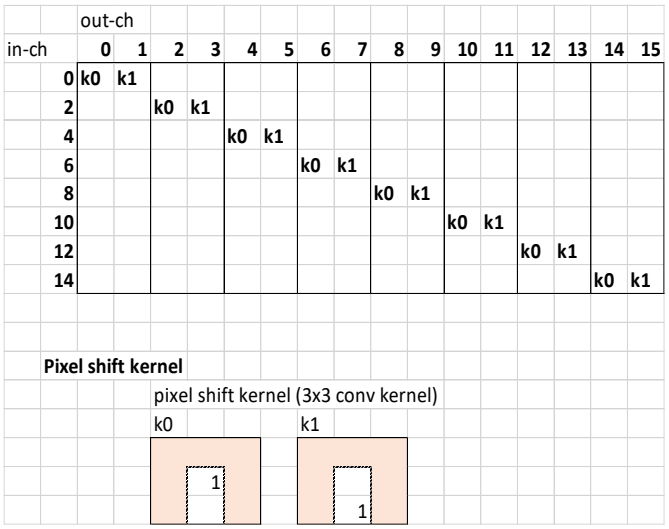



**Figure U33.57 Weight setting for 16bit pool 1x2 stride 1**

**Figure U33.57** shows the setting of the non-zero weight elements in case of setting N to 14. In this case, the weights are set for emulating 16bit overlapped pooling with 1x2 pooling kernel and stride 1. For this case, the fixed value 1 is used as weight.

**(i) Bilinear filter 2x2**

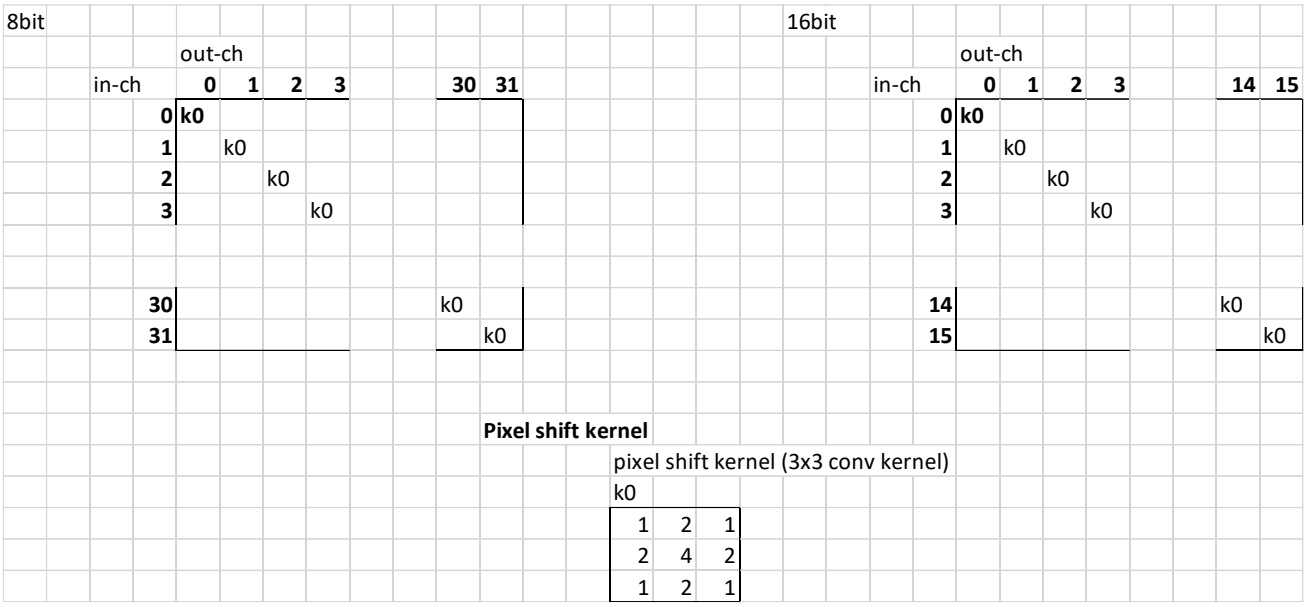



**Figure U33.58 Weight setting for bilinear filter 2x2**

**Figure U33.58** shows the setting of the non-zero weight elements in case of setting N to 15 (8bit) or 16 (16bit). In this case, the weights are set for processing the bilinear filter 2x2.

**(j) Bilinear filter 3x3**

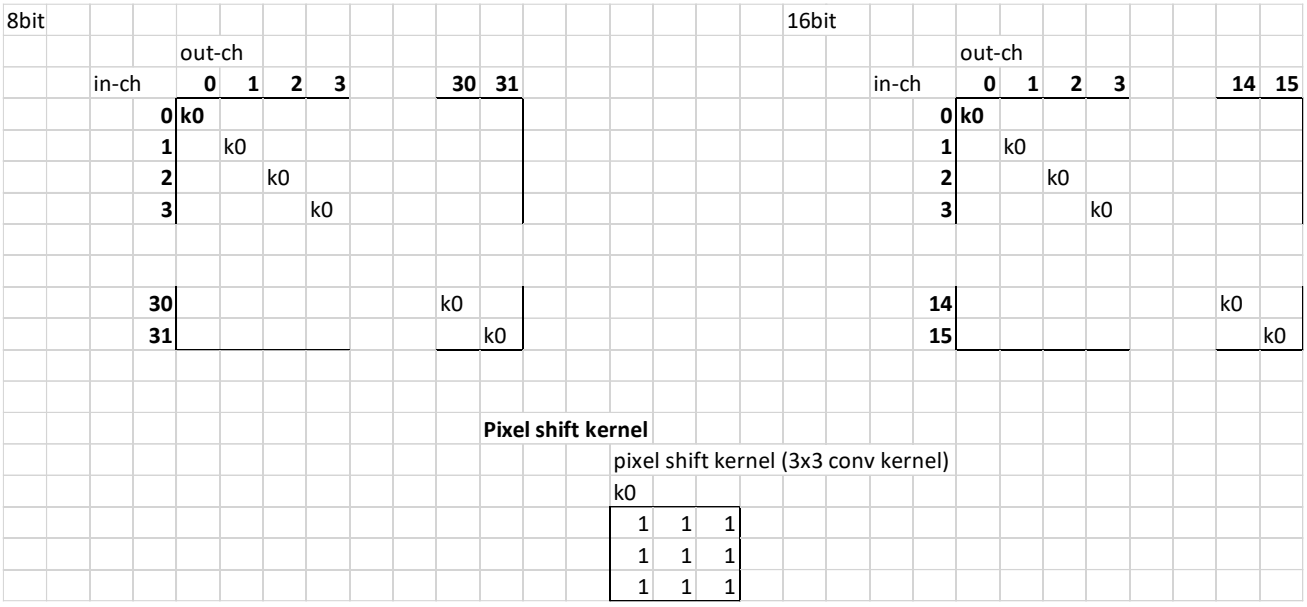



**Figure U33.59 Weight setting for bilinear filter 3x3**

**Figure U33.59** shows the setting of the non-zero weight elements in case of setting N to 17 (8bit) or 18 (16bit). In this case, the weights are set for processing the bilinear filter 3x3.

**(k) Bilinear filter 4x4**

8bit

| in-ch \ out-ch | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | k0 | | | | | | | |
| 1 | | k0 | | | | | | |
| 4 | | | k0 | | | | | |
| 5 | | | | k0 | | | | |
| 8 | | | | | k0 | | | |
| 9 | | | | | | k0 | | |
| 12 | | | | | | | k0 | |
| 13 | | | | | | | | k0 |

16bit

| in-ch \ out-ch | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | k0 | | | |
| 2 | | k0 | | |
| 4 | | | k0 | |
| 6 | | | | k0 |

**Pixel shift kernel**

pixel shift kernel (5x5 conv kernel)

k0

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 2 | 1 |
| 2 | 4 | 4 | 4 | 2 |
| 2 | 4 | 4 | 4 | 2 |
| 2 | 4 | 4 | 4 | 2 |
| 1 | 2 | 2 | 2 | 1 |

**Figure U33.60 Weight setting for bilinear filter 4x4**

**Figure U33.60** shows the setting of the non-zero weight elements in case of setting N to 19 (8bit) or 20 (16bit). In this case, the weights are set for processing the bilinear filter 4x4.

**(l) Bilinear filter 5x5**

8bit

| in-ch \ out-ch | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | k0 | | | | | | | |
| 1 | | k0 | | | | | | |
| 4 | | | k0 | | | | | |
| 5 | | | | k0 | | | | |
| 8 | | | | | k0 | | | |
| 9 | | | | | | k0 | | |
| 12 | | | | | | | k0 | |
| 13 | | | | | | | | k0 |

16bit

| in-ch \ out-ch | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | k0 | | | |
| 2 | | k0 | | |
| 4 | | | k0 | |
| 6 | | | | k0 |

**Pixel shift kernel**

pixel shift kernel (5x5 conv kernel)

k0

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**Figure U33.61 Weight setting for bilinear filter 5x5**

**Figure U33.61** shows the setting of the non-zero weight elements in case of setting N to 21 (8bit) or 22 (16bit). In this case, the weights are set for processing the bilinear filter 5x5.

### U33.3.9 CNN-IP processing flow

1. Complete the necessary settings for all DMAs and ARI for the next tile, execute SLP command, and only execute the start commands after WUP.
2. To start the next tile, if there is a weight transfer, please execute the SYNCS command with CNV bit cleared before starting DMAW and after executing SYNCS command with DMAW bit cleared before starting DMAI. If there is no weight transfer, please execute the SYNCS command with DMAI bit cleared before starting DMAI.
3. "WPR" command must be used for DMAWS/DMAIS/SLSP[*1], while “WPRNBSY” command should be used for other registers. The register setting cycles for each DMA transfer can be masked by setting the registers for each DMA after SLSP start and writing only to each DMA start registers after SYNCS. (Restrictions 4,5,6 must be observed.)"
4. When DMA3DC is not used, set DMA3DCE to 0 before writing to DMAIS register.
5. Except for ELTWISE MULT processing, if for a channel DMA3DC enable is set, ARI enable must be also set.
6. Comply with this restriction: ARI_LEN after XY pooling >=DMAOL.
7. When changing the setting for a DMAO channel from 1 to 0, be sure to set the BUS bit (bit 6) of the SYNCS command to 0 (not masked). If BUS busy is masked and DMAOE is switched for a channel from 1 to 0, the channel clock may be switched off while still bus accesses happen. Other setting, please refer to U33.4.7(b) Note .
8. All register settings (accept register settings described in code sequences in U33.4.1(b) and U33.5.5(8)) must be done between SLSP start and SYNCS DMAI or if at the beginning no SYNCS DMAI is available, before DMAW/DMAI start, whichever is earlier, after, only write to the register to start DMAx processings; no register setting is allowed between DMAI start and SLSP.
   a. If there is a weight transfer with DMAW, the preferred setup flow is:
   (Register setting)→SYNCS(DMAI/CNV)→WPR(DMAWS)→SYNCS(DMAW/DMA3DC/DMAO/ ARI/BUS)→WPR(DMAIS)→WPRNBSY(DMA3DCS ) → WPRNBSY(DMAOS)→ WPR(SLSP).

b. If there is a weight setting by CLW, the preferred setup flow is:
(Register setting)→SYNCS(DMAI/CNV)→CLW→WPR(DMAIS)→
SYNCS(DMA3DC/DMAO/ARI/BUS)→WPRNBSY(DMA3DCS ) →WPRNBSY(DMAOS)→ WPR(SLSP).
c. If there is no weight transfer, the preferred setup flow is:
(Register setting) → SYNCS(DMAI) → WPR(DMAIS) → SYNCS(DMA3DC/DMAO/ARI/BUS) →
WPRNBSY(DMA3DCS ) →WPRNBSY(DMAOS) → WPR(SLSP)."

9. In planar transfer mode, if only some DMAI channels must be enabled, set them continuously, not intermittently (to enable 6 of the 16 channels, e.g., set DMAIE to 0x003F).
10. In case a CLW is needed before weight loading by DMAWS (case a), please put it directly before writing to DMAWS.

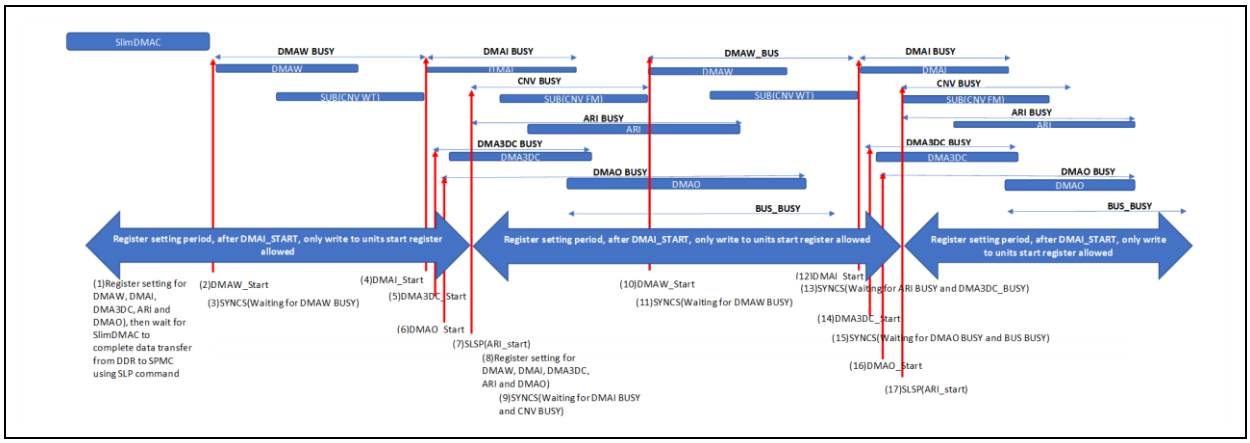



**Figure U33.62 Example CL when weight loading over DMAW**

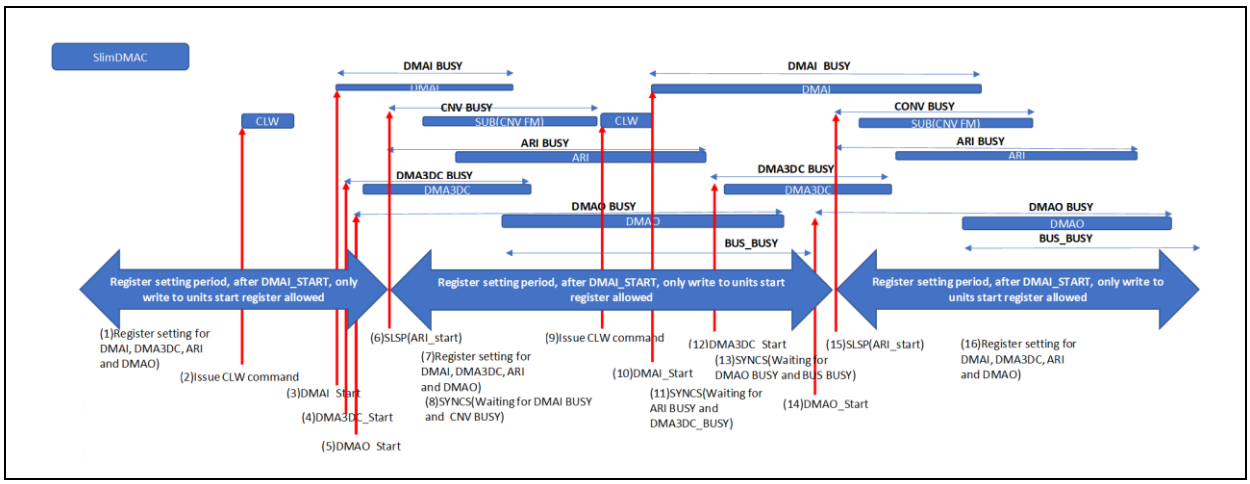



**Figure U33.63 Example CL when weight loading over CLW**

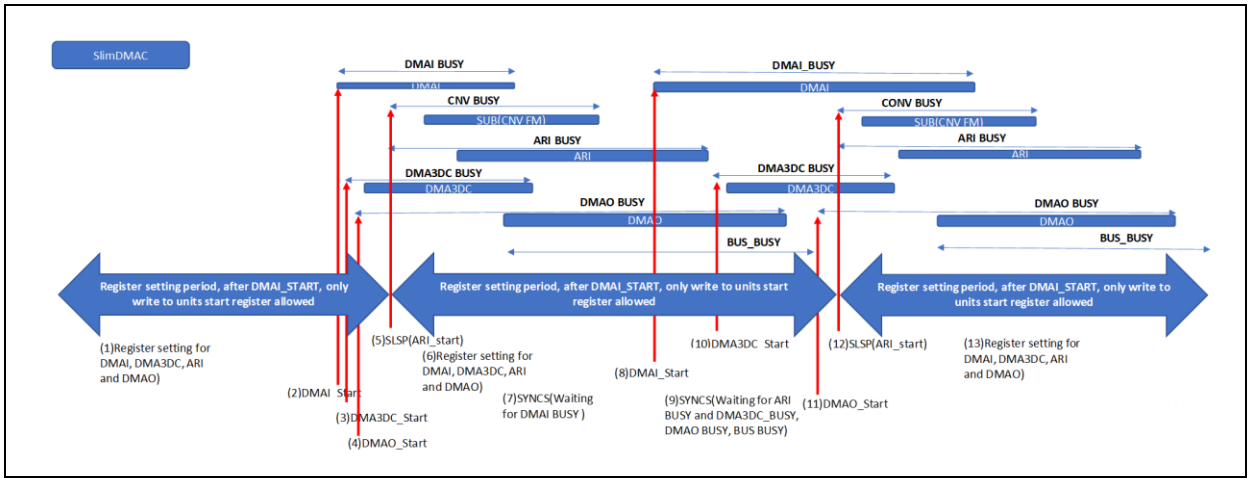



**Figure U33.64 Example CL when no weight loading**

Note 1: In case of SLSP register, if the following special command sequence (with or without writing to DMA3DCS register) is used, SLSP can be also written by using “WPRNBSY”:

```
0x8b0000xx      // SYNCS
0x55140501      // WPRNBSY DMA3DCS,1 (optional)
0x00000001      // data word (optional)
0x56180701      // WPRNBSY DMAOS,1
0xxxxxxxxx      // data word
0x56004401      // WPRNBSY SLSP,1
0x00000003      // data word
```

## U33.4 Command List

It is prohibited to use instructions not listed in this section.

**Table U33.53 Instruction list**

| Instruction | OP CODE | Description |
|---|---|---|
| CLW | H'4C | Clear weight buffers |
| WPR | H'55 | Writes a long value to the register |
| WPRNBSY | H'56 | Writes a long value to the register, does not wait till register value has been written |
| NOP | H'80 | No Operation |
| INT | H'81 | Generates an interrupt and waits for the release of the interrupt bit. |
| TRAP | H'88 | End of CL execution |
| SYNCS | H'8B | Waits for completion of all sub tasks of CNN unit. |
| JUMP | H'90 | Jumps to an address. |
| GOSUB | H'91 | Jumps to a sub routine. |
| RET | H'99 | Returns from a sub routine. |
| WUP | H'A0 | Activates an external unit. |
| SLP | H'A1 | Waits for an activation request from an external unit. |

### U33.4.1 CLW

**(a) Usage**

CLW

**(b) Function**

The CLW instruction clears all weight entries of weight buffer. Please refer to the Table U33.54 for the details of N setting. Regarding the field VAL used if N defined as 23(B'10111) or 24(B'11000), signed 16bit is used for 16bit processing, lower 8bit is used as signed 8bit for 8bit processing. For more details, refer to U33.3.8(7) .
The CLW instruction must be used always surrounded by 2 WPR instructions, which are enabling and disabling the usage of the CLW instruction. For enabling the usage of the CLW instruction, WPR should be used, while for disabling the usage of the CLW instruction, WPRNBSY can also be used. A correct usage sequence looks as follows:

```
0x55009401      ; enabling CLW instruction
0x00000001
0x4C000000      ; example CLW instruction with reset to all zero
0x56009401      ; disabling CLW instruction
0x00000000
```

**(c) Instruction Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0100_1100 (H'4C) | | | | | | | | Reserved (all 0) | | | N | | | | | VAL | | | | | | | | | | | | | | | |

**Table U33.54 Possible setting on N filed of CLW instruction**

| | Description |
|---|---|
| Setting in N field | Clear all weight entries of weight buffer<br>00000: Reset all with 0<br>00001: Reset for ELTWISE MAX 8bit reorder processing<br>00010: Reset for ELTWISE MAX 16bit reorder processing<br>00011: Reset for ELTWISE MAX 8bit through processing<br>00100: Reset for ELTWISE MAX 16bit through processing<br>00101: Reset for OverlapPool 4to1 3x3_2 2x2_1 8bit processing<br>00110: Reset for OverlapPool 4to1 3x3_2 2x2_1 16bit processing<br>00111: Reset for OverlapPool 4to1 3x1_1 8bit processing<br>01000: Reset for- OverlapPool 4to1 3x1_1 16bit processing<br>01001: Reset for OverlapPool 4to1 1x3_1 8bit processing<br>01010: Reset for OverlapPool 4to1 1x3_1 16bit processing<br>01011: Reset for OverlapPool 2to1 2x1_1 8bit processing<br>01100: Reset for OverlapPool 2to1 2x1_1 16bit processing<br>01101: Reset for OverlapPool 2to1 1x2_1 8bit processing<br>01110: Reset for OverlapPool 2to1 1x2_1 16bit processing<br>01111: Reset for Bilinear 2x2 8bit processing<br>10000: Reset for Bilinear 2x2 16bit processing<br>10001: Reset for Bilinear 3x3 8bit processing<br>10010: Reset for Bilinear 3x3 16bit processing<br>10011: Reset for Bilinear 4x4 8bit processing<br>10100: Reset for Bilinear 4x4 16bit processing |

| | |
|---|---|
| | 10101: Reset for Bilinear 5x5 8bit processing<br>10110: Reset for Bilinear 5x5 8bit processing<br>10111: Reset for Identity matrix with signed 8bit const value VAL for 8bit processing<br>11000: Reset for Identity matrix with signed 16bit const value VAL for 16bit processing |

### U33.4.2 WPR

#### (a) Usage

V3U_COMP_MODE == 0: WPR INDEX,ADD,N
V3U_COMP_MODE == 1: WPR ADD,N

#### (b) Function

The WPR instruction writes the number of values specified in the N field to N consecutive addresses starting from the address specified in the ADD field (data is in order from DATA 0 to DATA N - 1). The ADD field is set in 4byte units. Setting 0 in the N field is prohibited.
If V3U_COMP_MODE bit of V3U_COMP_MODE register is set to 0, the INDEX field is replacing the upper 3 address bits. The INDEX field specifies the number of the index register INDEXN. The value from the index register will be added to the offset specified in the DATA fields of this command. The range of INDEX is (0 to 7). It is advised to use INDEX0 in case of writing values to non-address registers while using INDEX1 till INDEX7 for address registers.

#### (c) Instruction Format

V3U_COMP_MODE == 0:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0101_0101 (H'55) | | | | | | | | INDEX | | | ADD | | | | | | | | | | | | | N(>=1) | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA N-1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

V3U_COMP_MODE == 1:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0101_0101 (H'55) | | | | | | | | ADD | | | | | | | | | | | | | | | | N(>=1) | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA N-1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U33.4.3 WPRNBSY

#### (a) Usage

V3U_COMP_MODE == 0: WPRNBSY INDEX,ADD,N
V3U_COMP_MODE == 1: WPRNBSY ADD,N

#### (b) Function

The WPRNBSY instruction writes the number of values specified in the N field to N consecutive addresses starting from the address specified in the ADD field (data is in order from DATA 0 to DATA N - 1). The ADD field is set in 4byte units.
Setting 0 in the N field is prohibited. In contrast to the WPR instruction, the WPRNBSY instruction does not wait till the previous instruction is finished. Therefore, in case dependencies exist to previously executed commands, WPRNBSY should not be used. Such dependency could be e.g., writing to the SLSP register to start layer processing which should only start after all parameter registers received their parameter values, here normal WPR instruction would be required.
If V3U_COMP_MODE bit of V3U_COMP_MODE register is set to 0, the INDEX field is replacing the upper 3 address bits. The INDEX field specifies the number of the index register INDEXN. The value from the index register will be added to the offset specified in the DATA fields of this command. The range of INDEX is (0 to 7). It is advised to use INDEX0 in case of writing values to non-address registers while using INDEX1 till INDEX7 for address registers.

#### (c) Instruction Format

V3U_COMP_MODE == 0:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0101_0101 (H'56) | | | | | | | | INDEX | | | ADD | | | | | | | | | | | | | N(>=1) | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

…

| DATA N-1 |
|---|

V3U_COMP_MODE == 1:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=0101_0101 (H'56) | | | | | | | | ADD | | | | | | | | | | | | | | | | N(>=1) | | | | | | | |
| DATA 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DATA 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

…

| DATA N-1 |
|---|

### U33.4.4 NOP

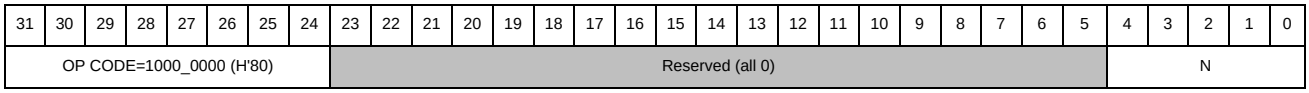
**(a) Usage**

NOP N

**(b) Function**

The NOP instruction causes the CNN unit to do nothing for the number of clock cycles specified in the N field before execution of the next instruction. Values up to 16 are supported.

**(c) Instruction Format**



### U33.4.5 INT

**(a) Usage**

INT CODE

**(b) Function**

Only when the INTE bit of the SRE register is 1, it is reflected in the INT bit of the SR register.
Regardless of the value of the INTE bit in the SRE register, the value specified in the CODE field is reflected in the INTCODE bits of the SR register.
While the INT bit in the SR register is 1, subsequent instructions are not started.

**(c) Instruction Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_0001 (H'81) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | CODE | | | | | | | |

### U33.4.6 TRAP

**(a) Usage**

TRAP CODE

**(b) Function**

TRAP terminates execution of the command list. Only when the TRAE bit of the SRE register is 1, it is reflected in the TRA bit of the SR register.
However, regardless of the value of the TRAE bit in the SRE register, the value specified in the CODE field is reflected in TRAPCODE bits of the SR register.

**(c) Instruction Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_1000 (H'88) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | CODE | | | | | | | |

### U33.4.7 SYNCS

**(a) Usage**

SYNCS N

**(b) Function**

If there is a CNN operation currently being executed, this instruction waits for completion of the execution of that CNN operation. After the CNN operation has been finished, the subsequent instruction is started.
If there is no CNN operation being executed, it is the same as NOP (the following instruction is started).
With the command, parameters can be specified. With the DMAW, DMAI, DMA3DC, ARI, DMAO and BUS bits, the module for which not to wait can be masked, multiple masks are supported. By default, all bits are set to 0, so no module is masked.
With the DMAW bit, the check whether DMAW processing has finished, can be masked.
With the DMAI bit, the check whether DMAI processing has finished, can be masked.
With the DMA3DC bit, the check whether DMA3DC processing has finished, can be masked.
With the DMAO bit, the check whether DMAO processing has finished, can be masked.

With the ARI bit, the check whether convolution and ARI processing has finished, can be masked.
With the CNV bit, the check whether convolution has finished, can be masked. After confirming the completion of CNV, the weight setting by DMAW or CLW command for the next FM process can be started without waiting for the completion of ARI.

With the BUS bit, the check whether the data transfer on the bus has finished, can be masked.
If DMA3DC/DMAI is reading in the next run the data the DMAO is in the current run writing out, the user must take care that preloading DMA3DC/DMAI data is not overtaking the data write out from DMAO by e.g., not masking the check whether the data transfer on the bus has finished.
If in the next run the data is not written to the same SPMC memory, so a change from SPMC0 to SPMC1 or from SPMC1 to SPMC0 happens, the bus bit must be checked additionally before starting the write out to the changed SPMC memory.
Note:
(1) If DMAI on the next layer reads the data that DMAO has written to SPMC0, wait for DMAO completion with both BUS (bit6) and DMAO (bit4) of the SYNCS instruction enabled.
(2) If DMA3DC on the next layer reads the data that DMAO has written to SPMC1, wait for DMAO completion with both BUS (bit6) and DMAO (bit4) of the SYNCS instruction enabled.
(3) If DMAO writes to SPMC0 in this layer and DMAO writes to SPMC1 in the next layer, or if DMAO writes to SPMC1 in this layer and DMAO writes to SPMC0 in the next layer, wait for DMAO completion with both BUS (bit6) and DMAO (bit4) of the SYNCS instruction enabled.
(4) When changing the setting for a DMAO channel from 1 to 0, be sure to set the BUS bit (bit 6) of the SYNCS command to 0 (not masked). If BUS busy is masked and DMAOE is switched for a channel from 1 to 0, the channel clock may be switched off while still bus accesses happen.
(5) If using in this layer DMAO in packed mode and in the next layer in planar fast mode and if not enabling channel 0,8,16 or 24 in the next layer, make sure to wait also for BUS busy before starting DMAO processing of next layer.

**(c) Instruction Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1000_1011 (H'8B) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | BUS | CNV | DMAO | ARI | DMA3DC | DMAI | DMAW |

### U33.4.8 JUMP

#### (a) Usage

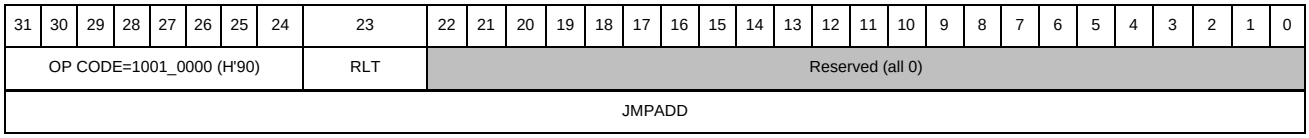
JUMP RLT

#### (b) Function

This is the command to jump to the specified address.

It allows jumping to the address specified in JMPADD field. Setting 1 to RLT field uses JMPADD as relative address to JUMP command address and setting 0 in RLT field uses JMPADD as absolute address of the command list. Specify 4byte unit command address in the JMPADD field.

#### (c) Instruction Format



### U33.4.9 GOSUB

#### (a) Usage

GOSUB RLT, CON

#### (b) Function

This is the command to save current CL address to the hardware buffer and jump to the specified address.
It allows jumping to the address specified in GSADD field and returning to the address next to the GOSUB command by RET command. Specify 4byte unit command address in the GSADD field, nesting of up to 4 level is supported. Setting 1 to RLT field uses GSADD as relative address to GOSUB command address and setting 0 in RLT field uses GSADD as absolute address of the command list. Setting 1 in CON field executed GOSUB conditionally and setting 0 in the CON field executed GOSUB unconditionally. Setting 1 in CON field and 0 in CLCNDGSBR bit of CLCNDGSBR register changes GOSUB command into NOP command. After the CLCNDGSBR register and used index register are set, a SYNCS command should be added before using the GOSUB command.

#### (c) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1001_0001 (H'91) | | | | | | | | RLT | CON | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | |
| GSADD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### U33.4.10 RET

#### (a) Usage

RET

#### (b) Function

The RET instruction returns to the address stored by the GOSUB instruction.

#### (c) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1001_1001 (H'99) | | | | | | | | Reserved (all 0) | | | | | | | | | | | | | | | | | | | | | | | |

### U33.4.11 WUP

#### (a) Usage

WUP CN

#### (b) Function

With this instruction, a WUP signal is sent to other modules. A WUP signal is issued to the modules defined by SYNCCm (m = 0 to 15) signals by setting bit 'm' of CN enable field to 1.
Be sure to insert 8 or more instructions other than the WUP instruction between two WUP instructions in the CL.

#### (c) Instruction Format

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP CODE=1010_0000 (H'A0) | | | | | | | | Reserved (all 0) | | | | | | | | CN | | | | | | | | | | | | | | | |

### U33.4.12 SLP

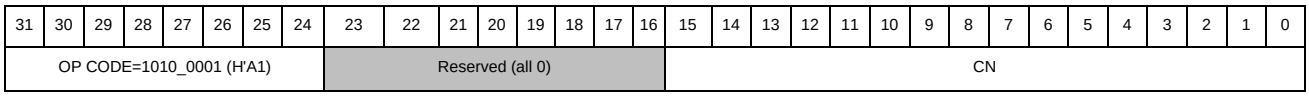
#### (a) Usage

SLP CN

#### (b) Function

With this instruction, a WUP signal is received from other modules. By executing the SLP instruction, the module does not start subsequent instructions until it receives the WUP signal from the module to be synchronized. Upon receipt of the WUP signal, execution of the subsequent instruction on the CL is continued.
By setting bit 'm' of the CN enable field to 1, the WUP signal from the module set to SYNCCm (m = 0 to 15) signal is awaited, selection of multiple bits is supported.

#### (c) Instruction Format



## U33.5 Usage Notes

### U33.5.1 Synchronized system between modules

For synchronized system between modules, refer to the **Section U34.IMP-X7_system, U34.10.5 Inter Module Synchronization of IMP-X7_System manual.**

### U33.5.2 Power shutdown

For power shutdown, refer to User's Manual: Hardware, section 58.7.1, Shutting Down the Power Supply.

### U33.5.3 Module Standby Mode

For module standby mode, refer to User's Manual: Hardware, section 58.7.2, Module Standby Mode.

### U33.5.4 CPG Software Reset

For software reset of CPG, refer to User's Manual: Hardware, section 58.7.3, Software Reset by the Clock Pulse Generator (CPG).

### U33.5.5 Limitation

#### (1) FM size limitation

The vertical size after magnification function (skip/repeat), that can be supported in one HW activation, is limited to 2048(SPMC0) or 1024(SPMC1), while the horizontal size is limited to 128.

#### (2) DMAISA setting limitation for planar fast type

The start address in the DMAISA register is limited to 16byte aligned settings when reading the data from multiple channels simultaneously by planar fast type (DMAIS_MODE bit = 1) and selecting a magnification function (skip/repeat) in horizontal direction. If no magnification is used, this limitation is not available.

#### (3) DMAIE setting limitation for 1*5x5 mode

The processing power of the architecture is not sufficient to support 1*5x5 mode for all input channels, therefore, the number of usable input channels is halved here, so that channel m and channel m+2 together are operating on channel m. The enable signals of channel m and m+2 must be set same.
8bit:
DMAIE[0]=DMAIE[2], DMAIE[1]=DMAIE[3], DMAIE[4]=DMAIE[6], DMAIE[5]=DMAIE[7]
DMAIE[8]=DMAIE[10], DMAIE[9]=DMAIE[11], DMAIE[12]=DMAIE[14], DMAIE[13]=DMAIE[15]
16bit:
DMAIE[0]=DMAIE[2], DMAIE[4]=DMAIE[6], DMAIE[8]=DMAIE[10],DMAIE[12]=DMAIE[14]

### (4) Weights setting limitation for 2*3x3 mode

When the DMAIS_PART bits are set to 2 or 3 in 2*3x3 convolution mode, the weights for the odd FM channels must be set to 0, because only the even FM channels hold valid data. In that way, invalid data can be prevented from getting summed up during the convolution sum-up phase.

### (5) Magnification and padding mode combination limitation

The combination of repeat constant (magnification mode) and padding identity (padding mode) in Y direction cannot be supported.

### (6) PADD_STRIDE_STARTPOS and ARI_LEN setting limitation

If PADD_STRIDE_STARTPOS.START_POS_X is set to a larger value than 0, ARI_LEN.ARI_XLEN must be set to larger value than 2.
If PADD_STRIDE_STARTPOS.STRY is set to a value larger than 1, ARI_LEN.ARI_XLEN must be set to a value larger than 4.

### (7) DMAO setting limitation for m16 and m32

DMAOS.DMAOS_RMW should be set to 0 when m16 (DMAOPARAM. DMAOFM=2) or m32 (DMAOPARAM. DMAOFM=3) is selected. Also, don't access SPMC1 with Slim-IMP DMAC or other non-CNN IP at the same time of the DMAO transfer with m16 and m32.

### (8) Required settings for special DMAIE values

If DMAIE[15:0] meets condition (a) to (d) below, additional settings are required for DMAI_CONFIG and DMAW_CONFIG. If multiple conditions (a) to (d) apply, set DMAI_CONFIG to the ORed value.

(a) When DMAIE[3:0] is 0x4, 0x8, or 0xC, set DMAI_CONFIG and DMAW_CONFIG as follows before and after setting SLSP.SLSP_START bit to 1.

```
0x56009001    // Write 0x0001 to DMAI_CONFIG[15:0] with WPRNBSY command
0x00000001
0x56009401    // Write 0x1 to DMAW_CONFIG with WPRNBSY command
0x00000001

(Setting SLSP.SLSP_START to 1)

0x55009001    // Write 0x0000 to DMAI_CONFIG[15:0] with WPR command
0x00000000
0x56009401    // Write 0x0 to DMAW_CONFIG with WPRNBSY command
0x00000000
```

(b) When DMAIE[7:4] is 0x4, 0x8, or 0xC, set DMAI_CONFIG and DMAW_CONFIG as follows before and after setting SLSP.SLSP_START bit to 1.

```
0x56009001    // Write 0x0010 to DMAI_CONFIG[15:0] with WPRNBSY command
0x00000010
0x56009401    // Write 0x1 to DMAW_CONFIG with WPRNBSY command
```

0x00000001

(Setting SLSP.SLSP_START to 1)

0x55009001 // Write 0x0000 to DMAI_CONFIG[15:0] with WPR command
0x00000000
0x56009401 // Write 0x0 to DMAW_CONFIG with WPRNBSY command
0x00000000

(c) When DMAIE[11:8] is 0x4, 0x8, or 0xC, set DMAI_CONFIG and DMAW_CONFIG as follows before and after setting SLSP.SLSP_START bit to 1.

0x56009001 // Write 0x0100 to DMAI_CONFIG[15:0] with WPRNBSY command
0x00000100
0x56009401 // Write 0x1 to DMAW_CONFIG with WPRNBSY command
0x00000001

(Setting SLSP.SLSP_START to 1)

0x55009001 // Write 0x0000 to DMAI_CONFIG[15:0] with WPR command
0x00000000
0x56009401 // Write 0x0 to DMAW_CONFIG with WPRNBSY command
0x00000000

(d) When DMAIE[15:12] is 0x4, 0x8, or 0xC, set DMAI_CONFIG and DMAW_CONFIG as follows before and after setting SLSP.SLSP_START bit to 1.

0x56009001 // Write 0x1000 to DMAI_CONFIG[15:0] with WPRNBSY command
0x00001000
0x56009401 // Write 0x1 to DMAW_CONFIG with WPRNBSY command
0x00000001

(Setting SLSP.SLSP_START to 1)

0x55009001 // Write 0x0000 to DMAI_CONFIG[15:0] with WPR command
0x00000000
0x56009401 // Write 0x0 to DMAW_CONFIG with WPRNBSY command
0x00000000

### (9) WPRNBSY limitation after SLSP command

There must be an interval of several cycles between the setting of the SLSP register and the setting of the register whose trigger is specified as "ARI" in Table U33.3 by the WPRNBSY command. Therefore, choose one of the following methods to make the interval.

(a) Insert a SYNCS command between the setting of the SLSP register and the setting of the ARI-triggered register.

Example:
1, WPR (SLSP register)

2, SYNCS
3, WPRNBSY (ARI-triggered register)

(b) Insert at least one WPR command between the SLSP register setting and the ARI-triggered register setting or use the WPR command to set the ARI-triggered register.

Example 1:
1, WPR (SLSP register)
2, WPR (Some register)
3, WPRNBSY (ARI-triggered register)

Example 2:
1, WPR (SLSP register)
2, WPR (ARI-triggered register)

(c) Insert the required number of WPRNBSY commands between the SLSP register setting and the ARI-triggered register setting. The number of WPRNBSY commands required depends on the setting of the N field of WPRNBSY.
N field =1 …counts as 2cycle
N field =2 …counts as 3cycle
N field =3 …counts as 4cycle
N field =4 …counts as 5cycle
N field =5 or more …counts as 6cycle or more
A number of WPRNBSY instructions should be inserted so that the sum of inserted clock cycles reaches at least 6 cycles.

Example:
Insert WPRNBSY with N field =1 (2cycle) and N field =3 (4cycle)
1, WPR (SLSP register)
2, WPRNBSY, N field =1 (Some register)
3, WPRNBSY, N field =3 (Some register)
4, WPRNBSY (ARI-triggered register)

### (10) Limitation overview

**Table U33.55 Limitation overview**

| module(DMAI, DMAW, SUB, DMA3DC, DMAO, ARI) | function | Limitation | Reference |
|---|---|---|---|
| HI | Register Access | Accessing addresses not listed in this register list is prohibited. Operation cannot be guaranteed when accessing an address, which is not listed. | U33.1.3 |
| HI | Register Access | Access size is 32-bit. Operation cannot be guaranteed when accessing by other than 32-bit. | U33.1.3 |
| HI | Register Access | Operation cannot be guaranteed when writing to a register from CPU during IMP CNN execution. | U33.1.3 |

| | | | |
|---|---|---|---|
| HI | Register Access | In the CNN register address area, offset addresses larger than 0x7000 are forbidden to access for read and write direction by WPR instruction and other IP. | U33.1.3 |
| HI | Register Access | Read-only. The write value should be the same as the read value. Only modify the target bit (Read Modify Write) because the initial value should not be changed. | U33.1.3 |
| IFID | CLW | CLW usage | U33.4.1 |
| IFID | WPRNBSY | WPRNBSY can be only used if existing dependencies are not violated. | U33.4.3<br>U33.5.5.9 |
| IFID | WPRNBSY | WPRNBSY limitation after SLSP command | U33.5.5.9 |
| IFID | SLSP | Required settings for special DMAIE values | U33.5.5.8 |
| IFID | WUP | Be sure to insert 8 or more instructions other than the WUP instruction between two WUP instructions in the CL. | U33.4.11<br>U33.3.8.2 |
| IFID | WUP | Between 2 wake-up signals to the same module a SLP command must be scheduled for handshaking to the same one. | U33.3.8.2 |
| IFID | SYNCS | If DMAO changes the memory to output make sure to wait for BUS busy. | U33.4.7.b |
| IFID | SYNCS | If disenabling a DMAO channel, make sure to wait for BUS busy. | U33.4.7.b |
| IFID | SYNCS | If changing DMAO output from packed to planar fast mode and disabling any of the channels 0,8,16 or 24 in the planar fast mode layer, make sure to wait for BUS busy. | U33.4.7.b |
| DMAW | Weight loading | start address is restricted to 512byte align, so the lower 9bit of DMAWSA should be set to 0. | U33.3.2.7 |
| DMAW | Weight loading | When loading weight in depthwise mode (DMAWS.DMAWS_TYPE=B'0001), the CLW command with N field=B'00000 must be executed before starting the transfer by setting DMAWS.DMAWS_START to 1. | U33.3.2.7 |
| DMAW | Weight loading | When wrap around is enable, keep following restriction<br>DMAWWA_START*512 <= DMAWSA < DMAWWA_END*512 | register map |
| DMAI | Kernel mode | Comply with the function limitation for each kernel mode | table U33.9 |
| DMAI | Kernel mode | Only 1 of the following 3 operations is supported at a time: convolutional striding, dilated convolution | table U33.10 |

| | | | |
|---|---|---|---|
| | | and magnification. | |
| DMAI | Transfer modes | Comply to the supported transfer modes for each kernel mode. | table U33.11 |
| DMAI | Picture parameter | Comply to the size setting restriction depending on magnification settings. | table U33.12<br>table U33.13<br>register map |
| DMAI | Picture parameter | Comply to the start address setting restrictions | table U33.14<br>table U33.17<br>table U33.18<br>register map |
| DMAI | Picture parameter | Comply to the stride setting restrictions | table U33.15<br>register map |
| DMAI | Picture parameter | DMAISA setting limitation for planar fast type | U33.5.5.2 |
| DMAI | Picture parameter | Comply to the size setting dependencies between ARI and DMAI for x and y direction | register map |
| DMAI | Picture parameter | In case of 16*1x1 kernel mode, DMAILX must not be set larger than half of the setting for other kernel modes. | register map |
| DMAI | Common/Individual Setting | This individual setup is only supported for the 1*5x5 kernel mode and planar type format. | U33.3.2.6 |
| DMAI | Common/Individual Setting | Set 0 to DMAICO in case of individual DMAISA setup. | U33.3.2.6 |
| DMAI | Channel offset | Comply to the channel offset setting restrictions | table U33.16<br>register map |
| DMAI | Wrap around | Wrap-around is supported only for the packed type format (DMAIS_MODE=B'10). | U33.3.2.8 |
| DMAI | Wrap around | Comply to the address settings when using wrap-around function | U33.3.2.8<br>table U33.20<br>register map |
| DMAI | Channel part | Comply to the limitations regarding part function usage for planar modes | U33.3.2.11 |
| DMAI | Channel part | Weights setting limitation for 2*3x3 mode | U33.5.5.4 |
| DMAI | Channel Enable | It is prohibited to run CNN with the DMAIE register set to 16'h0000. At least one channel must be enabled by setting DMAIE register. | U33.3.2.10 |
| DMAI | Channel enable | When DMAIS_8TO16BIT_CONV=1, only even bits of DMAIE are available, and odd bits should be set to 0. | U33.3.2.10 |
| DMAI | Channel enable | DMAIE setting limitation for 1*5x5 mode | U33.5.5.3<br>U33.3.2.10 |
| DMAI | Padding | Supported range of padding function per kernel | table U33.24 |

| | | | |
|---|---|---|---|
| | | mode | |
| DMAI | Conv Stride | Supported range of convolution striding function per kernel mode | table U33.26 |
| DMAI | Dilated conv | Supported range of dilated convolution per kernel mode (support only 1*5x5) | table U33.28 |
| DMAI | Start pos | PADD_STRIDE_STARTPOS and ARI_LEN setting limitation | U33.5.5.6 |
| DMAI | Start pos | Start position in x direction for 2*3x3 convolution mode requires setting the left padding to 0 | register map |
| DMAI | Start pos | Start position in y direction for 2*3x3 convolution mode requires setting the top padding to 0 | register map |
| DMAI | Start pos | Size in x-direction after magnification must be larger than start position in x direction. | register map |
| DMAI | Start pos | Size in y-direction after magnification must be larger than start position in y direction. | register map |
| DMAI | Padding | PADD_STRIDE_STARTPOS. PADDMODE=1 (identity padding) cannot be set together with DMAIPARAM.DMAIMM= 10 (repeat constant) | register map U33.5.5.5 |
| DMA3DC | Picture parameter | Comply to alignment of DMA3DC start address | register map |
| DMA3DC | Picture parameter | DMA3DCL must be same as ARI_LEN for X/Y direction | register map |
| DMA3DC | Picture parameter | DMA3DCST must be set in 16 pixels units for s8 and s16 format, 8 pixels units for m16 format, and 4 pixels units for m32 format. | U33.3.3.2 |
| DMA3DC | Format | Supported combinations of DMA3DCFM and ARIFM | U33.5.6 |
| DMA3DC | Format | When {ARICSR[25], ARICSR[31]} = 2B'00, if 3DC_INPUT_USAGE=2/3 is used, only DMA3DCFM=m16/m32 are supported. | register map |
| DMA3DC | Format | When {ARICSR[25], ARICSR[31]} = 2B'01, if 3DC_INPUT_USAGE=2/3 is used, only DMA3DCFM=m16 is supported. | register map |
| DMA3DC | Format | Combination DMA3DCFM=s16 and ARICSR.ARIFM=s8 is not supported. | register map |
| DMA3DC | Format | Combination DMA3DCFM=s8 and ARICSR.ARIFM=s16 is not supported. | register map |
| DMA3DC | Format | DMA3DCFM must be set same in ARICSR and DMA3DCPARAM | register map |
| DMA3DC | SPMC | DMA3DC input data is only supported for SPMC1, not SPMC0 | U33.3.3 |
| DMA3DC | Channel enable | When DMA3DC is not used (DMA3DCS is not set), the DMA3DCE register must be set to | U33.3.3.4 |

| | | | |
|---|---|---|---|
| | | 32H'0000_0000. | |
| DMA3DC | Channel enable | One of the bits 0-7 of DMA3DCEN must be ON when using DMA3DC. | U33.3.3.4 |
| DMA3DC | Channel enable | If a bit in the ARIE register is set to 0, the corresponding bit in the DMA3DCE register must also be set to 0 (exception: ELTWISE MULT 16bit). | U33.3.3.4 |
| DMA3DC | Shifter | No overflow check for SFTC, support of 32bit (8bit) and 41bit (16bit) for shifted output value | register map |
| SUB | Format | For convolution with s8/u8 FM and s16 WT.<br>-Setting 0x1 in the DMAIS.DMAIS_8TO16BIT_CONV register to enable 8bpp to 16bpp conversion for FM<br>-DMAIE can only be set to even bits, odd bits should be set to 0.<br>-ARICSR.DMAIFM register must be set to 0x2 (s16). | U33.3.4.3 |
| SUB | Format | When using DMAW to load weight data, DMAWS_FM register must be set to the same value as the ARICSR. ARIFM register. | U33.3.4.3 |
| SUB | Format | Convolution with DMAIPARAM.DMAIFM=2B'10 (s16) and DMAWS.DMAWS_FM=1B'0 (s8) is not supported | U33.3.4.3 |
| ARI | ALL subunits | ARIE bit 0 must be enabled | register map |
| ARI | Activation | Support for s8/s16/m16 with ARICSR.M16_VAL_AFTER_ACT set to 1.<br>Not support for m16 with ARICSR.M16_VAL_AFTER_ACT set to 0 and m32. | U33.3.5.2 |
| ARI | Activation | For RELU and BRELU and if not using an activation function, ACTIVATION_OUTPUT_PARAM and ACTIVATION_ZERO_POINT in post processing are not supported. | figure U33.32 figure U33.33 |
| ARI | Activation | SFTM_ZERO_POINT for RELU and BRELU only used if ARICSR.ARIFM_AFTER_ACTIVATION is s8.<br>User must set SFTM_ZERO_POINT to zero for s16/m16 with ARICSR.M16_VAL_AFTER_ACT set to 1. | table U33.35 |
| ARI | Activation | ACTIVATION_ZERO_POINT for Post Processing only used if ARICSR.ARIFM_AFTER_ACTIVATION is s8. | table U33.45 |

| | | | |
|---|---|---|---|
| | | User must set it to zero for s16/m16 with ARICSR.M16_VAL_AFTER_ACT set to 1. | |
| ARI | Activation | For EXP(SOFTMAX) mode, the input data should be negative values only. | U33.3.5.2e |
| ARI | Activation | For BRELU, only ARI_BND.ARI_UPPER_BND is used.<br>Leaky RELU, both ARI_BND.ARI_UPPER_BND and ARI_BND.ARI_LOWER_BND are used.<br>Other modes do not use ARI_BND.ARI_UPPER_BND and ARI_BND.ARI_LOWER_BND. | U33.3.5.2b<br>U33.3.5.2c |
| ARI | Activation | ACTIVATION_MULT_VAL and ACTIVATION_MULT_POST_SFT are only used for Leaky RELU. | U33.3.5.2 |
| ARI | Channel pooling | Support for s8/s16, m16/m32 are not supported | U33.3.5.3 |
| ARI | Channel pooling | The first channel of each channel pooling group must be enabled. | register map |
| ARI | XY pooling | Support for s8/s16, m16/m32 are not supported | U33.3.5.4 |
| ARI | XY pooling | When using XY average pooling, XPOOL and YPOOL bits of the ARICSR register must be the same | U33.3.5.4b |
| ARI | XY pooling | ARI_XLEN must be greater than XPOOL and ARI_YLEN must be greater than YPOOL | register map |
| ARI | Format | 16bit ARIFM and 8bit ARIFM_AFTER_ACTIVATION is not supported | table U33.48 |
| ARI | Format | {ARICSR.MIX_MODE_2,ARICSR.MIX_MODE} = 2B'01 must work with DMAO format = 2 (mixmode 16bit) | register map |
| ARI | Format | {ARICSR.MIX_MODE_2,ARICSR.MIX_MODE} = 2B'11 must work with DMAO format = 3 (mixmode 32bit) | register map |
| ARI | Statistics | Support for s8/s16, m16/m32 are not supported | U33.3.7 |
| ARI | Statistics | Performance counter do not support DMAW | register map |
| ARI | Statistics | Limited stall reason selection for DMAI and DMA3DC | register map |
| DMAO | Format | Only DMAO format conversion from s16 to s8 is supported, other combinations are not supported. | U33.3.6 |
| DMAO | Format | Planar mode output is only supported for format s8/s16, m16 and m32 are not supported. | table U33.47 |
| DMAO | Channel enable | DMAOE bit 0 must be enabled | U33.3.6 |

| | | | |
|---|---|---|---|
| DMAO | Channel enable | Only channels which are active after channel pooling can be selected by DMAOE. | register map |
| DMAO | Channel part | DMAO channel part is not supported with DMAO mixed mode processing | table U33.47 |
| DMAO | Channel offset | DMAO channel offset is not supported in DMAO packed mode | table U33.47 |
| DMAO | Wrap around | DMAO wrap-around is only supported for DMAO packed mode when accessing to SPMC0 | table U33.47 |
| DMAO | Wrap around | Comply to the address settings when using wrap-around function | table U33.47<br>U33.3.6 |
| DMAO | Picture parameter | CEILING(ARI_XLEN/(XPOOL+1),0) >= DMAO_X<br>CEILING(ARI_YLEN/(YPOOL+1),0) >= DMAO_Y | U33.3.9<br>register map |
| DMAO | Picture parameter | For SPMC1 output, 0 < y <= 1024 | U33.5.5.1 |
| DMAO | SPMC access | Mixed mode output is only supported for SPMC1, not SPMC0 | table U33.47 |
| DMAO | SPMC access | Register dump output is only supported for SPMC1, not SPMC0 | U33.3.7.3 |
| DMAO | SPMC access | Planar fast type of planar format output is only supported for SPMC0, not SPMC1 | table U33.47 |
| DMAO | Register dump | DMAO and DMARD cannot run at the same time. | U33.3.7.3 |
| DMAO | RmW | In case of DMAO_RMW=0 and packed mode, all not written data bytes of the 512byte data word are overwritten with 0, for planar mode, only not written data bytes of 8byte data words with at least 1 written data byte are overwritten with 0. | register map |
| DMAO | RmW / SPMC1 access | DMAOS.DMAOS_RMW must be 0 if using m16 or m32 and Slim-IMP DMAC or other non-CNN IP should not access at same time point. | U33.3.5.7 |
| DMARD | Channel enable | At least 1 DMAOE bit must be enabled | U33.3.7.3 |
| DMAI/DMAO/DMA3DC | packed | The horizontal stride DMAxST must be specified as multiple of 16 pixels. | U33.3.1.2 |
| DMAW, DMAI, DMA3DC, DMAO, ARI | processing flow | Comply to CNN-IP processing flow | U33.3.9 |

### U33.5.6 Format combinations

Table U33.56 shows all allowed format combinations.

**Table U33.56 Possible format combinations among DMAI, DMA3DC, ARI and DMAO**

| group | DMAI | | | DMA 3DC | ARI | | | | | | | DMAO | | | stored channel | | proc. channel | | comment |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMAIFM | DMAIS_8TO16BIT_CONV | DMAIS_PART | DMA3DCFM | ARIFM | MIXED_MODE | MIXED_MODE_2 | DMAIFM | DMA3DCFM | 3DC_INPUT_USAGE | ARIFM_AFTER_ACTIVATION | DMAOFM | ARIFM_AFTER_ACTIVATION | DMAOS_PART | packed input format | packed output format | processed input channel 1* | processed output channel | |
| bit signal | | | | | | | | | | | | | | | | | | | - |
| Possible Value Input/Output Format | 0-2 | 0-1 | 0-11 | 0-3 | 0-1 | 0-1 | 0-1 | 0-2 | 0-3 | 0-3 | 0-1 | 0-3 | 0-1 | 0-7 | 16/32 | 16/32 | 4/8/16/32 | 4/8/16/32 | - |
| s8,s8 | 0 | 0 | 0 | 0,2,3 | 0 | 0 | 0 | 0 | 0,2,3 | 0 | 0 | 0 | 0 | 0 | 32 | 32 | 32 | 32 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 32ch; DMAO: 32ch |
| s8,s8 | 0 | 0 | 0-3 | 0,2,3 | 0 | 0 | 0 | 0 | 0,2,3 | 0 | 0 | 0 | 0 | 0 | 32 | 32 | 8 | 32 | DMAI: 1x5x5: 8ch; ARI: 32ch; DMAO: 32ch |
| s8,s8 | 0 | 0 | 2,3 | 0,2,3 | 0 | 0 | 0 | 0 | 0,2,3 | 0 | 0 | 0 | 0 | 2,3 | 32 | 32 | 16 | 16 | DMAI: 2x3x3 mode, 16ch; ARI: 32ch; DMAO: 16ch |
| s8,s8 | 0 | 0 | 4-7 | 0,2,3 | 0 | 0 | 0 | 0 | 0,2,3 | 0 | 0 | 0 | 0 | 4-7 | 32 | 32 | 8 | 8 | DMAI: 2x3x3 mode, 8ch; ARI: 32ch; DMAO: 8ch |
| s8,m16 | 0 | 0 | 0 | 0,2,3 | 0 | 1 | 0 | 0 | 0,2,3 | 0 | 0 | 2 | 0 | 0 | 32 | 32 | 32 | 32 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 32ch; DMAO: 32ch, 16bit SPMC1 |
| s8,m16 | 0 | 0 | 0-3 | 0,2,3 | 0 | 1 | 0 | 0 | 0,2,3 | 0 | 0 | 2 | 0 | 0 | 32 | 32 | 8 | 32 | DMAI: 1x5x5: 8ch; ARI: 32ch; DMAO: 32ch, 16bit SPMC1 |
| s8,m32 | 0 | 0 | 0 | 0,2,3 | 0 | 1 | 1 | 0 | 0,2,3 | 0 | 0 | 3 | 0 | 0 | 32 | 32 | 32 | 32 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 32ch; DMAO: 32ch, 32bit SPMC1 |
| s8,m32 | 0 | 0 | 0-3 | 0,2,3 | 0 | 1 | 1 | 0 | 0,2,3 | 0 | 0 | 3 | 0 | 0 | 32 | 32 | 8 | 32 | DMAI: 1x5x5: 8ch; ARI: 32ch; DMAO: 32ch, 32bit SPMC1 |
| u8,s8 | 1 | 0 | 0 | 0,2,3 | 0 | 0 | 0 | 1 | 0,2,3 | 0 | 0 | 0 | 0 | 0 | 32 | 32 | 32 | 32 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 32ch; DMAO: 32ch |
| u8,s8 | 1 | 0 | 0-3 | 0,2,3 | 0 | 0 | 0 | 1 | 0,2,3 | 0 | 0 | 0 | 0 | 0 | 32 | 32 | 8 | 32 | DMAI: 1x5x5: 8ch; ARI: 32ch; DMAO: 32ch |
| u8,s8 | 1 | 0 | 2,3 | 0,2,3 | 0 | 0 | 0 | 1 | 0,2,3 | 0 | 0 | 0 | 0 | 2,3 | 32 | 32 | 16 | 16 | DMAI: 2x3x3 mode, 16ch; ARI: 32ch; DMAO: 16ch |
| u8,s8 | 1 | 0 | 4-7 | 0,2,3 | 0 | 0 | 0 | 1 | 0,2,3 | 0 | 0 | 0 | 0 | 4-7 | 32 | 32 | 8 | 8 | DMAI: 2x3x3 mode, 8ch; ARI: 32ch; DMAO: 8ch |
| u8,m16 | 1 | 0 | 0 | 0,2,3 | 0 | 1 | 0 | 1 | 0,2,3 | 0 | 0 | 2 | 0 | 0 | 32 | 32 | 32 | 32 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 32ch; DMAO: 32ch; 16bit SPMC1 |
| u8,m16 | 1 | 0 | 0-3 | 0,2,3 | 0 | 1 | 0 | 1 | 0,2,3 | 0 | 0 | 2 | 0 | 0 | 32 | 32 | 8 | 32 | DMAI: 1x5x5: 8ch; ARI: 32ch; DMAO: 32ch; 16bit SPMC1 |
| u8,m32 | 1 | 0 | 0 | 0,2,3 | 0 | 1 | 1 | 1 | 0,2,3 | 0 | 0 | 3 | 0 | 0 | 32 | 32 | 32 | 32 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 32ch; DMAO: 32ch; 32bit SPMC1 |
| u8,m32 | 1 | 0 | 0-3 | 0,2,3 | 0 | 1 | 1 | 1 | 0,2,3 | 0 | 0 | 3 | 0 | 0 | 32 | 32 | 8 | 32 | DMAI: 1x5x5: 8ch; ARI: 32ch; DMAO: 32ch; 32bit SPMC1 |
| s16,s16 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 0 | 16 | 16 | 16 | 16 | DMAI: 2x3x3, all 1x1 modes: 16ch; ARI: 16ch; DMAO: 16ch |
| s16,s16 | 2 | 0 | 0-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 0 | 16 | 16 | 4 | 16 | DMAI: 1x5x5: 4ch; ARI: 16ch; DMAO: 16ch |
| s16,s16 | 2 | 0 | 2,3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 2,3 | 16 | 16 | 8 | 8 | DMAI: 2x3x3 mode, 8ch; ARI: 16ch; DMAO: 8ch |
| s16,s16 | 2 | 0 | 4-7 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 4-7 | 16 | 16 | 4 | 4 | DMAI: 2x3x3 mode, 4ch; ARI: 16ch; DMAO: 4ch |
| s16,s8 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2,3 | 16 | 32 | 16 | 16 | DMAI: 2x3x3, all 1x1 modes: 16ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| s16,s8 | 2 | 0 | 0-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2,3 | 16 | 32 | 4 | 16 | DMAI: 1x5x5: 4 ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| s16,s8 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2,3 | 16 | 32 | 16 | 16 | DMAI: 2x3x3, all 1x1 modes: 16ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| s16,s8 | 2 | 0 | 0-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2,3 | 16 | 32 | 4 | 16 | DMAI: 1x5x5: 4 ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| s8,s16 | 0 | 1 | 2 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 0 | 32 | 16 | 32 | 16 | DMAI: all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch |
| s8,s16 | 0 | 1 | 2-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 0 | 32 | 16 | 16 | 16 | DMAI: 2x3x3 mode: 16ch; ARI: 16ch; DMAO: 16ch |
| s8,s16 | 0 | 1 | 4-11 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 0 | 32 | 16 | 4 | 16 | DMAI: 1x5x5 mode: 4ch; ARI: 4ch; DMAO: 16ch |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| s8,s8 | 0 | 1 | 2 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2-3 | 32 | 32 | 32 | 16 | DMAI: all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| s8,s8 | 0 | 1 | 2-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2-3 | 32 | 32 | 16 | 16 | DMAI: 2x3x3 mode: 16ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| s8,s8 | 0 | 1 | 4-11 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2-3 | 32 | 32 | 4 | 16 | DMAI: 1x5x5 mode: 4ch; ARI: 4ch; DMAO: 16ch, output upper s8 |
| s8,s8 | 0 | 1 | 2 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2-3 | 32 | 32 | 32 | 16 | DMAI: all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| s8,s8 | 0 | 1 | 2-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2-3 | 32 | 32 | 16 | 16 | DMAI: 2x3x3 mode: 16ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| s8,s8 | 0 | 1 | 4-11 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2-3 | 32 | 32 | 4 | 16 | DMAI: 1x5x5 mode: 4ch; ARI: 4ch; DMAO: 16ch, output lower s8 |
| s8,s16 | 0 | 0 | 0 | 2-3 | 0 | 0 | 0 | 0 | 2-3 | 2,3 | 1 | 1 | 1 | 0 | 32 | 16 | 32 | 16 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch |
| s8,s16 | 0 | 0 | 0-3 | 2-3 | 0 | 0 | 0 | 0 | 2-3 | 2,3 | 1 | 1 | 1 | 0 | 32 | 16 | 8 | 16 | DMAI: 1x5x5: 8ch; ARI: 16ch; DMAO: 16ch |
| s8,s8 | 0 | 0 | 0 | 2-3 | 0 | 0 | 0 | 0 | 2-3 | 2,3 | 1 | 0 | 1 | 2-3 | 32 | 32 | 32 | 16 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| s8,s8 | 0 | 0 | 0-3 | 2-3 | 0 | 0 | 0 | 0 | 2-3 | 2,3 | 1 | 0 | 1 | 2-3 | 32 | 32 | 8 | 16 | DMAI: 1x5x5: 8ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| s8,s8 | 0 | 0 | 0 | 2-3 | 0 | 0 | 0 | 0 | 2-3 | 2,3 | 1 | 0 | 0 | 2-3 | 32 | 32 | 32 | 16 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| s8,s8 | 0 | 0 | 0-3 | 2-3 | 0 | 0 | 0 | 0 | 2-3 | 2,3 | 1 | 0 | 0 | 2-3 | 32 | 32 | 8 | 16 | DMAI: 1x5x5: 8ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| u8,s16 | 1 | 1 | 2 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 0 | 32 | 16 | 32 | 16 | DMAI: all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch |
| u8,s16 | 1 | 1 | 2-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 0 | 32 | 16 | 16 | 16 | DMAI: 2x3x3 mode: 16ch; ARI: 16ch; DMAO: 16ch |
| u8,s16 | 1 | 1 | 4-11 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 1 | 0 | 32 | 16 | 4 | 16 | DMAI: 1x5x5 mode: 4ch; ARI: 4ch; DMAO: 16ch |
| u8,s8 | 1 | 1 | 2 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2-3 | 32 | 32 | 32 | 16 | DMAI: all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| u8,s8 | 1 | 1 | 2-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2-3 | 32 | 32 | 16 | 16 | DMAI: 2x3x3 mode: 16ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| u8,s8 | 1 | 1 | 4-11 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2-3 | 32 | 32 | 4 | 16 | DMAI: 1x5x5 mode: 4ch; ARI: 4ch; DMAO: 16ch, output upper s8 |
| u8,s8 | 1 | 1 | 2 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2-3 | 32 | 32 | 32 | 16 | DMAI: all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| u8,s8 | 1 | 1 | 2-3 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2-3 | 32 | 32 | 16 | 16 | DMAI: 2x3x3 mode: 16ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| u8,s8 | 1 | 1 | 4-11 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2-3 | 32 | 32 | 4 | 16 | DMAI: 1x5x5 mode: 4ch; ARI: 4ch; DMAO: 16ch, output lower s8 |
| u8,s16 | 1 | 0 | 0 | 2-3 | 0 | 0 | 0 | 1 | 2-3 | 2,3 | 1 | 1 | 1 | 0 | 32 | 16 | 32 | 16 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch |
| u8,s16 | 1 | 0 | 0-3 | 2-3 | 0 | 0 | 0 | 1 | 2-3 | 2,3 | 1 | 1 | 1 | 0 | 32 | 16 | 8 | 16 | DMAI: 1x5x5: 8ch; ARI: 16ch; DMAO: 16ch |
| u8,s8 | 1 | 0 | 0 | 2-3 | 0 | 0 | 0 | 1 | 2-3 | 2,3 | 1 | 0 | 1 | 2-3 | 32 | 32 | 32 | 16 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| u8,s8 | 1 | 0 | 0-3 | 2-3 | 0 | 0 | 0 | 1 | 2-3 | 2,3 | 1 | 0 | 1 | 2-3 | 32 | 32 | 8 | 16 | DMAI: 1x5x5: 8ch; ARI: 16ch; DMAO: 16ch, output upper s8 |
| u8,s8 | 1 | 0 | 0 | 2-3 | 0 | 0 | 0 | 1 | 2-3 | 2,3 | 1 | 0 | 0 | 2-3 | 32 | 32 | 32 | 16 | DMAI: 2x3x3, all 1x1 modes: 32ch; ARI: 16ch; DMAO: 16ch, output lower s8 |
| u8,s8 | 1 | 0 | 0-3 | 2-3 | 0 | 0 | 0 | 1 | 2-3 | 2,3 | 1 | 0 | 0 | 2-3 | 32 | 32 | 8 | 16 | DMAI: 1x5x5: 8ch; ARI: 16ch; DMAO: 16ch, output lower s8 |

Note:
1* Number of input FM channels one run processing.

### U33.5.7 Recommendations

#### (1) Channel enable setting in case of planar type of planar mode

It is recommended that consecutive input channels are enabled without disabling channels in between, because disenabled channels also consume time. This improves the performance as shown in following example. In case of 8x1x1 kernel in byte format and selecting 2 DMAI channels over DMAIE, case a and b is showing 2 examples with required cycles for loading the data for 1 row by DMAI.

a.) In case of DMAIE = 0x00000003, DMAI channel 0 and 1 are enabled, which means 16 channels (channel 0-15) are enabled and loading 1 row takes 16 cycles.
b.) In case of DMAIE = 0x00008001, DMAI channel 0 and 15 are enabled, which means 16 channels (channel 0-7 and 120-127) are enabled and loading 1 row takes 128 cycles.

#### (2) Channel enable setting in case of planar fast type of planar mode and packed mode

It is recommended that input channels are enabled in blocks or 4 channels (ch0-3, ch4-7, ch8-11, ch12-15), because blocks are getting loaded if at least 1 channel inside the block is enabled. This improves the performance as shown in following example. In case of 8x1x1 kernel in byte format and selecting 4 DMAI channels over DMAIE, case a and b is showing 2 examples with required cycles for loading 16byte per channel by DMAI.

a.) In case of DMAIE = 0x0000000f, DMAI channel 0 till 3 are enabled, which means 32 channels (channel 0-31) are enabled and loading 16 byte per channel takes 1 cycle.

b.) In case of DMAIE = 0x00001111, DMAI channel 0,4,8,12 are enabled, which means 32 channels (channel 0-7, 32-39, 64-71 and 96-103) are enabled and loading 16 byte per channel takes 4 cycles.

## U33.6 Safety Mechanism

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### U33.6.1 EDC protection.

EDC (Error Detection Code) protection is implemented to SRAM, Bus interface and dedicated CNN/SPMC in IMPCNN. EDC check is always enabled. Note that all FUSA errors from the IMP CNN are merged and finally output to the ECM (For details, please refer U34. IMP-X7: U34.10.6 Safety Mechanism) through IMPSLV. AXI slave BUS EDC errors, AXI master EDC errors, CNN SPMC interface EDC error are merged to **IMP-X7 bus error** and SRAM EDC errors are merged to **IMP-X7 SRAM EDC error** of ECM module.

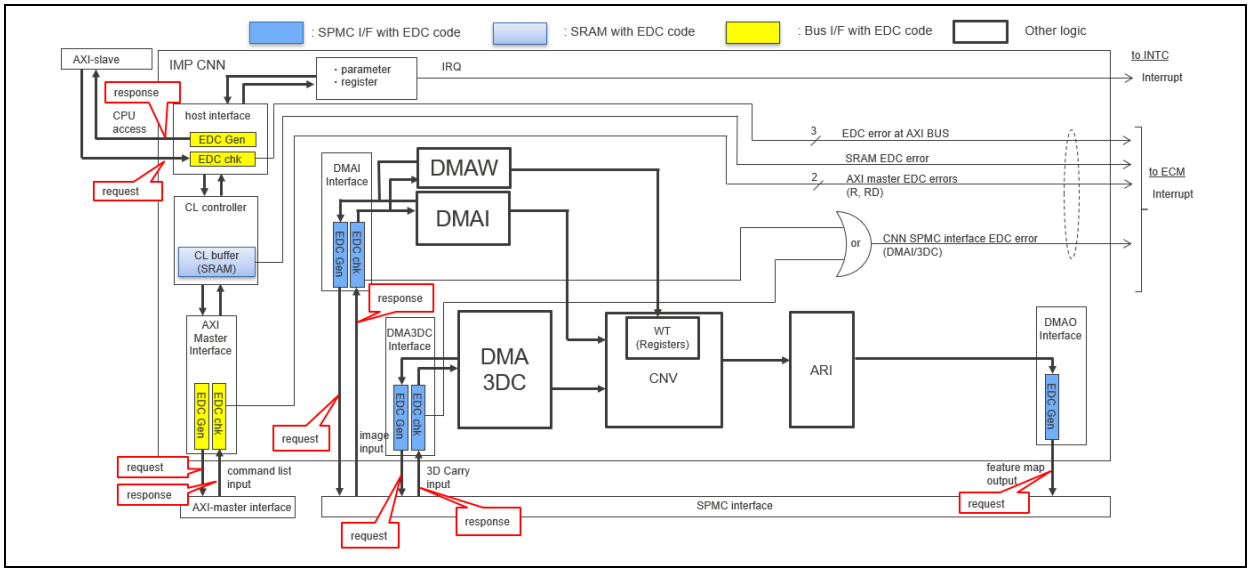



**Figure U33.65 Block Diagram of IMPCNN (FUSA)**

# U34. IMP-X7_system

## U34.1 Overview

The IMP-X7 consists of multiple modules and an Internal memory, each unit is connected by an IMP local bus. The IMP local bus allows each module to access any bank of internal memory (ScratchPad memory) at the same time. The configuration diagram of the IMP local bus is shown in Figure U34.1, and the address of each bank of the Internal memory is shown in Table U34.2.

Interrupts generated in each module can be controlled by the INT control logic of IMP-X7 and seven interrupts can be output to the interrupt controller (INTC). For the control method with INT control logic, refer to **U34.10 Operation**.

### U34.1.1 Features

The IMP local bus specification of IMP-X7 is shown in Figure U34.1.

**Table U34.1 IMP local bus Specification**

| Specification | (Module name in R-Car V4M) | | |
|---|---|---|---|
| | SPMI | SPMC (High speed domain) | |
| Operating frequency | 640 MHz | 800 MHz | |
| Number of units | 1 | 1 | |
| Bus width | 256 bits | 256 bits / CNN dedicated 4096 bits | |
| Capacity<br>(Internal memory) | 2 Mbytes<br>(256 Kbytes × 8 banks) *1 = 1 Mbytes<br>In each bank access protection (secure regions scheme) is applied. | 2 Mbytes<br>SPMC0<br>Access protection (secure regions scheme) is applied. | 0.5 Mbytes<br>SPMC1<br>Access protection (secure regions scheme) is applied. |

### U34.1.2 Block Diagram

Figure U34.1 shows the internal implementation of the Gen4 IMP-X7.VDSP DMAC can access SPMC/SPMI memory with incremental transaction. Possible transaction sizes are 128-byte/256-byte and alignment shall be 32-byte. For the base address and use of VDSP, refer to the VDSP HWM.

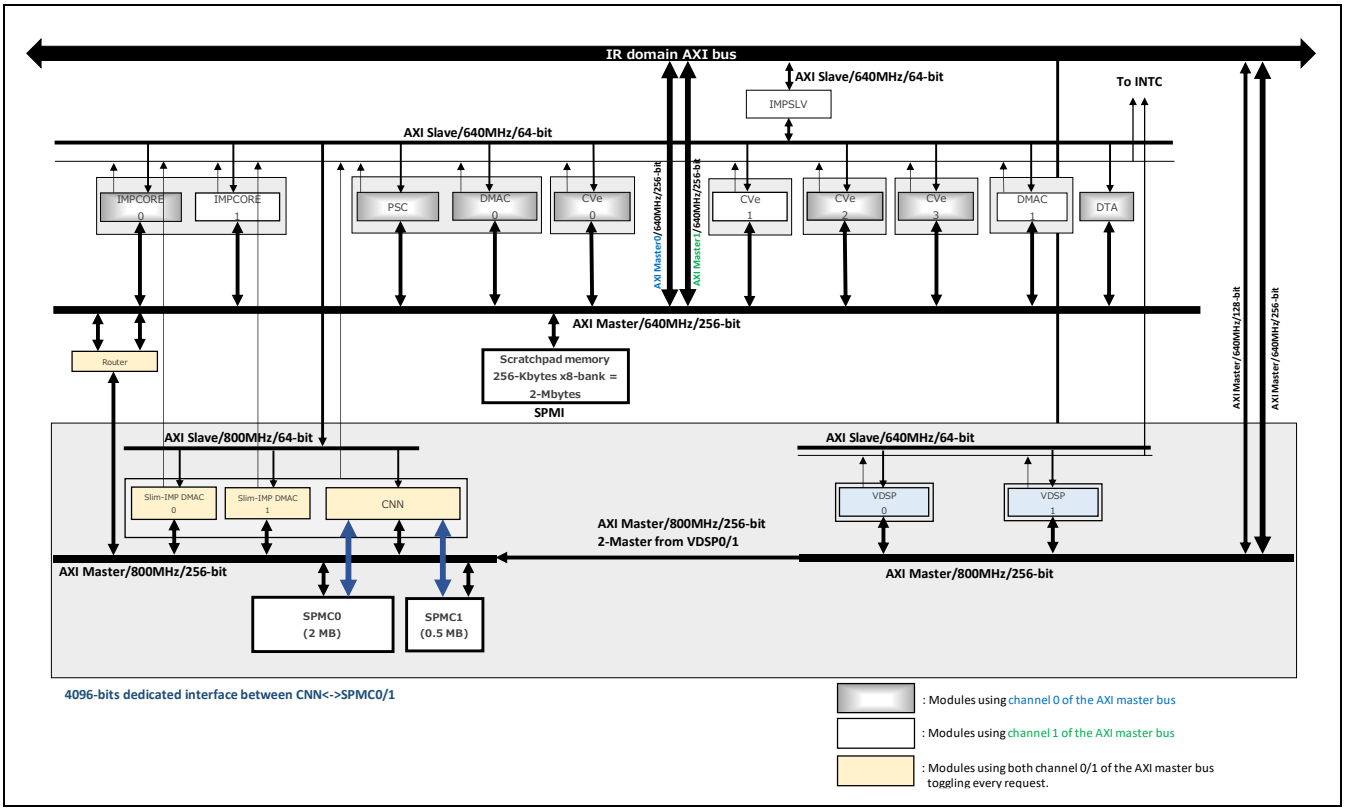



**Figure U34.1 IMP-X7 IMP local bus arrangement**

### U34.1.3 Address map of Scratchpad Memory

The address map of each bank of Scratchpad memory (multi-bank Internal memory of the IMP-X7) is shown in Table U34.2.
When accessing below addresses from host CPU, these are accessed as registers by default. For example, start addresses of Scratchpad memory in Table U34.2 and base addresses of SPM registers in U34.8 are same. If user access these addresses by CPU, then they are accessed as registers which are described in U34.8. If user need to access these addresses by CPU as a Scratchpad memory, refer to the setting in U34.10.3.
IMP-X7 IPs like IMPcore, DMAC, Slim-IMP DMAC, IMPPSC, CVe, IMPCNN always access these addresses as Scratchpad memory. When VDSP access Scratchpad memory (SPMI/SPMC0/1) in IMP-X7, initially set 0 to the bit 31 of SPMCTRL4 (0xFFAB0F18) register in SPMC. Do not change this bit after setting to 0.

**Table U34.2 Address map of Scratchpad memory (SPMI, SPMC0/1)**

| Module | Scratchpad memory Bank number | Start address | End address | Size |
|---|---|---|---|---|
| SPMI | 0 | 0xED200000 | 0xED23FFFF | 256 KB |
| | 1 | 0xED240000 | 0xED27FFFF | 256 KB |
| | 2 | 0xED280000 | 0xED2BFFFF | 256 KB |
| | 3 | 0xED2C0000 | 0xED2FFFFF | 256 KB |
| | 4 | 0xED300000 | 0xED33FFFF | 256 KB |
| | 5 | 0xED340000 | 0xED37FFFF | 256 KB |
| | 6 | 0xED380000 | 0xED3BFFFF | 256 KB |
| | 7 | 0xED3C0000 | 0xED3FFFFF | 256 KB |
| SPMC0 | - | 0xED400000 | 0xED5FFFFF | 2 MB |
| SPMC1 | - | 0xED600000 | 0xED67FFFF | 0.5 MB |

## U34.2 Register Configuration (IMPSLV bus Registers)

These are the registers implemented in IMPSLV module in IMP-X7 local bus. Do not access to any address other than those listed Register Configuration. Otherwise, correct operation is not guaranteed.

| Base Address Name | Base Address |
|---|---|
| IMPSLV | H'FF90 0000 |

[Legend]
Initial value : Register value after a reset (hardware reset : This reset is to reset all the logic including bus interface)
— : Undefined value
R/W : Readable/writable. The written value can be read.
R/WC1 : Readable/writable. Writing 1 initializes the bit. Writing 0 to the bit is ignored.
R : Read-only. The write value should always be 0.

—All the registers of IMPSLV can be reset by CPG software reset.
— For the bits that are not assigned:
a) For reserved bits of read only register, always read initial value before preset (initial setting).
b) For reserved bits of read/write register, always read initial value before preset (initial setting) and write value should always be read value after preset.

Refer to R-CarV4M_UM_U34_IMP-X7_System_IMPSLV_r0p40.xlsx.

## U34.3 Register Description (IMPSLV bus Registers)

Refer to R-CarV4M_UM_U34_IMP-X7_System_IMPSLV_r0p40.xlsx.

## U34.4 Register Configuration (SPMI bus Registers)

These are the registers implemented in SPMI module in IMP-X7 local bus. Do not access to any address other than those listed Register Configuration. Otherwise, correct operation is not guaranteed.

| Base Address Name | Base Address |
|---|---|
| SPMI | H'FFA8 C000 |

[Legend]
Initial value : Register value after a reset (hardware reset : This reset is to reset all the logic including bus interface)
— : Undefined value
R/W : Readable/writable. The written value can be read.
R/WC1 : Readable/writable. Writing 1 initializes the bit. Writing 0 to the bit is ignored.
R : Read-only. The write value should always be 0.

- All the registers of SPMI can be reset by CPG software reset.
- For the bits that are not assigned:
  a) For reserved bits of read only register, always read initial value before preset (initial setting).
  b) For reserved bits of read/write register, always read initial value before preset (initial setting) and write value should always be read value after preset.

Refer to R-CarV4M_UM_U34_IMP-X7_System_SPMI_r0p81.xlsx.

## U34.5 Register Description (SPMI bus Registers)

Refer to R-CarV4M_UM_U34_IMP-X7_System_SPMI_r0p81.xlsx.

## U34.6 Register Configuration (SPMC bus Registers)

These are the registers implemented in SPMC module in IMP-X7 local bus. Do not access to any address other than those listed Register Configuration. Otherwise, correct operation is not guaranteed.

| Base Address Name | Base Address |
|---|---|
| SPMC | H'FFAB 0000 |

[Legend]
Initial value : Register value after a reset (hardware reset : This reset is to reset all the logic including bus interface)
— : Undefined value
R/W : Readable/writable. The written value can be read.
R/WC1 : Readable/writable. Writing 1 initializes the bit. Writing 0 to the bit is ignored.
R : Read-only. The write value should always be 0.

— All the registers of SPMC can be reset by CPG software reset.
— For the bits that are not assigned:
  a) For reserved bits of read only register, always read initial value before preset (initial setting).
  b) For reserved bits of read/write register, always read initial value before preset (initial setting) and write value should always be read value after preset.

Refer to R-CarV4M_UM_U34_IMP-X7_System_SPMC_r0p81.xlsx.

## U34.7 Register Description (SPMC bus Registers)

Refer to R-CarV4M_UM_U34_IMP-X7_System_SPMC_r0p81.xlsx.

## U34.8 Register Configuration Scratchpad Memory (SPM Registers)

These are the registers implemented in Scratchpad Memory (SPM) module in IMP-X7. Do not access to any address other than those listed Register Configuration. Otherwise, correct operation is not guaranteed.

| Base Address Name | Base Address |
|---|---|
| SPMI bank0 – bank7<br><br>By default, 4-byte CPU access to Scratchpad memory as a register. | 0xED20_0000<br>0xED24_0000<br>0xED28_0000<br>0xED2C_0000<br>0xED30_0000<br>0xED34_0000<br>0xED38_0000<br>0xED3C_0000 |
| SPMC0/1<br>By default, 4-byte CPU access to Scratchpad memory as a register. | 0xED40_0000<br>0xED60_0000 |

[Legend]
Initial value : Register value after a reset (hardware reset : This reset is to reset all the logic including bus interface)
－ : Undefined value
R/W : Readable/writable. The written value can be read.
R/WC1 : Readable/writable. Writing 1 initializes the bit. Writing 0 to the bit is ignored.
R : Read-only. The write value should always be 0.
W : Write-only. The read value should always be 0.

- All the registers of SPM can be reset by CPG software reset.
- For the bits that are not assigned:
  a) For reserved bits of read only register, always read initial value before preset (initial setting).
  b) For reserved bits of read/write register, always read initial value before preset (initial setting) and write value should always be read value after preset.

Refer to R-CarV4M_UM_U34_IMP-X7_System_SPM_r0p70.xlsx.

## U34.9 Register Description Scratchpad Memory (SPM Registers)

Refer to R-CarV4M_UM_U34_IMP-X7_System_SPM_r0p70.xlsx.

## U34.10 Operation

### U34.10.1 Initial setting

IMP local bus needs following setting initially and again if it meets the preset condition in Table U34.3.

**Table U34.3** Initial setting IMP local bus

| Preset IMP local bus | Register name | Address | Initial set value | Preset condition |
|---|---|---|---|---|
| | PRESET Register [IMPSLV] | 0xFF90_2000 | 0x0000000A | Power on C4 or IMPSLV reset release |
| | PRESET Register [SPMI] | 0xFFA8_E000 | 0x0000000A | Power on C4 or SPMI reset release |
| | Master Bus ConTRoL120 [SPMI] | 0xFFA8_C868 | 0x00030000 | Power on C4 or SPMI reset release |
| | Master Bus ConTRoL130 [SPMI] | 0xFFA8_D028 | 0x00030000 | Power on A2CN0 or SPMC reset release |
| | PRESET Register [SPMC] | 0xFFAB_2000 | 0x00000009 | Power on A2CN0 or SPMC reset release |

### U34.10.2 Secure region for IMP-X7

Before using IMP subsystem, region ID related to IMP subsystem setting is necessary. For the region ID setting refer to the HWM for the 18.AXI-Bus secure protection using region ID.

### U34.10.3 Initialization of Scratchpad memory

Scratchpad Memory initialization is necessary after preset in U34.U34.10.1.
For initialization of the Scratchpad memory SPMI and SPMC0/1 all banks with single setting, refer to registers and setting value in Table U34.4. Read the INITSTS (0xED400004) register to confirm the initializing status. If read value of the memory initializing status register bit 1 is 1, the initialization is finished. INITSTS register shall be accessed as register (SPMI SPMCTRL4 bit 31 shall be 1) to confirm the Scratchpad memory initialization. When Scratchpad memory address is accessed by host CPU, it is accessed as a register in default setting.

Only SPMC0/1 Scratchpad Memory initialization setting is enough if only Pseudo-Power on A2CN0 and preset SPMC.

To access SPMI and SPMC0/1 as Scratchpad memory by CPU (Only 4-byte access is possible), set 0 to the SPMCTRL4 register bit 31 in SPMI. Before changing SPMCTRL4 register bit 31, confirm that there is no access to SPMI/SPMC0/1 memory and registers by outside HOST (CPU etc.) except IMP subsystem.

The order of register setting is SPMI initialization register and SPMC initialization register. In case of EDC/ECC mode first disable the ECM error register in ECM, initialize the memory and enable the ECM errors. In case user want to initialize the each scpatchpad memory bank, set the ScratchPadMemory ConTRL (SPMCTRL) register of each Scratchpad memory bank and read the InITializng STatuS (INITSTS) register by CPU to confirm the memory initialization. In this case, it is user responsibility to confirm that target bank of initialization is not in use.

**Table U34.4** Memory initialization registers

| Register name | Address | Setting value |
|---|---|---|
| ScratchPadMemory ConTRoL eXtra0 (SPMCTRLX0) [SPMI] | 0xFFA8_E300 | 0x80000000 |
| ScratchPadMemory ConTRoL eXtra0 (SPMCTRLX0) [SPMC] | 0xFFAB_2300 | 0x80000000 |

### U34.10.4 Interrupt control

In IMP-X7, number of interrupts output to the INTC is 3, as shown in Figure U34.2. With the INT control logic, it is possible to control interrupts generated in each module.

Interrupts generated in each module are reflected in the Interrupt Status Register (INTSTS) of INT control logic. Interrupts generated in each module can be grouped by selecting arbitrary modules of IMP-X7 with INT control logic. The modules to be grouped can be selected with the G00INTSEL to G02INTSEL registers.

For each group of interrupts, the GROUP00 to GROUP02 bits of the INTSTS register corresponding to the group are set to 1 when all interrupts of the grouped modules have occurred.

Further, if at least one interrupt of a grouped module is cleared, the GROUP00 to GROUP02 bits of the INTSTS register corresponding to the group becomes 0.

An example of operation of the GROUP00 bit when IMP CNN, Cve0, IMP core 0 is selected for the G00INTSEL register is shown below.

- When all three module interrupts have occurred, the G00INTS bit = "1".
- The GROUP00 bit = "0" when one of the three module interrupts is cleared.
- The value of the GROUP00 bit does not change when an interrupt occurs in a module other than 3 modules.
- The value of the GROUP00 bit does not change when an interrupt of a module other than 3 modules is cleared.

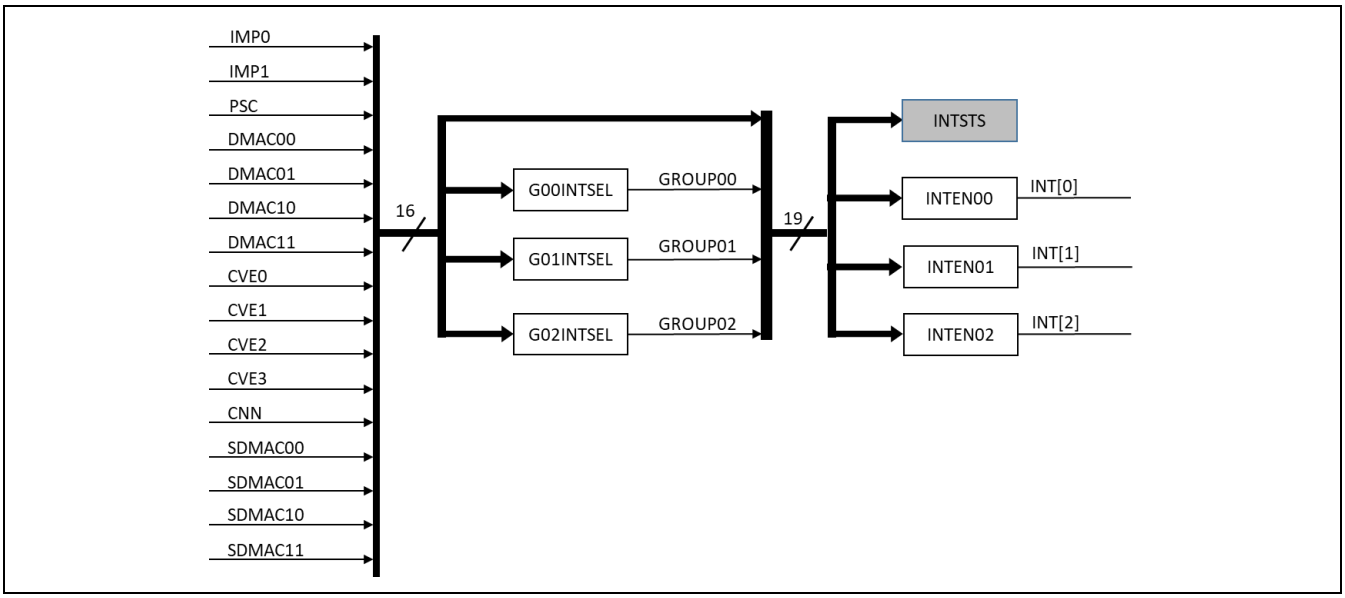



**Figure U34.2 INT control logic**

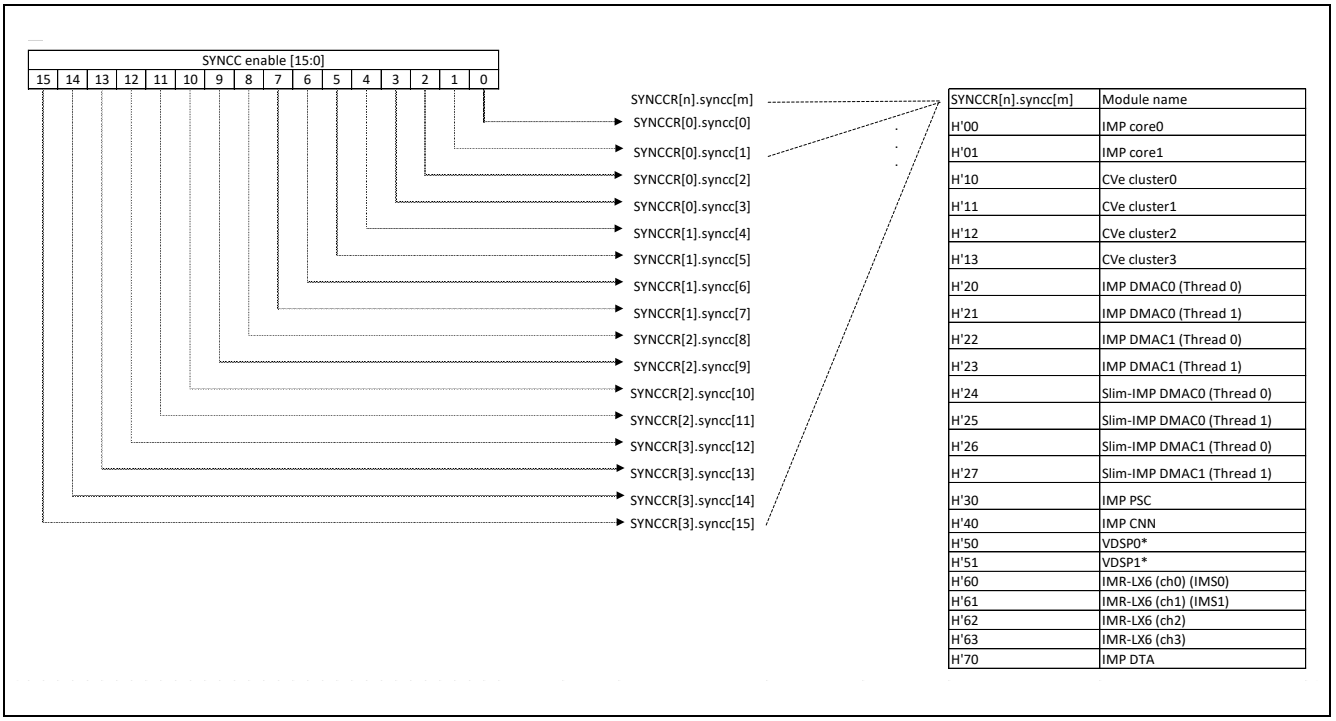
### U34.10.5 Inter Module Synchronization

The synchronization system between modules is the system that synchronizes the control of each CL Controller by specifying the target module and executing the WUP instruction and the SLP instruction for IMP core, CVe core, IMP DMAC, IMP PSC, IMP CNN, slim-IMP DMAC, VDSP and IMP DTA.

For the WUP instruction and the SLP instruction in for different IPs of IMP-X7 sub-system, refer to refer to the individual HWM of IP. For target module specification, refer to the next section.

#### U34.10.5.1 Target module specification

The target modules in WUP instruction and SLP instruction are specified by the SYNCC enable field as shown in Figure U34.3. The target module corresponds to the module specified by SYNCCR[n].syncc[m], which has 1 set in each bit of the SYNCC enable field. Settings not described in below figure are prohibited.




| SYNCCR[n].syncc[m] | Module name |
|---|---|
| H'00 | IMP core0 |
| H'01 | IMP core1 |
| H'10 | CVe cluster0 |
| H'11 | CVe cluster1 |
| H'12 | CVe cluster2 |
| H'13 | CVe cluster3 |
| H'20 | IMP DMAC0 (Thread 0) |
| H'21 | IMP DMAC0 (Thread 1) |
| H'22 | IMP DMAC1 (Thread 0) |
| H'23 | IMP DMAC1 (Thread 1) |
| H'24 | Slim-IMP DMAC0 (Thread 0) |
| H'25 | Slim-IMP DMAC0 (Thread 1) |
| H'26 | Slim-IMP DMAC1 (Thread 0) |
| H'27 | Slim-IMP DMAC1 (Thread 1) |
| H'30 | IMP PSC |
| H'40 | IMP CNN |
| H'50 | VDSP0* |
| H'51 | VDSP1* |
| H'60 | IMR-LX6 (ch0) (IMS0) |
| H'61 | IMR-LX6 (ch1) (IMS1) |
| H'62 | IMR-LX6 (ch2) |
| H'63 | IMR-LX6 (ch3) |
| H'70 | IMP DTA |

**Figure U34.3 Specification of target module in WUP instruction and SLP instruction**

For example, when the following are specified,

SYNCC enable field :B’0000000001001001
SYNCCR0.syncc0 :H’01
SYNCCR0.syncc3 :H’20
SYNCCR1.syncc6 :H’10

Target modules are IMP core1, IMP DMAC (Thread 0) and CVe cluster0.

By using WUP and SLP instructions, it is possible to synchronize between modules shown in Figure U34.3. It is possible to issue the WUP signal mutually with the module shown in Figure U34.3.

**Notes**

1. Specifying multiple identical modules is prohibited when specifying the target module.

   For example, as below, specifying IMP core1 more than one time as target module is prohibited.

   SYNCC enable field : B'0000000000001001
   SYNCCR0.syncc0 : H'01 (IMP core1)
   SYNCCR0.syncc3 : H'01 (IMP core1)

2. IMR-LX6 (IMR) can only WUP/SLP with IMPDMAC and Slim-IMP DMAC
3. SYNCCR register must be set before executing the WUP and SLP instruction.
4. For WUP sending, SYNCCR register of receiving module need to be set before executing the WUP instruction.

5. IMP component number for IMPDTA are as follows.

0 : IMP Core 0
1 : IMP Core 1
2 : CVe 0
3 : CVe 1
4 : CVe 2
5 : CVe 3
6 : IMPDMAC 0 (Thread 0)
7 : IMPDMAC 0 (Thread 1)
8 : IMPDMAC 1 (Thread 0)
9 : IMPDMAC 1 (Thread 1)
10 : Slim-IMPDMAC 0 (Thread 0)
11 : Slim-IMPDMAC 0 (Thread 1)
12 : Slim-IMPDMAC 1 (Thread 0)
13 : Slim-IMPDMAC 1 (Thread 1)
14 : IMPPSC
15 : IMPCNN

6. The policy of the WUP/SLP with VDSP is a little different than other modules. For the WUP/SLP with VDSP, follow the below guideline. IPs (For example: IMP core, CVe, IMP CNN, IMPPSC, IMPDMAC, Slim-IMP DMAC) communicate with VDSP as described in figure below. Selection of the communication type is user responsibility. Note that IMR and IMP DTA cannot communicate with VDSP. Refer to the VDSP HWM for the interrupt mapping.

   a) As shown in Figure U34.4, For example, if user select Polling-based communication between IMPCNN and VDSP0

      - First select the poling-based communication writing IMPSLV.DSP0CT[0] = 0.
      - Clear the DSP0CSTS[17] by writing 0 by CPU. Then start the VDSP0 and IMPCNN by CPU. When IMPCNN sends WUP to the VDSP0 then DSP0CSTS[17] status will become 1.
      - VDSP0 reads DSP0CSTS[17] status and starts its operation if DSP0CSTS[17] status = 1.
      - VDSP0 sends WUP to the IMPCNN and clears the DSP0CSTS[17] by writing DSP0CTRM[17] = 1 (Writing 0 to DSP0CTRM[17] is prohibited).
      - IMPCNN process will continue from SLP state when it receives WUP from the VDSP0.

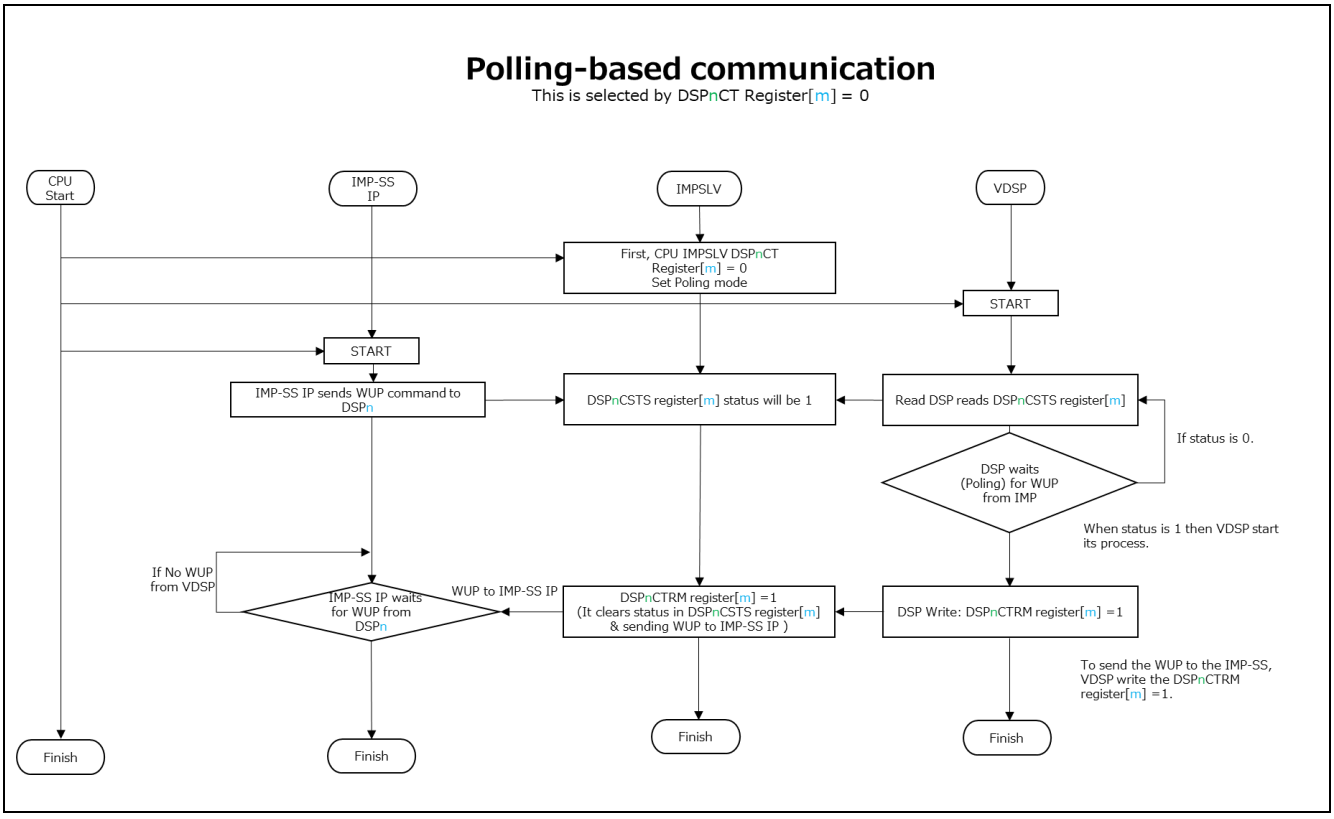



**Figure U34.4** Polling-based WUP/SLP communication with VDSP
[n=0~1] [m=0~12]

b) As shown in Figure U34.5, For example, if user select interrupt-based communication between IMPCNN and VDSP0

- First select the interrupt-based communication writing IMPSLV.DSP0CT[0] = 1.
- Clear the DSP0CSTS[17] by writing 0 by CPU. Then start the VDSP0 and IMPCNN by CPU. When IMPCNN sends WUP to the VDSP0 then DSP0CSTS[17] status will become 1 and an interrupt will assert to VDSP0.
- VDSP0 will receives the interrupt and starts its operation after handling the interrupt.
- VDSP0 sends WUP to the IMPCNN and clears the DSP0CSTS[17] by writing DSP0CTRM[17] = 1 (Writing 0 to DSP0CTRM[17] is prohibited).
- IMPCNN process will continue from SLP state when it receives WUP from the VDSP0.

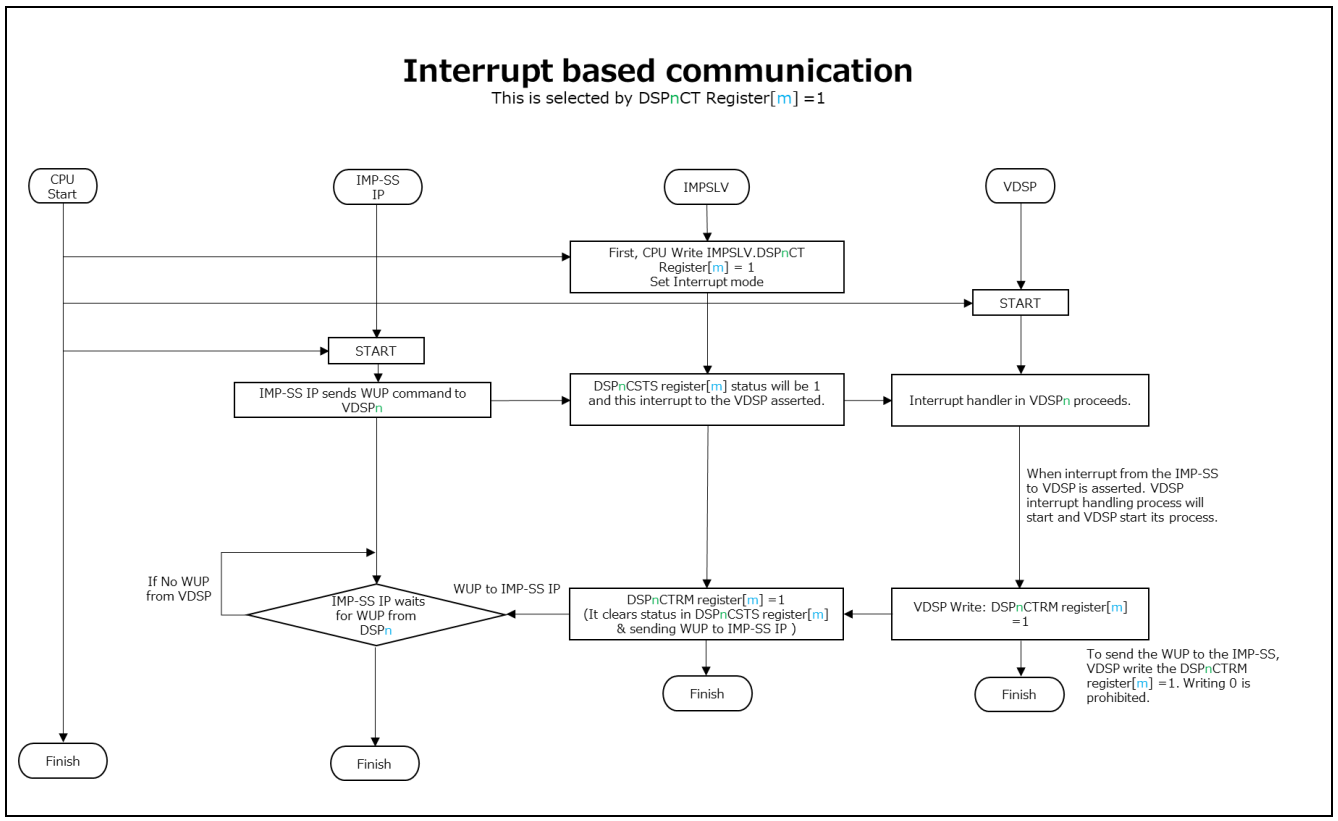



**Figure U34.5** Interrupt-based WUP/SLP communication with VDSP [n=0~1] [m=0~12]

### U34.10.5.2 Operations

Figure U34.6 and Figure U34.7 show examples of release from the sleep state in inter-module synchronization.

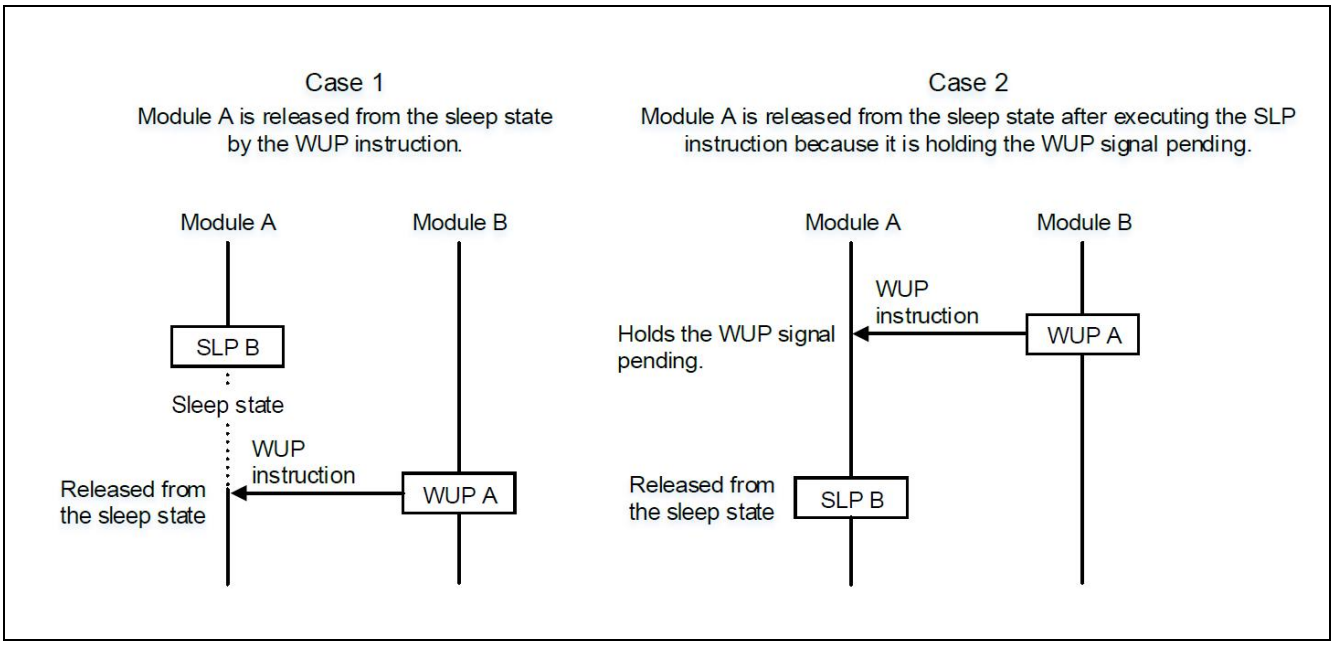



**Figure U34.6 Examples of Release from the Sleep State in Inter-Module Synchronization (Between Two Modules)**

Case 1 Figure U34.6 in is an example of releasing module A from the sleep state when module B executes the WUP instruction. Module A is placed in the sleep state by execution of the SLP instruction. Module A is then released from the sleep state once module B executes the WUP instruction.
Case 2 is an example of releasing module A from the sleep state by the pending WUP signal from module B. If module B executes the WUP instruction when module A is not in the sleep state, module A holds the WUP signal pending. When the subsequent SLP instruction is executed, module A is immediately released from the sleep state because the pending WUP signal is effective.

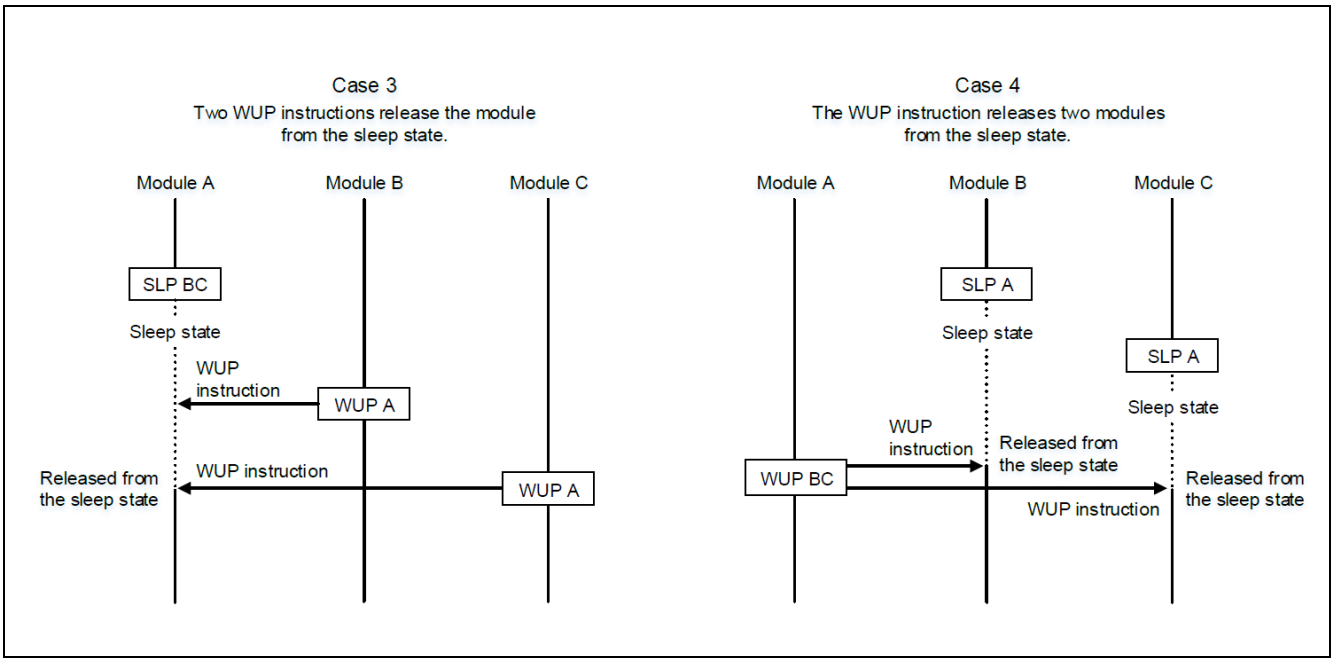



**Figure U34.7 Examples of Release from the Sleep State in Inter-Module Synchronization (between Three Modules)**

Case 3 in Figure U34.7 is an example of releasing module A from the sleep state when modules B and C execute the WUP instruction. Module A is placed in the sleep state by execution of the SLP instruction with modules B and C specified. Module A is then released from the sleep state in response to the WUP instructions executed by both modules B and C.
Case 4 is an example of releasing modules B and C from the sleep state when module A executes the WUP instruction. Modules B and C are placed in the sleep state by execution of the SLP instruction with module A specified. Modules B and C are then released from the sleep state when module A executes the WUP instruction with modules B and C specified.
These steps of release from the sleep state can also be applied to synchronization between four or more modules.

Figure U34.8 shows examples of synchronizing multiple modules in inter-module synchronization.

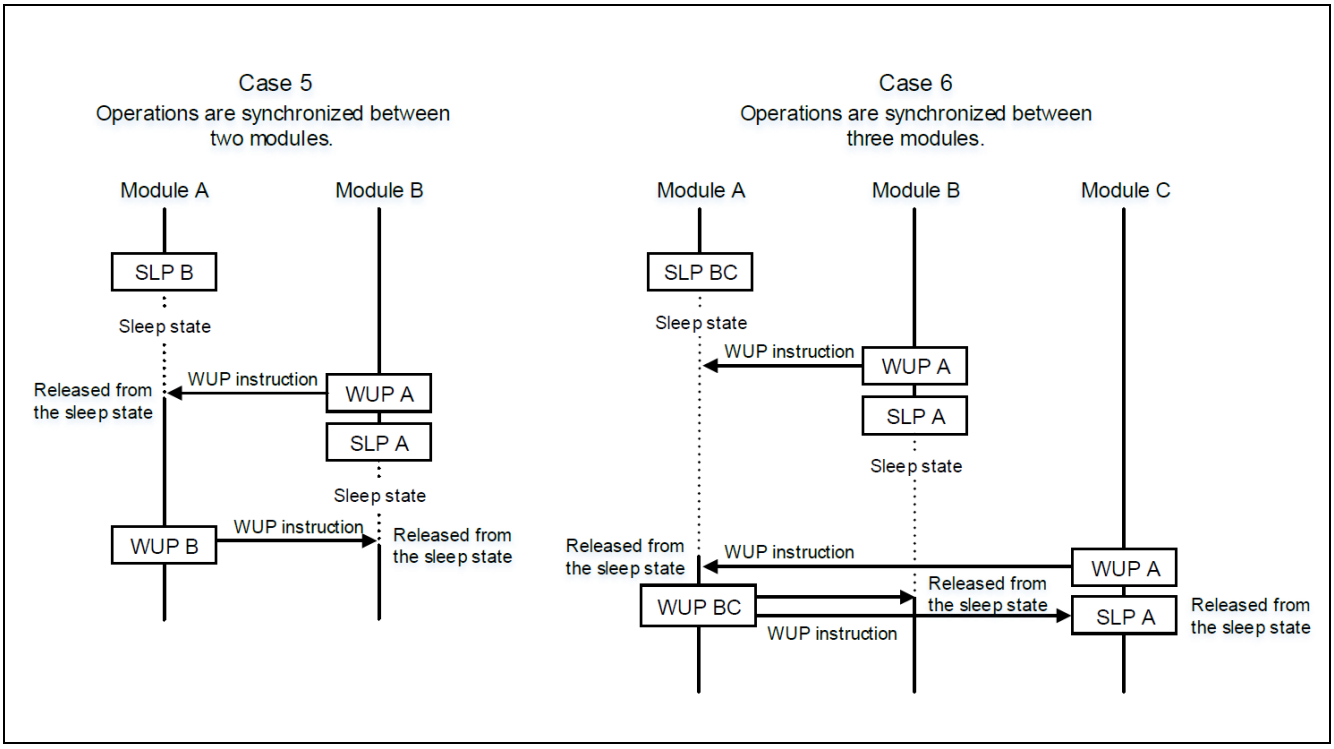



**Figure U34.8 Synchronization of Multiple Modules in Inter-Module Synchronization**

To synchronize operations of multiple modules, select a module as the key module and execute the following steps in the key module and the other modules.

- The key module executes the SLP instruction and then the WUP instruction with the other modules specified.
- The other modules execute the WUP instruction and then the SLP instruction with the key module specified.

Before the key module executes the WUP instruction, the key module has been released from the sleep state that was entered by execution of the SLP instruction and the other modules have executed the WUP instruction. The other modules then execute the SLP instruction, and the key module executes the WUP instruction; this synchronizes the operations of these modules.
Case 5 in Figure U34.8 is an example of synchronizing operations between two modules.
Module A is the key module and module B is the other module. Executing the following steps synchronizes operations of modules A and B.

- Module A executes the SLP instruction and then the WUP instruction with module B specified.
- Module B executes the WUP instruction and then the SLP instruction with module A specified.

Case 6 in Figure U34.8 is an example of synchronizing operations between three modules.
Module A is the key module and modules B and C are the other modules. Executing the following steps synchronizes the operations of modules A, B, and C.

- Module A executes the SLP instruction and then the WUP instruction with modules B and C specified.
- Modules B and C execute the WUP instruction and then the SLP instruction with module A specified.

The steps in case 6 can also be applied to synchronization between four or more modules.

Figure U34.9 shows a case where execution of the WUP instruction is made to wait in inter-module synchronization.

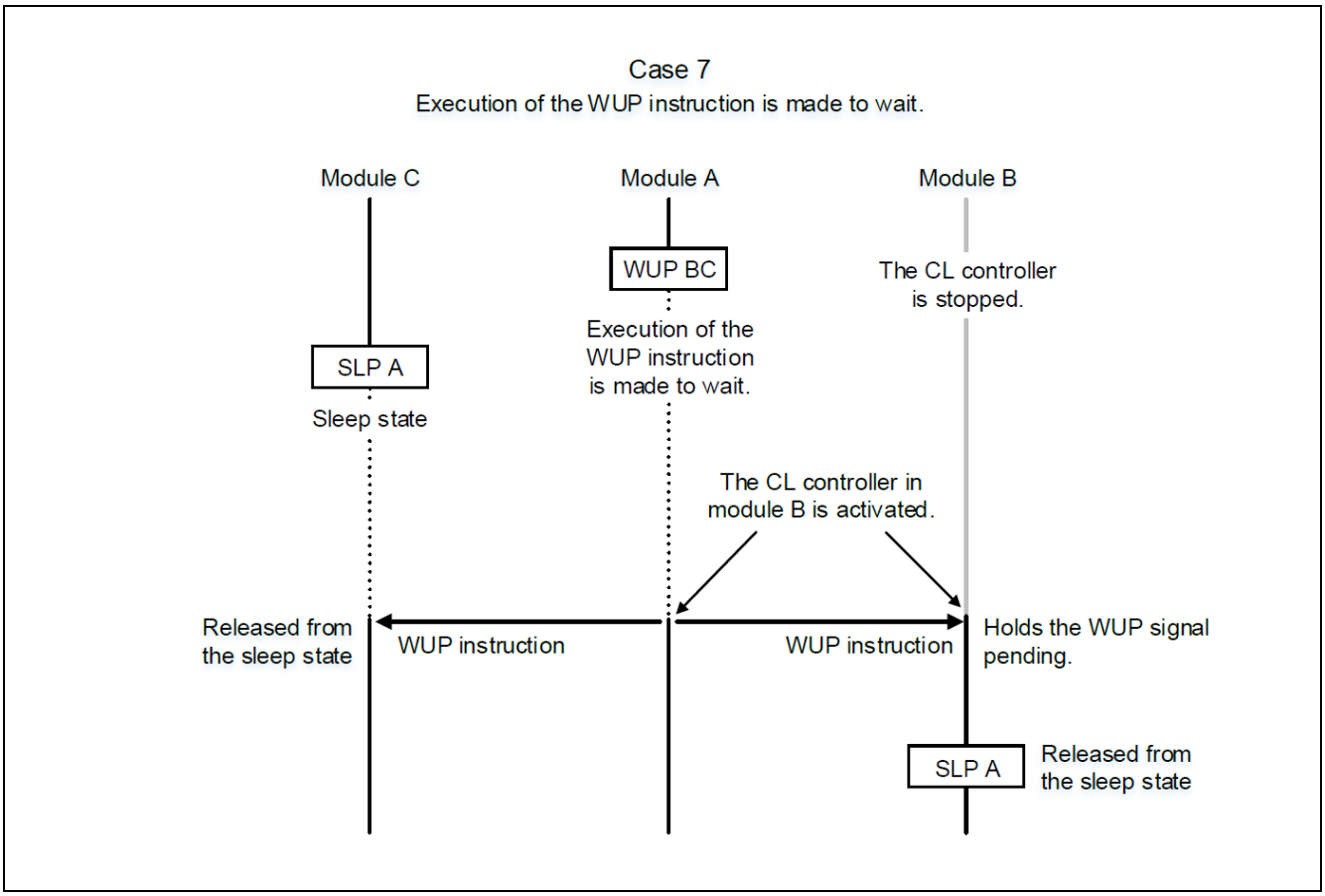



**Figure U34.9 Case where Execution of the WUP Instruction is Made to Wait**

In case 7 in Figure U34.9, execution of the WUP instruction is made to wait. Module A attempts to execute the WUP instruction with module B specified, but execution of the WUP instruction in module A is made to wait because the CL controller in module B is stopped. When the CL controller in module B is activated, module A can execute the WUP instruction.

### U34.10.5.3 Prohibitions

The following two operations are prohibited in inter-module synchronization. This section describes examples of the prohibitions.

- Prohibited operation 1: Execution of the WUP instruction for a module that is holding the WUP signal pending.
- Prohibited operation 2: Execution of the TRAP instruction in a module that is holding the WUP signal pending.

In inter-module synchronization, execution of the WUP instruction for a module that is holding the WUP signal pending is prohibited (prohibited operation 1). Figure U34.10 shows prohibited operation 1 and an example of a workaround in one situation.
While module B is holding the WUP signal pending after module A has executed the WUP instruction for module B, module A must not execute the WUP instruction again for module B before the pending WUP signal is cleared.
To execute the WUP instruction for a single module multiple times, synchronize the current module with the target module and then execute the second WUP instruction.

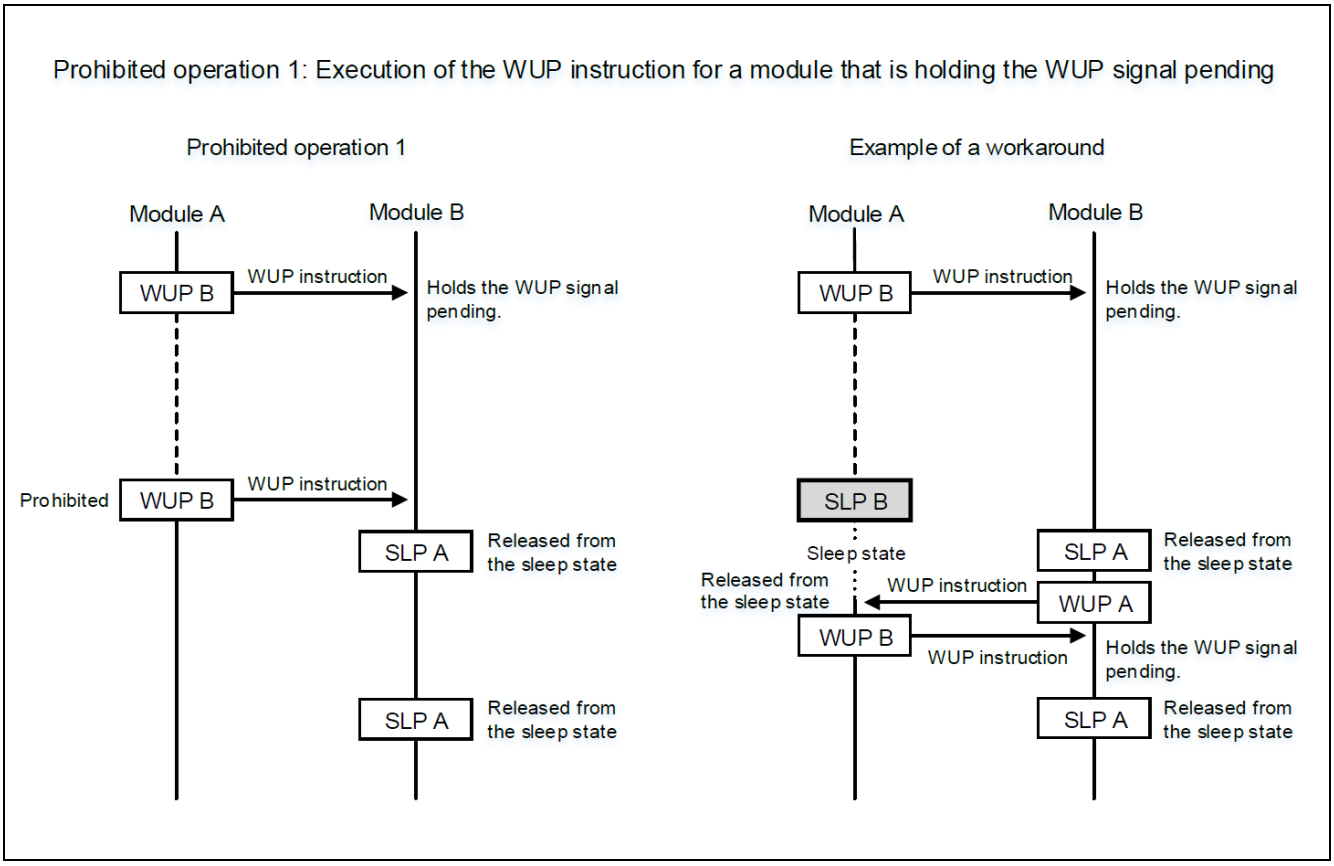



**Figure U34.10 Prohibited Operation 1 and Example of a Workaround**

In inter-module synchronization, execution of the TRAP instruction in a module that is holding the WUP signal pending is prohibited (prohibited operation 2). Figure U34.11 shows prohibited operation 2 and an example of a workaround in one situation.
While module A is holding the WUP signal pending after module B has executed the WUP instruction for module A, module A must not execute the TRAP instruction.
Module A must first execute the SLP instruction to clear the pending WUP signal and then execute the TRAP instruction.

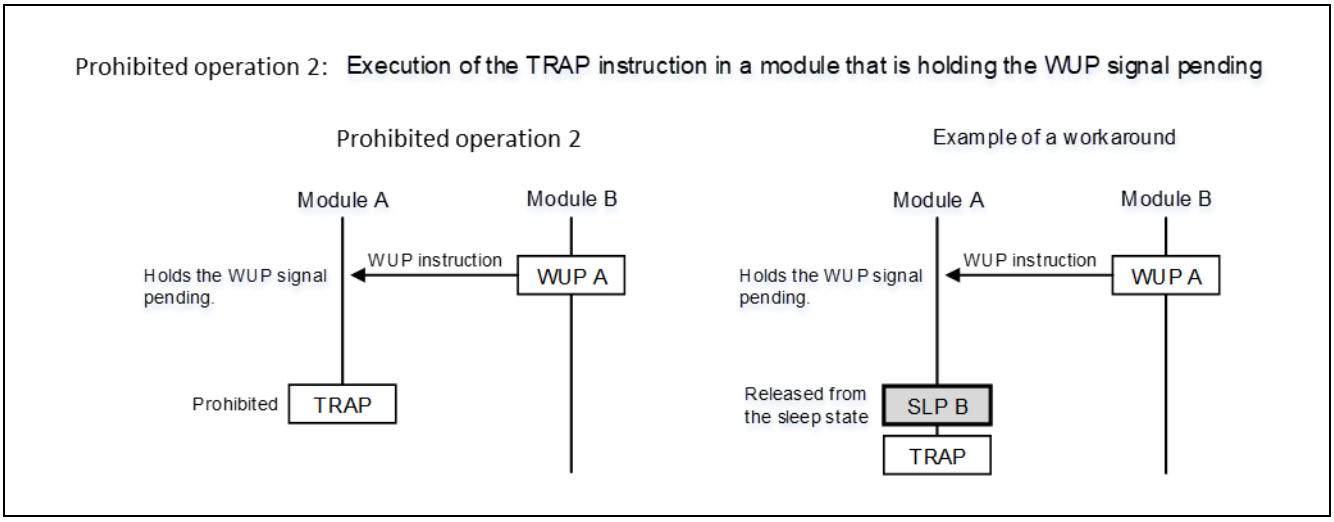



**Figure U34.11 Prohibited Operation 2 and Example of a Workaround**

### U34.10.5.4 Possible Combinations of Modules for Inter-Module Synchronization

| Receiving module / Transmitting module | IMP core0 | IMP core1 | IMP core2 | IMP core3 | CVe 0 | CVe 1 | CVe 2 | CVe 3 | IMP DMAC0.Th0 | IMP DMAC0.Th1 | IMP DMAC1.Th0 | IMP DMAC1.Th1 | Slim-IMP | Slim-IMP | Slim-IMP | Slim-IMP | IMP PSC | IMP CNN | VDSP0 | VDSP1 | IMR (ch0) | IMR (ch1) | IMR (ch2) Ext. | IMR (ch3) Ext. | IMP DTA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMP core0 | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ |
| IMP core1 | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ |
| CVe 0 | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ |
| CVe 1 | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ |
| CVe 2 | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ |
| CVe 3 | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ |
| IMP DMAC0.Th0 | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| IMP DMAC0.Th1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| IMP DMAC1.Th0 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| IMP DMAC1.Th1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Slim-IMP DMAC0.Th0 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Slim-IMP DMAC0.Th1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Slim-IMP DMAC1.Th0 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Slim-IMP DMAC1.Th1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| IMP PSC | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | √ | — | — | — | — | √ |
| IMP CNN | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | √ | — | — | — | — | √ |
| VDSP0 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | √ | — | — | — | — | — |
| VDSP1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | — | — |
| IMR (ch0) | — | — | — | — | — | — | — | — | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | — | √ | √ | √ | — |
| IMR (ch1) | — | — | — | — | — | — | — | — | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ | — | √ | √ | — |
| IMR (ch2) Ext. R&S | — | — | — | — | — | — | — | — | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ | √ | — | √ | — |
| IMR (ch3) Ext. R&S | — | — | — | — | — | — | — | — | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | √ | √ | √ | — | — |
| IMPDTA | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | — | — | — | — | — | — | — |

### U34.10.6 Safety Mechanism

#### U34.10.6.1 Block diagram of safety mechanism

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

Below figures describes the hardware description of IMP-X7 System safety mechanism. Refer to the Figure U34.13 for hardware description of redundant logic of register slice.

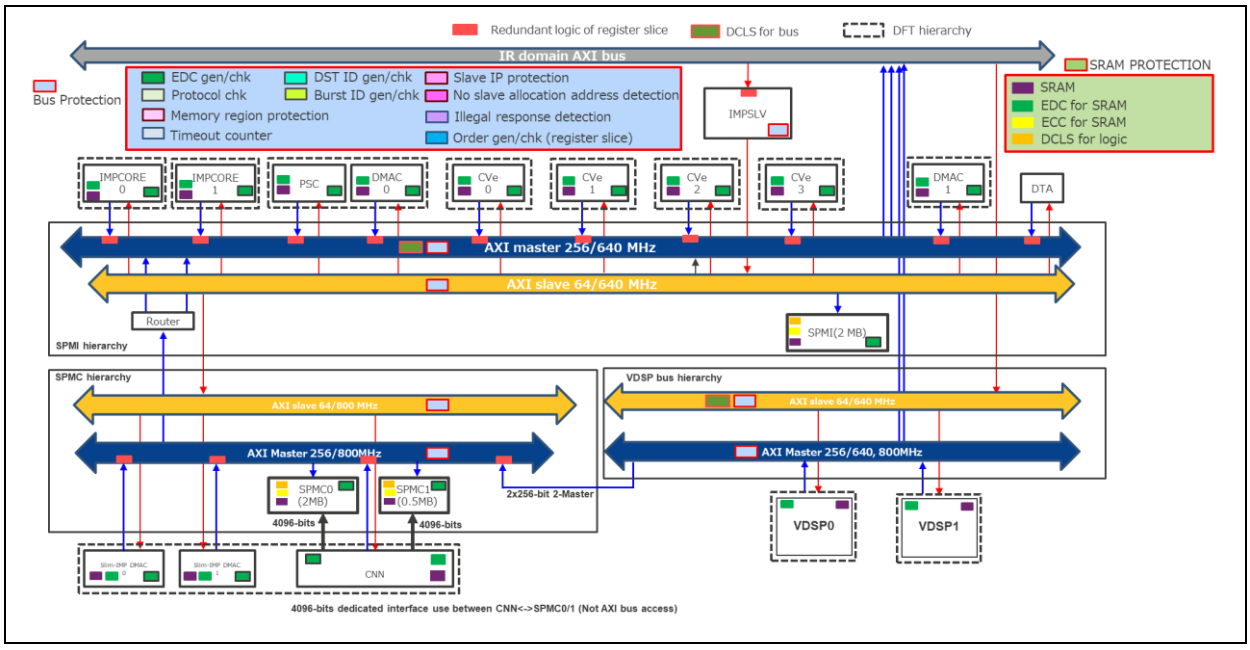



**Figure U34.12 Block Diagram of IMP-X7 (FUSA)**

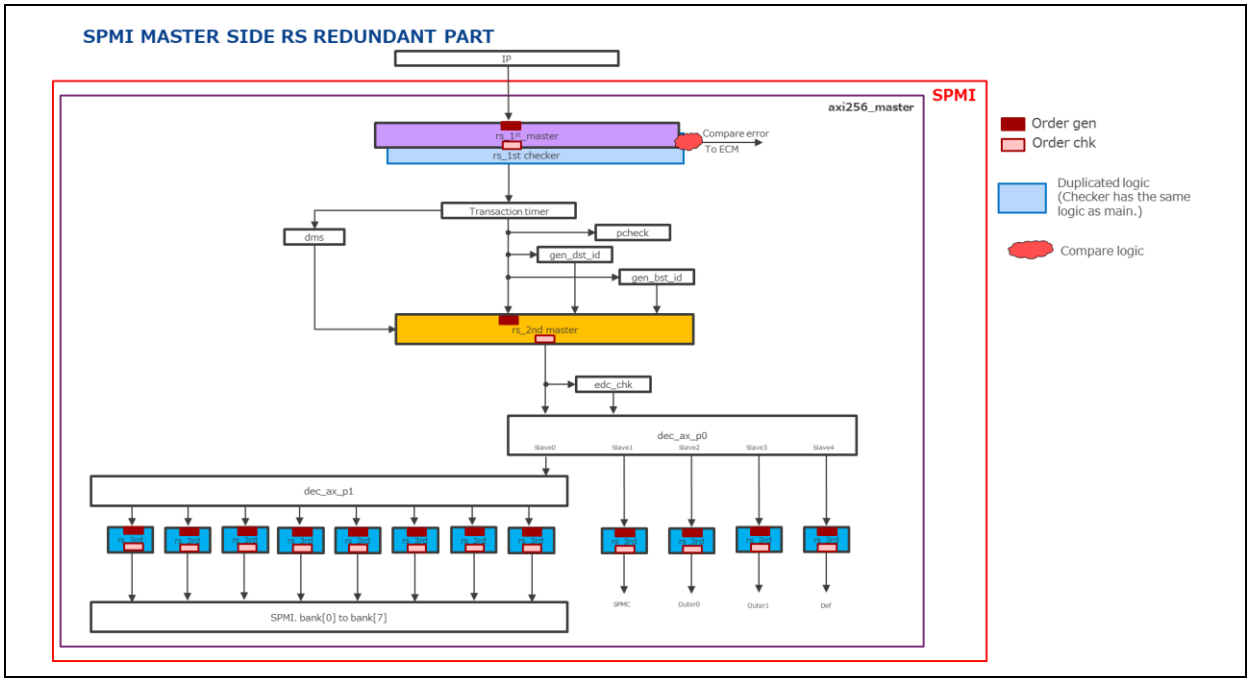



**Figure U34.13 Hardware description for Redundant logic of register slice (Example. SPMI hierarchy)**

### U34.10.6.2 Setting for functional safety

#### 1. Error Correction Code (ECC)

ECC protection is applied to the SRAM for Scratchpad memory (multi-bank Internal memory). To run in ECC mode initialization of the the Scratchpad memory is required. Refer to **U34.10.3** for details of Scratchpad memory initialization.

##### a) Error count threshold setting

Set the error threshold register for ECC error correction. For the module SPMI and SPMC following register setting is necessary.

**Table U34.5** Error thresholding registers

| Register name | Address |
|---|---|
| ScratchPadMemory ConTRoL4 (SPMCTRL4) -SPMI | 0xFFA8_CF18 |
| ScratchPadMemory ConTRoL4 (SPMCTRL4) -SPMC | 0xFFAB_0F18 |

#### 2. Timeout setting

The IMP subsystem local AXI bus has a Request and Response Timeout function that counts latency cycles from transaction request to transaction finish. When this counter exceeds a threshold waiting time, it is recognized as wrong operation of the bus controller and bus error corresponding to the domain to which the IP belongs will occur.

Below are the examples of Timeout function threshold setting.
If TIMESCALE[3:0] in SPMI register is set to N, the counter will count using the frequency obtained by dividing the SPMI clock by 2^N.
For example, if TIMESCALE[3:0]= 6 and SPMI is 640MHz, it counts at 640MHz/64 = 10MHz. Each waiting time in register is set to H’2710 (=10000) cycles at which the threshold at 10MHz is 1 ms.
If TIMESCALE0[3:0] in SPMC register is set to N, the counter will count using the frequency obtained by dividing the SPMC clock by 2^N.
For example, if TIMESCALE0[3:0]= 7 and SPMC is 800MHz, it counts at 800MHz/128 = 6.25MHz. Each waiting time in register is set to H’186a (=6250) cycles at which the threshold at 6.25MHz is 1 ms.

#### 3. IMP local AXI bus protection

There are following safety mechanisms inside IMP local AXI Bus system.

- EDC checker/generator on data, address, and side information for IMP local bus
- Protocol checker
- Memory region protection
- Timeout checker
- Destination ID generator/checker
- Burst ID generator/checker
- Slave IP protection for internal IMP modules
- No slave allocation address detection for IMP local bus
- Illegal response detection for IMP local bus
- Order generator/checker

For the detailed of the (above listed safety mechanisms) refer to the HWM AXI bus of safety mechanism of the **Safety Mechanism for AXI-bus.**

### U34.10.6.3 How to analyse the ECM error

All the FUSA errors are listed below (a total of 4 errors). To confirm the error occurred to the ECM, read the Functional Safety Error Board (FSERRBRD: H'FF9000C0) register bits 19 to 16, bit 7 and bits 1 to 0 of IMPSLV. Following errors can be analysed from the reading.

- IMP-X7 Scratchpad memory DCLS error
- IMP-X7 bus error
- IMP-X7 SRAM ECC error
- IMP-X7 SRAM EDC error

To confirm the ECM error of SPMI, SPMC and each IP, read the following registers in corresponding module.

- SPMI : Functional Safety Error Board (FSERRBRD)
- SPMI : Check the Error STatuS for IP[n] (IPnERRSTS) (n= 0-11) SPMI registers for IP side error.
- SPMI : Error SUMmary n (ERRSUMn) (n= 0-6)
- SPMI : ScratchPadMemory Error n (SPMERRn) (n= 0-1)

- SPMC : Functional Safety Error Board (FSERRBRD)
- SPMC : Check the Error STatuS for IP[n] (IPnERRSTS) (n= 0-3) SPMC registers for IP side error.
- SPMC : Error SUMmary n (ERRSUMn) (n= 0-6)
- SPMC : ScratchPadMemory Error n (SPMERRn) (n= 0-1)

Note: In case of SPMI/SPMC0/1, following signals are not compared by DCLS core between DCLS master instance and DCLS checker instance of Scratchpad memory.

- AXI256 read data
- DATA SRAM write data
- CNN dedicated interface read data.

### U34.10.6.4 Error injection and confirm the Error in ECM

IMP-X7 supports error injection for the in IMPSLV, SPMI and SPMC hierarchy so that error connected to the ECM can be confirmed by injecting the pseudo error. Note that all FUSA error from the IMP IPs (IMP core, IMP DMAC, Slim-IMP DMAC, IMPCNN, CVe), SPMI and SPMC are merged and finally output to the ECM through IMPSLV. Set the dms bit of SPMCTRLX0 register before error injection in scratchpad memory.

Figure U34.14 shows the general example of EDC/ECC error injection procedure in IMP-X7.

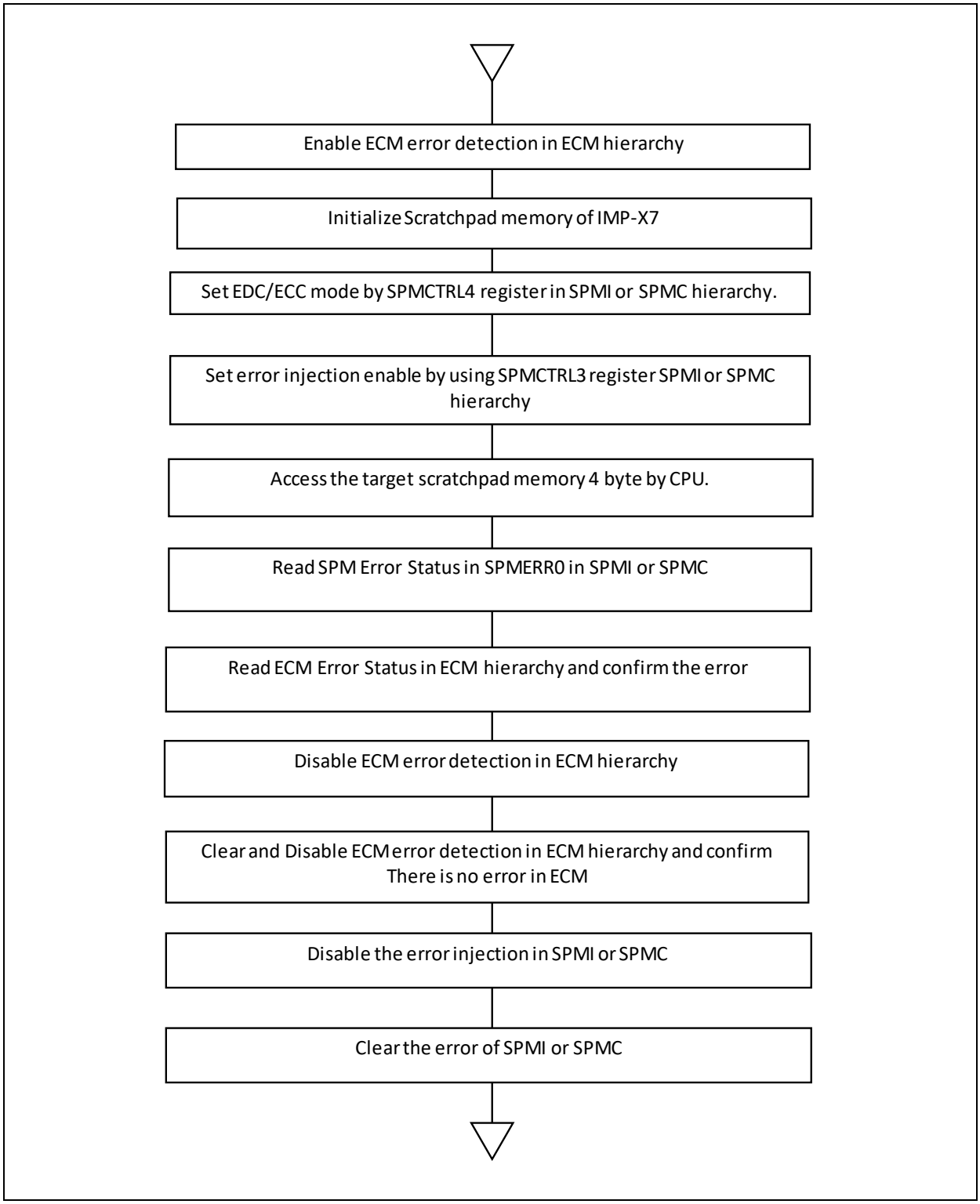



**Figure U34.14 General procedure of Error injection and confirm pseudo error to ECM**

## U34.11 Usage notes

### U34.11.1 Command list pre-fetch maximum data size information for IMP-X7 IP

IMP-X7 IPs (IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC, IMPCNN) has pre-fetch function of command list.
The maximum pre-fetch data size for IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC is 640-bytes. Therefore, CL must be placed in memory block so that 640-bytes space is available for maximum pre-fetch after the TRAP/RET/GOSUB/JUMP instruction.

In case of IMP CNN, the maximum pre-fetch data size for each IP is 36-Kbytes. Therefore, CL must be placed in memory block so that 36-Kbytes space is available for maximum pre-fetch after the TRAP/RET/GOSUB/JUMP instruction.

Note: It is prohibited to compress the Command List (CL), so place the CL including maximum pre-fetch data area only to the uncompressed region.

### U34.11.2 Command list pre-fetch alignment information for IMP-X7 IP

Pre-fetch of CL buffer in IMP-X7 IPs (IMP CNN, IMP core, CVe, IMP DMAC, Slim-IMP DMAC, IMP PSC) take place with 128-byte unit alignment. Therefore, to achieve efficient buffering and pre-fetch data transfer, it is recommended to use memory addresses with 128-byte unit alignment for CL start address, GOSUB address and JUMP address for these IPs. In the case of IMP CNN, additionally, pre-fetch of CL buffer take place with 4-Kbyte unit alignment. Therefore, the recommended memory address alignment is a 4-Kbyte unit for IMP CNN.
When using GOSUB or JUMP instructions, there is a possibility of a pre-fetch data invalidation of data which was already pre-fetched prior to the GOSUB or JUMP instructions. Therefore, pre-fetch data transfer efficiency might be affected by it.

# U36. Debug Trace Agent (IMP DTA)

## U36.1 Overview

### U36.1.1 Features

The IMP-X7 Debug and Trace Agent (IMP DTA) supports debugging of the IMP-X7 sub system. The IMP-X7 sub system consists of several independent components like CV engine, legacy IMP core, iDMA, Pyramidal Scaler. These components can synchronize each other by using a simple trigger mechanism. Each component can trigger another component. Each component can decide which component can trigger it.
To debug this synchronization between the different components the Debug and Trace Agent can interrupt processing of a component or of several components. Additionally, the debug and trace agent will inform the external debug module which component is now stopped and automatically send status updates to the STM trace module.
If IMPDTA is not enabled, each incoming WUP trigger will result in a KICK output trigger to the corresponding IMP component. If IMPDTA is enabled, each incoming WUP trigger will result in a KICK output trigger to the correspondent IMP component either in the next cycle or after the status had been successfully send out to the STM trace module, which can be connected over a direct connection to the DTA. This will reduce the influence of the debug and trace agent to the IMP subsystem behaviour.

### U36.1.2 Block diagram

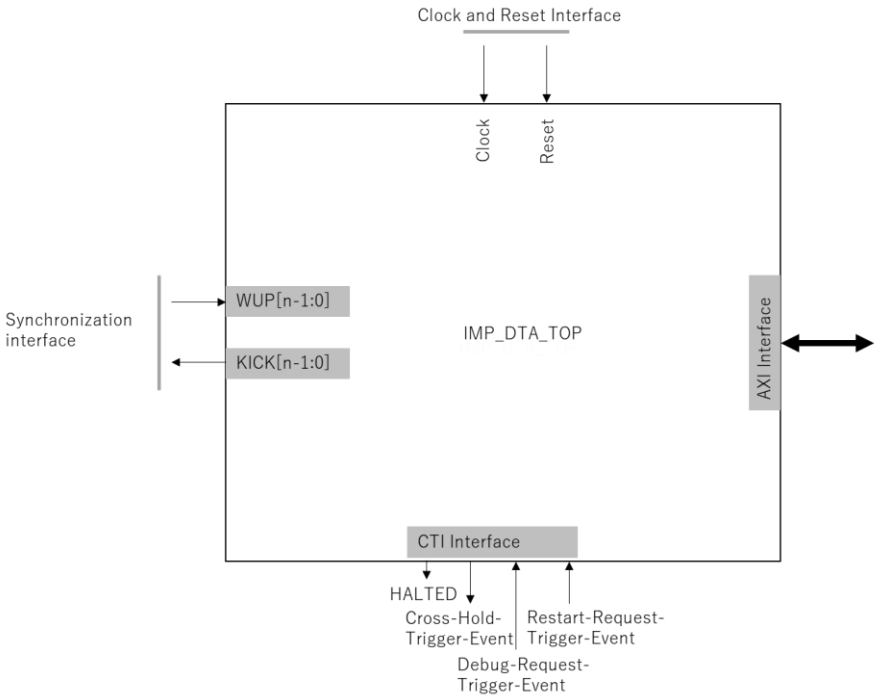



**Figure U36.1 IMP_DTA_TOP Block Diagram**

The Debug Trace Agent uses a standard AXI interface for register configuration. The AXI interface is also used for transferring the trace information to the CoreSight system. This trace information can be either sent out

over the general available 256bit AXI interface or over a 64-bit AXI interface to a bus directly connected to the STM.
The wakeup (WUP) and kick signals are used for the handshaking with the IMP-X7 components.

### U36.1.3 Configuration Process

Software process flow:

1. Reset Release
2. Configure Registers
   a. IMPDTAIDn
   b. IMPDTANDEVCNTLIMITn
   c. IMPDTADBCTRLn
   d. IMPDTACTRL
3. Set IMPDTACTRL.EN bit ‘1’
4. Wait until the configured components are stopped
5. To restart the components, use IMPDTACTRL.GLBKICK/IMPDTADBCTRLn.KICK

Note: Before disabling the component or the different debug modes, the SW must check the status of each component (IMPDTASTATn) to avoid any deadlock situation.

### U36.1.4 DTA Register Configuration

**Table U36.1 DTA Register Configuration Table for Third Generation R-Car Series Products**

| Register Name | Symbol | Access Possible From CP-R/ CP-W[*1] | Value after Reset | Offset Address | RESET |
|---|---|---|---|---|---|
| Control register | IMPDTACTRL | 1/1 | 0x0 | 0x0000 | ✓ |
| OPC value register | IMPDTAOPC | 1/1 | 0x0052E900 | 0x000C | ✓ |
| Break configuration register | IMPDTADBCTRLn[*2] | 1/1 | 0 | n * 0x0040 + 0x10 | ✓ |
| Event counter target register | IMPDTAEVCNTLIMITn[*2] | 1/1 | 0 | n * 0x0040 + 0x14 | ✓ |
| ID number register | IMPDTAIDn[*2] | 1/1 | 0 | n * 0x0040 + 0x18 | ✓ |
| Debug Status register | IMPDTASTATn[*2] | 1/1 | 0 | n * 0x0040 + 0x1C | ✓ |
| Event counter register | IMPDTAEVCNTVALn[*2] | 1/1 | 0 | n * 0x0040 + 0x20 | ✓ |

Note: 1. CP = central processor, R = read, W = write
2. n: IMP Component number, range is 0..17 (Refer to U34. IMP-X7 System, Inter module synchronization for the details about the component connections.)

## U36.2 Register Description

[Legend]
Initial value: Register value after a reset
-: Undefined value
R/W: Readable/writable. The written value can be read.
R: Read-only. The write value should be the same as the read value.
-/W: Write-only. The read value is undefined.

Function for control registers is not defined when using values other than the defined values or values outside the defined areas.

### U36.2.1 IMPDTACTRL register

This register configures the IMP_DTA IP.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | — | — | — | — | — | — | — | — | AXIM_SEL | GLBKICK | GLBBRK | GLBDBMODE | EN |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R/W | -/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 5 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. If a value other than 0 is written, correct operation cannot be guaranteed. |
| 4 | AXIM_SEL | 0 | R/W | AXI bus select signal<br>0: AXIM256 (Default)<br>1: AXIM64 (direct connection)<br>Selection must be done before DTA is used. |
| 3 | GLBKICK | 0 | W | Global KICK<br>0: no function<br>1: triggers a KICK pulse on all KICK signals. Switches to ‘0’ after 1 SYCLK cycle |
| 2 | GLBBRK | 0 | R/W | Global Break<br>0: disabled<br>1: all Components will be stopped |
| 1 | GLBDBMODE | 0 | R/W | Global Debug Mode<br>0: Disabled<br>1: Enabled |
| 0 | EN | 0 | R/W | IMPDTA enable:<br>0: disabled<br>1: enabled<br>IP does not start before setting this bit to ‘1’.<br>Other register configuration shall be done before. |

### U36.2.2 IMPDTAOPC register

This register contains the OPC value for the AXI interface, low and high part.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IMPDTAOPC | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | IMPDTAOPC | | | | | | | | | | | | | | | |
| Initial value: | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0 | IMPDTAOPC | All 0x0052E900 | R | Reserved.<br>These bits are always read as 0x0052E900. The write value should always be 0x0052E900. If a value other than 0x0052E900 is written, correct operation cannot be guaranteed. |

### U36.2.3 IMPDTADBCTRLn register

This register configures the debug behavior for component n.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | — | — | — | — | — | — | — | DLY_KICK | MSGEN | KICK | EVCNTRES | TraceEN | DBMODE |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | -/W | -/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 6 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. If a value other than 0 is written, correct operation cannot be guaranteed. |
| 5 | DLY_KICK | 0 | R/W | DLY_KICK<br>0: KICK is directly sent out<br>1: KICK is delayed till message has been successfully sent out over AXI bus to STM<br>Selection must be done before DTA is used. |
| 4 | MSGEN | 0 | R/W | Message EN<br>0: Only Trace information (32bit CNTVAL) will be sent if wake up trigger received and TraceEN set<br>1: User message and trace information will be sent if wake up trigger received and TraceEN set (Upper 8bit: message, lower 24bit: lower 24bit of 32bit CNTVAL) |
| 3 | KICK | 0 | -/W | Kick<br>1: Kick<br>0: No Kick |
| 2 | EVCNTRES | 0 | -/W | Event Counter Reset<br>1: Counter reset<br>0: no function<br>Switches to '0' after 1 SYCLK cycle. |
| 1 | TraceEN | 0 | R/W | Trace EN<br>0: disabled<br>1: Trace information will be sent if wake up trigger received |
| 0 | DBMODE | 0 | R/W | Individual Debug Mode:<br>0: Disabled<br>1: Enabled |

### U36.2.4 IMPDTAEVCNTLIMITn register

This register configures the event counter limit of component n.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LIMIT | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | LIMIT | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0, | LIMIT | 0 | R/W | Node counter target value.<br>The component will be stopped if the global and individual debug mode is enabled, and the event counter reaches the LIMIT. |

### U36.2.5 IMPDTAIDn register

This register configures the trace ID of component n.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | TRCMSG | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TRCID | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 24 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. If a value other than 0 is written, correct operation cannot be guaranteed. |
| 23 to 16 | TRCMSG | 0 | R/W | User trace message for this component |
| 15 to 0 | TRCID | 0 | R/W | Configures the trace ID for this component |

### U36.2.6 IMPDTASTATn register

This register shows the current debug status of component n.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | — | — | — | — | — | — | — | — | — | — | — | — | — | — | BRK | GLBBRK |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 2 | - | All 0 | R | Reserved.<br>These bits are always read as 0. The write value should always be 0. If a value other than 0 is written, correct operation cannot be guaranteed. |
| 1 | BRK | 0 | R | 1: Component is stopped due to individual break. Bit is cleared if KICK is executed |
| 0 | GLBBRK | 0 | R | 1: Component is stopped due to global break.<br>Bit is cleared if KICK or GLBKICK is executed |

### U36.2.7 IMPDTAEVCNTVALn register

This register shows the current event counter value of component n.

| Bit: | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CNTVAL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Bit: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | CNTVAL | | | | | | | | | | | | | | | |
| Initial value: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CP: R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |

| Bit | Bit Name | Initial Value | R/W | Description |
|---|---|---|---|---|
| 31 to 0, | CNTVAL | 0 | R | Shows the current event counter value. |

## U36.3 Operation

Each synchronization interface is assigned to 1 core process. The individual core processes are configured by the global configuration registers and by the individual configuration registers.

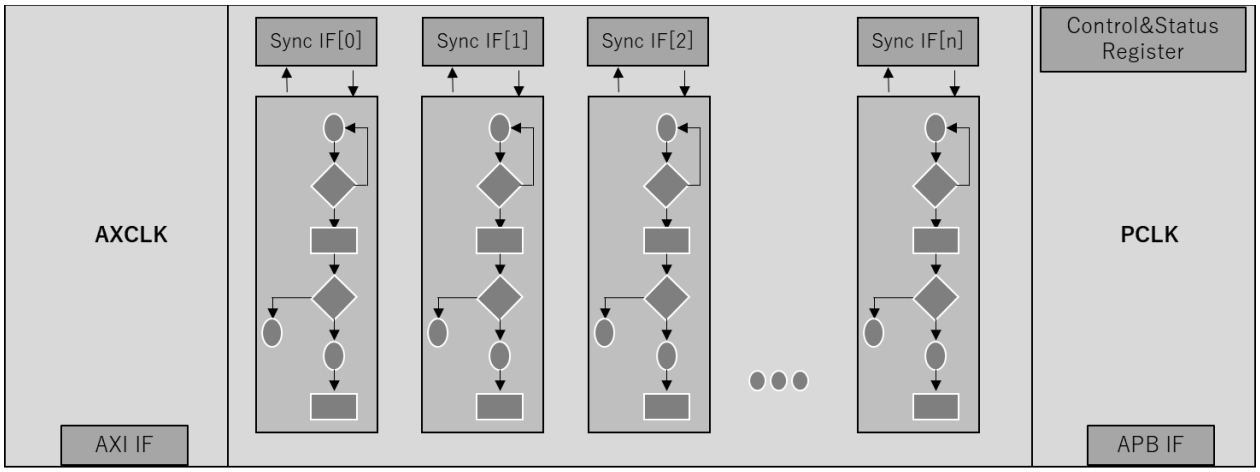



**Figure U36.2 Assignment of Synchronization Interfaces to Core Processes**

The core process is separated into 2 levels:

1) Configuration level
2) Break processing level

The configuration level is 1 single global process for all synchronization interfaces.

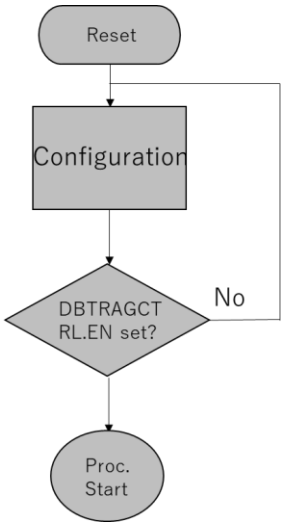



**Figure U36.3 Flow diagram of configuration level process**

The break processing level is implemented as several independent processes, 1 process for each synchronization interface. Figure U36.4 shows the break processing level for synchronization interface/process.

After initialization is done, the DTA is waiting for a WUP signal from other IMP sub modules. When such WUP is received, an internal event counter is increased, and a trace command is sent out to CoreSight system if desired. After, global and local debug mode settings are checked and depending on their settings and the setting of a global or local break point, the DTA unit goes in a global or local wait state from which it can be restarted by receiving a global or local KICK.

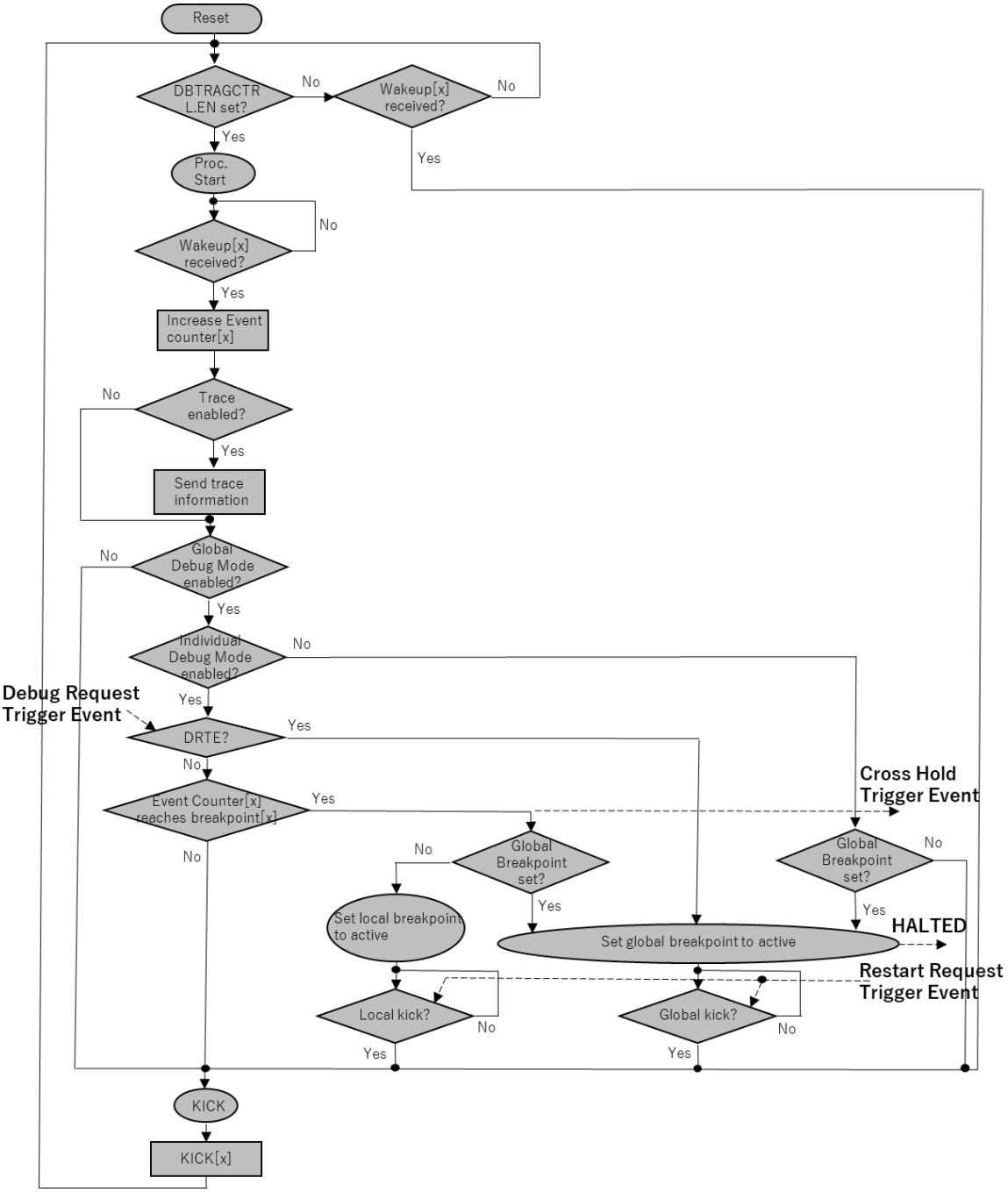



**Figure U36.4 Flow Diagram of break processing level process**

If the IMP DTA is not enabled (EN bit of IMPDTACTRL register is 0), it sends a KICK whenever a wakeup is received. For synchronization with the Cross Trigger Module, the following 4 signals were added:

Cross Hold Trigger Event: Informs Cross Trigger Module of a break of an individual debug mode enabled IMP-X7 component.
Debug Request Trigger Event: Global breakpoint request is set for all enabled IMP-X7 components.
HALTED: Informs Cross Trigger Module that no component is running anymore, they are either not enabled or stopped due to local or global breakpoint.
Restart Request Trigger Event: All enabled components waiting in local or global break are restarted again.

## U36.4 Usage Notes

### U36.4.1 Synchronized system between modules

The synchronization between DTA and other IMP sub modules works as follows. The other IMP sub modules using their WUP/SLP CL commands to get in synchronization with the DTA module. The DTA module does not have such WUP/SLP CL commands, instead, an acknowledge KICK to a received WUP signal is directly in the receiving following clock cycle sent back as WUP to the calling IMP sub module to wake up that sub module from SLP state. A SLP does not exist in the DTA unit.
NOTE: Please make sure that the DTA unit is started before receiving any WUP signals from other IMP sub modules, otherwise, it can cause a hang-up.

### U36.4.2 Power shutdown

For power shutdown, refer to the Section 58.7.1 Shutting Down the Power Supply of IMP-X7 manual.

### U36.4.3 Module Standby Mode

For module standby mode, refer to the Section 58.7.2 Module Standby Mode of IMP-X7 manual.

### U36.4.4 CPG Software Reset

For software reset of CPG, refer to the Section 58.7.3 Software Reset by the Clock Pulse Generator (CPG) of IMP-X7 manual.

## U36.5 Safety Mechanism

Note: The suitability of the safety mechanisms referenced in this manual will finally be judged during the safety analysis and safety assessment of the SoC. Therefore, the safety documentation shall always be considered as additional requirements for the system or software derived from the results of the safety analysis.

### U36.5.1 Block diagram for FuSa

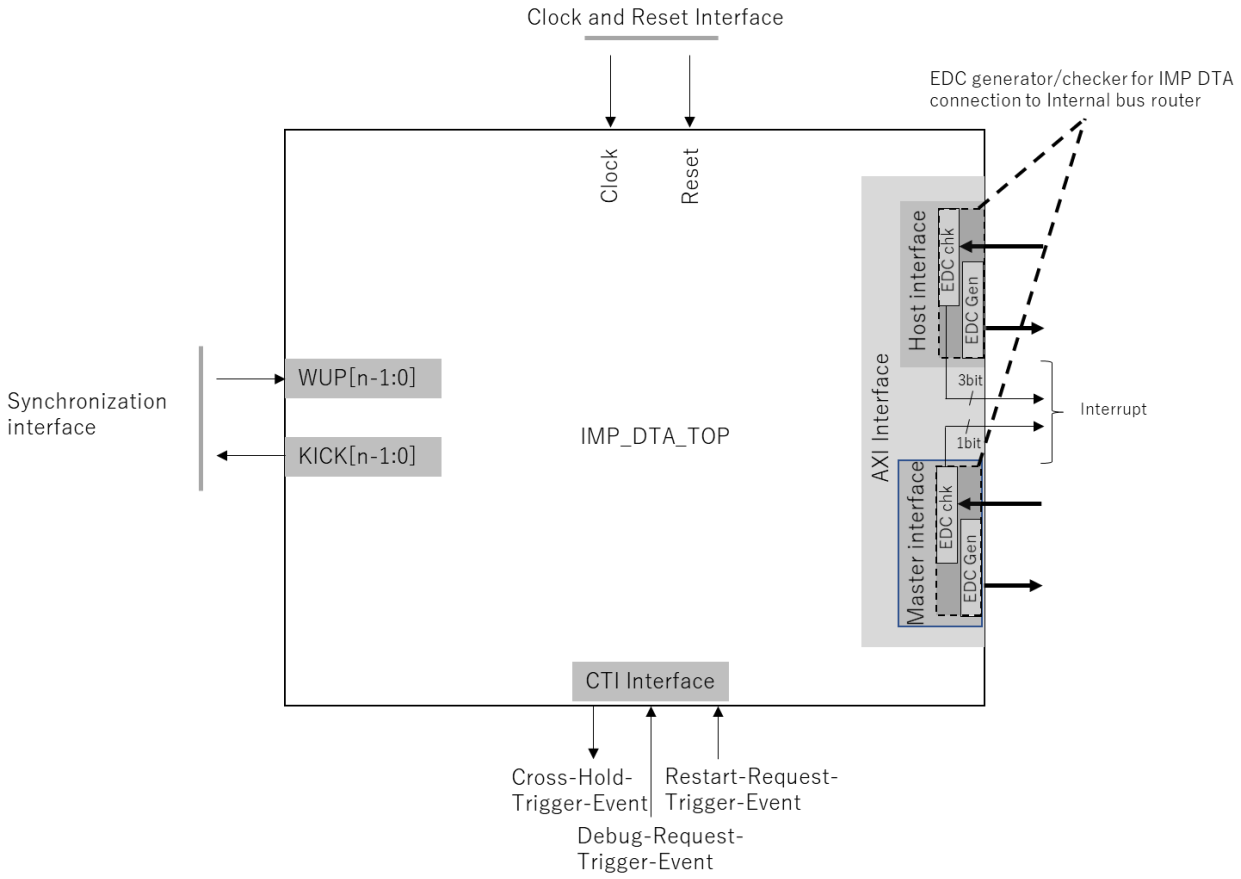



**Figure U36.5 DTA Block diagram for FuSa**

### U36.5.2 Bus I/F

Bus EDC is implemented inside DTA.

# U39. Vision DSP (VDSP)

This chapter describes the Vision DSP (VDSP).

The VDSP has parallel execution performance by 8-way VLIW architecture which allows it to handle a variety of applications.

The vector processing unit adopts the SIMD architecture, and up to 512 8-bit x 8-bit MACs (multiply-accumulate arithmetic units), includes up to 128 16-bit x 16-bit MACs, up to 32 single-precision floating-point MACs.

For the details refer to the implementation guide of CEVA-SP500™ core. Note that chapter number of the implementation guide of CEVA-SP500™ might be different according to the version of guide.

## U39.1 Overview

### U39.1.1 Features

VDSP supports the following features

- 2 VDSPs with CEVA-SP500™ core are available
- The following memories are implemented in each core of VDSP.
    1. 256KByte of internal data memory
    2. 8KByte of internal program memory
    3. 64KByte of instruction cache memory
- It can be started from the instructions from the external memory or its own instruction RAM by specifying the reset vector address
- VDSP PDMA (Program DMA) and DDMA (Data DMA)
- Can access IMP subsystem registers using 4 byte access.
- WatchDog Timer is implemented for functional safety (FUSA) .
- Supports the Cross Trigger Interface.
- All CEVA-SP500™ cores of the VDSP can be debugged using CEVA® toolchain
- Support Runtime Test and power down for each VDSP.
- VDSP can access to other VDSP's memory regions.

Below is the CEVA-SP500™ Configuration table.

Table U39.1 CEVA-SP500™ HW Configuration

| Configuration | Setting | Remarks |
|---|---|---|
| Type | CEVA-SP500™ | 512 MAC configration |
| BTB_WAY | 4 | Branch target buffer n-way |
| BTB_SIZE | 256 | btb total entries |
| BASIC_FULL | BASIC | BASIC + extensions.<br>FULL |
| VPU_FXP_UNITS | 512 | CEVA-SP500™ has 512 8x8 fix point mac |
| LVPU | TRUE | Light VPU as assitance to main VPU (LVCU Light vector computation unit) |
| SPU_FLP_Type | BOTH | 16 & 32 bit operations (HP, SP) |
| SPU_FLP_Units | 2 | |
| VPU_Non_Linear_Units | 16 | Unit count of INV, SQRT,... (Basic 16, Full-extended 32) |
| VPU_FLP_Units | SP32 | |
| VRF | 24 | # of 512b vectors (Basic 24, Full 32) |
| Program_TCM | 8KB | Internal program memory |
| Program_CACHE | 64KB | Instruction cache memory |
| Data_TCM | 256KB | Internal data memory |
| EDP_width | 256bit | External Data master AXI interface (EDP) width (bits) |
| EDAP_width | 128bit | External Data slave AXI interface (EDAP) width (bits) |
| EPP_width | 128bit | External Program master AXI interface (EPP) width (bits) |
| Additional_AXI_Masters | 1port | AXI Master internal-256 bit |
| ECC_WIDTH | 32bit | SRAM ECC is implemented for 32-bit alignment(7-bit) code |
| NUM_TIMERS | 4 | One VDSP has 4 Timers |
| RTT | Internal | Real Time Trace (excludes VCU); location of ETM-R4 trace bus interface |
| ENH_SAFETY | TRUE | Enhance safety. |
| Number of ICU | 2ch | Interrupt control unit |

### U39.1.1.1 **Block Diagram**

The following figures show the block diagram of the VDSP subsystem and one VDSP.

The VDSP subsystem consists of 2 VDSPs with CEVA-SP500 ™. The CPU can access the registers in the 2 VDSP via "AXI Slave / SVD1_IRϕ / 64-bit" of the IR domain AXI bus.

Each VDSP can access the DDR-SDRAM via "AXI Master / SVD1_IRϕ / 128-bit" for instructions, and "AXI Master / SVD1_IRϕ / 256-bit" for data.

In addition, each VDSP can access the SPMI/SPMC memory in IMP-subsystem to share data with IPs in the IMP-subsystem via "AXI Master / IMPAD1ϕ / 256-bit". Refer U34.IMP-X7_System HWM for IMP-subsystem.

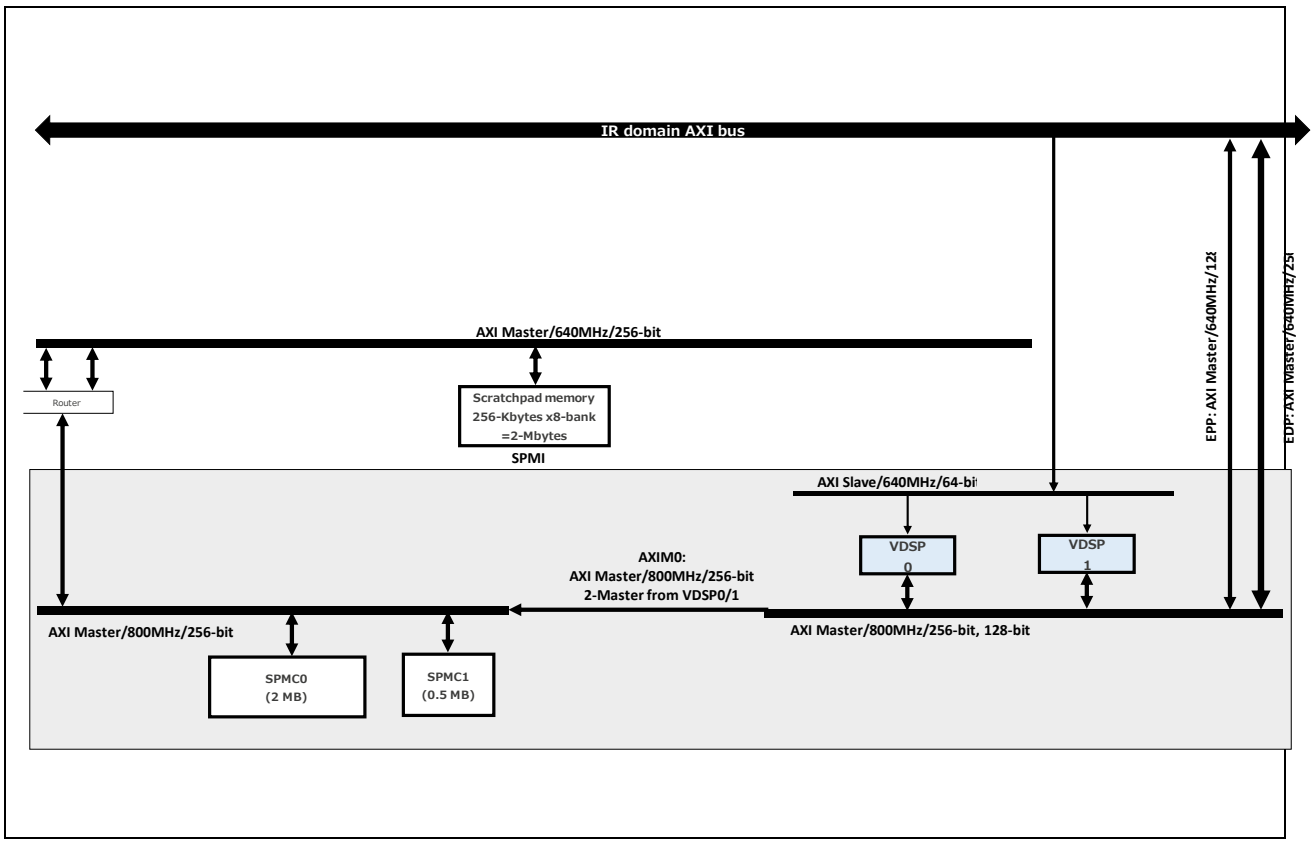



**Figure U39.1.1 VDSP subsystem block diagram**

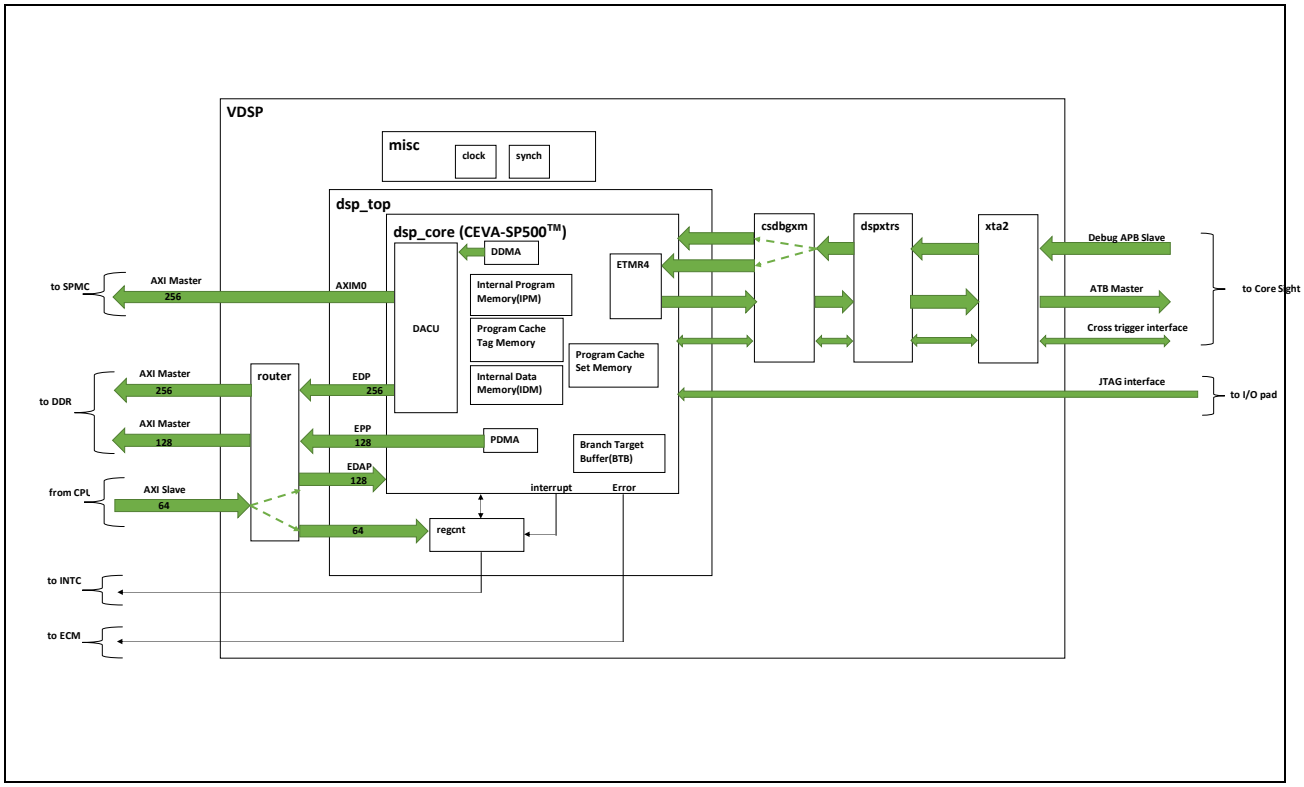



**Figure U39.1.2 VDSP block diagram**

### U39.1.1.2 **Base Address**

Table U39.2 Base Address

| Module | Address |
|---|---|
| VDSP0 | 0xF1400000 |
| VDSP1 | 0xF1600000 |

#### U39.1.1.2.1 VDSP Offset-Address Partitions

Table U39.3 VDSPn Offset-Address Partitions

**n=0-1**

| Module | Block | Sub-block | Offset-Address for CPU * | Offset-Address for CEVA-SP500™ * |
|---|---|---|---|---|
| VDSPn | dsp_core | Internal Data Memory in CEVA-SP500™ (256kByte) | 0x00000 – 0x3FFFF | 0x000000 – 0x03FFFF |
| | | Internal Program Memory in CEVA-SP500™ (8KByte) | 0x40000 – 0x42000 | 0x200000 – 0x202000 |
| | | Internal register in CEVA-SP500™ | 0x80000 – 0x82700 | 0x400000 – 0x402700 |
| | regcnt | Wrapper register | 0xC0000 – 0xCFFFF | - |
| | router | Router register | 0xD0000 – 0xDFFFF | - |

Note: Offset-Address for CEVA-SP500™[22:21] = Offset-Address for CPU [19:18],
Offset-Address for CEVA-SP500™ [20:18] = 0,
Offset-Address for CEVA-SP500™ [17:0] = Offset-Address for CPU [17: 0]}.
When accessing the Internal Data Memory (DTCM) of the DSP from the outside, It is necessary to initialize the all DTCM correctly which automatically add the necessary EDC/ECC code. Otherwise DTCM access from outside cannot be guarenteed. Internal Program Memory (PTCM) cannot be access from outside.

#### U39.1.1.2.2 Offset-Address Partitions for Internal register in CEVA-SP500™

Table U39.4 Offset-Address Partitions for Internal register in CEVA-SP500™

| **Programming Model Sections** | Offset-Address for CPU * | Offset-Address for CEVA-SP500™ * |
|---|---|---|
| MultiCore_Configuration | 0x00080000- 0x000800a8 | 0x00400000 - 0x004000a8 |
| Program_Memory_Subsystem | 0x00080400 - 0x0008057c | 0x00400400 - 0x0040057c |
| PMSS_Safety | 0x00080580 - 0x000805fc | 0x00400580 - 0x004005fc |
| Data_Memory_Subsystem | 0x00080600 - 0x000806ec | 0x00400600 - 0x004006ec |
| Watchdog | 0x00080700 - 0x0008074c | 0x00400700 - 0x0040074c |
| CORE_Safety | 0x00080750 - 0x0008075c | 0x00400750 - 0x0040075c |
| Error_Correction_Codes | 0x00080780 - 0x00080798 | 0x00400780 - 0x00400798 |
| DMSS_Safety | 0x000807b0 - 0x000807fc | 0x004007b0 - 0x004007fc |
| Interface_Configuration | 0x00080900 - 0x00080b28 | 0x00400900 - 0x00400b28 |
| Access_Protection | 0x00080c80 - 0x00080c88 | 0x00400c80 - 0x00400c88 |
| DBG | 0x00080d00 - 0x00080dcc | 0x00400d00 - 0x00400dcc |
| Power_Scaling_Unit | 0x00080e50 - 0x00080e64 | 0x00400e50 - 0x00400e64 |
| Round_Robin_Arbiter | 0x00080fa0 - 0x00080fb0 | 0x00400fa0 - 0x00400fb0 |

| | | |
|---|---|---|
| Queue_Descriptor | 0x00081100 - 0x00081120 | 0x00401100 - 0x00401120 |
| QMAN_Cntrl | 0x00081180 - 0x000811bc | 0x00401180 - 0x004011bc |
| Queue_Manager | 0x00081200 - 0x00081df8 | 0x00401200 - 0x00401df8 |
| Timers | 0x00082000 - 0x0008203c | 0x00402000 - 0x0040203c |
| Interrupt_Control | 0x00082100 - 0x000826fc | 0x00402100 - 0x004026fc |

Note: Offset-Address for CEVA-SP500™[22:21] = Offset-Address for CPU [19:18],
Offset-Address for CEVA-SP500™ [20:18] = 0,
Offset-Address for CEVA-SP500™ [17:0] = Offset-Address for CPU [17: 0]}.

### U39.1.1.3 **Register Configuration on the regcnt block**

The regcnt is a block equipped with registers that control the CEVA-SP500™. Accessing to addresses not listed in this register list is prohibited. Operation cannot be guaranteed when accessing to an address, which is not listed.

Table U39.5 Register Configuration on the regcnt block

**n=0-1**

| Register Name | Symbol | R/W | Address | Initial Value | Access Size |
|---|---|---|---|---|---|
| ICU_EXT_VEC | IEV | R/W | <VDSP[n]> + H'C0000 | H'00000000 | 32 |
| ICU_ACU_LOCK | IAL | R/W | <VDSP[n]> + H'C0004 | H'00000000 | 32 |
| TIMER_CNTRL | TC | R/W | <VDSP[n]> + H'C0008 | H'00000000 | 32 |
| PSU_CNTRL | PC | R/W | <VDSP[n]> + H'C000C | H'00000001 | 32 |
| RESET_VEC | RV | R/W | <VDSP[n]> + H'C0010 | H'00000000 | 32 |
| GPIO_IN | GPIOI | R/W | <VDSP[n]> + H'C0014 | H'00000000 | 32 |
| CORE_ID | COREID | R/W | <VDSP[n]> + H'C0018 | H'00000000 | 32 |
| DDMA_CNTRL | DMACNT | R/W | <VDSP[n]> + H'C001C | H'00000000 | 32 |
| MCACHE_INVLD | MCACHIVL | R/W | <VDSP[n]> + H'C0020 | H'00000000 | 32 |
| DSP_CONTROL | DSPCNT | R/W | <VDSP[n]> + H'C0038 | H'00000001 | 32 |
| JTAG_BYP_METH | JTBYME | R/W | <VDSP[n]> + H'C003C | H'00000000 | 32 |
| OUT_INT_EN_I | OIE_I | R/W | <VDSP[n]> + H'C0040 | H'00000000 | 32 |
| OUT_INT_CLR_I | OIC_I | W | <VDSP[n]> + H'C0044 | H'00000000 | 32 |
| OUT_INT_MSK_I | OIM_I | R/W | <VDSP[n]> + H'C0048 | H'00000000 | 32 |
| OUT_INT_STATE_I | OIS_I | R | <VDSP[n]> + H'C004C | H'00000000 | 32 |
| OUT_INT_EN_D | OIE_D | R/W | <VDSP[n]> + H'C0050 | H'00000000 | 32 |
| OUT_INT_CLR_D | OIC_D | W | <VDSP[n]> + H'C0054 | H'00000000 | 32 |
| OUT_INT_MSK_D | OIM_D | R/W | <VDSP[n]> + H'C0058 | H'00000000 | 32 |
| OUT_INT_STATE_D | OIS_D | R | <VDSP[n]> + H'C005C | H'00000000 | 32 |
| FUNC_TEST | FUNCTST | R | <VDSP[n]> + H'C0070 | H'00000000 | 32 |
| OCM_STATE | OCMSTS | R | <VDSP[n]> + H'C0078 | H'0000000F | 32 |
| PSU_STATE | PSUSTS | R | <VDSP[n]> + H'C007C | H'00000003 | 32 |
| GPIO_OUT | GPIOO | R | <VDSP[n]> + H'C0080 | H'00000000 | 32 |
| VERSION1 | VER1 | R | <VDSP[n]> + H'C0094 | Undefine | 32 |

Note : VERSION1 read value is V4M: H'20211220

## U39.2 Register Description

[Legend]

| | |
|---|---|
| Initial value | : Register value after a reset (hardware reset : This reset is to reset all the logic including bus interface) |
| - | : Reserved. The write value should always be 0. |
| R/W | : Readable/writable. The written value can be read. |
| W | : Write-only. The read value is not guaranteed. |
| R | : Read-only. The write value should always be 0. |

Refer : R-CarV4M_UM_U39_VDSP_r0p50.xlsx

Bits not listed in the referenced file above are reserved read-only bits.

# U39.3 Operation

## U39.3.1 Data DMA Debug Match for Breakpoint Generation

The data DMA (DDMA) can output a debug match indication when a configured address matches the address of the current DDMA transaction.

A match can be done on either an internal TCM address or an external AXI addresses. It can also be done for DDMA transfer direction (upload, download, or IIT).

The DDMA debug match indication is an output at the top level of the SensPro™ where it is called DDMA_MATCH.

The address match indication is cleared either by software writing to the register of the SensPro™ or by asserting an input at the top level of the SensPro™ where it is called DDMA_MATCH_ACK.

For more details about the DDMA_MATCH and DDMA_MATCH_ACK, see “3.9.8 DMA Breakpoint Generation” of “SensPro™ Architecture Specification Volume III (MSS)”.

To clear this indication useing DDMA_MATCH_ACK, set the DDMA_EXT_ACK bit in the DDMA_CNTRL register to 1 and then 0.

To also clear interrupts to INTC or other DSPs, set the OIC_ * _DDMA_DBG_MATCH bit in the OUT_INT_CLR_I or OUT_INT_CLR_D registers to 1.

The DDMA debug match indication also goes to the OCEM and is connected to the Stand-Alone External Breakpoint input (SA_EXT4). This means that the DDMA debug match can trigger a breakpoint in the OCEM, if the OCEM is configured to accept that external breakpoint.

If configured, the OCEM acknowledges the DDMA debug match indication automatically.

For more details about the configuration registers in the OCEM, see the SensPro™ Debug Unit Reference Guide in the release package.

The DDMA address match is triggered when the actual read or write has been completed.

For example, if triggering on the internal address during a DDMA download, the DDMA can perform the memory write over several cycles because of contention issues with higher priority sources on one or more banks.

In this case, the DDMA debug match indication is not asserted until the DDMA has completed the entire write to all banks.

**Figure U39.3.1** shows the DDMA_MATCH, the DDMA_MATCH_ACK, and the OCEM acknowledge.

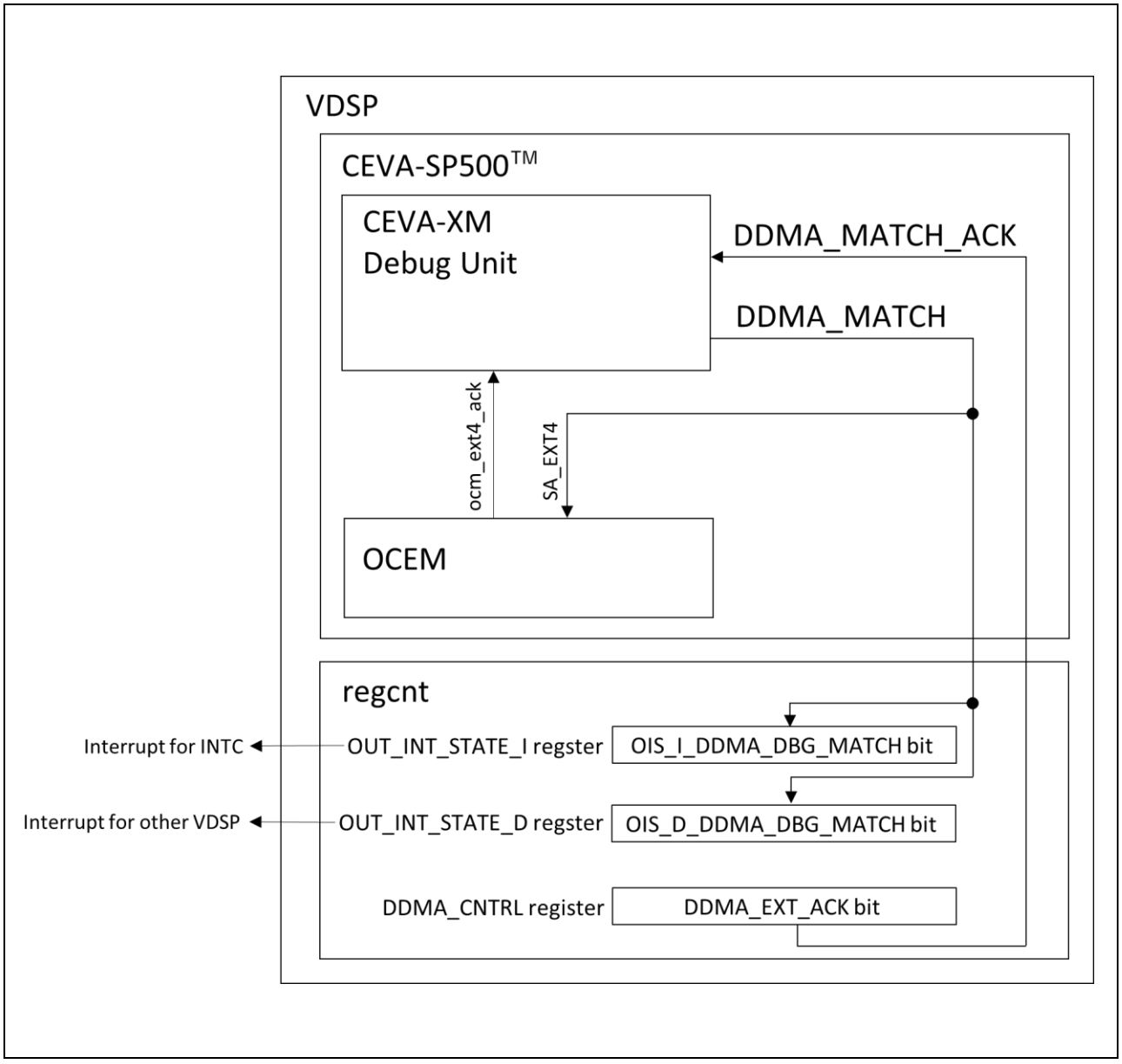



**Figure U39.3.1 DDMA Debug Match Indication**

**Figure U39.3.2** shows an example of a debug match indication and the subsequent acknowledgements.

Note: The input of the external acknowledgment DDMA_MATCH_ACK is used by the synchronized ext_ddma_dbg_match_ack to clear the match indication.

ddma_edp_rd_addr_r : Read Address signals to the EDP port by the DDMA transfer

ddma_edp_rd_r : Read-Address-valid signal to the EDP port by the DDMA transfer

edp_axb_rready : Read-Address-ready signal from the EDP port

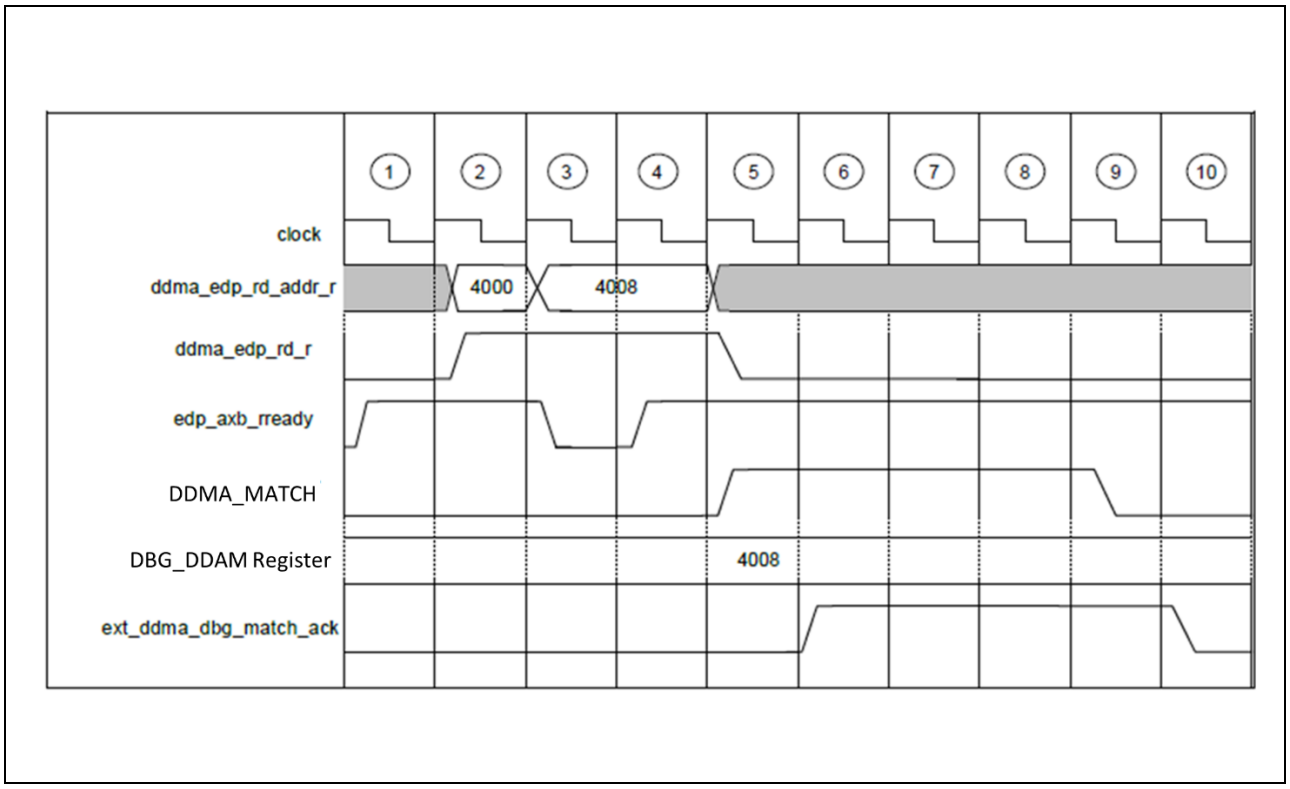



**Figure U39.3.2 DDMA Debug Match Indication Timing Diagram**

### U39.3.2 Program Memory Cache Invalidate Strap.

**Figure U39.3.3** shows the PMSS cache invalidate strap pin sequence.

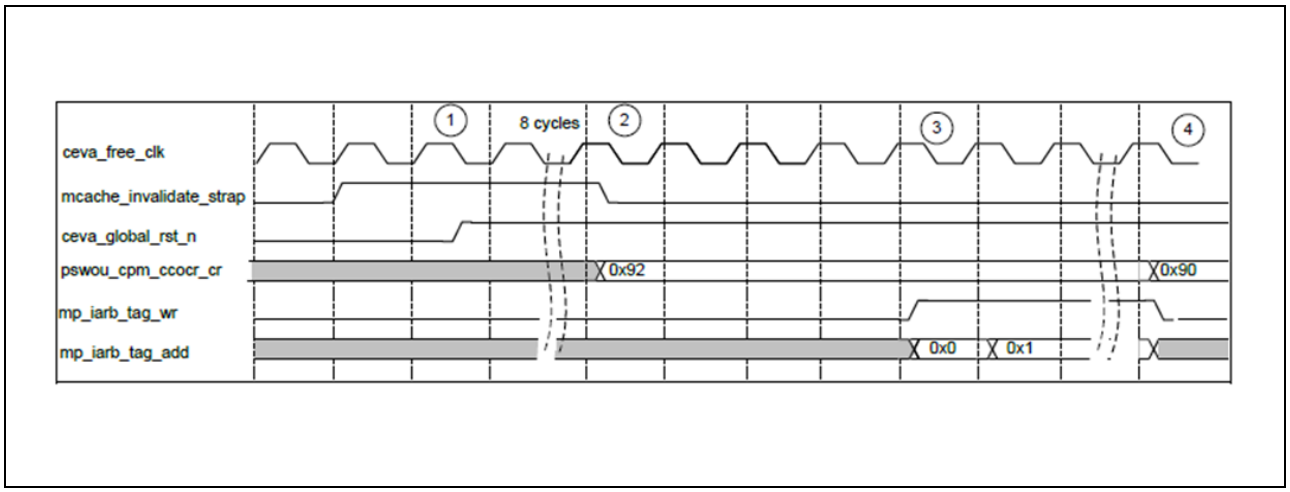



**Figure U39.3.3 PMSS Cache Invalidate Strap pin**

The transaction sequence is as follows:

1) Global reset is released.
2) After eight cycles, the PMSS CCOCR register is configured to start the entire cache invalidation process.
3) Four cycles later, the cache invalidation process begins.
4) When the entire cache has been invalidated, bit 1 in the CCOCRregister is reset, and the user can begin working with the cache

The PMSS cache invalidate strapping signal should be high and stable when the external reset is de-asserted. To enable the PMSS to start the cache invalidate process, the PMSS cache invalidate strap pin should be held for eight cycles after the reset is released. The process is done automatically by setting the software operation of the PMSS to invalidate the entire cache. This strap is active when either the global (ceva_global_rst_n) or system (ceva_sys_rst_n) resets are released.

### U39.3.3 SPMI/SPMC access Method through internal router (DACU)

Data DMA (DDMA) of VDSP can access Scratchpad memory (SPMI/SPMC) in IMP-X7. Accessing SPMI/SPMC other than DDMA is prohibited.

When DDMA access to SPMI/SPMC, it is necessary to set the AXIM0 port in D_ADD*_START register in CEVA-SP500™. VDSP cannot access SPMI/SPMC from the EDP port.

Refer SensPro™ Architecture Specification Volume III(MSS), 3.4 Data Access Control Unit for details on how to set the D_ADD*_START register. Refer to U34.1.3 Address map in U34.IMP-X7_system for the SPMI/SPMC address. When VDSP access Scratchpad memory (SPMI/SPMC0/1) in IMP-X7, initially set 0 to the bit 31 of SPMCTRL4 (0xFFAB0F18) registers in SPMC. Do not change this bit after setting to 0.

Refer SensPro™ Architecture Specification Volume III(MSS), 3.9 Data DMA for details on how to use DDMA. When using DDMA, follow the restrictions described in U39.4.13 AXI Usage.

An example of setting the D_ADD*_START register when using the entire SPMI/SPMC area (0xED30_0000 to 0xED5F_FFFF) is shown below.

1) Set D_ADD1_START register to 0x002ED300

| | | |
|---|---|---|
| REGION_START[19:0] | = | 0xED300 (Start address, 4KB unit) |
| REGION_MID[27:20] | = | 0x02 (AXIM0 port) |
| INACTIVE[28] | = | 0x0 (Active) |
| DPRAW[29] | = | 0x0 |

2) Set D_ADD2_START register to 0x080ED600

| | | |
|---|---|---|
| REGION_START[19:0] | = | 0xED600 (End address, 4KB unit) |
| REGION_MID[27:20] | = | 0x80 (Blank region) |
| INACTIVE[28] | = | 0x0 (Active) |
| DPRAW[29] | = | 0x0 |

### U39.3.3.1 DDMA SPMI/SPMC transfer data

Below are the example in which Data DMA (DDMA) of VDSP taranfer the data form SPMC to IDM of VDSP.

a) Linear data transfer from SPMC to IDM

Set the D_ADD*_START register to select the SPMC region. In below example, D_ADD3_START is used.

- First release the vdsp*_global DSP reset.
- Set **P_PDEA**[*1] = **0x0000 0000 // pload_stop=1->0**
- Set **ICU_ACU_LOCK**[*1] = **0x0000 0008 // External masters can configure the DACU**
- Set **D_ADD3_START**[*1] = **0x002E D400 // SPMC (AXIM0)**
- Set **MSS_DDTC**[*1] = **0x0000 0400 // (1024 bytes) data**
- Set **MSS_DDEA**[*1] = **0xED40 0000 // Data DMA SPMC used as source**
- Set **MSS_DDIA**[*1] = **0x1000 0000 // IDM address used by DMA as destination**
- Set **MSS_DDCL**[*1] = **0x0000 00A0 // (8 transfers (INCR)), Transfer from the external memory to the IDM**

Wait for the DMA transfer end.

Note 1 : Refer "SensPro™ Architecture Specification Volume III (MSS)" for details of these registers.

b) Two dimensional data transfer from SPMC to IDM

Set the D_ADD*_START register to select the SPMC region. In below example, D_ADD3_START is used.

- First release the vdsp*_global DSP reset.
- Set **P_PDEA**[*1] = **0x0000 0000 // pload_stop=1->0**
- Set **ICU_ACU_LOCK**[*1] = **0x0000 0008 // External masters can configure the DACU**
- Set **D_ADD3_START**[*1] = **0x002E D400 // SPMC (AXIM0)**
- Set **MSS_2DCFG1**[*1] = **0x0000 0001 // Select two dimensional transfer indication**
- Set **MSS_2DCFG2**[*1] = **0x0010 0040 // [WIDTHxHEIGHT(64x16=1024 bytes)] data**
- Set **MSS_2DCFG3**[*1] = **0x0040 0000 // Set EXT_STRIDE**
- Set **MSS_DDEA**[*1] = **0xED40 0000 // Data DMA SPMC used as source**
- Set **MSS_DDIA**[*1] = **0x1000 0000 // IDM address used by DMA as destination**
- Set **MSS_DDCL**[*1] = **0x0000 00A0 // (8 transfers (INCR)), Transfer from the external memory to the IDM**

Wait for the DMA transfer end.

Note 1: Refer "SensPro™ Architecture Specification Volume III (MSS)" for details of these registers.

# U39.4 Usage Notes

## U39.4.1 Software Reset Register Comparison table

Table U39.6 Software Reset Register Comparison table

| Register Name (9.2.1 Software Reset Register) | bit | Bit Name | CEVA-SP500™ Interface name | Notes |
|---|---|---|---|---|
| SRCR28 | 31 | SRT31 | - | VDSP1, Reset registers in regcnt |
| | 30 | SRT30 | - | VDSP1, Reset debugging interface |
| | 28 | - | - | - |
| | 27 | SRT27 | WR_RESET | VDSP0, CEVA-SP500™ reset RTT |
| | 26 | SRT26 | OCEM RESET | VDSP0, CEVA-SP500™ resert OCEM |
| | 25 | SRT25 | CORE RESET | VDSP0, CEVA-SP500™ stop |
| | 24 | SRT24 | SYS RESET | VDSP0, CEVA-SP500™ reset System |
| | 23 | SRT23 | GLOBAL RESET | VDSP0, CEVA-SP500™ reset all |
| | 22 | SRT22 | - | VDSP0, Reset registers in regcnt |
| | 21 | SRT21 | - | VDSP0, Reset debugging interface |
| | 4 | - | - | - |
| | 3 | - | - | - |
| | 2 | SRT2 | - | VDSP1, Reset AXI router |
| | 1 | SRT1 | - | VDSP0, Reset AXI router |
| SRCR29 | 23 | - | - | - |
| | 22 | - | - | - |
| | 21 | - | - | - |
| | 20 | - | - | - |
| | 19 | - | - | - |
| | 18 | - | - | - |
| | 17 | - | - | - |
| | 16 | - | - | - |
| | 14 | - | - | - |
| | 13 | - | - | - |
| | 12 | - | - | - |
| | 11 | - | - | - |
| | 10 | - | - | - |
| | 9 | - | - | - |
| | 8 | - | - | - |
| | 7 | - | - | - |
| | 5 | - | - | - |
| | 4 | SRT4 | WR_RESET | VDSP1, CEVA-SP500™ reset RTT |
| | 3 | SRT3 | OCEM RESET | VDSP1, CEVA-SP500™ resert OCEM |
| | 2 | SRT2 | CORE RESET | VDSP1, CEVA-SP500™ stop |
| | 1 | SRT1 | SYS RESET | VDSP1, CEVA-SP500™ reset System |
| | 0 | SRT0 | GLOBAL RESET | VDSP1, CEVA-SP500™ reset all |

### U39.4.2 INTC Comparison table

Table U39.7 INTC Comparison table

| INTC Interrupt Name | VDSP Interrupt output Name | Notes |
|---|---|---|
| dspp0 timer interrupt | intreq_dspp0timer | See “U39.4.4 Interrupt to INTC”. |
| dspp0 interrupt1 | intreq_dspp0[1] | |
| dspp1 timer interrupt | intreq_dspp1timer | |
| dspp1 interrupt1 | intreq_dspp1[1] | |

### U39.4.3 ECM Comparison table

Table U39.8 ECM Comparison table

| ECM Register Name | bit | VDSP output Name | Notes |
|---|---|---|---|
| ECMERRxxxR41 | 31 | VDSP1 errreq_iop_wdog_viol | See “U39.4.5 Interrupt to ECM”. |
| | 30 | VDSP1 errreq_epp_wdog_viol | |
| | 29 | VDSP1 errreq_precise_safety_uncorr | |
| | 27 | VDSP1 errreq_irrecoverable | |
| | 26 | VDSP1 errreq_dmss_corr | |
| | 25 | VDSP1 errreq_dmss_uncorr | |
| | 24 | VDSP1 errreq_corepmss_corr | |
| | 23 | VDSP1 errreq_corepmss_uncorr | |
| | 22 | VDSP0 errreq_icu_wdog_viol | |
| | 21 | VDSP0 errreq_sys_wd_int | |
| | 20 | VDSP0 errreq_axim0_wdog_viol | |
| | 19 | VDSP0 errreq_edp_wdog_viol | |
| | 18 | VDSP0 errreq_iop_wdog_viol | |
| | 17 | VDSP0 errreq_epp_wdog_viol | |
| | 16 | VDSP0 errreq_precise_safety_uncorr | |
| | 14 | VDSP0 errreq_irrecoverable | |
| | 13 | VDSP0 errreq_dmss_corr | |
| | 12 | VDSP0 errreq_dmss_uncorr | |
| | 11 | VDSP0 errreq_corepmss_corr | |
| | 10 | VDSP0 errreq_corepmss_uncorr | |
| | 9 | a3arb intreq_ecm | |
| ECMERRxxxR42 | 3 | VDSP1 errreq_icu_wdog_viol | |
| | 2 | VDSP1 errreq_sys_wd_int | |
| | 1 | VDSP1 errreq_axim0_wdog_viol | |
| | 0 | VDSP1 errreq_edp_wdog_viol | |

### U39.4.4 Interrupt to INTC

Table U39.9 Interrupt to INTC table.

**n=0-1**

| For INTC interrupt output Name (OR of unmasked bits) | VDSP regiser in the regcnt (Each enabled bit holds the entered "1".) | | CEVA-SP500™ Interrupt output |
|---|---|---|---|
| | Regiser neme | Bit Name | |
| intreq_dspp***n***timer | OUT_INT_STATE_I[n] | OIS_I_TIMER00 | TIMER0_OUT |

| | | | |
|---|---|---|---|
| | | OIS_I_TIMER01 | TIMER1_OUT |
| | | OIS_I_TIMER02 | TIMER2_OUT |
| | | OIS_I_TIMER03 | TIMER3_OUT |
| intreq_dspp***n***[1] | | OIS_I_DDMA_DBG_MATCH | DDMA_MATCH |

The intreq_dspp***n***timer interrupt to INTC outputs the OR of the OIS_I_TIMER00 to 03 bits of the OUT_INT_STATE_I[n] register, and the intreq_dspp***n***[1] interrupt outputs the OIS_I_DDMA_DBG_MATCH bit of the OUT_INT_STATE_I[n] register.

The interrupt output by CEVA-SP500™ is input to the OUT_INT_STATE_I[n] register in the REGCNT of each VDSPn when the corresponding bit is enabled by the OUT_INT_EN_I[n] register, and holds the input "1".

Each bit in the OUT_INT_STATE_I[n] register is cleared by writing "1" to the corresponding bit in the OUT_INT_CLR_I[n] register.
Note that the interrupt output factor of the CEVA-SP500™ must be cleared first in order to clear the interrupt.

Each bit in the OUT_INT_STATE_I[n] register can be excluded from the interrupt factor by setting the corresponding bit in the OUT_INT_MSK_I[n] register to "1".

### U39.4.5 Interrupt to ECM

Table U39.10 Interrupt to ECM table.

**VDSPn = VDSP0 - VDSP1**

| VDSP output Name | CEVA-SP500™ Interface name | Notes |
|---|---|---|
| VDSPn errreq_icu_wdog_viol | cevasp_icu_wdog_viol_r | CEVA-SP500™ Indicates ICU Watchdog violation |
| VDSPn errreq_sys_wd_int | cevasp_sys_wd_viol_r | CEVA-SP500™ System watchdog interrupt |
| VDSPn errreq_axim0_wdog_viol | cevasp_axim0_wdog_viol_r | CEVA-SP500™ Indicates AXIMX Watchdog timeout |
| VDSPn errreq_edp_wdog_viol | cevasp_edp_wdog_viol_r | CEVA-SP500™ Indicates EDP Watchdog timeout |
| VDSPn errreq_iop_wdog_viol | cevasp_iop_wdog_viol_r | CEVA-SP500™ Indicates IOP Watchdog timeout |
| VDSPn errreq_epp_wdog_viol | cevasp_epp_wdog_viol_r | CEVA-SP500™ Indicates EPP Watchdog timeout |
| VDSPn errreq_precise_safety_uncorr*3 | P_ECC_ERR | See "4.2.5 Internal Interrupt Map" of "SensPro™ Architecture Specification Volume III (MSS)" for details. |
| VDSPn errreq_irrecoverable | CNSIRE | |
| VDSPn errreq_dmss_corr*1 | CORR_DMSS_SAFETY_ERR | |
| VDSPn errreq_dmss_uncorr*1 | UNCORR_DMSS_SAFETY_ERR | |
| VDSPn errreq_corepmss_corr*2 | CORR_PMSS_SAFETY_ERR | |
| VDSPn errreq_corepmss_uncorr*2 | UNCORR_PMSS_SAFETY_ERR | |
| a3arb intreq_ecm | - | AXI bus error detected |

Notes:

1. Internal data Memory must be initialized before being read. Please refer to "2.2 Error Correction and Detection (ECC)" of SensPro Hardware Safety Mechanism Specification.
2. Internal Program Memory must be initialized before being read. Please refer to "2.2 Error Correction and Detection (ECC)" of SensPro Hardware Safety Mechanism Specification. Entire Program Cache must be invalidated by configuring P_CCOCR register before it is enabled. Please refer to "2.10.3.5.2 Entire-Cache Operation" of SensPro Architecture Spcification Volume III (MSS) for the operation flow.
3. Internal Program Memory must be initialized before being read. Please reer to "2.2 Error Correction and Detection (ECC)" of SensPro Hardware Safety Mechanism Specification.

### U39.4.6 Interrupts between VDSPs

Table U39.11 Interrupts between VDSPs table.

| VDSPn.CEVA-SP500™ Interrupt input Name | VDSP Interrupt output Name | Notes |
|---|---|---|
| VDSP1.EXT_INT[1] | dspp0_internal_int | See “Table U39.12 Interrupt to other VDSP table”. |
| VDSP0.EXT_INT[1] | dspp1_internal_int | |

Table U39.12 Interrupt to other VDSP table.

**n=0-1**

| For othe VDSP Interrupt output Name (OR of unmasked bits) | VDSP regiser in the regcnt (Each enabled bit holds the entered "1".) | | CEVA-SP500™ Interrupt output |
|---|---|---|---|
| | Regiser Neme | Bit Name | |
| dspp***n***_internal_int | OUT_INT_STATE_D[n] | OIS_D_TIMER00 | TIMER0_OUT |
| | | OIS_D_TIMER01 | TIMER1_OUT |
| | | OIS_D_TIMER02 | TIMER2_OUT |
| | | OIS_D_TIMER03 | TIMER3_OUT |
| | | OIS_D_DDMA_DBG_MATCH | DDMA_MATCH |

The dspp***n***_internal_int interrupt to the other VDSP outputs the OR of the all bits of the OUT_INT_STATE_D[n] register.

The interrupt output by CEVA-SP500™ is input to the OUT_INT_STATE_D[n] register in the REGCNT of each VDSPn when the corresponding bit is enabled by the OUT_INT_EN_D[n] register, and holds the input "1".

Each bit in the OUT_INT_STATE_D[n] register is cleared by writing "1" to the corresponding bit in the OUT_INT_CLR_D[n] register.
Note that the interrupt output factor of the CEVA-SP500™ must be cleared first in order to clear the interrupt.

Each bit in the OUT_INT_STATE_D[n] register can be excluded from the interrupt factor by setting the corresponding bit in the OUT_INT_MSK_D[n] register to "1".

## U39.4.7 VDSP startup method

This chapter describes how to start the DSP. There are the following 3 patterns for boot method.

1) Start from the vector address set in register RST_VEC_ADDR.
2) Start from internal program memory after fetching a program from external memory.
3) Preload a program into the program cache memory and boot from the cache memory.

Before booting the VDSP, turn on the power of the VDSP according to the following procedure.

1) Assert the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.

   (In no particular order)

   - WR_RESET
   - OCEM RESET
   - CORE RESET
   - SYS RESET
   - GLOBAL RESET

2) Turn on the power of VDSP by the register of SYSC.

For CPG and SYSC registers, refer to the HW manual for CPG and SYSC.

### U39.4.7.1 Start from the vector address set in register RST_VEC_ADDR

1) Release the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.

   (In no particular order)

   - Reset registers in regcnt
   - Reset debugging interface

   Then, confirm that the reset has been released in the CPG registers.

2) Set the PLOAD_STOP bit in register DSP_CONTROL to 0.
3) If it is necessary to set a unique Core ID for each VDSP, set the CORE_ID register.
4) Set the reset vector in the register RST_VEC_ADDR.

   - Initial value 0x00000000
   - External Memory address 0x40000000～0xBFFFFFFF (SDRAM Area 2GB)

   If RST_VEC_ADDR is set to initial value (0x00000000), the DSP boots from the Internal Program Memory, please refer to U39.4.7.2 for how to boot from the Internal Program Memory.

5) Release the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.

   (In no particular order)

   - GLOBAL RESET
   - SYS RESET
   - CORE RESET

   Then, confirm that the reset has been released in the CPG registers.

6) RST_VEC_ADDR register should not be changed during reset release.

### U39.4.7.2 Start from internal program memory after fetching a program from external memory

1) Release the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.
(In no particular order)
・Reset registers in regcnt
・Reset debugging interface
Then, confirm that the reset has been released in the CPG registers.
2) Set the PLOAD_STOP bit in register DSP_CONTROL to 0.
3) If it is necessary to set a unique Core ID for each VDSP, set the CORE_ID register.
4) Release the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.
(In no particular order)
・GLOBAL RESET
・SYS RESET
Then, confirm that the reset has been released in the CPG registers
5) Activate the PDMA in CEVA-SP500™ to transfer the boot program from external memory to Internal Program Memory. For more information on PDMA, see the “2.11 Program Direct Memory Access” chapter of “SensPro™ Architecture Specification Volume III (MSS)”. The maximum size of the program that can be transferred is 8KByte.
6) After confirming the completion of program download by PDMA in the register P_PDTC inside the DSP, release the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.
・CORE RESET
Then, confirm that the reset has been released in the CPG registers

### U39.4.7.3 Preload a program into the program cache memory and boot from the cache memory

1) Release the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.
(In no particular order)
・Reset registers in regcnt
・Reset debugging interface
Then, confirm that the reset has been released in the CPG registers.
2) Set the PLOAD_STOP bit in register DSP_CONTROL to 0.
3) If it is necessary to set a unique Core ID for each VDSP set the CORE_ID register.
4) Set the reset vector in the register RST_VEC_ADDR.
・Initial value 0x00000000
・External Memory address 0x40000000～0xBFFFFFFF (SDRAM Area 2GB)

If RST_VEC_ADDR is set to initial value (0x00000000), the DSP boots from the Internal Program Memory, please refer to U39.4.7.2 for how to boot from the Internal Program Memory.

5) Release the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.

(In no particular order)

- GLOBAL RESET
- SYS RESET

Then, confirm that the reset has been released in the CPG registers.

6) Transfer the boot program from external memory to program cache memory by controlling the registers of CEVA-SP500™.

a) Invalidate cache:

Set to P_L1ICO = '1' , P_OT = "0100", P_OS = '1' in P_CCOCR register (0x4e4).

b) Confirm completion of cache invalidation:

Read P_ACTIVE and P_L1ICO = '0' in P_CCOCR register (0x4e4).

c) Enable cache:

Set L1IC bit ='1' in P_ADD0_ATT0 registor (0x424).

d) Set the Start Address for pre-fetch to P_CCOSA in the P_CCOSAR register (0x4e0).

e) Start the pre-fetch:

Set to P_L1ICO = '1', P_OT = "0001", P_OS = '0', P_NOBPL = "0100" in P_CCOCR register (0x4e4).

f) Confirm completion of pre-fetch:

Read P_ACTIVE and P_L1ICO = '0' in P_CCOCR register (0x4e4).

7) Release the following VDSP resets by using CPG register. See Refer to Table U39.6 for details.

- CORE RESET

Then, confirm that the reset has been released in the CPG registers.

8) RST_VEC_ADDR register should not be changed during reset release.

### U39.4.8 VDSP stop method

This chapter describes how to stop the VDSP.

To stop the VDSP, transition the PSU (Power Scaling Unit) of CEVA-SP500™ core to the following special modes.

- Light Sleep Mode : The core clock is shut down while memories and ports are enabled to provide upload or download.
- Standby Mode : The core clock and memory clocks are shut down.

To enter Light Sleep Mode / Standby Mode, user need to execute a “psu {lightsleep / standby}” CEVA-SP500™ instruction.

PSU mode can be confirmed by each bit of VDSP wrapper register PSU_STATE as shown in the table below. Refer U39.2 Register Description for PSU_STATE register.

Table U39.13 PSU_STATE values in each PSU mode.

| PSU mode | PSU_STATE register |
|---|---|
| Light Sleep | When the transition to Light Sleep mode is completed, the LIGHT_SLP_MD bit becomes "1". |
| Standby | When the transition to Standby mode is completed, the STANDBY_MD bit becomes "1". |

VDSP can perform “U39.4.10 Power shutdown”, “U39.4.11 Module Standby Mode”, or “U39.4.12 CPG Software Reset” only in the Standby Mode of the PSU mode.

To transition to the Standby Mode of the PSU mode, follow the steps below.

1) Execute psu {standby} instruction of the CEVA-SP500™ in the VDSP.
2) Confirm that the STANDBY_MD bit of the VDSP wrapper register PSU_STATE set to "1".

Note:

For details of each mode and transition method, refer to the following Power Scaling Unit and "4.6 Power Scaling Unit" of "SensPro™ Architecture Specification Volume III (MSS)"

The following sections describe the Power Scaling Unit (PSU). The PSU has the following modes:

- **Free Run**: Related to unit clock control. All clocks to all units are toggling.
- **Dynamic Power Save (DPS)**: Related to unit clock control. Each unit asserts active and gets a clock according to its active status (this is the reset value).
- **Light Sleep**: Related to unit clock control. No core units get clocks.
- **Standby**: Related to unit clock control. No core units or system units (for example, DMSS or PMSS) get clocks.

The modes are changed by writing to the **PSVM** register in the PSU, via the *psu* instruction.

The interrupt can be a wakeup trigger to the PSU for recovery from Light Sleep and Standby mode.

For details about the restrictions and limitations for switching between these modes, see the SensPro™ *Architecture Specification*.

### U39.4.9 Bridge for APB / ATB / CTI (csdbgxm)

APB, ATB and Cross trigger interface go through a bridge called "csbgxm". See Chapter “170 Debug and Trace” for more information on this bridge.

### U39.4.10 Power shutdown

1. Do not attempt access to the registers of the modules in a domain to which the supply of power has been shut down.
2. Before shutting down the power supply to a domain, ensure that no modules in the target domain are operating. Operation cannot be guaranteed if the power supply is shut down while any of the modules is operating.
3. Before Power shutdown in the VDSP, confirm VDSP is Standby mode as describe in “U39.4.8 VDSP stop method”.

For power shutdown, refer to chapter 11. SYSC, Shutting Down the Power Supply.

### U39.4.11 Module Standby Mode

Table U39.6 Module Stop Control Comparison table

| Register Name (9.2.3 Module Stop Control Register) | bit | Bit Name | Notes |
|---|---|---|---|
| MSTPCR28 | 30 | vdsp1_csb | VDSP1, dsp_top block |
| | 21 | vdsp0_csb | VDSP0, dsp_top block |
| | 2 | vdsp1_bus | VDSP1, AXI router block |
| | 1 | vdsp0_bus | VDSP0, AXI router block |

1. Do not attempt access to the registers of the modules in a domain that is in the module standby mode.
2. Before releasing a domain from the module standby mode, be sure to reset the target domain by a software reset from the clock pulse generator. Otherwise, reset the modules once they are released from the module standby mode.
3. Before placing a domain in the module standby mode, ensure that none of the modules in the target domain are operating. Operation cannot be guaranteed if a target domain is placed in the module standby mode while any of its modules is operating.

For module standby mode, refer to chapter 09.Module_standby_Software_reset , Module Standby Mode.

### U39.4.12 CPG Software Reset

1. Do not attempt access to the registers of the modules in a domain to which a software reset is being applied by the clock pulse generator.
2. Before applying a software reset to a target domain by using the CPG, ensure that none of the modules in the domain are operating. A software reset applied to a target domain by the CPG is equivalent to a power-on reset. Accordingly, operation cannot be guaranteed if a domain is reset while any of its modules is operating.

For software reset of CPG, refer to chapter 09.Module_standby_Software_reset, Software Reset by the Clock Pulse Generator (CPG).

## U39.4.13 AXI Usage

### U39.4.13.1 Max Burst Length

D_ADD0_ATT1.DABSZ (CPM Address 0x928), D_ADDx_ATT1.DABSZ or MSS_DDCL (CPM Address 0x68c).BSZ should be set in such a manner that maximum burst length does not exceed the 8 transfers. P_SYS_OU_C (CPM Address 0x404).PDMA_BSZ should be set in such a manner that maximum burst size for PDMA transactions is B'10: 8 beats. Otherwise operation cannot be guaranteed. Refer "SensPro™ Architecture Specification Volume III (MSS)" for the details of D_ADD0_ATT1, D_ADDx_ATT1 and MSS_DDCL registers. Note that, when VDSP DMAC access SPMI/SPMC memory of IMP-Subsystem, only incremental transaction is allowed and possible transaction are 128-byte/256-byte and alignment shall be 32-byte. Refer U34.IMP-X7_System HWM for SPMI/SPMC address map. One transaction must always access within one BANK of SPMI/SPMC.

### U39.4.13.2 Max Outstanding (OS)

When using DDMA, the efficiency of data transfer may vary depending on the setting of read and write outstanding numbers at D_ADD0_ATT1.DADOL/D_ADD0_ATT1.DAUOL or D_ADDx_ATT1.DADOL/D_ADDx_ATT1.DAUOL or MSS_DACC.DOL/MSS_DACC.UOL registers. The recommended outstanding setting for maximum data transfer efficiency is 16-outstanding for both read and write transactions. An outstanding setting greater than 16 cannot be used.

When using PDMA, the efficiency of data transfer may vary depending on the setting of outstanding numbers at P_SYS_OU_C.PDMA_OS (CPM Address 0x404) register. The recommended outstanding setting for maximum data transfer efficiency is 8-outstanding for PDMA.

### U39.4.13.3 Prohibit Narrow Access with DDMA

D_ADD0_ATT1.EXACT_RD (CPM Address 0x928), D_ADDx_ATT1.EXACT_RD should be set to 0 to prevent Narrow access when using DDMA on the Ceva-SP500™. Refer "SensPro™ Architecture Specification Volume III (MSS)" for the details of D_ADD0_ATT1 and D_ADDx_ATT1 registers.

### U39.4.13.4 Other AXI attribute settings

External AXI bus does not support the exclusive access, therefore do not use AXI Exclusive Access for read and write when using the program or the PDMA/DDMA. For the details, refer "SensPro™ Architecture Specification Volume III (MSS), 3.13.2 External Memory Exclusive Access Synchronization".

Also, the external AXI bus must be accessed by a transaction with AxCACHE = 0x0010, Therefore always set L2A_WR and L2A_RD = 4'b0010 in the D_ADD0_ATT1 or D_ADDx_ATT1 register. And always set P_L2A = 4'b0010 in the P_ADD0_ATT0 or P_ADDx_ATT0 register.

The CEVA-SP500™ has two privilege levels, "Supervisor mode" and "User mode," which determine the value of AxPROT at the AXI Master Port. However, external AXI buses handle these privilege levels as "user mode," regardless of User (unprivileged) access or Supervisor (privileged) access from CEVA-SP500™.

Note: The External AXI bus explained in this chapter are EDP, EPP and AXIM0 described in Figure U39.1.1.

## U39.4.14 JTAG Usage

The JTAG interface of the VDSP is cascaded with up to 2 VDSPs. Before using the JTAG interface, the DBGEN input of the VDSP must be set to 1. Please refer to Chapter "170 Debug and Trace" for the setting method.

Also, it is necessary to set the JTAG_BYP_METH register (0xC003C) as shown in the table below, depending on the number of VDSPs implemented.

Table U39.15 Register settings for the JTAG interface

| Number of VDSP | JTAG_BYP_METH register (0xC003C) | |
|---|---|---|
| | JTAG_METHOD bit | JTAG_BYPASS[2:0] bits |
| 1 | VDSP0: 1’b1 | VDSP0: 3’b000 |
| 2 | VDSP0: 1’b0<br>VDSP1: 1’b0 | VDSP0: 3’b110<br>VDSP1: 3’b111 |

### U39.4.15 ATB Usage

Set the ETMTRACEIDR register of the ETM-R4 built into the VDSP to a value between 0x01 and 0x6F. The ETM-R4 can be configured via the APB interface. For more information about APB, Refer to Chapter "170 Debug and Trace" for details.

### U39.4.16 Interrupt Mapping

The mapping of the VDSP interrupt inputs is shown in the table below.

Table U39.16 Interrupt Mapping table

n = 0 - 1

| Interrupt Number | CEVA-SP500™ Interrupt Name | VDSPn Interrupt input | Notes |
|---|---|---|---|
| 0-8 | Internal Interrupt | | See "4.2.5 Internal Interrupt Map" of "SensPro™ Architecture Specification Volume III (MSS)" for details. |
| 9-32 | - | | - |
| 33 | EXT_INT[1] | VDSP1 : dspp0_internal_int<br>VDSP0 : dspp1_internal_int | See Table U39.12 Interrupt to other VDSP table.<br>See OUT_INT_STATE_D register in chapter "Register Description" for interrupt factors. |
| 34-35 | - | | |
| 36 | EXT_INT[4] | IMP core0 | See “U34.10.4 Interrupt control” of "U34.IMP-X7_system" for details. |
| 37 | EXT_INT[5] | IMP core1 | |
| 38 | - | | |
| 39 | - | | |
| 40 | EXT_INT[8] | CVe0 | |
| 41 | EXT_INT[9] | Light-CVe1 | |
| 42 | EXT_INT[10] | Light-CVe2 | |
| 43 | EXT_INT[11] | Light-CVe 3 | |
| 44 | EXT_INT[12] | IMPDMAC00 | |
| 45 | EXT_INT[13] | IMPDMAC01 | |
| 46 | EXT_INT[14] | IMPDMAC10 | |
| 47 | EXT_INT[15] | IMPDMAC11 | |
| 48 | EXT_INT[16] | Slim-IMP DMAC00 | |
| 49 | EXT_INT[17] | Slim-IMP DMAC01 | |
| 50 | EXT_INT[18] | Slim-IMP DMAC10 | |
| 51 | EXT_INT[19] | Slim-IMP DMAC11 | |
| 52 | EXT_INT[20] | IMPPSC | |
| 53 | EXT_INT[21] | IMPCNN | |
| 54-63 | - | | - |

### U39.4.17 DMA Queue manager (QMAN)

- Four queues are configurable per DSP
- Programmable queue location and size (either direct access to the IDM or access to the external memory) is supported.
- Separate descriptor push and activation stages are supported; tasks pushed to the queue are later enabled according to the system state.
- Descriptor activation is controlled by software activation.
- Task descriptors are pulled from queues.
- Tasks are split into programmable chunk sizes per queue.
- Priority of queues arbitrating for DDMA access is based either on preconfigured queue priority or on priority in the task descriptor.
- Queues with the same priority arbitrate according to the data packet order of execution.
- Queues with the same priority and same execution order arbitrate using a round-robin policy. Tasks are activated only if the source and destination are ready for a chunk/task transaction (using buffer management).
- Several tasks can be bundled into a task frame.

### U39.4.18 Unsupported CEVA-SP500™ features

VDSP does not support the following CEVA-SP500™ features. Do not use these functions.

- Exclusive external memory access synchronization, which uses an AXI interface.
- Multi-core messaging interface (MCCI)
- IDM Snooping mechanism
- Queue manager (QMAN) with VDSP multiple cores

## U39.5 Safety Mechanism

### U39.5.1 Functional Safety

VDSP is equipped with the following Safety Mechanism.

- The Runtime test detects a failure in the Logic section.
- For Memory, 1-bit error correction and 2-bit error detection are performed.
- Anomaly is detected by the watchdog timer.

The block diagram of FUSA is shown below.

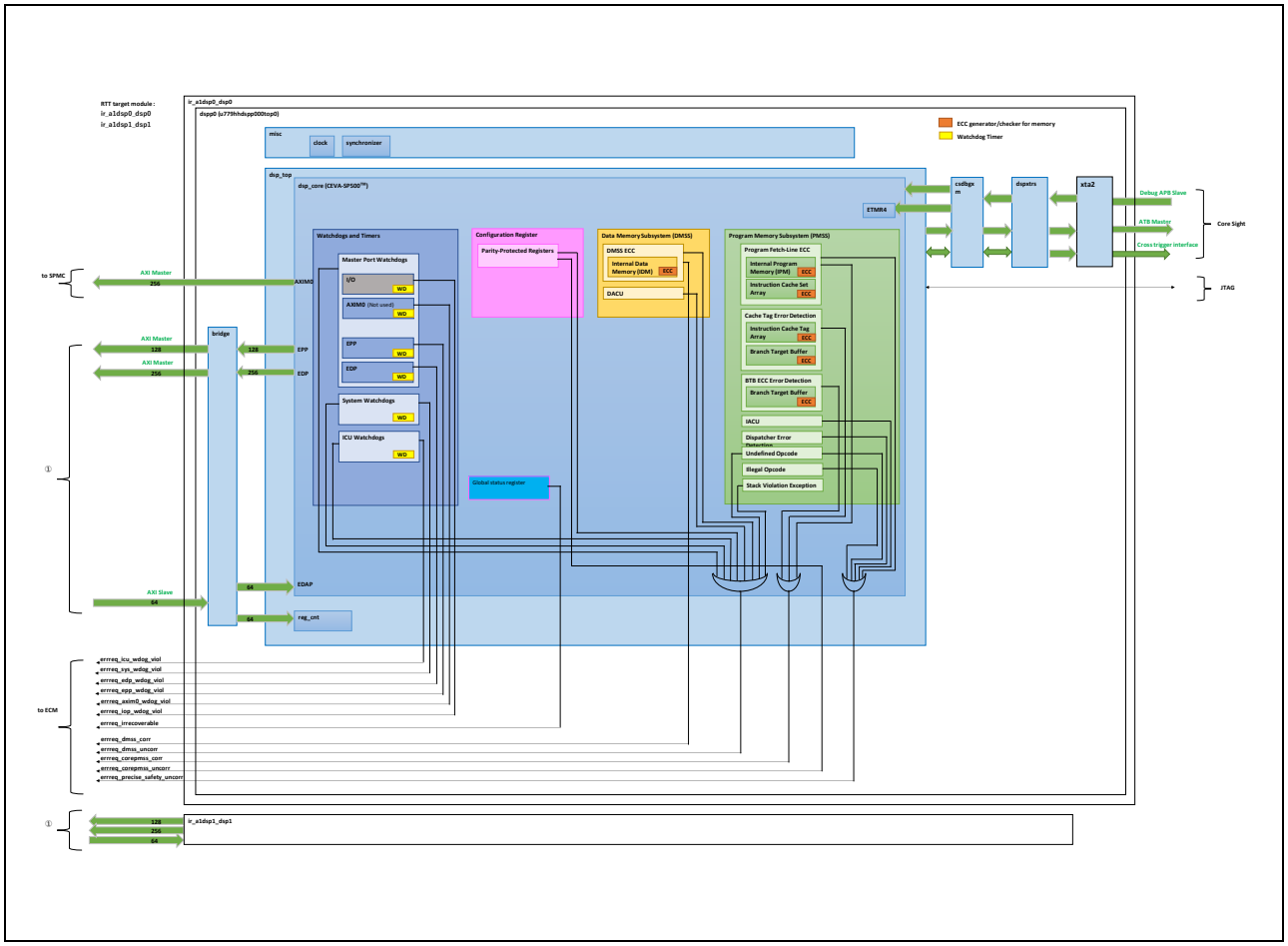



**Figure U39.5.1 FUSA block diagram(1)**

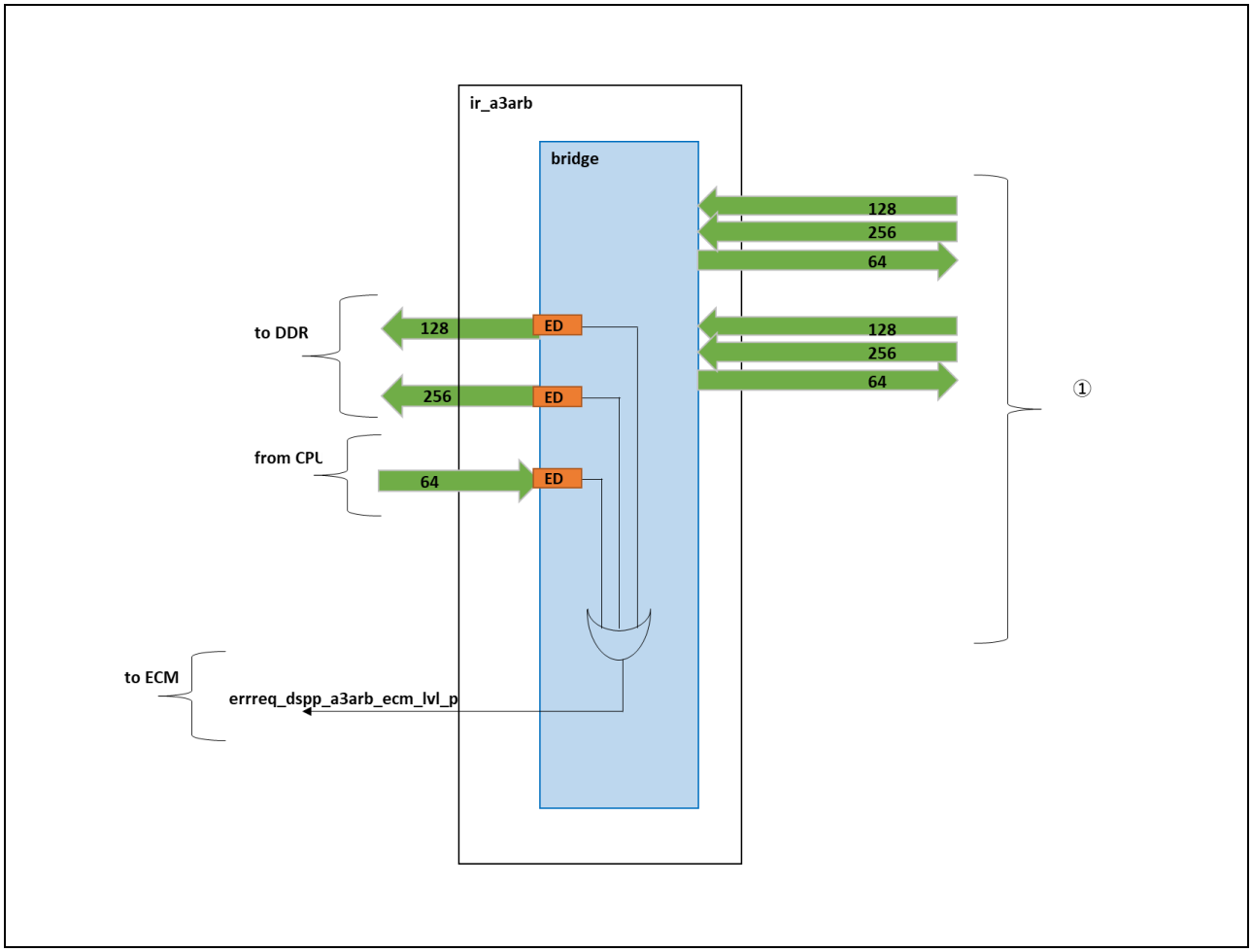



**Figure U39.5.2 FUSA block diagram(2)**

The following signal is output as an interrupt signal addressed to ECM.

Table U39.17 Table of ECM errors

| Error name | Description | Comments |
|---|---|---|
| errreq_corepmss_uncorr | Core and PMSS uncorrectable safety error | For a detailed description of these interrupt signals, please refer to “SensPro™ Hardware Safety Manual”. |
| errreq_corepmss_corr | Core and PMSS correctable safety error | |
| errreq_dmss_uncorr | DMSS uncorrectable safety error | |
| errreq_dmss_corr | DMSS correctable safety error | |
| errreq_irrecoverable | Irrecoverable error in non-secure mode | |
| errreq_precise_safety_uncorr | Core uncorrectable precise error | |
| errreq_epp_wdog_viol | Indicates EPP Watchdog timeout | |
| errreq_edp_wdog_viol | Indicates EDP Watchdog timeout | |
| errreq_axim0_wdog_viol | Indicates AXIM0 Watchdog timeout | |
| errreq_sys_wd_int | Indicates System Watchdog min/max violation | |
| errreq_icu_wdog_viol | cevasp_icu_wdog_viol_r | |
| a3arb intreq_ecm | Bus EDC error on a3arb | errreq_dspp_a3arb_ecm_lvl_p |

### U39.5.2 Region ID

For the VDSP subsystem secure protection, set the region ID using the RGID(R,W)_AXIDSPx registers / RGIDM_DSPxx registers located in the IR domain AXI bus interfaces, and the SPMI/SPMC slave region ID control registers located in the SPMI/SPMC bus interface of the IMP-subsystem.

To set the region ID on the slave side where the VDSP interfaces with the bus IR domain AXI bus via AXI Slave 64-bit, set the bit corresponding to the allowed region ID in the RGID(R,W)_AXIDSPx register to "1". Also write 0xFFFF to the RGID(R,W)_DSPPS register.

To set the region ID on the master side where the VDSP interfaces with the bus IR domain AXI bus via AXI Master 128-bit or AXI Master 256-bit, set the region ID value in the RGIDM_DSPx0 or RGIDM_DSPx1 registers.

The SPMC slave region ID control register need to be set when VDSP accesses to the SPMI/SPMC memory.

The following figures show the Region ID implementation for VDSP subsystem.

Refer to the chapter “User's Manual 18.2 Register description" for the details of RGID(R,W)_AXIDSPx , RGID(R,W)_DSPPS , and RGIDM_DSPxx registers. Refer to the chapter “User's Manual 18.5.9 Region ID protection” about the Region ID protection.

Refer to the chapter "U34.10.2 Secure region for IMP-X7" for the details of the SPMI/SPMC slave region ID control registers.

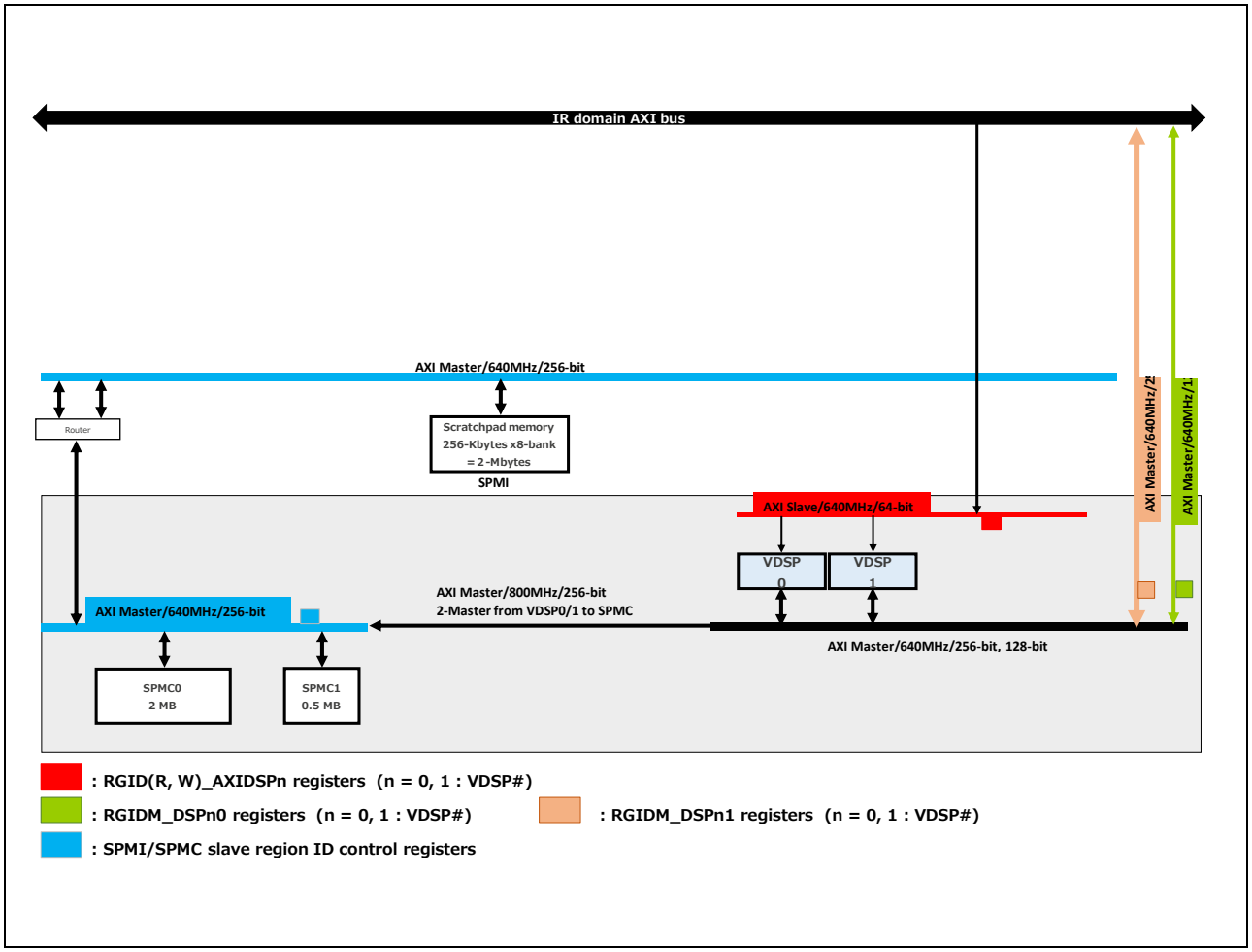



**Figure U39.5.2 Region ID registers implementation for VDSP**

# U61. DDR PHY

## U61.1 Overview

DDRPHY provides the functionality to interface DBSC5 (external bus controller) to a high-speed DRAM device - LPDDR5 / LPDDR4X. DDRPHY consists of 2channel block (DDRPHYn, n=0/1), and each channel has x16bitDQ.

### U61.1.1 Functions and Features

**(1) Basic Functions**

- Alignment of DRAM commands to the DRAM clock
- Timing control for write data and write data clock/strobe placement
- Read data capture using DRAM generated data strobe and re-synchronization
- Support multiple leveling/training function
- Support DBI mode only
- Support IO calibration
- Support DRAM Backup Mode
- Support DFS (Dynamic Frequency Scaling) function

**(2) LPDDR5 Configuration**

- Compliant with JEDEC JESD209-5B
- Supported maximum bit width is 32bit – x32bit x 1pcs
- Supported rank is 1rank and 2rank
- Supported Data Rate 1600Mbps(min) - 6400Mbps(max)@1rank / 5500Mbps(max)@2rank (T.B.D)
- Supported total memory size is up to 8GByte

**(3) LPDDR4X Configuration**

- Compliant with JEDEC JESD209-4C and JEDEC JESD209-4-1A
- Supported maximum bit width is 32bit – x32bit x 1pcs
- Supported rank is 1rank and 2rank
- Supported Data Rate is 1066Mbps(min) - 4266Mbps(max)@1rank / 4266Mbps(max)@2rank

  Supported total memory size is up to 8GByte

### U61.1.2 Block Diagram

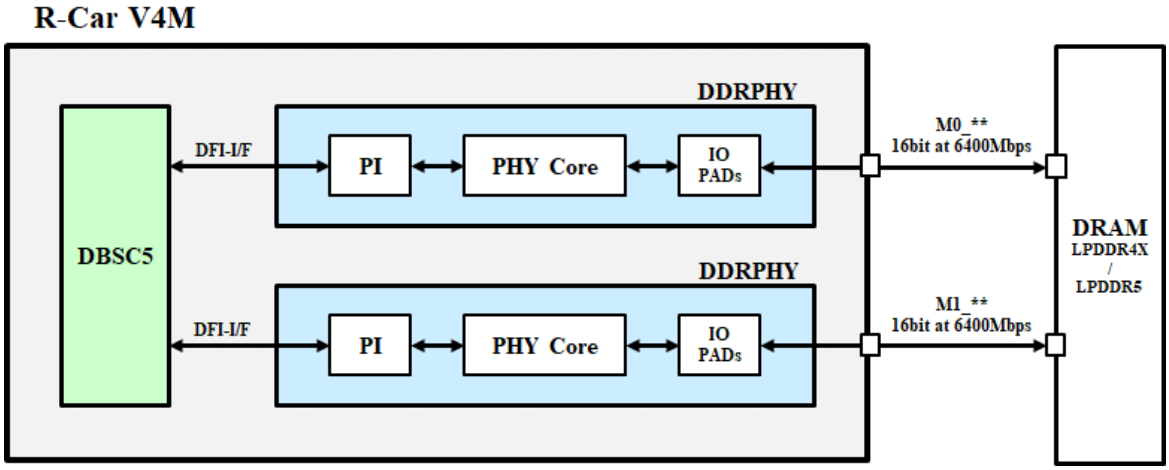



**Figure U61-1. Block Diagram of DDRPHY for R-Car V4M**

### U61.1.3 External Pins

**Table U61-1. DDRPHY External Pin List**

| Pin Name | I/O | Function | Target Device |
|---|---|---|---|
| M0_CK_T | O | Differential Clock T for Ch0 | LPDDR4X/LPDDR5 |
| M0_CK_C | O | Differential Clock C for Ch0 | LPDDR4X/LPDDR5 |
| M0_LP5CS0_LP4XCKE0 | O | Clock Enable for Ch0 - Rank0 : LPDDR4X<br>Chip Select for Ch0 - Rank0 : LPDDR5 | LPDDR4X/LPDDR5 |
| M0_LP5CS1_LP4XCKE1 | O | Clock Enable for Ch0 – Rank1 : LPDDR4X<br>Chip Select for Ch0 – Rank1 : LPDDR5 | LPDDR4X/LPDDR5 |
| M0_LP4XCS0 | O | Chip Select for Ch0 - Rank0 | LPDDR4X |
| M0_LP4XCS1 | O | Chip Select for Ch0 - Rank1 | LPDDR4X |
| M0_CA0 | O | Command/Address 0 for Ch0 | LPDDR4X/LPDDR5 |
| M0_CA1 | O | Command/Address 1 for Ch0 | LPDDR5 *1 |
| M0_CA2 | O | Command/Address 2 for Ch0 | LPDDR4X/LPDDR5 |
| M0_CA3 | O | Command/Address 3 for Ch0 | LPDDR4X/LPDDR5 |
| M0_CA4 | O | Command/Address 4 for Ch0 | LPDDR4X/LPDDR5 |
| M0_CA5 | O | Command/Address 5 for Ch0 | LPDDR4X/LPDDR5 |
| M0_CA6 | O | Command/Address 6 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ00 | I/O | Data Input/Output 00 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ01 | I/O | Data Input/Output 01 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ02 | I/O | Data Input/Output 02 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ03 | I/O | Data Input/Output 03 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ04 | I/O | Data Input/Output 04 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ05 | I/O | Data Input/Output 05 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ06 | I/O | Data Input/Output 06 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ07 | I/O | Data Input/Output 07 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ08 | I/O | Data Input/Output 08 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ09 | I/O | Data Input/Output 09 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ10 | I/O | Data Input/Output 10 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ11 | I/O | Data Input/Output 11 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ12 | I/O | Data Input/Output 12 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ13 | I/O | Data Input/Output 13 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ14 | I/O | Data Input/Output 14 for Ch0 | LPDDR4X/LPDDR5 |
| M0_DQ15 | I/O | Data Input/Output 15 for Ch0 | LPDDR4X/LPDDR5 |
| M0_WCK0_T | I/O<br>O | Differential Data Strobe T of Lower Byte for Ch0 : LPDDR4X<br>Differential Data Clock T of Lower Byte for Ch0 : LPDDR5 | LPDDR4X/LPDDR5 |
| M0_WCK0_C | I/O<br>O | Differential Data Strobe C of Lower Byte for Ch0 : LPDDR4X<br>Differential Data Clock C of Lower Byte for Ch0 : LPDDR5 | LPDDR4X/LPDDR5 |
| M0_WCK1_T | I/O<br>O | Differential Data Strobe T of Upper Byte for Ch0 : LPDDR4X<br>Differential Data Clock T of Upper Byte for Ch0 : LPDDR5 | LPDDR4X/LPDDR5 |
| M0_WCK1_C | I/O<br>O | Differential Data Strobe C of Upper Byte for Ch0 : LPDDR4X<br>Differential Data Clock C of Upper Byte for Ch0 : LPDDR5 | LPDDR4X/LPDDR5 |
| M0_RDQS0_T | I | Differential Read Data Strobe T of Lower Byte for Ch0 | LPDDR5 |
| M0_RDQS0_C | I | Differential Read Data Strobe C of Lower Byte for Ch0 | LPDDR5 |
| M0_RDQS1_T | I | Differential Read Data Strobe T of Upper Byte for Ch0 | LPDDR5 |
| M0_RDQS1_C | I | Differential Read Data Strobe C of Upper Byte for Ch0 | LPDDR5 |
| M0_DMI0 | I/O | Data Mask Inversion of Lower Byte Ch0 | LPDDR4X/LPDDR5 |
| M0_DMI1 | I/O | Data Mask Inversion of Upper Byte Ch0 | LPDDR4X/LPDDR5 |
| M0_ZQ | - | ZQ Calibration Reference Terminal for Ch0 | LPDDR4X/LPDDR5 |

| | | | |
|---|---|---|---|
| | | Connect to VSS through a 240ohm +/- 1% Resistor | |
| M0_ATB0 | I/O | Analog Test BUS0 for Ch0 | Open |
| M0_ATB1 | I/O | Analog Test BUS1 for Ch0 | Open |
| M01_BKUP | I | Buck Up Mode Control for Ch0/Ch1 | Power Device |
| M1_CK_T | O | Differential Clock T for Ch1 | LPDDR4X/LPDDR5 |
| M1_CK_C | O | Differential Clock C for Ch1 | LPDDR4X/LPDDR5 |
| M1_LP5CS0_LP4XCKE0 | O | Clock Enable for Ch1 - Rank0 : LPDDR4X<br>Chip Select for Ch1 - Rank0 : LPDDR5 | LPDDR4X/LPDDR5 |
| M1_LP5CS1_LP4XCKE1 | O | Clock Enable for Ch1 – Rank1 : LPDDR4X<br>Chip Select for Ch1 – Rank1 : LPDDR5 | LPDDR4X/LPDDR5 |
| M1_LP4XCS0 | O | Chip Select for Ch1 - Rank0 | LPDDR4X |
| M1_LP4XCS1 | O | Chip Select for Ch1 - Rank1 | LPDDR4X |
| M1_CA0 | O | Command/Address 0 for Ch1 | LPDDR4X/LPDDR5 |
| M1_CA1 | O | Command/Address 1 for Ch1 | LPDDR5 *1 |
| M1_CA2 | O | Command/Address 2 for Ch1 | LPDDR4X/LPDDR5 |
| M1_CA3 | O | Command/Address 3 for Ch1 | LPDDR4X/LPDDR5 |
| M1_CA4 | O | Command/Address 4 for Ch1 | LPDDR4X/LPDDR5 |
| M1_CA5 | O | Command/Address 5 for Ch1 | LPDDR4X/LPDDR5 |
| M1_CA6 | O | Command/Address 6 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ00 | I/O | Data Input/Output 00 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ01 | I/O | Data Input/Output 01 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ02 | I/O | Data Input/Output 02 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ03 | I/O | Data Input/Output 03 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ04 | I/O | Data Input/Output 04 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ05 | I/O | Data Input/Output 05 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ06 | I/O | Data Input/Output 06 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ07 | I/O | Data Input/Output 07 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ08 | I/O | Data Input/Output 08 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ09 | I/O | Data Input/Output 09 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ10 | I/O | Data Input/Output 10 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ11 | I/O | Data Input/Output 11 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ12 | I/O | Data Input/Output 12 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ13 | I/O | Data Input/Output 13 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ14 | I/O | Data Input/Output 14 for Ch1 | LPDDR4X/LPDDR5 |
| M1_DQ15 | I/O | Data Input/Output 15 for Ch1 | LPDDR4X/LPDDR5 |
| M1_WCK0_T | I/O<br>O | Differential Data Strobe T of Lower Byte for Ch1 : LPDDR4X<br>Differential Data Clock T of Lower Byte for Ch1 : LPDDR5 | LPDDR4X/LPDDR5 |
| M1_WCK0_C | I/O<br>O | Differential Data Strobe C of Lower Byte for Ch1 : LPDDR4X<br>Differential Data Clock C of Lower Byte for Ch1 : LPDDR5 | LPDDR4X/LPDDR5 |
| M1_WCK1_T | I/O<br>O | Differential Data Strobe T of Upper Byte for Ch1 : LPDDR4X<br>Differential Data Clock T of Upper Byte for Ch1 : LPDDR5 | LPDDR4X/LPDDR5 |
| M1_WCK1_C | I/O<br>O | Differential Data Strobe C of Upper Byte for Ch1 : LPDDR4X<br>Differential Data Clock C of Upper Byte for Ch1 : LPDDR5 | LPDDR4X/LPDDR5 |
| M1_RDQS0_T | I | Differential Read Data Strobe T of Lower Byte for Ch1 | LPDDR5 |
| M1_RDQS0_C | I | Differential Read Data Strobe C of Lower Byte for Ch1 | LPDDR5 |
| M1_RDQS1_T | I | Differential Read Data Strobe T of Upper Byte for Ch1 | LPDDR5 |
| M1_RDQS1_C | I | Differential Read Data Strobe C of Upper Byte for Ch1 | LPDDR5 |
| M1_DMI0 | I/O | Data Mask Inversion of Lower Byte Ch1 | LPDDR4X/LPDDR5 |
| M1_DMI1 | I/O | Data Mask Inversion of Upper Byte Ch1 | LPDDR4X/LPDDR5 |
| M1_ZQ | - | ZQ Calibration Reference Terminal for Ch1<br>Connect to VSS through a 240ohm +/- 1% Resistor | LPDDR4X/LPDDR5 |

| | | | |
|---|---|---|---|
| M1_RESET_N | O | RESET for Ch0/Ch1 | LPDDR4X/LPDDR5 |
| M1_ATB0 | I/O | Analog Test BUS0 for Ch1 | Open |
| M1_ATB1 | I/O | Analog Test BUS1 for Ch1 | Open |
| M1_PLLTEST | O | PLL Test Monitor for Ch0/Ch1 | Open |

*1: Mx_CA1 can't be used for LPDDR4X. Please use Mx_CA6 instead of Mx_CA1.

### U61.1.4 Connected Module

**Table U61-2. Connected Modules**

| Module Name | Connected Module Name | Connected Module Function |
|---|---|---|
| DDRPHY | DBSC5 | External Bus Controller for SDRAM |
| | CPG | Output clocks |
| | Module Standby | Control to stop clocks |
| | Software Reset | Execute software reset |

### U61.1.5 Register Configuration

DDRPHY Parameters are assigned to bit and bit fields of DDRPHY Registers, and access to DDRPHY Registers is through access to DBSC5 Registers.
Table U61-3 shows the DBSC5 Registers which are used to access to DDRPHY Registers.

**Table U61-3. DBSC Registers to access to DDRPHY Registers**

| Register Name | Abbreviation | Bit Name | R/W | Address | Initial value | Access Size |
|---|---|---|---|---|---|---|
| PHY Unit Register Address for DDRPHY0 | dbpdrga0 | pra0[15:0] | R/W | H'E679_0624 | H'0000 0000 | 32 |
| PHY Unit Register Access for DDRPHY0 | dbpdrgd0 | prd0[31:0] | R/W | H'E679_0628 | H'0000 0000 | 32 |
| PHY Unit Register Address for DDRPHY1 | dbpdrga1 | pra1[15:0] | R/W | H'E679_0664 | H'0000 0000 | 32 |
| PHY Unit Register Access for DDRPHY1 | dbpdrgd1 | prd1[31:0] | R/W | H'E679_0668 | H'0000 0000 | 32 |

## U61.2 Register Description

For detailed DDRPHY Register Description, refer to the following document.

- ***LPDDR5 : R-Car V4M Initialization Sequence of LPDDR5 interface***
- ***LPDDR4X : R-Car V4M Initialization Sequence of LPDDR4X interface***

## U61.3 Operation

### U61.3.1 Initialization and Training Sequence

**(1) Main Flow of Initialization Sequence**

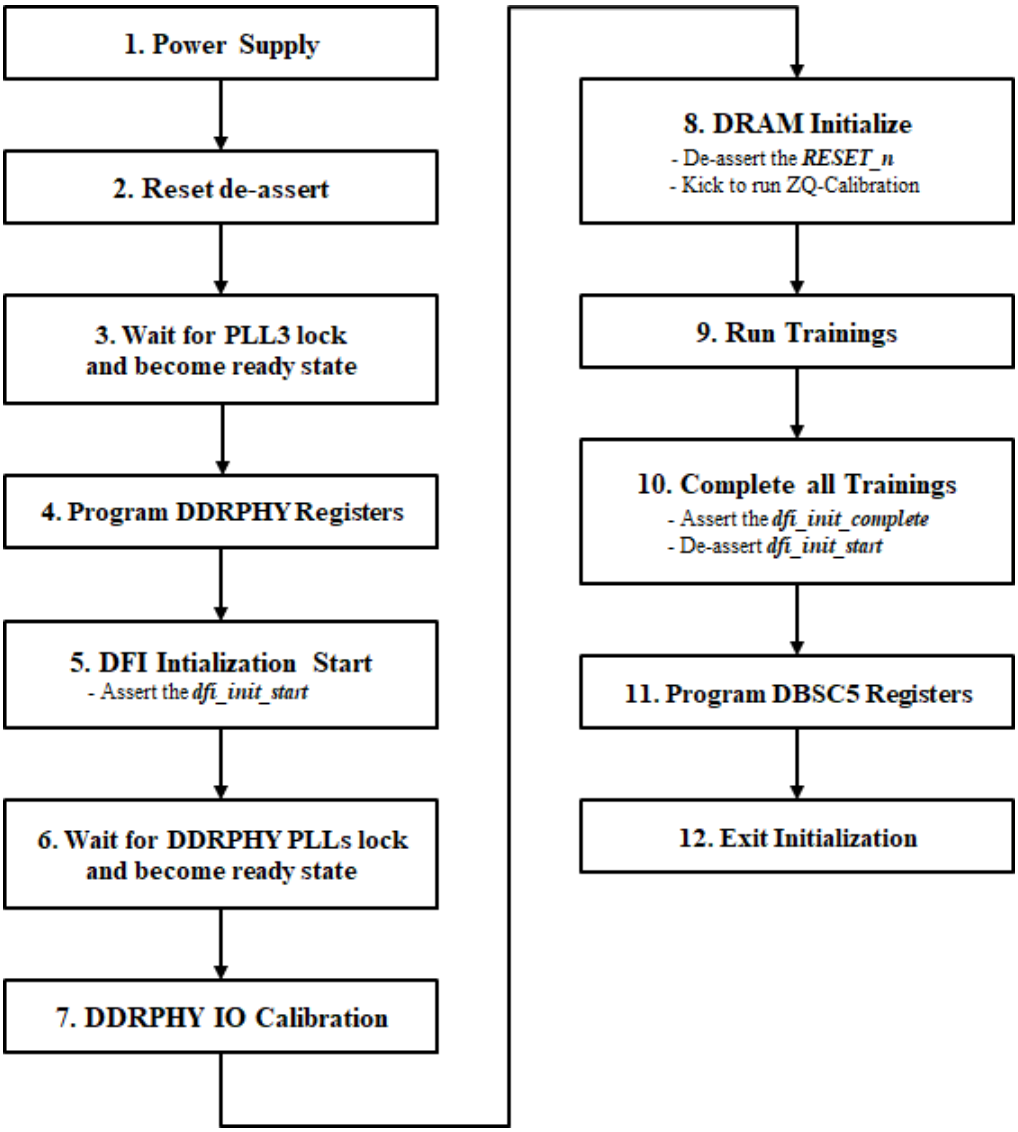



Figure U61-2. Main Flow of Initialization Sequence

**(2) Training Sequence**

Refer to the following Application Note.

- ***LPDDR5: R-Car V4M Initialization Sequence of LPDDR5 interface***
- ***LPDDR4X: R-Car V4M Initialization Sequence of LPDDR4X interface***

### U61.3.2 DRAM Backup Mode

DRAM: Self-Refresh + DDRPHY: Power off (VDDQ only)

DRAM devices have a function called Self Refresh, which automatically refreshes the memory contents without being controlled by other devices while there is no need to access DRAM. In this document, making DRAM enter the Self Refresh Mode is called the DRAM Backup function.
While DRAM is being backed up, DRAM cannot be accessed but the DRAM controller does not need to control DRAM refresh operations; the power supply to the controller can be turned off and the entire system can enter a low-power state. To shift DRAM to the Self Refresh mode, the DRAM controller should issue a self-refresh command to the target DRAM. Driving the CKE signal high or driving the RESET# signal low makes DRAM exit the Self Refresh Mode and cancels the DRAM Backup Mode.

For detailed DRAM Backup Mode function, refer to the following document.

- ***33.4.1 (3) SDRAM Power-supply Backup Mode***
- ***33.4.1 (4) Recovery from SDRAM Power-supply Backup Mode***

### U61.3.3 Clock Supply Path for DDRPHY

Figure U61-3 shows clock supply path to the external DRAM and the DFI-IF clock between DBSC5 and DDRPHY.

The clock to external DRAM is generated by PLL3 – ZB3φ. ZB3φ is multiplied by PLL in DDRPHY. PLL3 can be set in both 'Integer multiplication mode' and 'Fractional multiplication mode'.
For detailed PLL3 setting Description, refer to the following section.

- ***8.1.5.3 PLL3 Multiplication Ratio***

The clock for the DFI-IF is ZB3D2φ. It's generated by dividing ZB3φ. The divider ratio is set by DFICKM in the DBSC5 register domain.
- DFICKM (DBSYSCONF2A[9])
  0 : LPDDR4X - ZB3D2φ = ZB3φ x 1/2
  1 : LPDDR5 - ZB3D2φ = ZB3φ x 1
Note) Clock Monitor setting for ZB3D2φ need to be set according to DFICKM.

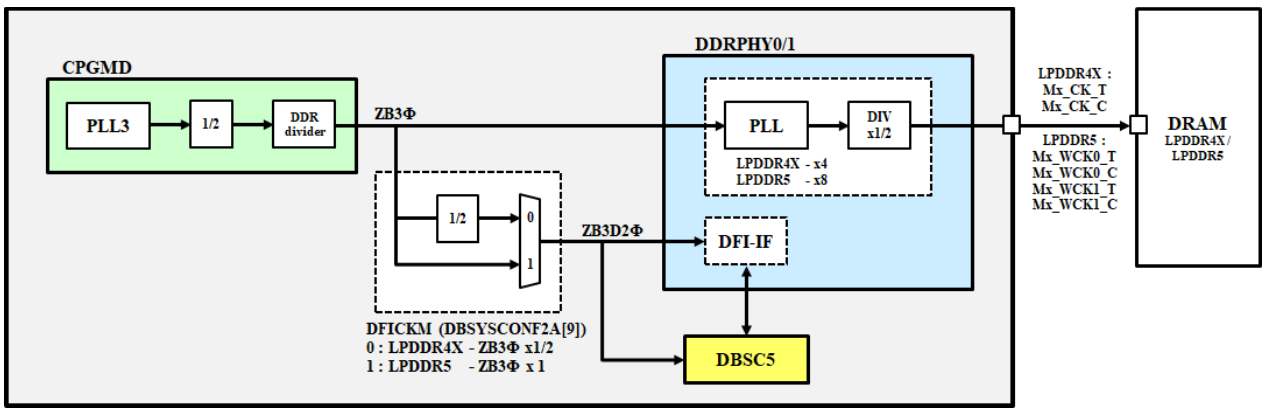



**Figure U61-3 Clock supply path to external DRAM**

### U61.3.4 DFS

#### (1) DFS Function Overview

DDRPHY supports DFS (Dynamic Frequency Scaling) function for Low Power mode. DRAM clock frequency changes between high-speed operation on Full-Run mode and medium speed operation on Low-Power mode automatically according to the system condition to optimize the power.

- DFS operation is realized by executing the sequence shown in 'Figure U61-5' by System-SW.
- In the case of using DFS function, initialization training is performed both at high speed and at medium speed. Both registers and training parameters are stored in DDRPHY and these parameters can be changed automatically according to the frequency mode during DFS operation.
- When the frequency is changed by DFS operation, it is necessary to enter the DRAM into Self-Refresh with Power-Down mode because the clock supply to the DRAM is temporarily stopped.
- It is necessary to execute some pre-processing before DFS operation.
  i.e) Disable DRAM access, Disable Periodic-training

#### (2) DFS Frequency Specification

Table U61-4 shows the frequency combinations that can be set in the DFS function. Frequency changing is realized by setting the DDR divider after the PLL3 on the CPGMD.

**Table U61-4. DFS Frequency**

| DRAM | Frequency Mode | Mode | | | |
|---|---|---|---|---|---|
| | | Full-Run Mode | | Low-Power Mode | |
| | | DDR divider N | High Speed | DDR divider N | Medium Speed |
| LPDDR5 | 6400Mbps | 2 | 6400Mbps | 8 | 1600Mbps |
| | 6000Mbps | 2 | 6000Mbps | 6 | 2000Mbps |
| | 5500Mbps | 2 | 5500Mbps | 6 | 1833Mbps |
| | 4800Mbps | 2 | 4800Mbps | 6 | 1600Mbps |
| LPDDR4X | 4266Mbps | 2 | 4266Mbps | 6 | 1422Mbps |
| | 3200Mbps | 2 | 3200Mbps | 4 | 1600Mbps |
| | 3733Mbps | 2 | 3733Mbps | 6 | 1244Mbps |

Note) LPDDR4X - 2133Mbps mode can't be used on DFS function.

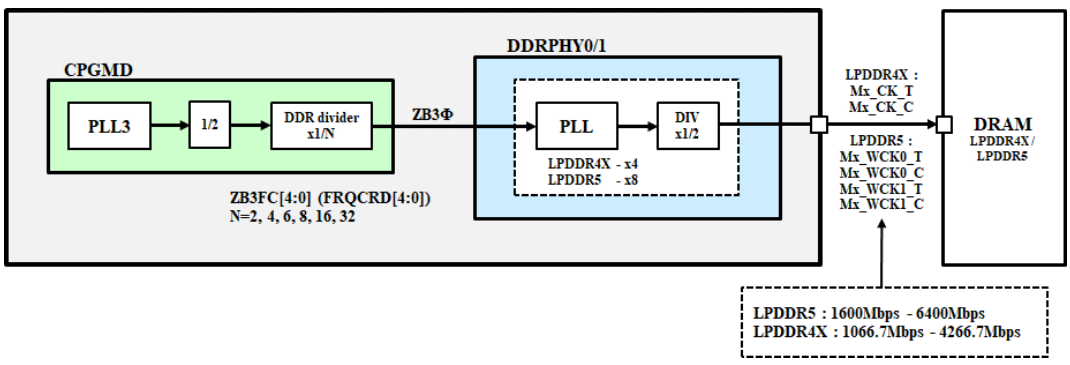



**Figure U61-4 DFS Frequency Generation**

### (3) DFS Sequence

Figure U61-5 shows the DFS sequence.

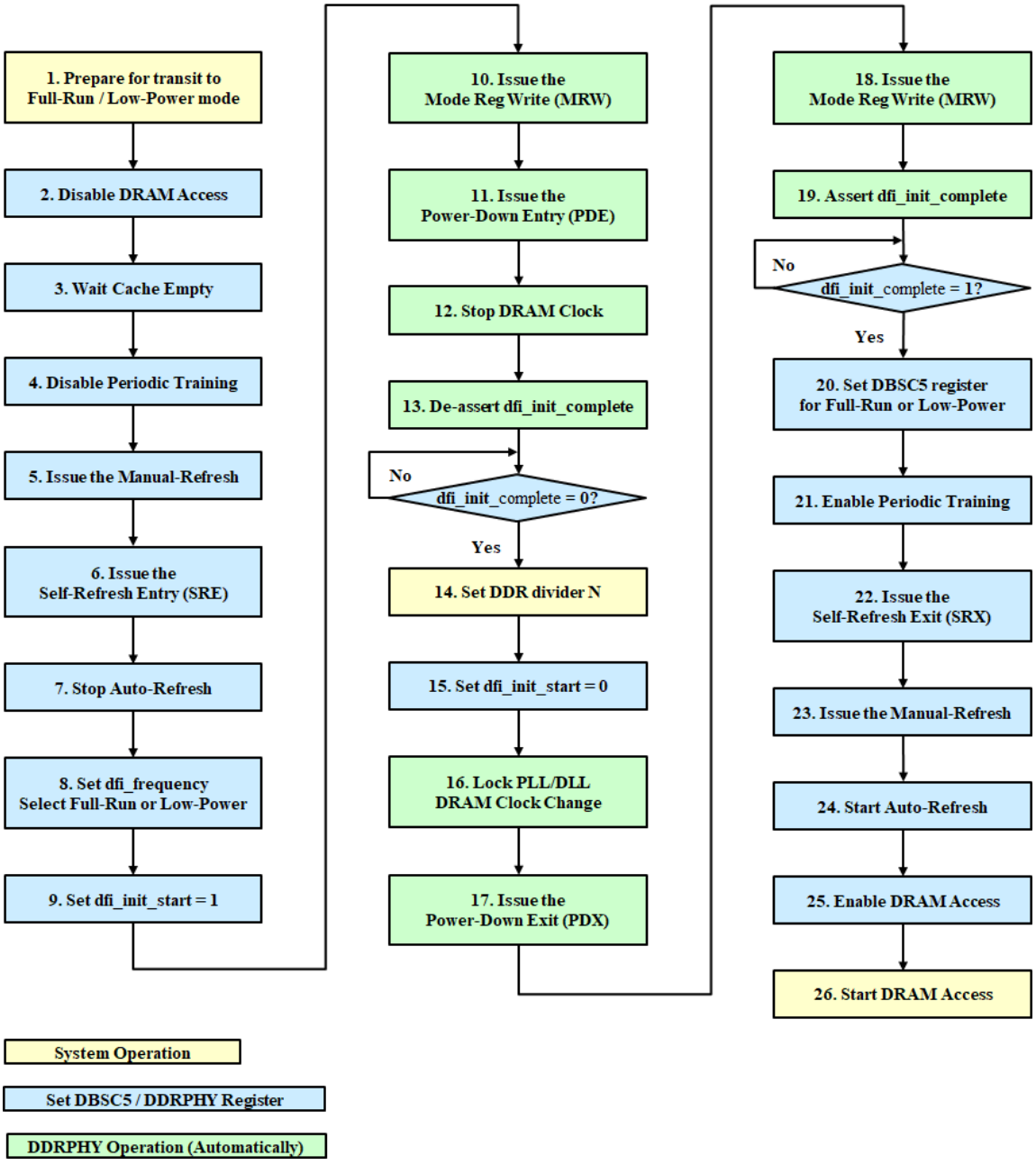



**Figure U61-5 DFS Sequence**

## U61.4 Usage Notes

### U61.4.1 Connection of DRAM

#### (1) Connection of LPDDR5

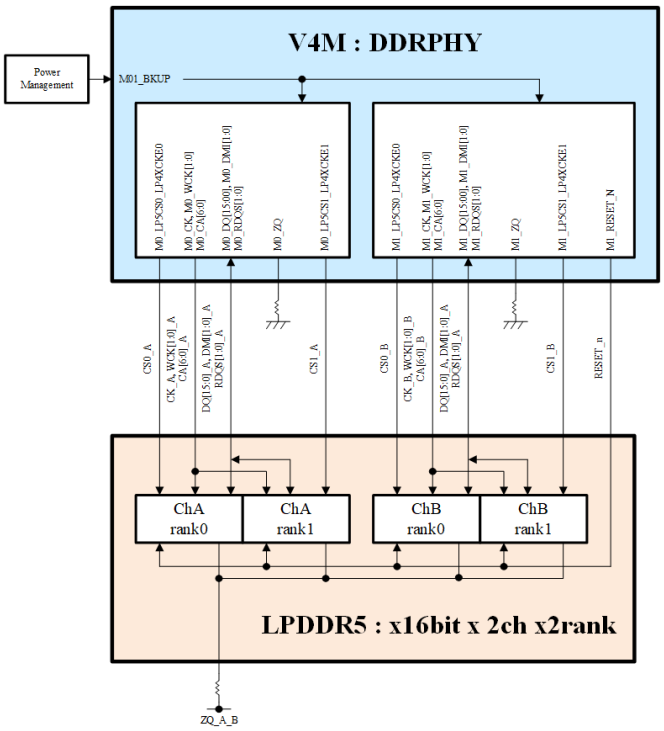



**Figure U61-6 LPDDR5 connection Example: x32bit x2rank x1pcs**

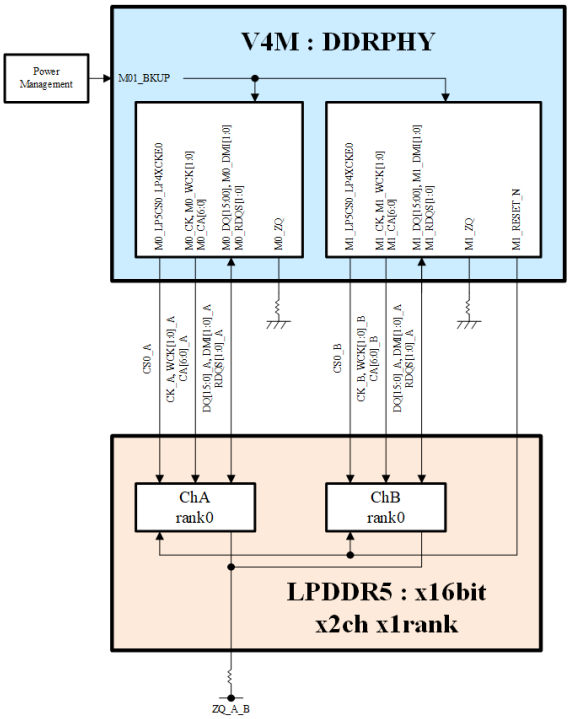



**Figure U61-7 LPDDR5 connection Example: x32bit x 1rank x1pcs**

### (2) Connection of LPDDR4X

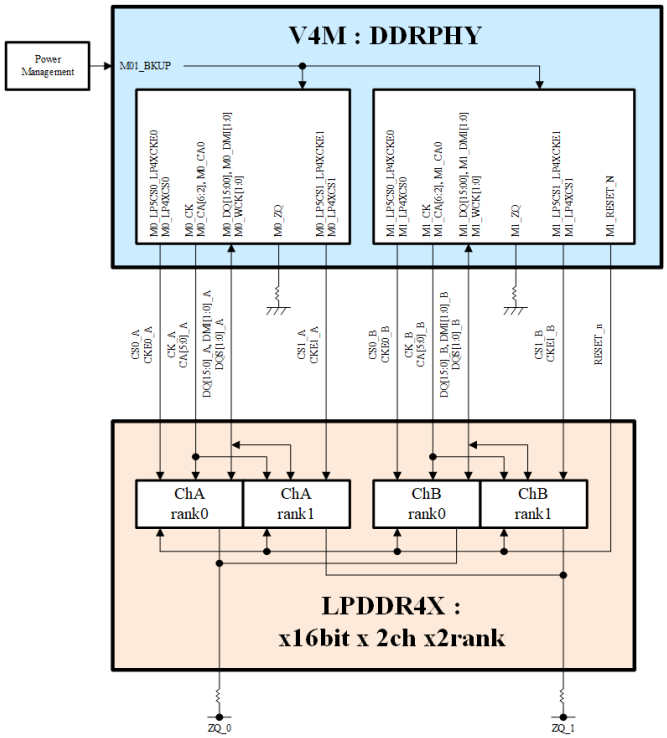



Figure U61-4-3 LPDDR4X connection Example: x32bit x2rank x1pcs

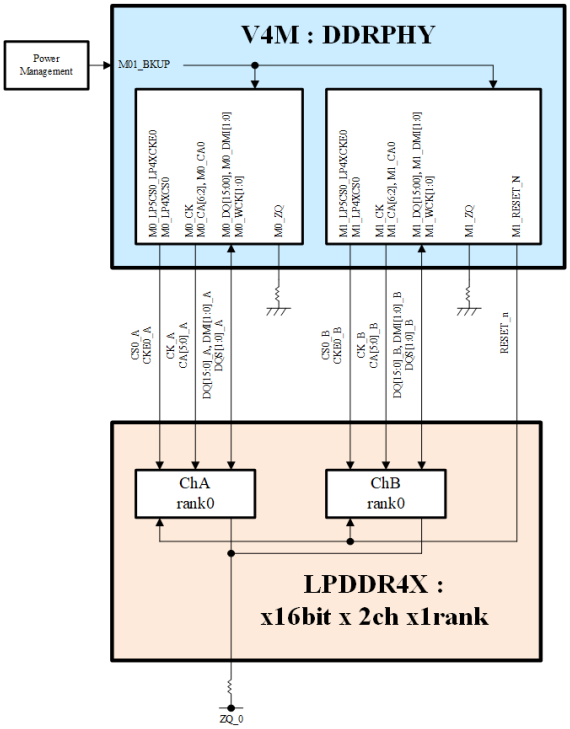



**Figure U61-8 LPDDR4X connection Example: x32bit x1rank x1pcs**

### U61.4.2 Mode Setting

#### (1) DRAM Mode

DRAM Mode is set by MD pin – MD22/MD27.

**Table U61-5. DRAM Mode Setting**

| MD27 | MD22 | DRAM Mode |
|---|---|---|
| 0 | 0 | LPDDR5 |
| 0 | 1 | LPDDR4X |
| 1 | 0 | Reserved |
| 1 | 1 | Reserved |

#### (2) DRAM Frequency Mode

DRAM Frequency Mode is set by MD pin – MD17/MD19. PLL. PLL3 and clock dividers for DDRPHY clock are set by CPG module according to EXTAL frequency and below table.

**Table U61-6. DRAM Frequency Mode Setting**

| MD19 | MD17 | LPDDR5 | LPDDR4X |
|---|---|---|---|
| 0 | 0 | 6400Mbps | 4266Mbps |
| 0 | 1 | 6000Mbps | 3733Mbps |
| 1 | 0 | 5500Mbps | 3200Mbps |
| 1 | 1 | 4800Mbps | 2133Mbps |

### U61.4.3 Module Stop and Software Reset

Module Standby (MSTP) and Software Reset (SRST) are supplied to pair of DDRPHY0/DDRPHY1 by Reset module. SRST initializes DDRPHY module by setting the software reset register bit. MSTP stops clock supply to DDRPHY by setting module stop register bit.

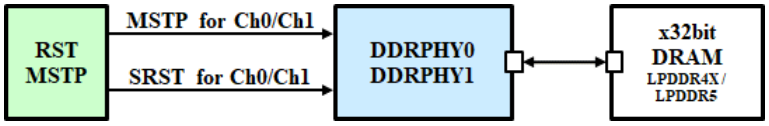



**Figure U61-9 Module Stop and Software Reset supply to DDRPHY**

### U61.4.4 Signal Swapping Function

The following rules apply to swapping data signaling between DDRPHY and DRAM connection.

- DQ Byte group can be swapped within Ch. All signals related to a Byte group are re-mapped when DQ Byte swapping is enabled. DQ Byte group is consisted of 8xDQ/1xDMI/1xWCK/1xRDQS.
- DQ bits and DMI can be swapped within DQ Byte group.
- CA signals can be swapped within Ch.
- CS signals can be swapped within Ch.
- CKE signals can be swapped within Ch.

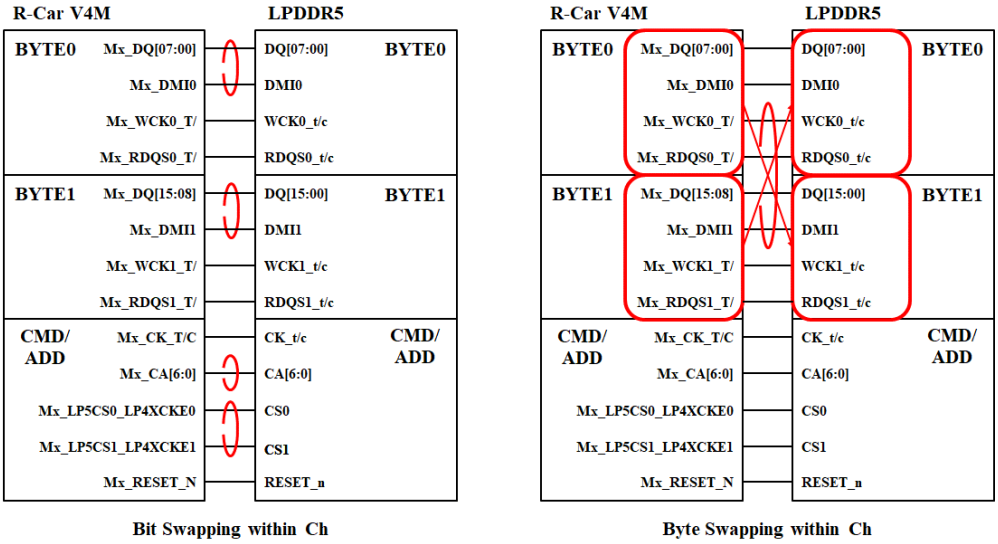
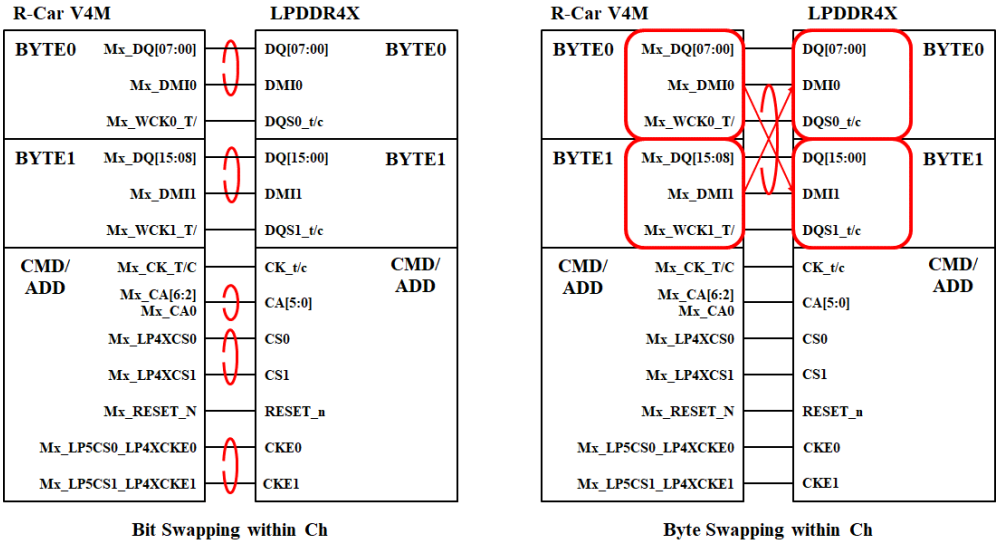



**Figure U61-10 LPDDR5 Swapping Function**




Note) : Mx_CA1 can't be used for LPDDR4X. Please use Mx_CA6 instead of Mx_CA1.

**Figure U61-11 LPDDR4X Swapping Function**

### U61.4.5 DBI Mode

DBI (Data Bus Inversion) is supported in DDRPHY. DDRPHY requires DBSC5 to perform DBI generation and decode, DDRPHY only passes the DBI information between DBSC5 and DRAM devices. DBI mode should be enable from the point of view of supply noise reduction.

## U61.5 Safety Mechanisms

DDRPHY has no safety mechanisms. DRAM access errors are protected by ECC protection function which DBSC5 performs.

For detailed ECC protection function, refer to the following section.

- ***33.5 Safety Mechanism***

R-Car V4M series User's Manual: Hardware

Publication Date: Rev.0.40 Dec. 29. '22
Rev.0.51 Oct. 31. '23
Rev.0.70 Jul. 24. '24
Rev.0.80 Jan. 31. '25
Rev.0.81 May 30. '25

Published by: Renesas Electronics Corporation

# R-Car V4M series User's Manual: Hardware

Renesas Electronics Corporation

R19UH0217EJ0081